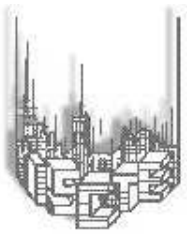


Instituto Superior
de Ciências do Trabalho e da Empresa

Um Modelo de Escalonamento Multi-Agente na Empresa Estendida

Joaquim António Marques dos Reis
(Mestre em Engenharia Mecânica)

Dissertação para obtenção do Grau de Doutor
em Ciências e Tecnologias de Informação
(especialidade de Inteligência Artificial)

Orientador:
Nuno João Neves Mamede (IST/UTL)

Co-orientador:
Henrique José Rocha O'Neill (ISCTE)

Lisboa, Fevereiro de 2000

(versão revista e corrigida em Abril de 2002)

à Laura

Agradecimentos

Ao meu orientador, Professor Nuno Mamede, agradeço toda a orientação técnica e científica dada durante o desenvolvimento do presente trabalho, bem como o apoio que proporcionou, com os seus conselhos e experiência prática.

Ao meu co-orientador, Professor Henrique José Rocha O' Neill, agradeço pelo apoio e sugestões proporcionados no enquadramento prático das ideias progressivamente desenvolvidas no trabalho, nomeadamente no que respeita à realização na prática do modelo da Empresa Estendida e em particular no projecto AITEAR.

Aos meus colegas de trabalho Professores Luís Botelho, Pedro Ramos e Filipe Santos, agradeço as ideias, as sugestões e as inúmeras conversas e discussões sobre temas variados, em particular Inteligência Artificial e Inteligência Artificial Distribuída.

Agradeço também aos Professores João Pavão Martins e Ernesto Marques Morgado, do Instituto Superior Técnico, a quem devo o meu interesse e envolvimento na área de Inteligência Artificial e ao Professor Helder Ferreira Coelho, da Faculdade de Ciências de Lisboa, a quem devo uma perspectiva abrangente da Inteligência Artificial Distribuída.

Por fim, agradeço ao Instituto Superior de Ciências do Trabalho e da Empresa, instituição onde trabalho, pela infra-estrutura e recursos disponibilizados.

Sumário

Este trabalho descreve um modelo de escalonamento de tarefas logísticas no contexto da cadeia de fornecimento (*supply chain*) de produção e distribuição, numa perspectiva multi-agente de Inteligência Artificial Distribuída e assumindo o enquadramento Empresa Estendida.

O escalonamento, ou *scheduling*, convencionalmente definido como o problema da afectação de recursos no tempo a um conjunto de tarefas a realizar de forma coordenada e tradicionalmente limitado ao contexto das actividades produtivas intra-empresa, é estendido a um contexto inter-empresa, ou inter-organizacional.

O modelo desenvolvido é organizado em dois níveis: o *nível físico* e o *nível virtual*. No nível físico modelam-se aspectos interessantes para o escalonamento convencional, nomeadamente tempo, capacidade, recursos e tarefas de produção e distribuição, enquadrados no contexto da cadeia de fornecimento cooperativa. O nível virtual é um nível de decisão e contém agentes que representam as empresas participantes na cadeia. Os agentes são responsáveis por gerir o tempo e a capacidade dos recursos do nível físico, estão interligados por relações cliente-fornecedor e podem comunicar entre si. Resumidamente, no nível físico ocorrem tarefas e fluxos de produtos através dos recursos da cadeia e de, e para, o exterior da cadeia; no nível virtual ocorrem fluxos de informação para controlo e coordenação das tarefas e fluxos de produtos pelos agentes.

É definido um protocolo de interacção de alto nível entre os agentes do nível virtual do modelo, apropriado para escalonamento multi-agente. É também proposto um mecanismo de coordenação específico que permite, a cada agente, localmente percepcionar restrições temporais globais rígidas de problemas de escalonamento. Este mecanismo é integrado no protocolo de interacção de alto nível e permite identificar problemas de escalonamento temporalmente super-constrangidos e orientar e limitar a actividade de re-escalonamento.

Mostram-se resultados de aplicação do modelo desenvolvido, assumindo certas restrições. Os resultados são apresentados através de exemplos que demonstram a aplicação do mecanismo de coordenação para escalonamento multi-agente acima referido.

Abstract

This work describes a model for the scheduling of logistic tasks in a production and distribution supply chain context, from a Distributed Artificial Intelligence multi-agent perspective and assuming an Extended Enterprise framework.

Scheduling is conventionally defined as the problem of assigning resources along time to a set of tasks to be executed in a coordinated way. Frequently associated to the intra-enterprise productive activities, scheduling is extended, in the current work, to an inter-entreprise, or inter-organizational, environment.

The model developed is organised in two levels: the *physical level* and the *virtual level*. In the physical level, interesting aspects of conventional scheduling, namely time, capacity, resources and production and distribution tasks, are modeled in a cooperative supply chain context. The virtual level is a decison level and contains agents that represent the enterprise partners in the chain. The agents are responsible for managing the time and the capacity of the resources in the physical level, they are interconnected through client-supplier relationships and they can communicate. Briefly, at the physical level, tasks are executed and product flows occur inside the chain and from, and to, the outside of the chain; at the virtual level, information flows occur for the control and co-ordination of the tasks and the product flows by the agents.

A high level interaction protocol, suitable for multi-agent scheduling, is defined for the agents in the virtual level. A specific co-ordination mechanism is also proposed, that allows agents to perceive locally hard temporal global constraints of scheduling problems. The co-ordination mechanism is integrated with the high level interaction protocol and allows the agents to recognize temporally over-constrained scheduling problems, as well as guiding them, and setting boundaries, in the re-scheduling activity.

Results of the model developed, under some restrictions, are shown. These results are presented through examples that demonstrate the application of the co-ordination mechanism for multi-agent scheduling referred above.

Um Modelo de Escalonamento Multi-Agente na Empresa Estendida

Joaquim António Marques dos Reis

Dissertação para obtenção do Grau de Doutor
em
Ciências e Tecnologias de Informação
(especialidade de Inteligência Artificial)

Instituto Superior de Ciências do Trabalho e da Empresa

Departamento de Ciências e Tecnologias da Informação

Lisboa, Fevereiro de 2000

(versão revista e corrigida em Abril de 2002)

Índice de Matérias

página

Índice de Figuras

página

página

Índice de Tabelas

Glossário de Termos

Português	**Inglês**
(Algoritmo) A-qualquer-tempo	*Anytime*
Actividade-em-arco	*Activity-on-arc*, AOA
Actividade-em-nó	*Activity-on-node, AON*
Acumulador, área de armazenamento intermédia	*Buffer*
Agregação de lotes	*Batching*
Aprendizagem superficial	*Shallow learning*
Armazenamento, existências	*Stock*
Arrefecimento simulado	*Simulated annealing*
Atraso	*Tardiness*
Atraso relativo	*Lateness*
Atributo (de um enquadramento)	*(Frame) Slot*
Avanço	*Earliness*
Bloqueio mútuo (entre tarefas)	*Gridlock*
Cadeia de Fornecimento	*Supply chain*
Chão-de-fábrica	*Shop-floor*
Colocação de restrições	*Constraint posting*
(Procura) Com salto, (procura) em fronteira	*Jumptracking*
Comprimento (do escalonamento)	*Makespan*
Congestionamento	*Bottleneck*
Congestionamento errante	*Shifting bottleneck*
Data de chegada, tempo de chegada	*Ready date, ready time*
Data de lançamento, tempo de lançamento	*Release date, release time*
Data de fim, tempo de fim	*Completion date, completion time*
Data limite mais cedo	*Earliest due date, EDD*
Data limite, data de entrega	*Due date*
Descida do monte	*Hill-descending*
Dimensionamento de lotes	*Lot-sizing*
Dinâmica dos congestionamentos	*Bottleneck dynamics*

(Sistema de produção) Em-tempo	*Just-in-time*
Encaminhamento	*Routing*
Enquadramento	*Frame*
Escalonamento	*Scheduling; schedule*
Escalonamento activo	*Active schedule*
Escalonamento construtivo	*Constructive scheduling*
Escalonamento de lotes	*Lot-scheduling*
Escalonamento por partição de conflitos	*Conflict partition scheduling*, CPS
Escalonamento por reparação	*Repair-based scheduling*
Escalonamento preemptivo	*Preemptive scheduling*
Escalonamento sem-atraso, escalonamento de despacho	*Non-delay schedule, dispatch schedule*
Escalonamento semi-activo	*Semi-active schedule*
Fonte de conhecimento	*Knowledge source*
Gerar-e-testar	*Generate-and-test*
(Estratégia) Glutona	*Greedy*
Gráfico de Gantt, mapa de Gantt	*Gantt chart*
Grafo disjuntivo	*Disjunctive graph*
Heurísticas de despacho míopes	*One-pass myopic dispatch heuristics*
Heurísticas de despacho simples	*Simple dispatch heuristics*
Heurísticas de passagem múltipla	*Multi-pass heuristics*
Heurísticas de uma passagem	*One-pass heuristics*
Heurísticas simples baseadas em congestionamentos	*Simple bottlenect heuristics*
Imposição de consistência	*Consistency enforcement*
Inteligência Artificial Distribuída, IAD	*Distributed Artificial Intelligence, DAI*
Lógica difusa	*Fuzzy logic*
Linha temporal	*Time line*
Lote de processo	*Release batch*
Lote de tarefa	*Operation batch*
Lote de transferência	*Transfer batch*
Manutenção de verdade	*Truth maintenance*
(Sistema de) Manutenção de verdade baseado em assumpções	*Assumption based truth maintenance (system)*
Mapa temporal	*Time map*

(Heurística) Mínimo-de-conflitos	*Min-conflicts*
Olhar-para-a-frente	*Look-ahead*
Planeamento de Requerimentos de Materiais	*Material Requirements Planning, MRP*
Planeamento de Requerimentos de Capacidade	*Capacity Requirements Planning, CRP*
Preempção	*Preemption*
Problema de optimização com restrições, POR	*Constrained optimization problem, COP*
Problema de satisfação de restrições, PSR	*Constraint satisfaction problem, CSP*
Processamento aberto	*Open shop*
Processamento contínuo	*Flow shop*
Processamento intermitente	*Job shop*
Processo	*Job*
Procura em faixa	*Beam search*
Procura em largura	*Breadth-first search*
Procura em profundidade	*Depth-first search*
Procura heurística com restrições distribuída	*Distributed constrained heuristic search*, CHS
Procura pelo melhor	*Best-first search*
Procura tabu	*Tabu search*
Produção Integrada por Computador	*Computer Integrated Manufacturing, CIM*
Quadro preto	*Blackboard*
Raciocínio baseado em casos	*Case based reasoning, CBR*
Ramificar-e-limitar	*Branch-and-bound*
(Regra da) Razão crítica	*Critical ratio, CR*
Rearranjo dinâmico da procura	*Dynamic search rearrangement*
Rede semântica	*Semantic network*
Regra despacho, regra de prioridade	*Dispatch rule, priority rule*
Retrocesso	*Backtracking*
Retrocesso cronológico	*Chronological backtracking*
(Procura com) Salto-para-trás	*Backjumping*
Símbolo	*Token*
Sem-espera	*No-wait*
Sistema de produção	*Production system*

Sistema pericial	*Expert system*
Subdivisão de lotes	*Lot-splitting*
Subida do monte	*Hill-climbing*
Tarefa	*Operation, task, activity*
Tecnologia de Produção Optimizada	*Optimized Production Technology, OPT*
Tempo de fluxo	*Flow time*
Tempo de folga restante, (regra da) folga mínima	*Slack time remaining, STR*
Tempo de preparação, preparação	*Setup time, setup*
Tempo de processamento mais curto, (regra do) menor tempo de processamento	*Shortest processing time, SPT*
Trabalhos em curso	*Work-in-process, work-in-progress, WIP*
Valor menos limitativo	*Least constraining value, LCV*
Verificação para a frente	*Forward checking*

Convenções Tipográficas

(i) De um modo geral usam-se caracteres especiais de uma fonte "Courier" em expressões ou fórmulas que se destacam assim do texto normal. Objectos simples são denotados por variáveis compostas por letras deste tipo de fonte. Exemplos: A, a, W, w, p, r, t, q.

(ii) Objectos compostos (tuplos ou conjuntos) são denotados por variáveis compostas por letras de um tipo de fonte "Colonna MT". Exemplos: $\mathbb{pq}$, $\mathbb{E}$, $\mathbb{O}$, $\mathbb{P}$, $\mathbb{v}$, $\mathbb{pq}$, $\mathbb{E}$, $\mathbb{O}$.

(iii) Tuplos (pares ordenados, triplos ordenados, etc.) são denotados por variáveis compostas por letras de um tipo de fonte "Colonna MT" em texto regular. Exemplos: $\mathbb{pq}$, $\mathbb{E}$, $\mathbb{O}$. Usam-se os símbolos de menor e maior (< e >) para delimitar os elementos de um tuplo e vírgulas (,) para separar esses elementos. Exemplos: <p,q>, <$\mathbb{V}_x$,$\mathbb{V}_y$,$\mathbb{pm}$>, <$\mathbb{e}_s$,$\mathbb{e}_e$,cval>.

(iv) Conjuntos são denotados por variáveis compostas por letras de um tipo de fonte "Colonna MT" em texto "itálico". Exemplos: $\mathbb{P}$, $\mathbb{O}$, $\mathbb{v}$, $\mathbb{r}$. Usam-se chavetas ({ e }) para delimitar os elementos de um conjunto e vírgulas (,) para separar esses elementos. Exemplos: {p_1,p_2,p_3}, {$\mathbb{O}_1$,$\mathbb{O}_2$,...,$\mathbb{O}_m$}, {v_1,...,v_i,...,v_n}, {...,r_i,...}.

(v) Objectos a que é associado um valor variável no tempo são denotados por uma variável seguida de um símbolo denotando tempo (geralmente t) delimitado por parêntesis. Exemplos: C(t), $\mathbb{c}$(t), c(t).

(vi) Usam-se sub-índices para denotar objectos diferentes da mesma natureza, ou do mesmo grupo. Quando existe mais de um índice eles são separados por vírgulas (,). Exemplos: $b_{i,j}$, $\mathbb{E}_{x,y}$.

(vii) Usam-se super-índices para denotar objectos diferentes da mesma natureza que são componentes de objectos compostos diferentes, referindo-se o super-índice ao objecto composto de que o objecto é componente (sub-índices e super-índices poderão, naturalmente, coexistir na mesma variável). Exemplos: $\mathbb{r}^x$, $\mathbb{p}^x$, r_i^x, $\mathbb{pq}_e^x$, p_s^x.

(viii) Certos índices podem ser usados para denotar valores particulares associados a certos objectos. Por exemplo, os índices e, s e max denotam fim, início e máximo, respectivamente. Exemplos: t_e, t_s, $W_{max}(t)$, $A_{max}(t)$.

1 Introdução

Este capítulo descreve o problema tratado e o seu contexto, descreve os objectivos, resume os capítulos subsequentes e as contribuições dadas no presente trabalho.

1.1 Introdução

Este trabalho tem como objectivo principal o desenvolvimento de um modelo de escalonamento de tarefas logísticas no contexto das redes cooperativas de produção e distribuição, numa perspectiva inter-empresa e multi-agente.

O problema do escalonamento, ou *scheduling* (ver, por exemplo, [Baker 1974], [Kan 1976], [Morton 1993] ou [Blazewicz 1994]), é convencionalmente definido como sendo o problema da afectação de recursos no tempo a um conjunto de tarefas a realizar de forma coordenada. Este problema envolve recursos e tempo limitados e tarefas que, ao serem realizadas consomem recursos e tempo. Tipicamente, as limitações referidas, apelidadas de restrições, dizem respeito às capacidades restritas dos recursos, que se traduzem na disponibilidade ou não disponibilidade dos recursos e em durações maiores ou menores na realização de cada tarefa, e às restrições temporais sobre os tempos de início e de fim de cada tarefa, sejam elas exógenas (como a data limite imposta para a conclusão de um conjunto de tarefas, ou a data actual) ou endógenas (como uma ordem predeterminada para a realização de uma sequência de tarefas imposta por razões tecnológicas). Visto que, dado o conjunto de tarefas a realizar, o conjunto de recursos necessários para as realizar e o conjunto das restrições, o problema pode ter várias soluções possíveis e que algumas umas dessas soluções podem ser preferíveis a outras, usualmente é definido um critério de avaliação das soluções, que permite distinguir melhores de piores soluções e decidir que solução, ou soluções, são adoptadas. O estudo de problemas de escalonamento remonta ao tempo da 2ª Guerra Mundial, cujas necessidades logísticas desencadearam uma fase de preocupação com os problemas de planeamento e de tomada de decisão. Áreas de actividade que reconhecidamente levantam problemas de escalonamento são as do Planeamento e Controlo da Produção e do Planeamento de Projectos.

No ambiente de produção e distribuição existem empresas que possuem recursos físicos utilizados para produzir, transportar e armazenar produtos. Uma empresa é um agente autónomo (*i.e.*, com capacidades de decisão e acção próprias) que possui e explora recursos com o objectivo de tornar disponíveis produtos específicos a outras empresas, ou ao consumidor final. Normalmente a empresa consome produtos de outras empresas, ou extrai-os da natureza, para serem utilizados como componentes ou matérias-primas a incorporar nos seus produtos. Regra geral, uma empresa é especializada num grupo restrito de produtos específicos e não produz um produto final (disponível ao cliente, ou consumidor final) a partir de matérias-primas que extrai da natureza, porque não detém todos os recursos necessários para realizar todas as tarefas de todas as fases que é necessário levar a cabo entre a obtenção de matérias-primas e o produto final. A regra geral é, então, uma empresa depender de outras: das empresas clientes, que consomem os seus produtos e das empresas fornecedoras que abastecem a empresa dos produtos que ela consome. O aspecto da interdependência entre empresas é, portanto, um aspecto importante e a tomar em conta. Outro aspecto igualmente importante é o da descentralização geográfica. Com a redução das limitações ao comércio internacional e a harmonização dos padrões e das leis, cada vez mais se pode dizer que qualquer produto pode ser vendido para qualquer ponto do globo. O mesmo se pode dizer com a produção, já que os meios de produção são cada vez mais acessíveis em qualquer ponto do globo: é cada vez mais frequente os produtos serem projectados num país, a engenharia do processo de produção ser realizada noutro país e a produção ainda noutro [Hunt 1997].

Para a produção e distribuição da maior parte dos bens tangíveis é necessário existir uma rede de actividades coordenadas. Tradicionalmente, existem dois modelos básicos segundo os quais estas actividades pode ser organizadas: o mercado e a hierarquia [Williamson 1975], [Malone 1988], [Fox 1988], [Busby 1993]. Num modelo de mercado há grupos de especialistas que negociam entre si num mercado aberto. Acordam preços e estabelecem contratos entre si que normalmente limitam as obrigações mútuas a um período de tempo. Após esse período são livres de se associarem a outros especialistas competidores. Num modelo de hierarquia, a rede de actividades está contida numa estrutura de comando hierárquica. O fluxo de produtos é planeado de forma centralizada e não há nenhuma competição ou redundância na estrutura.

Como resultado da globalização dos mercados e da pressão da competição as empresas, principalmente pequenas e médias empresas vêm-se constrangidas, cada vez mais, a especializarem-se, a cooperarem, a planear e a controlar de forma coordenada as suas actividades e a organizarem-se em cadeias, ou redes de fornecimento (*supply chains*), procurando ter tempos de resposta cada vez mais curtos frente às necessidades do consumidor. A tendência é no sentido de maior coordenação e integração das actividades destes "agentes de rede" tanto ao nível da configuração como do controlo [Thomas 1996], cada vez mais apoiados nas tecnologias de informação e das comunicações. Também, a evolução recente destas tecnologias e o recurso cada vez maior a elas transforma os produtos, os processos, as empresas e a competição entre as empresas [Porter 1985].

Actualmente, a maior parte dos processos produtivos é executada por uma rede de múltiplas empresas. Antes limitada tipicamente a domínios particulares de negócio, como é o caso da indústria automóvel, a organização em rede cooperativa é hoje uma tendência que se generalizou e estendeu a outros domínios, abrangendo a produção, a distribuição e os serviços. No entanto, note-se que, relativamente ao planeamento coordenado das actividades logísticas em redes de produção e distribuição existem trabalhos interessantes a partir de meados do século XX.[1] Mesmo assim, só mais recentemente (na década de 90) se tem vindo a dar atenção mais dedicada ao problema de escalonamento em contextos descentralizados (ver [Fox 1993], [Tate 1995], [Rabelo 1996a], [Hildum 1997], [Arnold 1997] ou [Kjenstad 1998], por exemplo).

Uma rede cooperativa de produção e distribuição pressupõe um conjunto de empresas ligadas por relações cliente-fornecedor, que desempenham actividades de produção e distribuição — actuando como fornecedores, produtores, distribuidores, retalhistas — e cooperam na colocação de um produto ou produtos finais no mercado. Os nós da rede cooperativa correspondem a empresas especializadas, que desempenham vários tipos de tarefas, fases ou estágios (de produção, de armazenamento, de transporte) complementares do processo de produção e distribuição. Ao especializarem as suas competências e organizarem-se em rede cooperativa as empresas podem obter vantagens competitivas (*i.e.*, superioridade em termos de preferência do cliente ou consumidor) relativamente a empresas competidoras, pois podem partilhar informação técnica e comercial, actuar de forma coordenada na introdução de novos produtos, reduzir custos que advêm da necessidade de segurança face à incerteza (como os do armazenamento intermédio de produtos ou das inspecções de controlo de qualidade) e reduzir tempos de resposta face a uma procura variável. Os objectivos da rede podem ser genericamente traduzidos pela máxima da logística: *colocar o produto certo, na quantidade*

[1] Esta literatura é vasta e surge associada à Gestão e à Investigação Operacional principalmente a partir dos anos 50 e 60. Atente-se, por exemplo, na quantidade (e nas datas) de artigos referidos em [Graves 1993], nos capítulos que dizem respeito ao assunto (alguns destes artigos, bem como outros, são referidos no Capítulo 2).

certa, no momento certo, no local certo [Christopher 1993], [Carvalho 1996], [Gattorna 1996]. Esta máxima pode traduzir um objectivo, não só para os nós junto do cliente final, que tornam disponíveis os produtos finais, mas também para cada um dos nós da rede que tornam disponíveis os produtos de estágios intermédios.

A organização das empresas em redes cooperativas combina as vantagens de ambos os modelos de organização, mercado e hierarquia [Busby 1993], [Thorelli 1986], [Camarinha-Matos 1999b]. Por um lado a rede cooperativa é uma organização relativamente durável, menos volátil e transiente do que os contratos negociados a intervalos frequentes entre os participantes num modelo de mercado. Permite, por isso, um maior planeamento e controlo, frequentemente associados ao modelo da hierarquia, mas evita a inércia e a rigidez deste modelo. Permite a competição e a transparência de preços que caracterizam o modelo de mercado, mas evita a adversidade e miopia que se podem originar num contexto de mercado. Esta combinação, juntamente com o recurso ao suporte das tecnologias de informação e das comunicações, está na origem de paradigmas de gestão como Resposta Rápida (*Quick Response*), Resposta Precisa (*Accurate Response*), Gestão Integrada da Cadeia de Fornecimento (*Integrated Supply Chain Management*), Produção Ágil (*Agile Manufacturing*), Empresa Virtual (*Virtual Enterprise*), Empresa Estendida (*Extended Enterprise*), ver, por exemplo, [Ross 1996], [Fisher 1994], [Fox 1993], [Parunak 1996a], [Camarinha-Matos 1999a] e [O'Neill 1996], respectivamente.

Para ter sucesso, uma empresa deve reduzir *tempos de resposta*, para além de aumentar a *qualidade* e a *flexibilidade* nos seus produtos e serviços. Mas para isso deve estar *ligada a todos os estágios do processo de negócio* em que está envolvida, não apenas dentro das suas próprias fronteiras, mas na cadeia de fornecimento da indústria [O'Neill 1996]. Também devem considerar-se importantes, não só a produção e distribuição coordenada de produtos, mas também o desenvolvimento cooperativo de novos produtos. Isto é o que se pretende com paradigmas como o da Empresa Estendida, ou o da Empresa Virtual.

A *Empresa Estendida* define-se como uma fusão configurável e transiente dos *processos de negócio* de diferentes *unidades de negócio* para formar um sistema que coloca num mercado determinados produtos [Busby 1993], [Sackett 1994], [O'Neill 1994], [Browne 1996], [O'Neill 1996]. Este sistema é uma rede cujos nós são empresas especializadas, actuando simultaneamente como clientes e fornecedores, onde a coordenação é assegurada através de comunicação inter-empresa, suportada por redes electrónicas e recurso às tecnologias de informação e das comunicações. O conceito de Empresa Estendida (EE) é tipicamente realizável num domínio que é o das unidades de negócio de produção ou serviços, em geral geograficamente distribuídas, capazes de operar como um sistema controlado e planeado para colocação de produtos ou serviços num mercado. Segundo o modelo da EE, a coordenação das tarefas da rede de EE numa perspectiva global (*i.e.*, abrangendo toda a rede) deve ser assegurada por uma *unidade de supervisão*, para a qual contribuem todos os nós intervenientes na rede (a unidade de supervisão é uma equipa onde cada empresa envolvida está representada). Esta unidade de supervisão funciona como um *nó virtual*, ou agente coordenador de rede que integra as perspectivas individuais dos nós numa perspectiva global da rede. Pressupõe-se ainda uma infra-estrutura que possibilita comunicações rápidas e a partilha de dados comuns (rede electrónica de comunicações, bases de dados) necessários à coordenação.[2]

[2] Um paradigma similar ao da Empresa Estendida é o de Empresa Virtual, definido como uma aliança temporária de empresas que juntam competências e recursos para melhor responder a oportunidades de

1.2 Definição do Contexto e do Problema

No presente trabalho assume-se um contexto, ou ambiente de escalonamento em que existe uma rede de produção e distribuição multi-produto (com múltiplos produtos finais), cujas actividades são coordenadas por uma EE. Cada um dos nós da rede de produção e distribuição pode entregar, ou tornar disponíveis, um conjunto limitado de produtos, à custa do consumo de certos materiais ou componentes e executando tarefas de produção e distribuição (produção, armazenamento ou transporte). Os nós são *macro-recursos* que correspondem a fábricas, armazéns ou unidades de transporte cuja capacidade é gerida, de forma coordenada, pelas empresas participantes da EE.[3]

As empresas de uma EE são tratadas como *agentes autónomos* e a actividade de escalonamento das tarefas logísticas é vista pela perspectiva do paradigma *Multi-Agente* da *Inteligência Artificial Distribuída* (IAD, ou *Distributed Artificial Intelligence*, DAI, [O' Hare 1996], [Bond 1988], [Weiss 1999], [Ferber 1999]). Esta actividade envolve uma *coordenação multi-agente*, em que os agentes cooperam para realizar objectivos conjuntos de satisfação da procura final ao longo do tempo, respeitando restrições de capacidade e restrições temporais de escalonamento. Existe no entanto margem de manobra para actividade competitiva no escalonamento: os agentes poderão competir na satisfação de objectivos ou preferências de escalonamento individuais.[4]

Podem distinguir-se dois níveis neste contexto:

a) O nível dos recursos físicos, constituído pela rede de produção e distribuição. Esta rede contém nós (os macro-recursos), interdependentes devido a relações de cliente-fornecedor e especializados em tarefas logísticas de produção, armazenamento e transporte. Para a satisfação de pedidos, ou encomendas, do exterior são criadas tarefas, a escalonar nos nós da rede. Um processo de rede é um grupo de tarefas ligadas por relações de precedência temporal, que advêm das relações cliente-fornecedor entre os nós — no mesmo processo, cada tarefa de um nó fornecedor deve preceder a tarefa do nó cliente — que é gerado para satisfazer um encomenda do exterior. O escalonamento de uma tarefa num nó ocorre por afectação da capacidade do nó à tarefa num intervalo de tempo. Cada nó tem uma capacidade (de produção, de armazenamento ou de transporte) limitada para afectar a tarefas. Este nível, em que ocorrem basicamente fluxos físicos de produtos, é apelidado de *nível físico*;

b) O nível de decisão, constituído por agentes cooperantes. Estes agentes representam as empresas participantes da EE e gerem a capacidade dos nós da rede de produção e distribuição. Cada agente tem a seu cargo a gestão da capacidade de um nó da rede. As relações de cliente-fornecedor entre os nós são estendidas aos agentes: um agente é simultaneamente fornecedor e cliente de outros agentes da EE, com os quais pode comunicar. Cada agente pode receber pedidos (encomendas) de agentes clientes. Para a

---

negócio e cuja cooperação é suportada por redes de computadores [Camarinha-Matos 1999a]. Embora não haja definições claramente aceites, em regra, a Empresa Virtual é entendida como uma organização de configuração mais volátil (os participantes podem mudar) do que a Empresa Estendida.

[3] O paradigma da EE tenta dar resposta aos desafios actuais colocados às empresas. Para o presente trabalho, este paradigma contribuiu em parte, tanto do ponto de vista de suporte conceptual, como com a experiência da sua aplicação prática, nomeadamente num caso prático de uma cadeia de empresas na área dos têxteis e vestuário, no contexto do projecto Esprit AITEAR [AITEAR 1997a], [AITEAR 1997b].

[4] Havendo lugar para esta competição, tratar-se-á de um ambiente *semi-cooperativo*, ou *parcialmente adversarial/parcialmente cooperativo*, parafraseando [Yokoo 1990] e [Yokoo 1991].

satisfação dos pedidos o agente escalona as tarefas apropriadas no nó por ele gerido. Estas tarefas permitem entregar os produtos nas quantidades e nas datas expressas nos pedidos. Adicionalmente, o agente envia os pedidos adequados a agentes fornecedores para assegurar os fornecimentos necessários àquelas tarefas. Este nível, em que ocorrem basicamente fluxos de informação, é apelidado de *nível virtual*;

Dado este contexto, o problema de escalonamento a que este trabalho se refere, corresponde basicamente a uma generalização do problema de escalonamento convencional, com actividade de escalonamento descentralizada (distribuída pelos agentes) e comunicação inter-agente. Isto quer dizer que, em primeiro lugar, as decisões de escalonamento não são atribuídas a um único agente central, mas são distribuídas por um conjunto de agentes decisores, cada um com capacidade de decisão sobre um único recurso; adicionalmente, para a coordenação das decisões, estes comunicam entre si. Dado o contexto, o problema é definido como se segue.

Problema de Escalonamento em Rede de EE

É dado um *pedido* do exterior. A um pedido está associado um *produto final*, um *nó terminal* da rede, uma *quantidade* do produto, uma *data*, ou *tempo, de entrega* e um *horizonte temporal de escalonamento*. Adicionalmente, são dadas *restrições temporais de datas limite*, na forma de um intervalo definido por uma data de início mais cedo e uma data de fim mais tarde, durante o qual deverão ocorrer todos os eventos e todas as tarefas necessários à satisfação do pedido. *Restrições temporais de precedência* entre as tarefas necessárias, definidas pelas relações cliente-fornecedor entre os executantes, impõem uma ordem de execução que deve ser respeitada. Também, as *restrições de capacidade* dos executantes das tarefas devem ser respeitadas, de modo que nenhum executante deverá exceder a capacidade disponível para execução de tarefas.

Deve determinar-se qual o *escalonamento*, ou *plano temporal*, das tarefas necessárias para a satisfação do pedido, de modo a satisfazerem-se as restrições.

1.3 Objectivos

Um objectivo muito geral, que norteia o presente trabalho é o de produzir um modelo que permita tratar o problema de escalonamento numa perspectiva abrangente, que inclua não só o escalonamento convencional mas também a decisão distribuída, os agentes decisores de escalonamento e a comunicação necessária entre os agentes para coordenação da actividade de escalonamento, num contexto de escalonamento multi-agente cooperativo.

O trabalho desenvolvido pode, então, ser resumido de forma sumária pelas seguintes palavras chave:

- Escalonamento multi-agente, escalonamento cooperativo;
- Comunicação inter-agente;
- Gestão da cadeia de fornecimento, Empresa Estendida.

Os objectivos gerais são:

1. Abordar o problema de escalonamento numa perspectiva que abrange a comunicação entre agentes de escalonamento, num contexto de escalonamento multi-agente cooperativo;

2. Desenvolver um modelo de escalonamento, de acordo com a perspectiva em 1, para escalonamento de tarefas logísticas de produção e distribuição, aplicável num contexto de gestão da cadeia de fornecimento e segundo a perspectiva do paradigma da Empresa Estendida de gestão integrada da cadeia de fornecimento.

Os objectivos específicos são:

1. Modelar os aspectos interessantes para o escalonamento convencional, nomeadamente tempo, capacidade, recursos e tarefas, enquadrados no contexto de rede de produção e distribuição;
2. Modelar os aspectos interessantes do escalonamento multi-agente e da comunicação inter-agente para coordenação da actividade de escalonamento multi-agente no contexto da gestão da cadeia de fornecimento, nomeadamente agentes intervenientes, informação trocada entre os agentes e protocolo de interacção entre eles;
3. Desenvolver um sistema computacional no qual os aspectos em 1 e 2 possam ser representados e testados e baseado no qual seja possível desenvolver trabalho futuro, dando continuidade ao trabalho actual.

O trabalho desenvolvido consiste na elaboração de um modelo de escalonamento baseado em dois níveis, o *nível físico* e o *nível virtual*, e uma realização computacional do modelo. Os objectivos específicos 1 e 2 são satisfeitos pela elaboração dos níveis físico e virtual do modelo, descritos nos capítulos 3 e 4. O objectivo específico 3 apenas é parcialmente satisfeito já que a realização computacional do modelo está, na data de conclusão do trabalho, incompleta. Apesar de incompleta, esta realização computacional é referida por ter ajudado a refinar, testar e consolidar ideias durante o desenvolvimento do modelo. Para a demonstração da validade de um aspecto importante do modelo — um método proposto para a resolução de problemas de escalonamento multi-agente cooperativo, que inclui um mecanismo de coordenação inter-agente — recorre-se à simulação de casos exemplificativos.

1.4 Resumo dos Capítulos

O presente trabalho é constituído por seis capítulos, quatro apêndices e uma secção contendo referências. Nos apêndices incluem-se matérias complementares às dos capítulos. Os conteúdos dos capítulos subsequentes ao Capítulo 1 e os dos apêndices são, a seguir, resumidos.

Capítulo 2, Trabalho Relacionado - Neste capítulo descreve-se o estado do conhecimento, ou estado da arte, nas áreas que o problema tratado abrange, bem como alguns dos trabalhos mais recentemente realizados. São incluídos elementos do estado do conhecimento relativos ao problema de escalonamento convencional e ao planeamento coordenado de redes de produção e distribuição, com abordagens das áreas de Investigação Operacional (IO), Inteligência Artificial (IA) e Inteligência Artificial Distribuída (IAD).

Capítulo 3, Um Modelo para Escalonamento Multi-Agente — Nível Físico - Neste capítulo descreve-se a parte do modelo de escalonamento multi-agente relativa ao *nível físico*. Definem-se componentes do modelo como nós (recursos), redes e tarefas no contexto de redes de produção e distribuição. O trabalho descrito conjuntamente no Capítulo 3 e no Capítulo 4 (complementar do Capítulo 3) estabelece as entidades básicas para a construção de um ambiente computacional no qual se possa simular a actividade de escalonamento num

contexto multi-agente das redes cooperativas de produção e distribuição e testar mecanismos de coordenação no que respeita a essa actividade.

Capítulo 4, Um Modelo para Escalonamento Multi-Agente — Nível Virtual - Este capítulo é complementar do Capítulo 3. Nele se descreve o *nível virtual* do modelo. Definem-se componentes do modelo como pedidos, agentes e rede de EE e descrevem-se os mecanismos para troca de informação que são necessários à coordenação inter-agente da actividade de escalonamento. Propõe-se um método de resolução de problemas de escalonamento multi-agente que se apoia num mecanismo de coordenação da actividade de escalonamento e re-escalonamento dos agentes e definem-se protocolos de interacção de alto nível entre os agentes.

Capítulo 5, Exemplos - Neste capítulo mostram-se resultados de exemplos de aplicação do modelo desenvolvido, assumindo certas restrições. Apresentam-se simulações de casos exemplificativos que demonstram a validade do mecanismo de coordenação de escalonamento multi-agente sugerido no Capítulo 4.

Capítulo 6, Conclusões e Trabalho Futuro - Neste capítulo apresentam-se as conclusões e contribuições relativas ao presente trabalho e descreve-se o trabalho futuro a desenvolver.

Apêndice A — Diagramas de Classes - Neste apêndice são exibidos diagramas de classes de objectos do modelo descrito nos Capítulos 3 e 4.

Apêndice B — Inteligência Artificial e Resolução de Problemas através de Procura - Neste apêndice faz-se uma breve introdução à resolução de problemas por procura, método a que se recorre frequentemente na resolução de problemas difíceis, em particular na área de Inteligência Artificial, nomeadamente quando aplicada ao problema de escalonamento (como se refere no Capítulo 2).

Apêndice C — O Problema de Satisfação de Restrições - Neste apêndice faz-se uma breve introdução ao Problema de Satisfação de Restrições (PSR) já que, actualmente, em várias abordagens ao escalonamento é frequente o problema de escalonamento ser definido em termos do PSR (como se refere no Capítulo 2).

Apêndice D — Inteligência Artificial Distribuída - Neste apêndice faz-se uma breve introdução à área de Inteligência Artificial Distribuída (IAD), a qual foi importante na elaboração do presente trabalho.

1.5 Resumo das Contribuições

Resumem-se aqui as contribuições contidas no modelo que o presente trabalho expõe. Estas contribuições são descritas com maior detalhe no Capítulo 6.

Propõe-se uma organização do modelo em dois níveis:

- O *nível físico* que modela a parte física do ambiente de escalonamento. Esta parte do modelo diz respeito a tempo, quantidades, produtos, recursos e capacidade, redes de recursos e relações cliente-fornecedor entre recursos, tarefas e escalonamento de tarefas nos recursos, fluxos de produtos que podem ocorrer entre os recursos de uma rede, processos de rede;
- O *nível virtual* que modela a parte não física do ambiente de escalonamento. Esta parte do modelo diz respeito aos agentes que gerem as capacidades dos recursos de uma rede, à interacção entre os agentes quando na presença de problemas de escalonamento multi-agente para os quais uma solução deve ser encontrada e à coordenação da actividade de escalonamento dos agentes.

O Nível Físico

No nível físico são modelados os componentes físicos de escalonamento multi-agente. Os aspectos básicos relevantes do nível físico do modelo são:

- A informação associada aos acontecimentos logísticos relevantes para a actividade de escalonamento é modelada de uma forma integrada por meio de objectos denominados *eventos*. Os eventos servem basicamente para definir tarefas e caracterizam o que acontece no início ou no fim de uma tarefa escalonada num nó de rede.

Os componentes principais do nível físico do modelo são a *rede* física de produção e distribuição e *tarefas*. A rede de produção e distribuição inclui *nós*, que representam os recursos de capacidade limitada da rede e *arcos*, que representam as relações de cliente-fornecedor entre os nós e, portanto, os possíveis *fluxos* físicos de *produtos* entre os nós. Nós podem classificar-se em *nós de capacidade*, *nós de retalho* e *nós de matéria-prima*. Estas duas últimas classes de nós são artificiais e apenas servem para representar uma fronteira física da rede com o exterior. As características relevantes dos nós de capacidade são:

- Terem um conjunto de *produtos de saída*, que podem entregar (são nós multi-produto) e um conjunto de *produtos de entrada*, que podem consumir. A entrega e o consumo de produtos estão associados a tarefas do nó;
- Terem *capacidade* afectável a tarefas limitada. A capacidade é classificada em *capacidade de acumulação*, definida com base numa quantidade de produto, e *capacidade de processamento*, definida com base numa taxa de processamento, equivalente a uma quantidade de produto por unidade de tempo. A quantidade de capacidade necessária para o escalonamento de uma tarefa num nó depende do nó e do produto de saída da tarefa, sendo proporcional à quantidade de produto. Cada nó inclui uma representação da *capacidade máxima* e da *capacidade utilizada* em cada produto de saída ao longo do tempo;
- Os nós de capacidade são classificados em *acumuladores*, que dispõem de capacidade de acumulação e *processadores*, que dispõem de capacidade de processamento. Os processadores classificam-se ainda em *produtores* e *transportadores*.

As tarefas são criadas para serem escalonadas em nós de rede. Podem existir *tarefas de capacidade*, *tarefas de retalho* e *tarefas de matéria-prima*, para escalonamento em nós de capacidade, nós de retalho e nós de matéria-prima, respectivamente. As duas últimas classes de tarefas são artificiais e apenas servem para representar fronteiras temporais de processos de rede. Tarefas de capacidade são classificadas em *tarefas de acumulação*, *tarefas de produção*

e *tarefas de transporte*, escalonáveis em acumuladores, produtores e transportadores, respectivamente. As características relevantes das tarefas de capacidade são:

- Uma tarefa é criada para ser escalonada num determinado nó de capacidade, com a finalidade de entregar um produto de saída do nó, consumindo certos produtos de entrada do nó em quantidades predeterminadas no nó e utilizando capacidade do nó durante o intervalo de tempo em que é escalonada;
- Uma tarefa pode ser decomposta em *sub-tarefas* e escalonada em intervalos de tempo distintos.

O Nível Virtual

O nível virtual é o nível da decisão, em que são modelados os agentes intervenientes na actividade de escalonamento e os fluxos de informação necessários à coordenação desta actividade. Neste nível, que pressupõe o nível físico, definem-se *pedidos*, *agentes* e *rede de EE*.

No nível virtual de uma rede de EE existem *agentes de rede*, cada um associado a um nó de rede do nível físico. Os agentes de rede classificam-se em *agentes gestores* (ou *agentes de capacidade*), que estão associados a nós de capacidade, e *agentes de retalho* e *agentes de matéria-prima*, associados a nós de retalho e de matéria-prima, respectivamente. Os agentes gestores podem ainda ser classificados, de acordo com os nós que gerem, em *agentes acumuladores* e *agentes processadores* e os agentes processadores em *agentes produtores* e *agentes transportadores*. Para além dos agentes de rede existe um agente adicional, *o agente supervisor*, que serve de interface com o exterior e tem a função de introduzir trabalho na rede de EE, *i.e.*, novos problemas de escalonamento multi-agente. Os aspectos mais relevantes no nível virtual do modelo são:

- A definição de *arquitecturas de agente*, uma para cada classe de agente, apropriadas à actividade de escalonamento dinâmico multi-agente e à comunicação entre agentes para coordenação daquela actividade;
- A coordenação da actividade de escalonamento multi-agente assenta na interacção por comunicação de *mensagens* entre agentes. Definem-se *protocolos de interacção* para as classes de agentes comunicantes (pares de agentes cliente-fornecedor, essencialmente). Estes protocolos contemplam a troca de mensagens para o estabelecimento problemas de escalonamento novos (incluindo comunicação de pedido de problema novo e respostas de rejeição ou de aceitação), a modificação, a desistência e a satisfação de pedidos associados a problemas de escalonamentos existentes;
- A unidade de troca de informação relevante para escalonamento (e re-escalonamento) multi-agente é o *pedido*. Pares de agentes gestores cliente-fornecedor transmitem entre si a informação associada a problemas de escalonamento, através da transmissão de mensagens contendo pedidos. Do ponto de vista de estrutura de informação, pedidos são eventos;
- O modelo contempla *re-escalonamento* (modificação de uma solução de escalonamento) multi-agente, para resolução de *conflitos* (violação de restrições temporais ou de capacidade) numa solução de problema de escalonamento;
- O modelo contempla a *desistência* de um problema de escalonamento multi-agente, para poder permitir remover conflitos não resolúveis.

Adicionalmente, propõem-se:

- Uma *taxonomia* e uma *caracterização* dos *conflitos temporais* possíveis em problemas de escalonamento multi-agente;
- Um *conjunto de regras*, para uso dos agentes gestores, para detecção e identificação de certos tipos de conflitos temporais;
- Um *conjunto de tratamentos* diferenciados para cada tipo de conflito temporal detectado pelas regras referidas, para agentes gestores processadores e para agentes gestores acumuladores;
- Um *mecanismo de coordenação* para escalonamento, baseado em folgas temporais, que permite coordenar a actividade de escalonamento e re-escalonamento multi-agente. O mecanismo permite que cada agente gestor disponha da informação necessária para desistir de problemas de escalonamento quando estes são temporalmente super-constrangidos e, adicionalmente, que os agentes mantenham informação sobre limites temporais de escalonamento das suas tarefas. Em caso de ser necessário recorrer ao re-escalonamento, esta informação permite que os agentes excluam opções de escalonamento das tarefas que inviabilizam soluções possíveis;
- Um *método de resolução* cooperativa de problemas de escalonamento multi-agente, que prescreve um comportamento individual cooperativo faseado para os agentes gestores. Neste método enquadram-se o mecanismo de coordenação para escalonamento multi-agente, as regras para detecção e identificação de conflitos temporais e os respectivos tratamentos, acima referidos.

2 Trabalho Relacionado

Neste capítulo descreve-se o problema de escalonamento e o estado do conhecimento, abrangendo o escalonamento convencional e o escalonamento enquadrado no contexto mais lato das redes de produção e distribuição e da gestão da cadeia de fornecimento (*supply chain*).

2.1 O Escalonamento — Definição e Caracterização do Problema

O *escalonamento*, ou *scheduling*,[1] é motivado por questões que surgem em particular no domínio da produção industrial, no âmbito do planeamento da produção. No entanto, em geral, estas mesmas questões levantam-se em todas as situações em que existem recursos que devem ser afectados a tarefas a serem executadas ao longo do tempo. Tipicamente, as decisões do escalonamento afectam a capacidade disponível de recursos (máquinas, equipamento, pessoal, espaço) a certas actividades no contexto de um qualquer sistema de processamento (fábrica, armazém, estaleiro, escola, hospital, etc.), ao longo do tempo. As decisões têm como resultado um *escalonamento* (*schedule*),[2] que pode ser entendido como um plano que indica a sequência das tarefas decidida e os tempos de afectação dos recursos a cada uma das tarefas [Vollmann 1997]. O escalonamento tem sido definido como a actividade que tem como objectivo *a afectação de recursos, no tempo, necessários para executar um conjunto de tarefas* [Baker 1974], sendo uma preocupação importante no escalonamento a da *afectação óptima* dos recursos escassos às tarefas no tempo [Lawler 1993].

2.1.1 Perspectiva Histórica e Definições Informais

Historicamente, a ideia da eficiência da utilização dos recursos e do uso económico do tempo aparece com a noção de tempo objectiva e mensurável com precisão. A gestão do tempo e dos recursos passa a ocupar um lugar de primordial importância na organização económica dos processos industriais de produção pela altura da Revolução Industrial (séc. XVIII), reconhecendo-se que só uma exploração eficiente dos factores de produção (máquinas, por exemplo) garante o máximo de retorno do capital investido. Charles Babbage (séc. XVIII) foi um dos primeiros a analisar a economia da produção industrial, na sua obra *On the Economy of Machines and Manufactures* (1832), baseando-se em análises empíricas do processos industriais e fábricas do seu tempo (que chegaram a influenciar os trabalhos de Karl Marx e John Stuart Mill). Taylor, no seu livro *Principles of Scientific Management* (1911), lançou as

---

[1] Os termos "programação", "calendarização", "agendamento" ou "geração de horário" são, por vezes, também usados com o mesmo significado (frequentemente em domínios diferentes). Embora os termos "programação" e "programa" possam ser fortemente conotados com programação de computadores (usando linguagens de programação) não é invulgar serem usados na literatura de Gestão da Produção com o mesmo sentido de "escalonamento" (frases como "Planeamento e Programação da Produção", "programa de operações" ou "programa de actividades" não são invulgares). O termo "calendarização" sugere um pressuposto que é o da existência de um calendário, isto é, um referencial temporal com uma estrutura cíclica (baseada em períodos de tempo como, anos, meses, semanas, dias) que, numa formulação mais formal de problemas de escalonamento nem sempre é pressuposta. Por vezes é também usado o termo "agendamento", como sinónimo de escalonamento, e "marcação", com um significado que é o do escalonamento de uma actividade individual. Os termos "escalonamento" ou "escala" (que vagamente se assemelha ao termo inglês "*schedule*"), "programa" ou "programação" (por exemplo, a "programação do canal de televisão" para determinado dia), "calendário", "horário" ou "agenda", são frequentemente usados para designar o produto da actividade do escalonamento.

[2] No presente trabalho optou-se por fazer corresponder o termo português "escalonamento" ao termo "*schedule*", usado na literatura anglo-saxónica (ver, por exemplo, [Rabelo 1997], em que a opção é idêntica). Outras alternativas possíveis a "escalonamento" estão descritas na nota 1. Portanto, note-se que, neste trabalho, "escalonamento" tanto poderá significar a actividade de escalonar, como o resultado desta actividade, dependendo o significado do contexto em que o termo é empregue.

bases científicas da organização racional do trabalho. As necessidades logísticas da 2ª Guerra Mundial desencadearam uma fase de preocupação com os problemas de planeamento e de tomada de decisão, surgindo a ciência da Investigação Operacional (IO).[3] O método *Simplex,* base matemática para exprimir muitos problemas de optimização de forma rigorosa, é inventado em 1947. Surgem várias abordagens à resolução de problemas de escalonamento usando o método *Simplex*. Uma evolução que a Guerra Mundial propulsionou foi também a dos computadores electrónicos (a partir de 1950): problemas de gestão de recursos em sistemas operativos de computadores e de escalonamento de processadores, estimularam igualmente a análise matemática de problemas de escalonamento [Froeschl 1993].

Como preâmbulo para uma definição mais formal do problema de escalonamento (ver mais adiante) apresentam-se a seguir algumas definições informais de investigadores envolvidos na área:

- Segundo [Baker 1974] o escalonamento tem como objectivo a afectação de recursos no tempo necessários para executar um conjunto de processos;
- Para [Lawler 1989] o escalonamento preocupa-se com a afectação óptima de recursos limitados a tarefas no tempo;
- Segundo [Fox 1994] o escalonamento é a selecção de planos alternativos e a afectação de recursos e tempos a cada tarefa, de modo a obedecer a restrições de tempo nas tarefas e a limitações de capacidade de um conjunto de recursos partilhados;
- De acordo com [Smith 1994] trata-se da afectação de recursos a tarefas de múltiplos processos independentes ao longo do tempo de modo a optimizar um conjunto de objectivos e preferências;
- Em [Le Pape 1994] o escalonamento é um processo de decisão para afectação de recursos ao longo do tempo com vista a realizar uma colecção de tarefas, sujeito a restrições (datas limite, duração e precedência das tarefas, tempos de movimentação e preparação, disponibilidade e partilha de recursos) e preferências ou restrições relaxáveis (relacionadas com datas limite, produtividade, frequência de troca de ferramentas, níveis de existências e estabilidade da fábrica);
- [Sadeh 1994] diz que o escalonamento trata da afectação de recursos (máquinas, ferramentas, operadores humanos) ao longo do tempo a um conjunto de tarefas, atendendo a uma variedade de restrições e objectivos;
- Em [Muscettola 1994a] o escalonamento tem a ver com as tarefas numa base diária. Dados um conjunto de objectivos instanciam-se os planos e atribui-se a cada acção uma fatia de tempo para uso exclusivo dos recursos necessários. O resultado é uma predição de um curso específico de acção que, ao ser seguido, assegura que se atingem todos os objectivos respeitando as restrições físicas do sistema;
- Para [McDermott 1995] trata-se de um caso especial importante do planeamento para o qual, dado um conjunto de acções a serem executadas, se tem de produzir a ordem pela qual elas devem ser executadas. Cada uma das acções requer um conjunto de recursos de capacidade finita e haverá uma preferência de certas ordens em relação a outras;
- Em [Chase 1995], um escalonamento é um esquema para a realização de actividades, que requer recursos ou atribui instalações. A finalidade do escalonamento das actividades, na

[3] Uma boa e abrangente introdução à Investigação Operacional é [Hillier 1990].

instalação funcional, é desagregar o programa director de produção em actividades faseadas por semana, dia ou hora, ou, por outras palavras, especificar, em termos precisos, a carga de trabalho do sistema produtivo planeada a muito curto prazo;

- Para [Burke 1994] o escalonamento é o problema de decidir *o que fazer* (tarefas), *quando* (durações, tempos) e *onde* (recursos), em princípio com o objectivo de maximizar o lucro da empresa. Todos estes elementos estão sujeitos a mudanças (mesmo a forma como o objectivo é conseguido pode variar conforme a conjuntura de mercado). Por isso é melhor construir um sistema de escalonamento baseado em *satisfação*, em vez de optimização, e reactivo, em vez de apenas predictivo, que seja capaz de manter um escalonamento satisfatório num mundo aberto e dinâmico.

Como se verifica pelas definições apresentadas, de um modo geral, nos problemas de escalonamento assume-se a necessidade de executar um certo número de tarefas (também designadas por actividades, ou operações) que requerem o uso de um certo número de recursos. As tarefas podem ser organizadas em grupos, cada um dos quais consiste numa dada sequência de tarefas a executar pela ordem especificada na sequência. É usual designar estes grupos por *processos* (*jobs*),[4] em domínios em que a sua execução se repete (por exemplo, na produção industrial) e por *projectos*, em domínios em que isso, em geral, não acontece. O processamento, ou execução, de uma tarefa requer o uso de um recurso particular durante um certo intervalo de tempo e cada recurso tem uma capacidade limitada (por exemplo, só pode executar uma tarefa de cada vez). É fácil verificar que os problemas de escalonamento ocorrem em contextos variados se se pensar que as tarefas e os recursos podem ser, respectivamente, operações de produção e máquinas de uma fábrica, actividades de construção e máquinas de construção civil, tratamento de pacientes e pessoal ou equipamento hospitalar, afectações de espaço e armazéns, aulas e professores numa escola, operações de manutenção de navios e docas de um estaleiro, programas de computador e computadores, refeições e cozinheiros num restaurante, cidades e caixeiros viajantes, etc..

Na afectação dos recursos escassos para a produção de um certo tipo de produto, ou realização de um serviço específico, podem distinguir-se duas fases: a fase de *planeamento* e a fase de *escalonamento* [Hildum 1994]. Na fase de planeamento é escolhido o conjunto de tarefas para a produção do produto (ou a realização do serviço, se disso se tratar) particular, quer dizer, é estabelecido um *plano de processo* adequado que especifica o tipo de tarefas e o tipo de ordem, ou sequência das tarefas.[5] A um nível operacional, esta fase resume-se a um procedimento de escolha do processo apropriado, entre um conjunto de processos previamente definidos, um para cada produto ou serviço que pode ser tornado disponível. Quer dizer, quando há um pedido (uma encomenda de um produto, ou uma solicitação de um serviço, específico) é seleccionado e instanciado um plano de processo adequado para satisfação do pedido. Em geral, num domínio de escalonamento existirão, em cada momento, múltiplos processos em execução para a satisfação de vários pedidos. A fase de escalonamento combina os aspectos de *sequenciação* e *geração de horário*.[6] O aspecto da sequenciação tem a ver com o estabelecimento da sequência da execução das tarefas de diferentes processos em cada recurso (tarefas competidoras pela utilização do mesmo recurso); o aspecto da geração de horário tem a ver com a atribuição de tempos de início (e de fim) a cada tarefa a executar.

---

[4] O termo processo é empregue aqui com mesmo significado do termo *job*, empregue na literatura anglo-saxónica da área de escalonamento.

[5] Esta ordem poderá, inclusive, ser uma ordem parcial, ou nenhuma ordem.

[6] *Timetabling*, na literatura anglo-saxónica. Na formulação do problema de escalonamento, no entanto, é frequente os dois aspectos não serem explicitamente distintos.

Uma sequenciação que permita que a sequência de tarefas de cada processo seja processada na ordem correcta é dita sequência possível, executável ou realizável e poderão existir preferências por certas sequências possíveis (por exemplo, se houver prioridades associadas aos processos). Um escalonamento contém uma sequência possível de tarefas e especifica os tempos de início (e de fim) exactos para todas as tarefas [Kan 1976]. Na actividade de escalonar pretendem-se encontrar escalonamentos, em particular aqueles que são óptimos, isto é, que minimizam, ou maximizam, o valor de uma determinada *função objectivo*.

Obras sobre o escalonamento numa perspectiva convencional (a qual se assume na presente secção) são, por exemplo, [Baker 1974], [Kan 1976], [Lawler 1989], [Blazewicz 1994], [Brucker 1998] ou [Morton 1993] (esta última apresenta o escalonamento numa perspectiva mais abrangente do que as restantes).

A teoria do escalonamento, classicamente enquadrada na área de IO, é caracterizada por um número muito grande de tipos de problemas e abordagens. Esta investigação pode classificar-se quanto à natureza do tipo de problema de escalonamento a que diz respeito segundo duas dimensões: a da natureza estática ou dinâmica e a da natureza determinística ou estocástica dos problemas [Vollmann 1997].

Quanto à natureza estática/dinâmica dos problemas podem ter-se:

- Problemas de *escalonamento estático* - Neste tipo de problemas existe um conjunto fixo de tarefas a serem escalonadas e assume-se que todas elas estão prontas a serem executadas num determinado momento que é, em geral, o momento de início do horizonte temporal de escalonamento, e que todos os recursos estão disponíveis nesse momento;
- Problemas de *escalonamento dinâmico* - Neste tipo de problemas novas tarefas a serem executadas surgem continuamente ao longo do tempo.

Quanto à natureza determinísitca/estocástica dos problemas temos:

- Problemas de *escalonamento determinístico* - Neste tipo de problemas toda a informação que define o problema de escalonamento, em particular as durações das tarefas (tempos de processamento) é conhecida previamente com precisão;
- Problemas de *escalonamento estocástico* - Aqui, a informação que define o problema é conhecida mas com um certo grau de incerteza, assumindo-se que está sujeita a variações aleatórias.

A maior parte da investigação clássica foi orientada para problemas de *escalonamento de máquinas determinístico* (*deterministic machine scheduling* problems) [Lawler 1993]. Nestes problemas, para além da restrição respeitante ao determinismo, o tipo de recursos, apelidados de máquinas, têm restrições de capacidade que, tipicamente, impõem a execução de, no máximo, uma tarefa de cada vez em cada máquina e além disso cada tarefa individual pode ser executada por, no máximo, uma máquina.

2.1.2 Definição e Caracterização dos Problemas de Escalonamento

De um ponto de vista formal, os problemas de escalonamento determinístico, para os quais se orienta a presente secção, são parte de uma classe mais geral de problemas: os *problemas de optimização combinatória*, e estes, por sua vez, são uma classe de *problemas de optimização*. Um problema de optimização é aquele que pode ser estruturado em termos de um conjunto de

variáveis de decisão e de um *conjunto de restrições* e para o qual se pretendem encontrar soluções óptimas [Reeves 1993a]. De um modo geral, um problema de optimização pode ser formulado do seguinte modo [Hillier 1990], [Reeves 1993a]:

- Minimizar (ou maximizar) $f(\mathbb{x})$;
- Sujeito às restrições $g_i(\mathbb{x}) \leq b_i \ (i=1,\ldots,m)$.

Em que $\mathbb{x}$ é um vector de variáveis de decisão e f e g_i são funções. f é chamada a *função de objectivo* e um vector de valores para $\mathbb{x}$ que não violem as restrições indicadas é chamado uma *solução possível*. Num problema de optimização pretende-se encontrar uma solução possível que minimize, ou maximize, conforme a formulação, o valor de f. Este tipo de solução é apelidada de *solução óptima* do problema. Se o problema é posto em termos de minimização de f, esta função é normalmente designada por *função de custo*. Classes específicas destes problemas podem ser obtidas colocando restrições sobre as funções f e g_i e sobre o conjunto de valores possível para as variáveis de $\mathbb{x}$.[7] Nos problemas de optimização combinatória as variáveis de decisão são restritas a valores discretos, sendo a solução um conjunto, ou uma sequência, de inteiros ou outros objectos discretos.[8]

Tendo enquadrado o problema do escalonamento em classes mais gerais de problemas, adopta-se agora uma definição mais formal do problema de escalonamento. O problema de escalonamento é posto do seguinte modo (adaptado de [Kan 1976], de [Blazewicz 1994], de [Sadeh 1994] e de [Roldão 1995]):

- Existe um conjunto de k *tarefas*, *operações* ou *actividades*, $\mathcal{O}=\{O_1, O_2, \ldots, O_k\}$;
- Existe um conjunto de m *processadores* ou *máquinas*, $\mathcal{P}=\{P_1, P_2, \ldots, P_m\}$;
- Opcionalmente consideram-se recursos de um conjunto de s *recursos adicionais* $\mathcal{R}=\{R_1, R_2, \ldots, R_s\}$, caso as tarefas necessitem de recursos adicionais para além dos processadores.[9]

O *escalonamento* consiste na afectação dos processadores de $\mathcal{P}$ (e, possivelmente, de recursos de $\mathcal{R}$) às tarefas de $\mathcal{O}$, associando-se a cada uma destas um tempo de início, de modo a completar todas as tarefas sob as restrições impostas. Em escalonamento clássico determinístico há duas restrições gerais [Lawler 1989], [Blazewicz 1994]:

- Cada tarefa é processada por, no máximo, um processador de cada vez e cada processador pode processar, no máximo, uma tarefa de cada vez (restrição de recurso);
- Toda a informação que define o problema é previamente conhecida com precisão.

---

[7] Por exemplo, nos *problemas de programação linear*, f e g_i são restritas a funções lineares das variáveis de decisão.

[8] Alguns exemplos deste tipo de problemas muito conhecidos são o *problema da atribuição* (*assignment problem*), o *problema do empacotamento* (*knapsack problem*) e o *problema da determinação de rotas* (*routing problem*) [Reeves 1993a].

[9] No ambiente de produção estes recursos podem corresponder a operários, ferramentas, materiais, dispositivos de transporte, etc. (e originam, quando tomados em consideração, um modelo mais complexo do problema de escalonamento).

2.1.3 Caracterização dos Processadores

Os processadores podem classificar-se de acordo com as funções que desempenham, do seguinte modo [Blazewicz 1994]:

- *Processadores paralelos* - Se todos os processadores de $\mathcal{P}$ podem desempenhar as mesmas funções (podendo ser distinguidos pelas suas velocidades de processamento);
- *Processadores dedicados* - Se cada processador de $\mathcal{P}$ é especializado na execução de certas tarefas.

Os recursos adicionais podem ser classificados [Blazewicz 1994] do seguinte modo:

- *Tipo de recurso* - De acordo com a funcionalidade preenchida pelo recurso: recursos do mesmo tipo são os que preenchem a mesma funcionalidade;
- *Categoria de recurso* - Podem classificar-se segundo dois pontos de vista: quanto às *restrições de recurso* (ou *restrições de capacidade*) e quanto à *divisibilidade do recurso*.

Quanto às restrições de recurso, um recurso pode ser:

- *Renovável* - Aquele em que apenas a sua disponibilidade temporária, ou uso total, em cada momento, é restrita, isto é, ao ser usado por uma tarefa, o recurso ficará novamente disponível após a tarefa o libertar;
- *Não renovável* - Aquele em que apenas a sua disponibilidade integral, ou consumo total, até um certo momento, é restrita, isto é, uma vez usado por uma tarefa o recurso não mais pode ser usado;
- *Duplamente constrangido* - Aquele em tanto o uso total, em cada momento, como o consumo total, até um certo momento, são restritos.

Um recurso pode ser, quanto à sua *divisibilidade*:

- *Discreto* - Se é discretamente divisível, isto é, pode ser afectado a tarefas numa das quantidades discretas de um conjunto finito de afectações possíveis (de que a afectação de uma única unidade de recurso por tarefa é um caso particular);
- *Contínuo* - Se é continuamente divisível, isto é, pode ser afectado numa quantidade arbitrária, inferior ou igual a uma certa quantidade máxima.

2.1.4 Caracterização dos Processos

No caso de *processadores dedicados*, o conjunto de tarefas é tipicamente particionado em subconjuntos denominados *processos* (*jobs*), colocando-se o problema assim:

- Há um conjunto de n *processos* $\mathcal{J}=\{J_1, J_2, \ldots, J_n\}$;
- Cada processo J_j é um conjunto de k_j *tarefas* $J_j=\mathcal{O}^j$, com $\mathcal{O}^j=\{O^j_1, O^j_2, \ldots, O^j_{k_j}\}$;
- As tarefas usam, ou são processadas por, *processadores* de um conjunto de m processadores $\mathcal{P}=\{P_1, P_2, \ldots, P_m\}$.

Os seguintes parâmetros são normalmente usados para caracterizar os processos e as suas tarefas:[10]

- Cada processo J_j tem um *tempo de chegada*, *data de chegada*, ou *data de lançamento* (*ready time*, *ready date*, ou *release date*) r_j, correspondente ao tempo a partir do qual J_j estará disponível para ser processado;[11]
- Cada processo J_j tem uma *data limite*, ou *data de entrega* (*due date*) d_j, que é o tempo até ao qual o processamento de J_j deverá terminar, sendo $r_j \leq d_j$;
- A cada processo pode associar-se uma *prioridade*, *peso*, ou *importância*, w_j (significa uma criticalidade, ou urgência relativa, em terminar a execução do processo);
- Para cada tarefa O_i^j há uma *duração*, ou *tempo de processamento*, fixa, p_i^j;
- Poderá ser especificada uma *ordem*, ou *restrição temporal de precedência*, entre tarefas de cada processo; para indicar que O_a^j *precede* O_b^j (e que, portanto, O_b^j só poderá iniciar-se após O_a^j terminar) escreve-se $O_a^j \ll O_b^j$.

Os parâmetros r_j, d_j e p_i^j são números inteiros[12] e w_j é um número positivo. As restrições temporais de precedência, quando existem, podem ser expressas na forma de um grafo acíclico direccionado cujos nós representam tarefas e cujos arcos representam as precedências temporais a respeitar, sem arcos transitivos, denominado *grafo de precedências*. Na Figura 2-1 e na Figura 2-2 dão-se exemplos de grafos de precedências. Esta forma de representação usada no grafo de precedências é designada por actividade-em-nó (*activity-on-node*) ou AON. Uma forma alternativa de representação, por vezes utilizada, é a de actividade-em-arco (*activity-on-arc*) ou AOA, em que os arcos do grafo correspondem a tarefas e os nós correspondem a tempos de eventos de início e de fim de tarefas.

Temos ainda a considerar que:

- Podem existir *acumuladores* ou *áreas de armazenamento intermédias* (*buffers*) entre os processadores em cada processo, que se assume terem capacidade de acumulação ilimitada (um processo, após terminado o processamento num processador pode esperar até que o processamento no processador seguinte se inicie); alternativamente pode assumir-se a propriedade (dos processos) de *sem-espera* (*no-wait*) se os acumuladores têm capacidade zero (neste caso, o processos não podem esperar entre dois processadores consecutivos);

---

[10] Os parâmetros aqui descritos para processos aplicam-se também a tarefas individuais, nomeadamente quando o conjunto de tarefas não é particionado em processos (nesse caso será $k=n$).

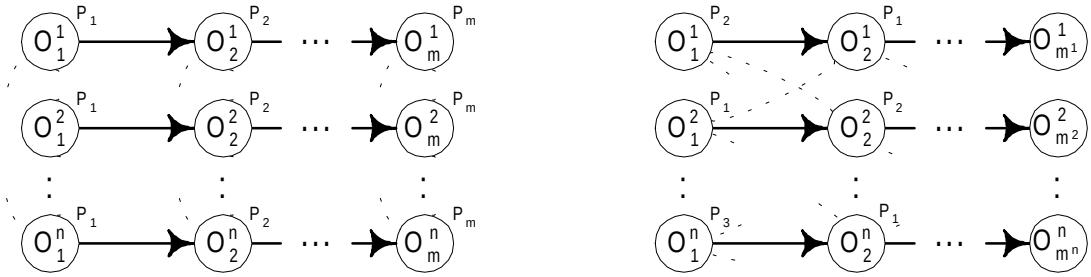
[11] Na literatura de escalonamento as designações *ready date* (ou *ready time*) e *release date* (ou *release time*) são usadas, aparentemente, com o mesmo significado (ver, por exemplo, [Morton 1993], [Blazewicz 1994], [Rabelo 1997], [Brucker 1998]). Em [Hildum 1994], página 4, no entanto, *ready time* é definido como sendo o tempo no qual *se pretende* que um processo esteja disponível para processamento (em complementaridade, a *due date* é definida como sendo o tempo no qual *se pretende* que um processo termine o seu processamento) e *release time* é definido como sendo o tempo no qual um processo *fica efectivamente* disponível para processamento (em complementaridade, a *completion time* é definida como sendo o tempo no qual um processo termina efectivamente o seu processamento).

[12] Como é dito em [Blazewicz 1994] esta suposição não é tão restritiva como pode parecer, já que é equivalente a permitir, para aqueles parâmetros, valores racionais arbitrários.

- As tarefas podem, ou não, ser interrompidas e retomadas mais tarde (no mesmo processador ou noutro). No primeiro caso o escalonamento é dito *preemptivo*, ou *com preempção*, no outro caso *não preemptivo*, ou *sem preempção*;
- Cada tarefa necessita de um único processador de $\mathbb{P}$ (para o qual poderão, ou não, existir várias alternativas) e, possivelmente, de recursos adicionais de $\mathbb{R}$, cujos requerimentos são predeterminados.

Podemos ainda caracterizar os processos quanto ao modelo de processamento. Para processadores dedicados existem, basicamente, dois modelos possíveis [Kan 1976], [Blazewicz 1994], [Brucker 1998]:[13]

1. *Processamento contínuo* (*flow shop*) - O número de tarefas por processo k_j, é igual ao número de processadores m, e as tarefas usam os processadores sempre pela mesma ordem. Quer dizer, para cada processo J_j, é executada primeiro a tarefa O_1^j no processador P_1, depois a tarefa O_2^j no processador P_2, etc., por último a tarefa O_m^j no processador P_m. Uma variante deste modelo designada por *processamento aberto* (*open shop*) não tem restrições de precedência predeterminadas;
2. *Processamento intermitente* (*job shop*) - O número e a ordem de execução das tarefas pode variar de processo para processo (embora seja fixo à partida).



a) processamento contínuo (m processadores)

b) processamento intermitente (m processadores; $m^1, m^2, \ldots, m^n \leq m$)

Figura 2-1- Grafos de precedências de processamento contínuo e de processamento intermitente. Cada nó O_i^j representa uma tarefa de um processo J_j. Os arcos direccionados representam relações de precedência temporal entre tarefas (arcos conjuntivos). Os arcos não direccionados adicionalmente incluídos, a traço leve, ligam nós de tarefas que usam o mesmo processador (arcos disjuntivos).

No caso dos processadores dedicados o grafo de precedências é limitado a uma *cadeia*, já que a ordem de execução das tarefas é uma *ordem linear*.[14] No contexto típico da produção industrial, o processamento contínuo é o tipo de processamento característico em linhas de montagem e indústrias de processo (refinarias, siderurgias, etc.) e o processamento intermitente o da produção por lotes e produção unitária. Os dois modelos são graficamente exemplificados, na forma de grafos de precedências (do tipo AON), na Figura 2-1.

---

[13] Com maior detalhe, poderiam ainda considerar-se outros casos possíveis, além dos descritos, como os de ambientes mistos e ambientes flexíveis (com alto nível de automatização e integração entre máquinas, transporte e armazenamento).

[14] Excepção feita ao modelo de processamento aberto.

Um terceiro modelo de processamento, envolvendo qualquer tipo de recursos e qualquer tipo de restrições de precedência, pode ainda considerar-se:

3. *Processamento com recursos constrangidos e precedências temporais gerais* - Podem ser requeridos outro tipo de recursos (processadores dedicados ou não, recursos com capacidade de executar mais de uma tarefa de cada vez, recursos adicionais), sendo as restrições de precedência quaisquer.

O escalonamento frequentemente designado por escalonamento de *projecto*[15] enquadra-se neste modelo. Na Figura 2-2 exemplifica-se este modelo através de um grafo de precedências (do tipo AON) de um projecto.

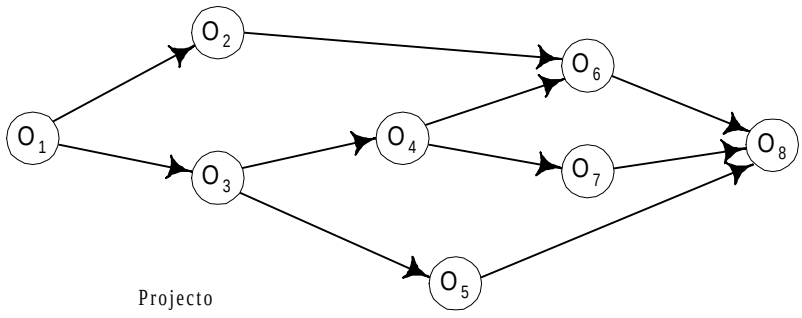


Figura 2-2- Grafo de precedências de um projecto. Os nós $O_1, \ldots, O_8$ representam tarefas do projecto.

Este terceiro modelo pode ser considerado uma extensão natural do processamento intermitente em que se consideram envolvidos quaisquer tipos de recursos e quaisquer restrições de precedência [Morton 1993]. Embora este modelo genérico de processamento seja complexo[16] não é, no entanto, difícil conceber aspectos realistas que possam ampliar ainda mais a sua complexidade, como sendo:

- Os processos poderem ter sequências de tarefas alternativas;
- A mesma tarefa ter processadores alternativos: por exemplo, quando uma tarefa deve ser executada numa qualquer máquina de um grupo de máquinas com características semelhantes (podendo, ou não, existir preferências por algumas uma delas);
- Ter em conta tempos de preparação de processadores e de recursos antes da execução de uma tarefa nele; os tempos de preparação poderão, ou não, variar conforme a sequência de tarefas escalonada no processador ou recurso; também os tempos de movimentação entre cada tarefa de um processo poderão ter de ser considerados;
- Uma tarefa poder ser interrompida (por exemplo, devido a uma avaria) e mais tarde concluída (escalonamento preemptivo);

---

[15] Abrangido pelo que é frequentemente designado por *planeamento e programação de projecto*, *planeamento e controlo de projecto* ou, simplesmente, *planeamento de projecto*.

[16] A complexidade de processamento depende aqui fundamentalmente do número e ordem das tarefas. O modelo referido é aquele que mais desafia os métodos clássicos de obtenção de uma solução para o problema de escalonamento, por conduzir a modelos de optimização combinatória enormes e difíceis (ver [Graves 1981], [Shapiro 1993], [Blazewicz 1994]).

- O conjunto de processos não ser fixo, ou o conjunto de processadores não ser fixo.

2.1.5 Escalonamento e Representação de Escalonamentos

Para escalonar completamente todas as tarefas, há que associar a cada tarefa O_i^j um tempo de início s_i^j, um valor inteiro a determinar ao construir o escalonamento. Um *escalonamento* é uma afectação de processadores do conjunto $\mathbb{P}$ (e, possivelmente de recursos adicionais do conjunto $\mathbb{R}$) a todas as tarefas dos conjuntos $\mathcal{O}^j$, com tempos de início s_i^j determinados para cada tarefa, que satisfazem as seguintes restrições [Blazewicz 1994]:

- Em cada momento cada processador está afectado no máximo a uma tarefa (se não está desocupado) e cada tarefa é processada por, ou usa, no máximo, um processador;
- Cada tarefa O_i^j é processada no intervalo $[r_j, \infty]$ (isto é: $s_i^j \in [r_j, \infty]$) e termina;
- Para cada par de tarefas O_a^j e O_b^j relacionadas por uma restrição de precedência $O_a^j \ll O_b^j$, o processamento de O_b^j não pode iniciar-se antes de O_a^j terminar (isto é, tem de ser: $s_a^j + p_a^j \leq s_b^j$);
- No caso de *escalonamento não preemptivo* nenhuma tarefa é interrompida; no caso de *escalonamento preemptivo* o número de interrupções de cada tarefa é finito;
- São satisfeitas as restrições de recurso que houver.

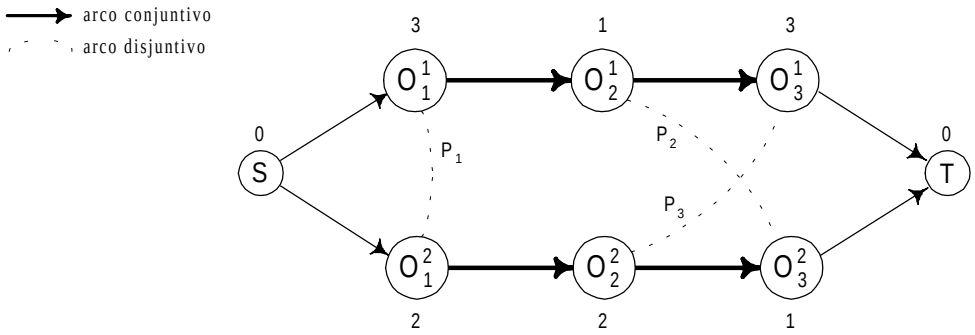
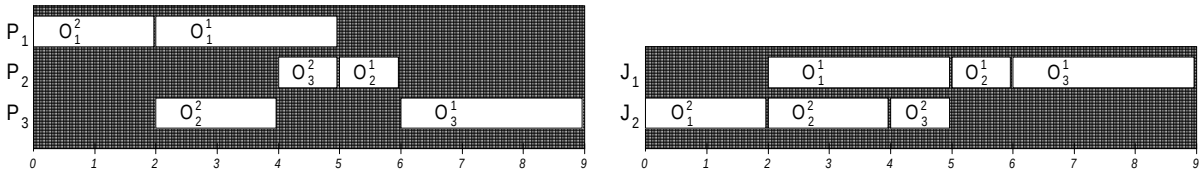


Figura 2-3- Grafo disjuntivo de um problema de escalonamento com processadores P_1, P_2 e P_3, processos $J_1=\{O_1^1, O_2^1, O_3^1\}$ e $J_2=\{O_1^2, O_2^2, O_3^2\}$. As relações de precedência são: $O_1^1 \ll O_2^1$, $O_2^1 \ll O_3^1$, $O_1^2 \ll O_2^2$ e $O_2^2 \ll O_3^2$. Os requerimentos de recurso são: O_1^1 e O_1^2 requerem 3 e 2 unidades de tempo em P_1; O_2^1 e O_3^2 requerem ambas 1 unidade de tempo em P_2; O_3^1 e O_2^2 requerem 3 e 2 unidades de tempo em P_3.

Em problemas de escalonamento de processadores dedicados com restrições de precedência pode representar-se o problema de escalonamento através de um grafo denominado *grafo disjuntivo* [Kan 1976], [Blazewicz 1994], [Brucker 1998].[17]

[17] Para medidas regulares (ver adiante), este grafo contém, implicitamente, a solução óptima para o problema de escalonamento.

Um grafo disjuntivo é um grafo de precedências que contém, adicionalmente, arcos não direccionados ligando todos os nós de tarefas que requerem o mesmo processador. Estes arcos são designados *arcos disjuntivos*; os arcos que representam as relações de precedência são também designados *arcos conjuntivos*. É usual incluir no grafo disjuntivo duas tarefas fictícias, de duração zero, cujos tempos de início delimitam o horizonte temporal de escalonamento. A título de exemplo representa-se, na Figura 2-3, o grafo disjuntivo de um problema de escalonamento com três processadores e dois processos; nesta figura, os nós S e T representam as tarefas fictícias acima referidas e os números junto de cada nó representam as durações das tarefas. Também, na Figura 2-1 sugerem-se grafos disjuntivos (nos quais não estão incluídas as tarefas fictícias).



a) Gráfico de Gantt de máquina. b) Gráfico de Gantt de processo.

Figura 2-4- Gráficos de Gantt.

Para representar escalonamentos são usados *gráficos de Gantt*, ou *mapas de Gantt*. Estes têm tipicamente a forma de um gráfico de barras, com um dos eixos representado a dimensão tempo e o outro representado as máquinas (gráfico de Gantt de máquina) ou, alternativamente, os processos (gráfico de Gantt de processo). Na Figura 2-4-a) e na Figura 2-4-b) representam-se, respectivamente, o gráfico de Gantt de máquina e o gráfico de Gantt de processo para um escalonamento que é uma solução para o problema acima descrito.

2.1.6 Medidas de Avaliação

Para avaliar a qualidade de um escalonamento e comparar escalonamentos alternativos existem uma série de medidas primárias, ou critérios, aplicáveis a processos,[18] que podem ser utilizados como função objectivo a minimizar, isto é, como função de custo. Em geral, no entanto, os escalonamentos são avaliados por medidas agregadas, ditas *objectivos de escalonamento*, obtidas a partir das medidas primárias, que resumem o desempenho de um escalonamento para o conjunto de todos os processos de um problema de escalonamento [Morton 1993].

Uma medida é dita *medida regular* se, para cada dois escalonamentos para o mesmo problema de escalonamento, valoriza sempre mais o escalonamento no qual qualquer tarefa termina mais cedo, estando as restantes escalonadas de forma igual nos dois escalonamentos. Pelo contrário, uma medida é dita *medida não regular* se isso não acontece.[19]

A uma medida primária está normalmente associado um significado físico, mais ou menos intuitivo, que tem a ver com uma determinada perspectiva de avaliar o desempenho. As medidas primárias são descritas a seguir (ver [Morton 1993], [Kan 1976], [Blazewicz 1994]):

---

[18] Cada medida primária é aqui apresentada como sendo aplicável a um processo, mas pode ser aplicada também a tarefas individuais.

[19] Para uma medida regular é irrelevante a distinção entre sequência e escalonamento (ver [Kan 1976]).

- *Tempo de fim* (*completion time*) C_j - É o tempo no qual o processo J_j termina (é igual ao tempo de fim da última tarefa do processo). A minimização de objectivos baseados em C_j corresponde à realização de um conjunto de tarefas no menor tempo possível, também libertando os processadores de modo a permitir iniciar outras tarefas mais cedo, conduzindo à maximização da utilização dos processadores;
- *Tempo de fluxo* (*flow time*) F_j - É a quantidade de tempo que o processo J_j demora a ser processado. É dado por $F_j=C_j-r_j$. Fisicamente mede a resposta do sistema de processamento a um pedido individual, já que é a duração do intervalo de tempo que decorre entre a chegada ao, e a saída do, sistema;[20]
- *Atraso relativo* (*lateness*) L_j - É a quantidade de tempo, positiva ou negativa, na qual o processamento de J_j excede a data limite. É dado por $L_j=C_j-d_j$. Fisicamente mede em quanto o processo respeita a data limite respectiva;
- *Atraso* (*tardiness*) T_j - É igual ao atraso relativo se este é positivo, ou a zero se ele é negativo. É dado por: $T_j=\max\{0,C_j-d_j\}$. Em situações em que há que ter em conta penalidades ou custos devidos a atrasos relativos positivos, mas não há penalidades ou benefícios se o atraso relativo for negativo, o uso desta medida é apropriado;[21]
- *Avanço* (*earliness*) E_j - É igual a zero se o atraso relativo é positivo, ou ao valor absoluto do atraso relativo se este é negativo. É dado por: $E_j=\max\{0,d_j-C_j\}$. É apropriada em situações em que há uma penalidade por ter terminado as tarefas mais cedo que a data limite.[22]

Todas as medidas descritas são regulares, à excepção do avanço. Relativamente às medidas agregadas a utilizar como função objectivo, existem três tipos de agregações mais comuns [Morton 1993]:

a) Médias ponderadas de uma medida primária (usam-se como pesos as prioridades w_j que reflectem a importância relativa de cada processo J_j);

b) O valor máximo ou o valor mínimo de uma medida primária;

c) Uma combinação das anteriores.

Algumas das medidas agregadas mais comuns, para utilização como funções objectivo (a minimizar), são (w_j, w_{N_j}, w_{T_j} e w_{E_j} significam pesos diferentes):

- *Comprimento* (*makespan*) C_{max} - Tempo de fim escalonado para o último processo a terminar. É dado por: $C_{max}=\max_j\{C_j\}$;
- *Soma ponderada dos tempos de fluxo* F_w - É dado por: $F_w=\sum_j w_j F_j$;
- *Soma ponderada dos atrasos relativos* L_w - É dado por: $L_w=\sum_j w_j L_j$;
- *Soma ponderada dos atrasos* T_w - É dado por: $T_w=\sum_j w_{T_j} T_j$;

---

[20] Está directamente relacionada com aquilo que é referido, num ambiente de produção industrial, por quantidade de trabalhos em curso (*work-in-process*, *work-in-progress* ou WIP): menores tempos de fluxo significam um menor volume de trabalhos em curso.

[21] Por exemplo, situações em que o cliente se pode fazer pagar pelo atraso na entrega de uma encomenda, ou realização de um serviço (não raro de forma proporcional ao atraso), ou pode mesmo recusá-los.

[22] Por exemplo, situações em que o cliente só aceita a entrega de uma encomenda após a data de entrega estipulada, havendo então, que suportar um custo de posse relativo ao produto acabado durante o tempo até àquela data.

- *Tempo de fluxo máximo* F_{max} - É dado por: $F_{max}=\max_j\{F_j\}$;
- *Atraso relativo máximo* L_{max} - É dado por: $L_{max}=\max_j\{L_j\}$;
- *Atraso máximo* T_{max} - É dado por: $T_{max}=\max_j\{T_j\}$;
- *Soma ponderada do número do processos com atraso* N_w - Dado por: $N_w=\sum_j w_{N_j}\delta(T_j)$ (com δ tal que: $\delta(x)=1$ se $x>0$ e $\delta(x)=0$ no caso contrário);
- *Somas ponderadas do avanço e do atraso* ET_w - Dado por: $ET_w=\sum_j(w_{E_j}E_j+w_{T_j}T_j)$.

2.1.7 Problema de Escalonamento, Algoritmos e Escalonamentos

Um *problema de escalonamento* é um conjunto de parâmetros que especificam processadores e processos (alguns deles sem valores numéricos atribuídos),[23] juntamente com uma medida de avaliação [Blazewicz 1994]. Para encontrar soluções para problemas de escalonamento usam-se *algoritmos de escalonamento*. Um algoritmo de escalonamento constrói um escalonamento para o problema de escalonamento determinando os valores para os tempos de início de todas as tarefas do problema.

Um escalonamento que satisfaz as restrições indicadas na secção 2.1.5 é dito *escalonamento possível*, ou *realizável*; se isso não acontece, o escalonamento é dito *escalonamento impossível* ou *não realizável*. Um *escalonamento semi-activo*, é um escalonamento possível tal que é impossível reduzir o tempo de início de qualquer tarefa sem alterar a ordem das tarefas em algum processador. Para um determinado conjunto de sequências fixas de tarefas nos processadores, um escalonamento semi-activo é o que tem os tempos de início menores possíveis para cada tarefa. Para um objectivo regular, o conjunto de escalonamentos semi-activos é um subconjunto do conjunto de escalonamentos possíveis interessante pelo facto de conter o conjunto de escalonamentos óptimos. Portanto, na procura de solução óptima podem pôr-se de parte, à partida, todos os escalonamentos que não são semi-activos. Note-se, no entanto, que o número de escalonamentos semi-activos pode ser muito grande, mesmo para um problema pequeno: para n processos e m processadores há $(n!)^m$ escalonamentos semi-activos. Por isso há normalmente interesse em restringir a procura de solução óptima a subconjuntos mais restritos de escalonamentos. Um *escalonamento activo* é um escalonamento semi-activo no qual não é possível reduzir o tempo de início de qualquer tarefa sem aumentar o de, pelo menos, uma outra tarefa. O conjunto de escalonamentos activos para um dado problema contém escalonamentos óptimos para esse problema para qualquer medida regular [Kan 1976], [Morton 1993]. Um *escalonamento sem-atraso*, ou *escalonamento de despacho*, é um escalonamento activo em que cada processador não tem tempos de paragem, isto é, os tempos de não ocupação são zero, à excepção de antes de iniciar, ou após terminar, o processamento das tarefas afectadas ao processador. Um escalonamento de despacho, embora seja um escalonamento activo, pode não ser um escalonamento óptimo.

[23] Uma *instância* do problema é obtida quando se especificam valores particulares para todos os parâmetros do problema.

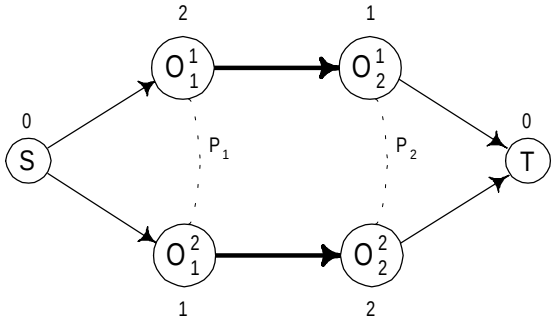
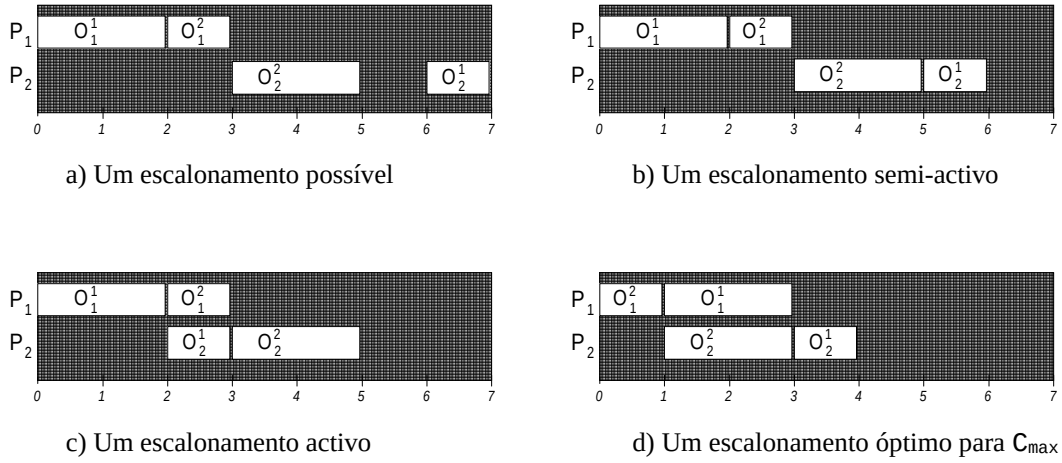


Figura 2-5- Grafo disjuntivo para um problema de escalonamento com dois processadores e dois processos.

A título de exemplo, representa-se na Figura 2-5 o grafo disjuntivo de um problema com 2 processos com 2 tarefas cada (com durações indicadas junto dos nós), com 2 processadores dedicados.



a) Um escalonamento possível

b) Um escalonamento semi-activo

c) Um escalonamento activo

d) Um escalonamento óptimo para C_{max}

Figura 2-6- Gráficos de Gantt de máquina de escalonamentos que são soluções possíveis para o problema da Figura 2-5.

Para este problema de escalonamento mostram-se na Figura 2-6 quatro escalonamentos possíveis. Destes, o escalonamento a) não é semi-activo e o escalonamento b) é semi-activo; o escalonamento c) é um escalonamento activo e o escalonamento d) é um escalonamento óptimo para o objectivo C_{max}; os escalonamentos c) e d) são escalonamentos sem-atraso.

2.2 Abordagens ao Problema do Escalonamento

Formalmente, os problemas de escalonamento são, em geral, colocados como problemas de optimização, mais especificamente como problemas de optimização combinatória, como já foi dito. Para encontrar a solução de um problema de escalonamento pode usar-se um *método exacto* que faz uso de um *algoritmo de optimização*. Infelizmente, devido à inerente

complexidade de muitos problemas de optimização nem sempre é possível encontrar, através de um algoritmo de optimização, a solução óptima do problema em tempo útil. Este é o caso geral do problema de escalonamento. Efectivamente o problema de escalonamento é um dos problemas que pertence à classe de problemas *NP-difícil* [Garey 1979]. Como se viu, para um problema com n processos e m processadores, no conjunto de todos os escalonamentos possíveis existe um subconjunto de $(n!)^m$ escalonamentos interessantes, o dos escalonamentos semi-activos, onde estão contidas as soluções óptimas. Mesmo para valores relativamente pequenos de n e m o número de escalonamentos semi-activos é elevado mas, para além disso, esse número aumenta numa razão exponencial em relação à dimensão do problema (número de processadores e processos). Estes factores tornam o problema intratável do ponto de vista computacional: não se descobriu até agora método algorítmico algum que consiga encontrar a solução óptima para qualquer problema de escalonamento em tempo polinomial relativamente à dimensão do problema, embora para certos casos particulares possam existir tais algoritmos de optimização. A *enumeração completa* dos escalonamentos possíveis para um determinado caso de problema de escalonamento — construir e avaliar cada escalonamento possível segundo o objectivo especificado, e concluir por fim devolvendo o melhor dos escalonamentos — um método exacto possível que pode parecer intuitivamente natural, mesmo limitando-se à enumeração dos escalonamentos semi-activos parece, assim, fora de questão.[24] Segundo [Blazewicz 1994], pode-se optar pelas seguintes abordagens possíveis:

- Usar um *método aproximado*, introduzindo *simplificações* no problema original por relaxação de algumas das restrições nele impostas (por exemplo, permitir preempção, mesmo se o problema não é de escalonamento preemptivo; assumir tarefas com duração unitária, mesmo não sendo o caso no problema original; assumir certo tipo de grafos de precedência como árvores, ou cadeias, quando o grafo é qualquer no problema original). Isto equivale a resolver uma versão simplificada do problema real, cuja solução (sub-óptima) pode ser, ou não, uma boa aproximação da solução do problema original;
- Usar *métodos aproximados* que fazem uso de *algoritmos heurísticos*. Estes algoritmos *tendem* a encontrar um escalonamento óptimo, de uma forma mais expedita, mas sem sucesso garantido. Enquadram-se aqui o algoritmo de escalonamento com *fila de prioridade*, aplicado à sequenciação das tarefas nas máquinas, ou os *métodos heurísticos*, descritos mais adiante;
- Usar *algoritmos exactos* se o problema é resolúvel por um algoritmo de optimização *pseudopolinomial*, cuja função de complexidade, no pior caso, é limitada por um polinómio do maior valor presente na instância do problema. Para números razoavelmente pequenos o algoritmo pode mesmo comportar-se bem na prática, podendo mesmo ser usado num programa de computador;

Na secção seguinte descrevem-se várias abordagens ao problema do escalonamento, classificadas nas da área de IO, incluindo aqui também métodos heurísticos (e em particular alguns métodos heurísticos mais modernos apelidados de meta-heurísticas) e nas da área de

[24] Por exemplo, para $n=5$ e $m=5$, $(n!)^m=24883200000$; para $n=5$ e $m=6$, $(n!)^m=2985984000000$; para $n=5$ e $m=7$, $(n!)^m=358318080000000$. Um sistema que possa gerar e avaliar 100000 escalonamentos por segundo, precisaria de cerca de 3 dias, 1 ano e 114 anos, em cada caso, respectivamente, para poder determinar qual o escalonamento óptimo. A aplicabilidade de um método de enumeração completa é, portanto, limitada a casos especiais de problemas com dimensão reduzida, sem interesse prático.

Inteligência Artificial (IA) e de Inteligência Artificial Distribuída (IAD).[25] Embora esta classificação seja frequentemente adoptada — veja-se, por exemplo, [Hildum 1994], [Rabelo 1996a] ou [Kjenstad 1998] — e muito embora as abordagens de IA se distingam das de IO no emprego de conhecimento específico do domínio e na ênfase colocada no aspecto da representação das várias características e restrições presentes nos problemas práticos de escalonamento, deve fazer-se notar que ela é um pouco artificial já que, na prática, nos sistemas de escalonamento modernos se misturam e empregam uma grande variedade de métodos, oriundos de ambas as áreas.

2.2.1 Abordagens de Investigação Operacional

Nesta secção descrevem-se abordagens ao problema de escalonamento convencional em duas subsecções compreendendo métodos exactos e métodos heurísticos. Uma terceira subsecção é dedicada ao problema de escalonamento no contexto das redes de produção e distribuição.

2.2.1.1 Métodos Exactos

Os métodos descritos nesta secção caracterizam-se por tentarem encontrar soluções óptimas exactas para o problema de escalonamento entendido como um problema de optimização.

2.2.1.1.1 Programação Matemática

Um método que merece destaque, pela forma matemática e elegante com que permite formular o problema é o da *programação matemática*.[26] A partir dos anos 60 o uso do computador pôde ser cada vez mais explorado e isto permitiu a formulação de problemas de escalonamento segundo modelos de programação matemática e a sua resolução, em princípio, exacta (veja-se [Kan 1976], Capítulo 3, para exemplos de formulação de problemas de escalonamento segundo este tipo de modelos e [Morton 1993], Capítulo 2, para referências) usando, por exemplo, usando o método *Simplex*.

Na prática, a aplicação dos métodos de programação matemática a problemas de escalonamento reais origina frequentemente modelos complexos, isto é, com um elevado número de variáveis e de restrições, mesmo para um problema pequeno; além disso, os métodos de programação matemática não tiram partido da estrutura especial dos problemas de escalonamento [Kan 1976]. A título de exemplo, um modelo de programação inteira mista para o problema de escalonamento processamento intermitente apresentado em [Shapiro 1993], num caso com `10` processos de `10` tarefas cada um e com `8` máquinas dá origem a `2580` restrições, `201` variáveis contínuas e `240` variáveis indicadores (com valores possíveis `0` ou `1`). Para contornar a dificuldade computacional na resolução do problema de escalonamento,

---

[25] Não houve a intenção de rever, analisar ou classificar detalhada e exaustivamente os métodos de solução propostos pela metodologia clássica da área de IO para cada tipo de problema de escalonamento. Esta revisão, análise ou classificação dos métodos de solução clássicos poderia ser feita consultando obras como [Blazewicz 1994], [Kan 1976], ou [Lawler 1989], aliás citadas ao longo do texto. [Morton 1988] dá uma perspectiva mais abrangente e mais moderna do escalonamento, embora se oriente mais para métodos heurísticos baseados em *regras de despacho* (ver adiante) e simulação. Relativamente aos métodos mais recentes e originários da área de IA descrever-se-ão métodos e trabalhos considerados mais relevantes.

[26] Esta secção refere-se, mais especificamente, à *programação linear*. O termo *programação matemática* é, no entanto, em geral, considerado mais abrangente e inclui *análise de redes*, *programação dinâmica*, *teoria dos jogos* e *programação não linear*, para além da programação linear inteira [Hillier 1990].

vários investigadores sugerem combinações do método de *ramificar-e-limitar* (ver mais adiante), métodos heurísticos e limites inferiores para o valor de critério de avaliação de escalonamentos obtidos a partir de versões relaxadas/simplificadas, facilmente optimizáveis, do modelo do mesmo problema [Shapiro 1993].

2.2.1.1.2 Ramificar-e-Limitar

Os métodos apelidados de *ramificar-e-limitar* (*branch-and-bound*) foram inicialmente desenvolvidos e usados no contexto da resolução de problemas de programação inteira mista [Land 1960] e do problema do caixeiro viajante [Eastman 1959], sendo dos que mais foram usados para resolver problemas de optimização combinatória. Num método de ramificar-e-limitar é seguido um princípio de *enumeração implícita*, isto é, consideram-se certas soluções possíveis apenas indirectamente e sem as avaliar explicitamente (enumera parcialmente, portanto). Um outro princípio subjacente é o da *divisão do problema em subproblemas*, independentes e mais simples. O método consiste em considerar subconjuntos sucessivamente mais pequenos do conjunto de soluções possíveis (correspondendo a subproblemas do problema original cada vez mais pequenos) até obter conjuntos com uma única solução, ou conjuntos que, por certo, não contêm a solução óptima e que são postos de parte na procura da solução. Há três aspectos básicos neste método:

- O procedimento de *ramificar* (*branch*) - Partição de um problema grande em dois ou mais subproblemas;
- O procedimento de *limitar* (*bound*) - Calcula um limite inferior do valor de critério (função de custo) da solução óptima para cada subproblema;
- A estratégia de procura usada - Que subproblemas que são escolhidos para tomar em consideração em cada passo.

A execução do procedimento ramificar pode ser representada por uma *árvore de procura*, em que o problema original corresponde ao nó da raiz e os subproblemas de um problema aos nós filhos de um nó do nível imediatamente anterior. É mantida uma lista de nós não processados, correspondentes aos subproblemas ainda não eliminados, ou àqueles cujos subproblemas não foram ainda gerados. O modo como esta lista é gerida e escolhido o próximo nó para expandir em nós filhos depende da estratégia usada no método. É também memorizado o valor de critério para a melhor solução encontrada até ao presente. Este valor serve como *limite superior* para soluções, ou conjuntos de soluções, em subproblemas que se encontrarem nos passos seguintes. O procedimento de limitar permite calcular um *limite inferior* para o valor de critério para cada subproblema. Sempre que este limite é maior que o limite superior conclui-se que não vale a pena explorar o nó correspondente, isto é, não vale a pena gerar os seus subproblemas pois, através deles, as soluções que se iriam encontrar teriam um valor maior (pior, portanto) do que uma já encontrada. Isto significa que um (possivelmente grande) número de soluções possíveis não óptimas foi posto de parte, na procura da solução.

Uma estratégia possível é a de procura em árvore em profundidade (*depth-first*), em que os nós de um ramo da árvore de procura são explorados até ao fim (até gerar uma solução completa, possível ou não), retrocedendo, se necessário, para ir expandir nós em níveis anteriores. Uma estratégia alternativa é a de *procura com salto* ou procura em fronteira (*jumptracking*, ver [Blazewicz 1994]), que mantém uma fila de nós ainda não processados, escolhendo em cada passo aquele que tiver menor limite inferior. Esta última estratégia origina uma procura muito semelhante à do algoritmo conhecido por *procura em faixa* (*beam search*), que é uma variante da *procura pelo melhor* (*best-first search*), frequentemente referido na

literatura de IA (ver o Apêndice B). O método da procura pelo melhor mantém uma fila de *todos* os nós ainda não processados da árvore de procura ordenados pelo valor de uma função de avaliação que indica em quanto cada nó é mais promissor, no sentido de conduzir por um caminho mais curto a uma solução, *i.e.*, uma função que é heurística; a procura em faixa, mantém apenas um pequeno número desses nós. A diferença entre o método de ramificar-e-limitar e a procura em faixa é que, enquanto que o ramificar-e-limitar não explora aqueles ramos da árvore de procura que são *por certo* inúteis, a procura pelo melhor não explora aqueles que *parecem* ser inúteis, de acordo com a função de avaliação. Portanto, a procura em faixa pode ser descrita como um método de ramificar-e-limitar aproximado.

A simplicidade do princípio básico do ramificar-e-limitar, a sua fácil realização e eficiência computacionais parecem ser as razões básicas da sua popularidade [Lenstra 1975]. No entanto, pela sua natureza, este tipo de métodos tem um comportamento computacional imprevisível (não há nenhuma maneira de prever, em geral, quantos nós não serão examinados pelo facto de terem associados limites inferiores suficientemente grandes), para além de permanecerem intratáveis, pois o número de passos do algoritmo aumenta numa razão exponencial em relação ao tamanho do problema dado [Kan 1976], [Blazewicz 1994].

2.2.1.1.3 Programação Dinâmica

A *programação dinâmica* é também um método exacto de enumeração implícita e baseado no princípio da subdivisão do problema em subproblemas. O método foi desenvolvido nos anos 50 por Bellman [Bellman 1957], [Bellman 1962]. Segundo o método da programação dinâmica o problema de escalonamento (ou outro problema de optimização) é visto como um processo de decisão em vários estágios. Em cada estágio deve ser tomada uma decisão que tem impacto nas decisões a tomar em estágios subsequentes. O princípio de optimalidade de Bellman, que diz que, começando-se num estágio qualquer, a política óptima para os estágios subsequentes é independente da aplicada nos estágios anteriores, serve para obter uma equação recursiva que descreve o valor óptimo de critério num dado estágio em função do valor prévio. Este princípio é aplicável a muitos problemas de optimização.

Este método leva à exploração de um grafo de procura, numa forma similar à da árvore de procura do ramificar-e-limitar, fazendo uso também de um limite inferior dado pelos valores de custos até ao estágio actual e pondo de parte conjuntos de alternativas possíveis. À semelhança do que foi dito para o método de ramificar-e-limitar, também o método da programação dinâmica não é computacionalmente atractivo para problemas grandes. Nesses casos, o tamanho do grafo de procura é, em geral, grande e o mecanismo para pôr de parte conjuntos de soluções possíveis sub-óptimas pode ser bastante fraco [Kan 1976].

2.2.1.1.4 Métodos Baseados em Análise de Redes (PERT e CPM)

PERT (*Program Review and Evaluation Technique*, Revisão de Programa e Técnica de Avaliação) e CPM (*Critical Path Method*, ou Método do Caminho Crítico) são métodos de apoio ao planeamento de projectos. Estes métodos são usualmente apelidados de *métodos tipo PERT*, ou *métodos de caminho crítico* e baseiam-se na representação de projectos na forma de redes, frequentemente grafos do tipo AOA. São usados para, por exemplo, determinar a duração mínima do escalonamento de um projecto, determinar a probabilidade de satisfazer prazos predeterminados, identificar as tarefas mais críticas ou avaliar o efeito de alterações ou desvios no escalonamento de um projecto [Hillier 1990].

Ambos os métodos, PERT e CPM, se baseiam inicialmente na determinação de um caminho entre o início e o fim da rede de projecto denominado *caminho crítico*. Este caminho contém

as *tarefas críticas*, *i.e.*, as tarefas do projecto que, pela sua duração, impõem um comprimento mínimo do escalonamento do projecto. É frequente, na aplicação simplificada dos métodos de caminho crítico, considerar que as durações das tarefas são fixas e ignorar restrições de recurso, para determinar o caminho crítico. Em projectos com um número de tarefas relativamente pequeno esta abordagem é simples e presta-se a uso não automatizado. Numa fase posterior, podem tomar-se em consideração durações variáveis, ou durações prováveis, para as tarefas e restrições de recurso mais ou menos complexas [Morton 1993]. No caso do método CPM, a ideia básica é a de reduzir o comprimento do escalonamento do projecto reduzindo a duração das tarefas críticas, através da ampliação da capacidade dos recursos que aquelas tarefas requerem. Como a ampliação da capacidade dos recursos envolve custos é, então, realizada uma análise em que se pondera investir mais, ou menos, na capacidade dos recursos para obter uma menor, ou maior, duração total do projecto. No caso do método PERT, considera-se que as durações das tarefas não são conhecidas de forma exacta, mas têm uma distribuição de probabilidade conhecida. Estimativas da distribuição de probabilidade da duração das tarefas são usadas para obter a distribuição de probabilidade para a duração total do projecto. No que respeita à escolha de um dos métodos, o método CPM é particularmente apropriado quando as durações das tarefas são bem conhecidas, podem ser ajustadas por variação da capacidade dos recursos e o compromisso entre a duração e o custo do projecto é importante; o método PERT é adequado quando há uma grande incerteza na duração das tarefas e é importante controlar o escalonamento do projecto.

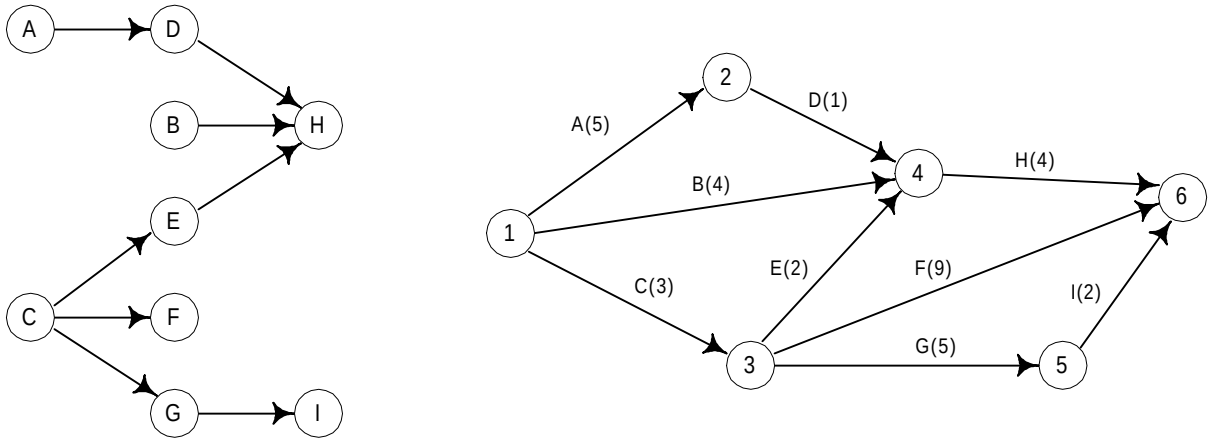
Uma descrição dos métodos de caminho crítico pode ser encontrada em [Hillier 1990] (no capítulo 10), ou em [Schroeder 1993] (no capítulo 15). Para um grau de detalhe maior (incluindo a tomada em consideração das restrições de capacidade dos recursos e uso de heurísticas) ver, por exemplo, [Morton 1993] (nos capítulos 17 e 18). A seguir descrevem-se, de forma resumida, os conceitos básicos associados a estes métodos bem como a sua aplicação, através de um exemplo simples envolvendo a determinação do caminho crítico e do comprimento mínimo do escalonamento de um projecto, com durações de tarefas fixas e sem restrições de recurso.

Para uma aplicação manual dos métodos de caminho crítico é mais frequente o grafo de precedências das tarefas do projecto ser representado por uma rede do tipo AOA. Neste tipo de redes, os nós estão associados a eventos de início e de fim de tarefas e os arcos estão associados a tarefas, com um sentido do evento de início para o evento de fim; cada tarefa deve ser representada por um único arco, não devendo existir mais de uma tarefa associada ao mesmo par de eventos de início e fim e a rede deve ter eventos de início e fim únicos (para que se verifiquem estes requisitos não é raro terem de ser introduzidas, nas redes do tipo AOA, tarefas fictícias cuja duração é nula); aos arcos associam-se "comprimentos", que representam a duração das tarefas; considera-se que o comprimento de um caminho na rede (respeitando os sentidos dos arcos) é a soma dos comprimentos dos arcos que o constituem. Entre todos os caminhos possíveis do evento de início até ao evento de fim, o caminho mais comprido representa o caminho crítico do projecto. O caminho crítico, que poderá não ser único, impõe uma duração mínima para o projecto e é composto por uma sequência de tarefas que, assumindo um escalonamento semi-activo, não têm flexibilidade temporal, *i.e.*, todas as tarefas do caminho crítico têm folga temporal nula. Pelo contrário, as tarefas que não pertencem a um caminho crítico, têm flexibilidade temporal.

Tome-se, como exemplo, o projecto constituído por nove tarefas com as precedências e as durações dadas pela Tabela 2-1. Na Figura 2-7-a) representa-se a rede AON e na Figura 2-7-b) a rede AOA para o mesmo projecto. Para determinar o caminho crítico do projecto considerem-se, no grafo da Figura 2-7-b), os caminhos possíveis partindo do nó 1 (correspondente ao evento 1, o evento de início do projecto) ao nó 6 (correspondente ao evento 6, o evento de fim do projecto). É fácil verificar, neste caso, que existe apenas um caminho mais comprido, o caminho que segue pelos arcos C e F, com comprimento 12. Portanto, existe um único caminho crítico, constituído pelas tarefas C e F (tarefas críticas) e a duração mínima do projecto é de 12 unidades de tempo.

Tabela 2-1- Tarefas, precedências e durações de um projecto.

tarefa	tarefas antecessoras	duração
A	-	5
B	-	4
C	-	3
D	A	1
E	C	2
F	C	9
G	C	5
H	B, D, E	4
I	G	2



a) Rede AON. b) Rede AOA.

Figura 2-7- Redes AON e AOA de um projecto. Na rede AOA os nós (1,...,6) representam eventos; cada arco é etiquetado pelo identificador e pela duração da tarefa que representa.

Neste exemplo, dado que o problema é pequeno, a determinação do caminho crítico não envolve dificuldade. No entanto, para problemas maiores (com elevado número de tarefas) esta determinação tem de ser realizada de uma forma mais sistemática, não sendo raro recorrer-se ao apoio de um programa de computador apropriado. A seguir exemplifica-se, para o exemplo em questão, a determinação manual do caminho crítico através de um algoritmo que sistematiza a determinação do caminho crítico, introduzindo previamente definições de parâmetros frequentemente empregues nos métodos de caminho crítico.

A cada evento são associados dois parâmetros temporais:

TC - O tempo mais cedo. É o menor tempo em que o evento pode acontecer;

TT - O tempo mais tarde. É o maior tempo em que um evento pode acontecer sem que origine atrasos.

A cada tarefa associam-se quatro parâmetros temporais:

TC_i - O tempo mais cedo do evento de início; TC_f - O tempo mais cedo do evento de fim;

TT_i - O tempo mais tarde do evento de início; TT_f - O tempo mais tarde do evento de fim.

Adicionalmente, para cada tarefa definem-se:

IC - O tempo de início mais cedo. Dado por: $IC=TC_i$;

FC - O tempo de fim mais cedo. Dado por: FC=IC+d (d é a duração da tarefa);

FT - O tempo de fim mais tarde. Dado por $FT=TT_f$;

IT - O tempo de início mais tarde. Dado por: IT=FT-d.

É usual definirem-se, para cada tarefa, as seguintes folgas temporais:

F_{total} - Folga total. Dada por: $F_{total}=FT-FC$. Significa o máximo atraso da tarefa não havendo atraso no caminho passando pela tarefa. É a folga mais utilizada;

F_{seg} - Folga de segurança. Dada por: $F_{seg}=IT-TT_i$. Significa o máximo atraso da tarefa se as tarefas antecessoras utilizarem toda a folga;

F_{livre} - Folga livre. Dada por: $F_{livre}=TC_f-FC$. Significa o máximo atraso da tarefa se as tarefas sucessoras utilizarem toda a folga;

F_{ind} - Folga independente. Dada por: $F_{ind}=max\{0,TC_f-(TT_i+d)\}$. Significa o máximo atraso da tarefa se as tarefas antecessoras e as sucessoras utilizarem toda a folga.

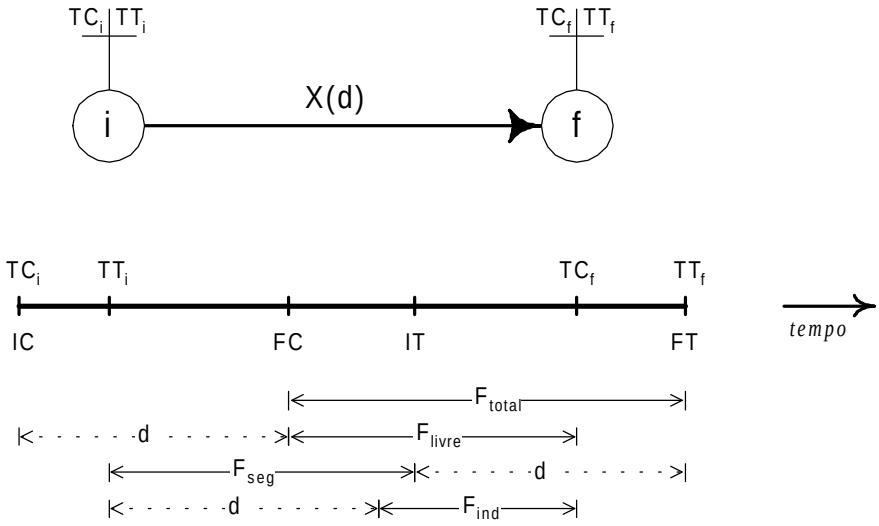


Figura 2-8- Parâmetros de escalonamento associados aos métodos de caminho crítico, para uma tarefa X com duração d.

Todos os parâmetros definidos são representados na Figura 2-8 na forma de um arco de rede AOA de uma tarefa genérica X e, adicionalmente, representados numa linha temporal.

Para a determinação do caminho crítico é usado o algoritmo a seguir descrito, que serve para calcular, de forma sistemática, os tempos TC e TT de todos os eventos do projecto. Para uso correcto do algoritmo, os nós da rede devem ter sido previamente numerados de modo que o número do nó de partida de qualquer arco seja menor do que o número do nó de chegada.

1. Cálculo dos TCs (passagem para a frente):

1.1. O TC do nó inicial é igual a 0;

1.2. Para cada nó, e por ordem crescente do número de nó, adicionar ao TC de cada nó antecessor o comprimento do arco entre o antecessor e o nó. O TC do nó é igual ao máximo dos valores resultantes;

2. Cálculo dos TTs (passagem para trás):

2.1. O TT do último nó é igual ao seu TC;

2.2. Para cada nó, e por ordem decrescente do número de nó, subtrair ao TT de cada nó sucessor o comprimento do arco entre o nó e o sucessor. O TT do nó é igual ao mínimo dos valores resultantes.

Na Figura 2-9 representa-se graficamente o resultado do cálculo efectuado através deste algoritmo, para o projecto do exemplo, conforme é sugerido na Figura 2-8.

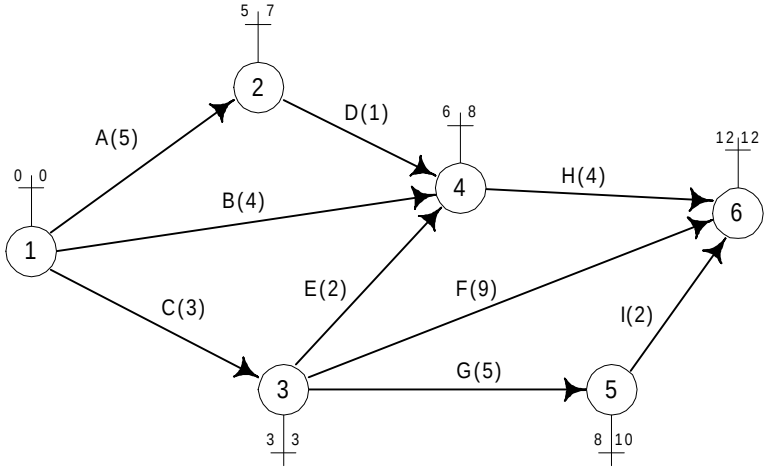


Figura 2-9- Rede AOA de projecto com TCs e TTs calculados.

A seguir calculam-se as folgas temporais para cada tarefa. A Tabela 2-2 resume os resultados.

Tabela 2-2- Resultados do cálculo dos tempos de eventos e das folgas temporais de um projecto.

tarefa	duração	TC_i	TT_i	TC_f	TT_f	IC	FC	FT	IT	F_{total}	F_{seg}	F_{livre}	F_{ind}
A	5	0	0	5	7	0	5	7	2	2	2	0	0
B	4	0	0	6	8	0	4	8	4	4	4	2	2
C	3	0	0	3	3	0	3	3	0	0	0	0	0
D	1	5	7	6	8	5	6	8	7	2	0	0	0
E	2	3	3	6	8	3	5	8	6	3	3	1	1
F	9	3	3	12	12	3	12	12	3	0	0	0	0
G	5	3	3	8	10	3	8	10	5	2	2	0	0
H	4	6	8	12	12	6	10	12	8	2	0	2	0
I	2	8	10	12	12	8	10	12	10	2	0	2	0

Um caminho crítico é composto por uma sequência de tarefas cujas folgas são todas nulas e corresponde, numa rede AOA anotada com TCs e TTs como a da Figura 2-9, a um caminho composto pela sequência de arcos que unem nós cujos TCs são iguais aos TTs. No caso do

projecto do exemplo pode ver-se (pela Tabela 2-2 ou pela Figura 2-9) que existe um caminho crítico composto pela sequência de tarefas C e F, que tem a duração de 12 unidades de tempo.

2.2.1.2 Métodos Heurísticos

Embora todo o trabalho teórico de índole matemática orientado para métodos exactos tenha trazido vários avanços ao longo dos anos para o problema do escalonamento, ele não é completamente bem sucedido em aplicações realistas. A complexidade revela-se um obstáculo com algoritmos de optimização combinatória, salvo em casos em que se introduzam simplificações que, não raro, são tão drásticas afastam o modelo do problema real. Este aspecto tem sido referido na literatura clássica e moderna sobre escalonamento, por vários investigadores.

Por exemplo, em [Graves 1981] reconhecia-se que "... há um distanciamento entre teoria e prática do escalonamento da produção" e que "o desenvolvimento de melhores algoritmos e heurísticas é essencial para que a teoria do escalonamento possa continuar a melhorar a prática do escalonamento". Para além disto, e no que respeita a direcções futuras (da investigação, em 1981) este autor dizia haver uma grande necessidade "... de modelos de escalonamento mais realistas e de uma melhor compreensão da dinâmica inerente ao ambiente de escalonamento".

Também segundo [Dorn 1994a] há três tipos de problemas com os modelos analíticos e formais:

- Os algoritmos são demasiado complexos para aplicações do mundo real;
- Os modelos exigem o conhecimento exacto de durações e restrições técnicas;
- O esforço necessário para formalizar um novo problema é considerável;

Em face destes aspectos na abordagem a problemas reais de escalonamento recorre-se frequentemente a métodos aproximados, de tipo heurístico. Nesta secção descrevem-se estes métodos. Segundo [Reeves 1993a], sendo as heurísticas em geral mais flexíveis e capazes de tratar funções objectivo e restrições mais complexas, tornam possíveis modelos exactos, ou pelo menos, muito mais aproximados do problema real. Enquanto que com os outros métodos (clássicos) nos é possível, em geral, obter soluções aproximadas de um modelo exacto do problema real (havia um modelo exacto, por exemplo de programação matemática, do problema, mas ao qual era necessário introduzir simplificações), com métodos heurísticos poderemos modelar o mundo real de forma mais precisa (ter um modelo exacto ou, pelo menos, tanto quanto possível exacto) e obter soluções que poderão (ou não) ser aproximadas [Reeves 1993a].

2.2.1.2.1 Heurísticas

Certas heurísticas são tão específicas e dependentes do tipo de problema que só podem ser usadas para um tipo de problema particular. Este é o caso do escalonamento baseado em *regras heurísticas*, *regras despacho*, ou *regras de prioridade* [Baker 1974], [Kan 1976], [Morton 1993] que têm tradicionalmente servido para produzir boas soluções para problemas de escalonamento de processamento intermitente. Tratam-se de regras heurísticas de decisão que produzem soluções boas, possivelmente sub-óptimas, em que a construção de um escalonamento é feita escalonando as tarefas em cada processador para a frente no tempo.

No escalonamento baseado em regras de despacho há, para cada processador, uma fila de tarefas por escalonar e, em cada passo, é escolhida para processar a primeira tarefa da fila. O

desempenho deste método aproximado depende da ordem pela qual as tarefas a escalonar são colocadas na fila e do objectivo de escalonamento [Blazewicz 1994]. A ordem das tarefas a escalonar na fila é prescrita por uma regra de prioridade. Várias regras de prioridade podem ser empregues e estudos de simulação demonstram que, embora certas regras sejam, em geral, melhores que outras, nenhuma regra é melhor em qualquer circunstância [Kan 1976], [Brown 1995]. Descrevem-se a seguir algumas destas regras (para maior detalhe e outras regras ver [Kan 1976], [Chase 1995] ou [Roldão 1995]).

- EDD (*earliest due date*, ou data limite mais cedo) - Regra da *menor data limite*: prioridade maior à tarefa do processo com menor data limite;
- STR (*slack time remaining*, ou tempo de folga restante) - Regra da *folga mínima*: prioridade maior à tarefa do processo com menor folga. A *folga* é a diferença entre o tempo restante antes da data limite e o tempo de processamento restante;
- SPT (*shortest processing time*, ou tempo de processamento mais curto) - Regra do *menor tempo de processamento*: prioridade à tarefa com o tempo de processamento mais curto. Considerando um peso, ou importância, para cada processo, temos a regra WSPT (*weighted shortest processing time*) que dá prioridade à tarefa com maior quociente entre o peso do processo e tempo de processamento da tarefa;
- FIFO (*first in first out*, ou por ordem de chegada) - Regra *primeiro a entrar-primeiro a sair*: prioridade maior à tarefa que primeiro fica disponível para processamento;
- RANDOM (por sorteio) - A prioridade é dada a uma tarefa escolhida aleatoriamente;
- CR (*critical ratio*, ou razão crítica) - Regra da menor *razão crítica*: prioridade à tarefa com menor razão crítica. A *razão crítica* é o quociente entre o tempo restante antes da data limite e o tempo de processamento (haverá atraso se a razão crítica for menor que 1);
- NOP (número de tarefas) - Prioridade à tarefa cujo processo tem menor número de tarefas por executar.

Segundo [Kan 1976], experiências feitas em larga escala indicam que as regras de prioridade funcionam bem na obtenção de escalonamentos sem atraso, quando é importante respeitar datas limite e em problemas de escalonamento dinâmico. Estas razões justificam talvez a sua popularidade nos ambientes de produção industrial. Segundo [Roldão 1995], e para ambientes de produção industrial, a regra SPT funciona bem em ambientes congestionados (quando há processadores críticos, isto é, com muita procura pelas tarefas) e tem bom desempenho quando os processos em curso não estão com atraso relativamente às datas limite; a regra NOP é eficiente quando o número de tarefas é elevado (devido à maior percentagem de tempos de espera, que são proporcionais ao número de tarefas); as regras EDD, STR e CR só são boas quando os níveis de carga são baixos; as regras EDD e STR são pouco aplicáveis quando há tempos de espera curtos entre tarefas e há pouca folga (data limite muito próxima); a conjugação das regras SPT e NOP origina frequentemente bons resultados.

Numa apreciação geral pode dizer-se que as regras de prioridade são muito especializadas, originando uma optimização muito local do escalonamento, insuficiente em problemas de escalonamento complexos. Usando regras de prioridade, as decisões de escalonamento são tomadas na ordem da sua execução (são regras de despacho) e não são revogáveis (são procedimentos de uma única passagem) [Kan 1976]. Daqui que o escalonamento obtido unicamente através destes métodos não seja, em geral, um escalonamento globalmente óptimo. Em [Panwalkar 1977] é apresentada uma revisão das regras de despacho de escalonamento.

Em [Morton 1993] é apresentado, com considerável detalhe, trabalho mais recente sobre regras de despacho e sua aplicação a variados problemas de escalonamento estáticos e dinâmicos, para várias funções objectivo de escalonamento. Numa classificação apresentada naquela obra, as regras de despacho são enquadradas numa classe mais geral apelidada de *heurísticas* que são divididas em:

- *Heurísticas de uma passagem* (*one-pass heuristics*) - Operam construindo passo a passo uma única solução completa. Enquadram-se aqui as regras de despacho clássicas, tal como as que até aqui foram descritas;
- *Heurísticas de passagem múltipla* (*multi-pass heuristics*) - Consistem na aplicação repetida de heurísticas de uma passagem e tentam obter melhores soluções gerando muitas. São mais sofisticadas e têm, em geral, um custo computacional bastante mais elevado.

Dentro das regras das heurísticas de uma passagem distinguem-se:

- *Heurísticas de sequenciação de permutações múltiplas* - Atribuem uma sequência previamente fixada dos processos a cada máquina; para evitar situações de *bloqueio mútuo* entre tarefas (*gridlock*)[27] essa regra é aplicada para sequenciar só as tarefas em fila de espera, isto é, assim que uma máquina ficar livre não esperará por uma tarefa que não esteja disponível para processamento;
- *Heurísticas de despacho simples* (*simple dispatch heuristics*) - Funcionam dando prioridade, numa máquina disponível, à tarefa em espera do processo com maior prioridade; se não há fila de espera uma tarefa que chegue é imediatamente processada. Estas prioridades podem ser: a) *estáticas*, se só são calculadas uma vez, no início (por exemplo, EDD, SPT ou WSPT); b) *dinâmicas*, se devem ser recalculadas de cada vez que a tarefa tem de ser escolhida para processamento (por exemplo, STR); c) *globais*, se não dependem da posição (máquina) no sistema (por exemplo, EDD, menor tempo de chegada de processo); d) *locais*, se só dependem da situação local (máquina) no sistema (por exemplo, SPT) ou e) *previsionais*, se dependem da situação local e de uma previsão do que vai acontecer ao processo até estar acabado (por exemplo, CR);
- *Heurísticas de despacho míopes* (*one-pass myopic dispatch heuristics*) - Tomam uma decisão numa máquina de modo a (aproximadamente) optimizar como se o problema fosse de uma só máquina. Estudos empíricos comprovam que estas heurísticas funcionam quase sempre muito bem. No entanto, se o objectivo de escalonamento se baseia em datas limite isso implica determinar uma estimativa de uma data limite de tarefa para a terminação de cada tarefa na máquina; a regra produzirá tanto melhores resultados quanto melhor a estimativa;[28]
- *Heurísticas simples baseadas em congestionamentos* (*simple bottleneck heuristics*) - Resultam de heurísticas de despacho míopes melhoradas para terem em conta máquinas congestionadas a jusante da máquina considerada, usando métodos baseados em

[27] Esta situação pode acontecer porque em processamento intermitente os processos podem ter caminhos diferentes pelas máquinas do sistema de processamento, sendo possível que, com sequências fixas em cada máquina, uma máquina tenha de esperar por uma tarefa que está em fila de espera noutra máquina que por sua vez está esperando por uma tarefa em fila de espera da primeira máquina.

[28] Estas estimativas podem ser obtidas recorrendo a um historial de valores, tendo ou não em conta a quantidade de trabalhos no sistema de processamento ou, se forem usadas regras de passagem múltipla, recorrendo ao valor da última iteração.

congestionamentos. Nestes métodos considera-se mais crítico e prioritário o escalonamento no recurso mais congestionado, isto é, no recurso com maior utilização.

Dentro das regras heurísticas de passagem múltipla descritas em [Morton 1993] distinguem-se a *procura em faixa guiada localmente* e o *despacho iterativo* ou o *despacho baseado em congestionamentos*. O método usado no primeiro caso consiste basicamente em utilizar a procura em faixa para conduzir uma simulação do sistema de processamento, explorando um número limitado de escolhas possíveis de tarefas em cada máquina e usando, como função de avaliação das escolhas, uma heurística de despacho míope. Nos últimos casos trata-se de métodos iterativos que se baseiam na melhoria progressiva de estimativas de tempos de processamento restante e/ou de valores heurísticos atribuídos às máquinas (ditos custos de recurso) que indicam o recurso mais congestionado a cada passo.

Duas abordagens heurísticas a problemas de escalonamento com a característica especial de os recursos terem, ou capacidade variável (podendo empregar maior ou menor capacidade numa tarefa), ou capacidade múltipla (podendo executar mais de uma tarefa simultaneamente), são as descritas em [Boctor 1996] e [Nuijten 1996].

Em [Boctor 1996] descreve-se uma abordagem heurística ao problema de escalonamento de projectos com restrições de capacidade nos recursos, em que a duração das tarefas depende da quantidade de capacidade de recurso utilizada na tarefa. O método proposto tem a particularidade de associar a cada tarefa durações alternativas, cada uma correspondendo a um requerimento de capacidade do recurso requerido diferente; considera então algumas combinações possíveis das tarefas por escalonar, juntamente com os respectivos pares duração-capacidade requerida, escolhendo a melhor combinação de acordo com um critério (o objectivo é, no caso, minimizar a duração de um projecto).

Em [Nuijten 1996] introduz-se o problema de escalonamento em *job shop* com capacidade múltipla (uma generalização do problema de escalonamento em *job shop* em que as máquinas podem processar mais de uma tarefa de cada vez). O problema é tratado como um Problema de Satisfação de Restrições (PSR, ver o Apêndice C) usando-se heurísticas para escolher, em cada passo, a tarefa a escalonar e o tempo de início para a tarefa.

2.2.1.2.2 Escalonamento Baseado em Congestionamentos

Uma filosofia possível de escalonamento é ver o escalonamento segundo a perspectiva dos processos, em que há um conjunto de processos com prioridades previamente atribuídas a escalonar nos recursos, escalonando-se processo a processo, do processo mais prioritário até ao menos prioritário. Esta filosofia centra a atenção nos processos e é tipicamente orientada para a redução do comprimento de escalonamento dos processos mais prioritários (processos menos prioritários poderão ter de esperar por recursos que estão ocupados por processos mais prioritários). Certa classe de abordagens ao problema do escalonamento de processamento intermitente é dita baseada em congestionamentos por dar prioridade, durante a construção de um escalonamento, ao escalonamento de tarefas, não do processo mais prioritário, mas no recurso mais prioritário, *i.e.*, o mais congestionado. Esta é uma filosofia que vê o escalonamento sob a perspectiva dos recursos e da sua utilização e pode resumir-se no escalonamento recurso a recurso, do recurso mais prioritário até ao menos prioritário.[29]

[29] Estas duas filosofias podem combinar-se de modo a obter uma *perspectiva baseada na tarefa*, usando de alguma forma as prioridades de recursos e as prioridades de processos para estabelecer prioridades de pares tarefa-recurso (prioridade ao par tarefa-recurso mais crítico) [Morton 1993]. Como adiante se verá, uma

O sistema de escalonamento da produção OPT (*Optimized Production Technology*, Tecnologia de Produção Optimizada) [Jacobs 1984], [Vollmann 1997] é um destes sistemas baseado em congestionamentos. O OPT é um programa de computador que, numa primeira passagem, realiza um escalonamento para trás, a partir das datas limite dos processos, assumindo capacidade infinita nos recursos, para determinar qual o recurso mais congestionado. Depois de identificado o recurso mais congestionado e de modo a optimizar a sua utilização, escalona para a frente as tarefas nele tendo em conta a capacidade finita e a seguir escalona as tarefas nos outros recursos (até ao recurso congestionado e depois deste) tentando minimizar as existências de trabalhos em curso. O fluxo de materiais é facilitado através da utilização de dois tipos de lotes: o *lote de tarefa* e o *lote de transferência* [Jacobs 1984], [Jacobs 1989] que diferem, em geral, dos *lotes de processo*,[30] *i.e.*, das quantidades que vão sendo introduzidas no sistema e que advêm de encomendas do exterior. O lote de tarefa é a quantidade de peças produzidas numa tarefa num recurso de cada vez (se houver tempos de preparação envolvidos "de cada vez" significa com um só tempo de preparação [31]); o lote de transferência é a quantidade de peças movida entre duas tarefas seguidas em dois recursos e é sempre menor ou igual ao lote de tarefa. Uma tarefa pode iniciar-se mesmo sem a tarefa antecessora, no mesmo processo, terminar se for usada a subdivisão de lotes (*lot-splitting*) de um lote de tarefa em vários lotes de transferência de uma tarefa para a outra. Isto leva uma redução do número de trabalhos em curso e do tempo de fluxo de um processo. Havendo tempos de preparação envolvidos, a utilização de lotes de tarefa maiores (agregando, inclusive, lotes de processos diferentes) origina menores perdas de tempo em tempos de preparação o que é preferido para o caso de tarefa em recursos congestionados; os recursos não congestionados têm tempos de inactividade e podem utilizar-se lotes de tarefa menores.

Embora o OPT seja criticado por não poder acomodar vários congestionamentos simultaneamente e congestionamentos que mudam de posição, por ser necessário determinar uma solução recomeçando a partir do início sempre que há uma correcção a fazer ao escalonamento, por ter uma fraca interactividade com o utilizador e por o algoritmo ser propriedade privada, este sistema gera soluções que se aproximam às dos modelos de programação matemática e foi um dos primeiros sucessos comerciais de escalonamento. Para além de ser um sistema de escalonamento que considera a capacidade dos recursos finita, teve o mérito de chamar a atenção da indústria para conceitos chave num sistema de produção como congestionamentos, existências de trabalhos em curso e tempo total de produção (*lead time*) variável.

A abordagem de *congestionamento errante* (*shifting bottleneck*) [Adams 1988] ao problema de escalonamento de processamento intermitente pode considerar-se uma sofisticação do método do OPT, mais intensivo do ponto de vista computacional, em que se considera poder existir mais de um recurso congestionado. Os recursos são escalonados por um processo iterativo, pela ordem do mais congestionado para o menos congestionado. Em cada iteração, num primeiro passo, mantém-se o escalonamento dos recursos já escalonados fixo, determina-se qual o recurso mais congestionado dos recursos que ainda não foram escalonados e é estabelecida a sequência óptima das tarefas nesse recurso. Para decidir qual é o recurso

---

filosofia de escalonamento baseada nos recursos, ou baseada nos processos, enquadra-se dentro das perspectivas de escalonamento apelidadas de macro-oportunistas e o uso de uma filosofia baseada em tarefa dentro das perspectivas micro-oportunistas.

[30] Designado por lote de lançamento (*release batch*) em [Jacobs 1989].

[31] O tempo de preparação é o tempo que uma peça gasta à espera que um recurso seja preparado para depois a processar.

mais congestionado considera-se o problema simplificado, resolvendo-o como se fosse um problema de um só recurso, para cada um dos recursos a escalonar; o recurso mais congestionado é aquele cujo problema simplificado tem o maior valor da função objectivo[32] e a sequência de tarefas óptima é a determinada na resolução desse problema. Num segundo passo da iteração, é re-optimizada a sequência de cada recurso já anteriormente escalonado, o que tenta resolver possíveis conflitos que apareçam entre recursos previamente escalonados e o recurso recentemente escalonado. A essência deste método assemelha-se à ideia clássica da programação não linear: concentrar-se sobre uma variável e optimizar segundo essa variável e fazer isto repetidamente para todas as variáveis do problema [Morton 1993]. Para um problema discreto em que há apenas um número finito de escolhas para cada variável este procedimento garante a convergência para um óptimo local, pelo menos, com um número finito de iterações. Se for possível, de algum modo, optimizar segundo cada variável, por ordem de importância decrescente das variáveis, é possível que o óptimo local encontrado seja o óptimo global, ou próximo dele. Os resultados desta abordagem são bastante bons para problemas com até 10 recursos e 500 tarefas. Um factor de peso no custo computacional é, obviamente, o do número de recursos. O procedimento do congestionamento errante foi também testado e comparado com um conjunto de dez heurísticas padrão e produziu os melhores resultados em 95% dos problemas de teste, com o mesmo custo computacional. Mais recentemente, em [Ivens 1996], foi proposta uma extensão do congestionamento errante que tem em conta atrasos e tempos de transporte, bem como tempos de preparação dependentes da sequência de tarefas em cada recurso, acomodando também datas limite, processos com estruturas convergentes e divergentes, máquinas paralelas e sobreposição de tarefas (através da subdivisão do lote de tarefa em lotes de transferência).

A abordagem de *dinâmica dos congestionamentos* (*bottleneck dynamics*) [Morton 1993], baseia-se em estimativas de custos de atraso de cada tarefa e em custos de atraso para cada recurso no sistema. Estes custos podem servir para calcular, por exemplo, que tarefa origina o maior quociente benefício/custo ao ser escalonada a seguir num recurso (decisão de sequenciação), ou que recurso deve ser escolhido para a execução de uma tarefa de modo a minimizar a soma dos custos de atraso de recurso e de tarefa (decisão de encaminhamento, por um dos recursos alternativos, ou *routing*). Outras decisões para que esta abordagem pode servir são a determinação de datas de lançamento de processos, o dimensionamento de lotes (*lot-sizing*) e a agregação de lotes (*batching*). A estimativa do custo de atraso de uma tarefa por um período de tempo é dada pelo aumento do atraso que ela provocará na data limite do processo respectivo (e, portanto, no aumento da não satisfação do cliente).[33] A estimativa do custo de atraso de um recurso por um período de tempo é dada pelo total dos custos de atraso relativos às tarefas no período corrente de ocupação do recurso (*i.e.*, não só as tarefas na fila de espera do recurso mas também aquelas que chegarão antes desta fila ficar vazia).[34]

---

[32] Na versão apresentada em [Adams 1988] do congestionamento errante a função objectivo é o comprimento do escalonamento.

[33] Há aqui, obviamente, dificuldade de estimar o tempo total de produção, pois ele depende de decisões de sequenciação ainda não tomadas. Se o custo marginal do atraso da entrega ao cliente é constante, este tempo não é relevante, pois o custo de atraso da tarefa é constante. Para o caso mais geral em que o cliente não se importa com uma entrega mais cedo que a data limite mas não tolera atrasos, o tempo total de produção pode ser estimado através de valores de um historial do tempo de processamento restante, ou a decisão de uma pessoa com base em experiência anterior no mesmo sistema produtivo, ou iterando várias simulações de modo a obter estimativas progressivamente melhores, como foi antes referido na nota 28.

[34] São necessárias, portanto, as estimativas dos custos de atraso das tarefas, das tarefas do período corrente de ocupação do recurso e do período corrente de ocupação do recurso.

Num primeiro estudo publicado de uma aplicação da dinâmica dos congestionamentos, o sistema SCHED-STAR [Morton 1988] (ver também [Morton 1993]), os princípios referidos são utilizados para, em vez de tentar optimizar o comprimento do escalonamento ou o atraso máximo ou o atraso médio, tratar directamente com custos explícitos e receitas, e custos implícitos como custos de atraso, através do uso de uma função objectivo de valor actualizado líquido.

2.2.1.2.3 Meta-Heurísticas

Na década de 80 a investigação orientou-se para métodos heurísticos que usam heurísticas tão gerais que podem ser usadas numa larga quantidade de problemas, podendo mesmo ser usadas em combinação com outros métodos (daí serem muitas vezes referidos na literatura por *meta-heurísticas*), ver [Reeves 1993a]. Uma característica comum destes métodos é a de convergirem para uma solução que é um óptimo global no espaço do problema, para a maior parte dos casos, mas em que isso não é garantido para algum caso em particular. Estes métodos heurísticos, que são aplicáveis a problemas de optimização combinatória e portanto, também ao escalonamento, descrevem-se resumidamente a seguir.

Arrefecimento Simulado (*Simulated Annealing*)

O método de *arrefecimento simulado* (*simulated annealing*) baseia-se na noção de vizinhança de uma solução, no espaço de soluções possíveis do problema e pode ser visto como uma variante do método heurístico de *procura local*, em que um subconjunto das soluções possíveis é explorado indo repetidamente de uma solução para uma solução vizinha. Trata-se de um tipo de métodos que é frequentemente apelidado de *melhoria iterativa* por repetidamente procurar passar de uma solução a uma outra que seja melhor. Em geral, uma solução razoavelmente boa é encontrada ao fim de algumas iterações. Um caso simples destes métodos é o algoritmo designado por *subida do monte* (*hill-climbing*), na versão de maximização, ou *descida do monte* (*hill-descending*), na versão de minimização, de que o arrefecimento simulado pode ser considerado uma versão aperfeiçoada. O método de arrefecimento simulado [Kirkpatrick 1983], [Van Laarhoven 1988], [Aarts 1989] faz uso de uma analogia da Termodinâmica Estatística, no que respeita ao fenómeno da mudança do estado de energia de um sólido quando é submetido a um processo de arrefecimento até convergir para um estado final; este estado final depende do modo como é feito o arrefecimento. Em 1983 Kirkpatrick [Kirkpatrick 1983] propõe usar um algoritmo, proposto em 1953 por Metropolis e outros investigadores para simular aquele fenómeno, para procura de soluções de um problema de optimização.

No algoritmo de Metropolis, cada iteração dá-se para um valor fixo da temperatura do material sólido, começando-se a uma dada temperatura. É gerada uma perturbação no estado do material e calcula-se a alteração de energia correspondente. Se a energia diminuiu o sistema muda para este novo estado, caso contrário o novo estado é aceite mediante o valor de uma dada função de probabilidade. Isto é repetido um número determinado de vezes dentro da mesma iteração (para a mesma temperatura), após o que a temperatura é diminuída, repetindo-se de seguida um novo ciclo até que o material "congele" num estado de final de equilíbrio, de energia mínima. A função de probabilidade é tal que a probabilidade diminui com o aumento da energia e diminui à medida que a temperatura diminui, reflectindo a tendência natural de o material preferir estados de energia mínima e de essa preferência ser cada vez mais forte à medida que a temperatura diminui.

Na analogia com o processo físico, um estado do sólido corresponde a uma solução do problema no problema de optimização; o nível de energia do sólido corresponde ao valor da função objectivo para uma solução e a temperatura a um parâmetro de controlo que varia (diminui) em cada iteração, servindo basicamente de contador de iterações. O estado final corresponde à solução encontrada. O método começa com uma solução sub-óptima do problema com um certo valor do parâmetro de controlo (a temperatura). Em cada iteração este parâmetro mantém-se constante. É escolhida uma solução vizinha da solução actual (aleatoriamente entre as soluções vizinhas) que será aceite como solução actual se o seu custo for menor. Caso contrário (custo maior) a solução escolhida pode mesmo assim ser aceite se um valor aleatoriamente gerado for menor que o valor da função de probabilidade (o uso de aleatoriedade leva a que o arrefecimento simulado seja frequentemente referido como um método de Monte Carlo). Isto é repetido um número determinado de vezes dentro da mesma iteração, após o que o parâmetro de controlo é decrementado, repetindo-se de seguida um novo ciclo até que o parâmetro de controlo seja zero.

Um dos problemas com os algoritmos de melhoria iterativa é o de se poderem encaminhar para soluções que correspondem a pontos mínimos locais, e não globais, do espaço de valores da função de custo. Correm assim o risco de terminarem com soluções que não são óptimas; este é o problema dos algoritmos do tipo subida ou descida do monte. O arrefecimento simulado tenta evitar este problema, ao permitir aceitar encaminhar-se para soluções piores do que a actual, mas mais em iterações iniciais, onde o valor da função de probabilidade é mais alto. À medida que o parâmetro da temperatura diminui a liberdade para mover-se para uma solução pior diminui, de modo que o comportamento esperado do algoritmo será o de dar inicialmente alguns "saltos" entre pontos afastados no espaço de soluções possíveis, em princípio obtendo uma solução próxima do óptimo, concentrando-se de seguida na exploração da vizinhança dessa solução até obter a solução óptima. Para isto, no entanto, é necessário afinar certos parâmetros, nomeadamente o perfil de evolução da temperatura (valores inicial e final, número de iterações para cada temperatura e decréscimo da temperatura de iteração para iteração). Para cada tipo particular de problema haverá que definir previamente a função de custo e a vizinhança (ver [Dowsland 1993]).

Para além das vantagens já descritas para os métodos heurísticos, os métodos do tipo subida ou descida do monte, como é o arrefecimento simulado, têm as vantagens de serem computacionalmente económicos em termos de espaço de memória necessário, devido a manterem em memória apenas o estado actual (não "olham" mais à frente do que os estados sucessores dele e não mantêm estados antecessores); este aspecto é mais vantajoso em espaços de soluções densos, em particular. Uma grande desvantagem é a de serem demorados (consomem muito tempo) e de haver necessidade de afinação dos parâmetros. Pouca aplicação terão em situações em que o caminho pelo qual a solução é atingido é relevante, já que não mantêm informação de estados antecessores.

Grande parte das aplicações deste método referidas no campo da IO envolvem problemas de escalonamento e geração de horários [Dowsland 1993]. Muitas das aplicações ao escalonamento têm a ver com a determinação da sequência óptima para um certo número de processos num dado conjunto de máquinas de modo a minimzar o comprimento do escalonamento. O espaço de soluções é constituído por todas as sequências possíveis e podem ser definidas uma série de tipos de vizinhança, como as que se obtêm por mudança da posição de um processo, ou troca das posições de dois processos. Na geração de horários a vizinhança de uma solução é obtida por troca da afectação de dois eventos (ou aulas, ou sessões) em tempos diferentes, ou cancelando um e substituindo-o. A função de custo mede a não aceitabilidade de uma solução podendo envolver uma soma ponderada de penalidades

associadas, por exemplo, várias formas de contabilização de violações de restrições, ou outros factores não aceitáveis em maior ou menor grau num horário.

[Dowsland 1993] descreve resumidamente aplicações com sucesso do arrefecimento simulado à sequenciação no problema de processamento contínuo ([Osman 1989], [Ogbu 1990], [Ogbu 1991]), problemas de produção de peças envolvendo vários estágios com minimzação de custos de preparação e de existências ([Kuik 1990]), escalonamento de sistemas de produção flexível, em que é tido em conta, para além dos processos e das máquinas, também o sistema de transporte entre as máquinas ([Brandimarte 1987]), escalonamento dos transportes ([Wright 1989]) e geração de horários ([Eglese 1987], [Dowsland 1990], [Abramson 1991]).

Procura Tabu

O método de *procura tabu* (*tabu search*) à semelhança do arrefecimento simulado faz também uso da noção de vizinhança e de uma analogia com um fenómeno natural que é, no caso, o uso de memória.

O princípio básico do procura tabu [Glover 1986], [Hansen 1986], [Glover 1993] é o de uma procura heurística do tipo subida/descida do monte, guiada para explorar o espaço de soluções de um modo a não se encaminhar para soluções localmente óptimas que já tenha previamente encontrado. Para isto é usada uma estrutura de memória designada por *lista tabu*, que contém em cada momento um número de soluções proibidas determinadas por uma história relativa a um certo número de iterações anteriores. Alternativamente, a lista tabu pode memorizar, não as próprias soluções proibidas, mas os movimentos no espaço de soluções que levam a elas. Esta forma alternativa pode economizar memória para armazenamento da lista tabu e economizar tempo de procura, mas poderá revelar-se demasiadamente restritiva proibindo soluções para as quais não havia essa intenção.

No método de procura tabu, em cada iteração passa-se da solução actual para a solução na vizinhança que não esteja na lista tabu e que mais melhore o valor da função objectivo, ou a que menos o piora, se não houver nenhuma melhor que a actual. Para além disto, existe um *critério de aspiração* que determina se um movimento é admissível apesar de figurar na lista tabu. Podem existir, no contexto deste método, estratégias de *intensificação*, para concentrar a procura em regiões do espaço de soluções mais promissoras, ou de *diversificação*, para conduzi-la para regiões não exploradas.

Outras características são o uso de *listas de candidatos*, usadas para reduzir o número de movimentos investigados em cada iteração, quando a determinação da vizinhança, ou a avaliação das soluções têm um custo computacional elevado; o uso de *soluções de elite* (que são soluções boas, óptimos locais ou próximas do óptimo global) para guiar estratégias de intensificação; o emprego da *frequência* de aparecimento de certos elementos na solução para guiar estratégias de diversificação [Ribeiro 1996].

Este método requer o uso intensivo da memória e uma realização computacional muito cuidada (há um número excessivo de parâmetros a estabelecer) mas oferece as vantagens de poder produzir boas soluções em tempos relativamente pequenos, se for bem aplicado [Ribeiro 1996].

Em [Glover 1993] a procura tabu é descrita em pormenor e descrevem-se resumidamente várias aplicações com sucesso do procura tabu a variados problemas, inclusive problemas de escalonamento. Em [Barnes 1995] faz-se um resumo da investigação sobre a procura tabu aplicada ao escalonamento da produção.

Algoritmos Genéticos

Os métodos baseados em *algoritmos genéticos* fazem uso de uma analogia com o fenómeno natural da evolução dos organismos vivos. A ideia base dos algoritmos genéticos [Holland 1975], [Goldberg 1989], [Reeves 1993b] é a de simular o desenvolvimento de populações de indivíduos de uma espécie que adquire características cada vez melhores, ou mais desejáveis, face a uma selecção natural. Estas características são determinadas ao nível genético pelo modo como são combinados os cromossomas dos progenitores dos indivíduos.

O algoritmo começa com uma população inicial de um certo número de cromossomas de progenitores potenciais, cujos valores de qualidade foram determinados. O valor de qualidade corresponde ao valor de uma *função de avaliação* usada para medir o grau de aptidão de um cromossoma. Há depois uma *selecção* dos cromossomas de maior qualidade para darem origem a uma nova geração e aplicam-se-lhes *operadores genéticos* para obtenção dos cromossomas da geração seguinte. Estes operadores podem ser o *cruzamento*, ou a *mutação*. O primeiro consiste na troca de genes de dois cromossomas progenitores; o segundo na alteração de genes de um cromossoma. O processo é repetido até uma certa *condição de terminação* se verificar. Esta condição pode ser por exemplo, ter decorrido um certo número de iterações/gerações, ou o melhor valor de qualidade na população ser constante ao longo de um certo número de gerações, ou a população ser dominada por um certo número de indivíduos. Este método pode ser visto como uma procura aleatória no espaço de soluções possíveis, conduzida basicamente pelos operadores genéticos que são aplicados e pela forma de selecção.

Os cromossomas são representados por uma sequência de dígitos binários, ou números. Factores a considerar num algoritmo genético são o tamanho da população, a população inicial, os mecanismos de selecção, a função de avaliação e os operadores genéticos.

Ao aplicar um algoritmo genético à resolução de um problema de optimização, cada cromossoma representa uma solução possível e os genes representam os elementos ou variáveis da solução. O valor de qualidade corresponde ao valor da função objectivo.

Em [Reeves 1993b] relatam-se aplicações dos algoritmos genéticos à sequenciação no problema de escalonamento de processamento contínuo, ao escalonamento estocástico de processamento contínuo e ao problema de escalonamento de processamento intermitente. No primeiro caso fizeram-se comparações do desempenho do algoritmo genético com a de uma heurística tipo arrefecimento simulado, altamente eficiente, concluindo-se por uma equiparação dos dois métodos em geral, mas um desempenho crescente do algoritmo genético à medida que a dimensão do problema aumentava. Também outro estudo relatado conclui por um desempenho equiparável à do método de procura tabu. Em [Mulkens 1994] descrevem-se resultados experimentais da aplicação de algoritmos genéticos à resolução de problemas de escalonamento de processamento contínuo com duas, três e mais máquinas e incluem-se comparações com heurísticas específicas. Em [Filipic 1993] é descrita uma aplicação de algoritmos genéticos ao escalonamento de uma fábrica de têxteis com o objectivo de optimizar o consumo de energia das máquinas. Em [Muller 1994] é proposta uma arquitectura em que vários algoritmos genéticos cooperam para resolver um problema de escalonamento, propondo e aperfeiçoando soluções completas, que são partilhadas por todos, e que são obtidas usando funções de avaliação diferentes e abordando o problema de perspectivas diferentes.

Os métodos heurísticos descritos nesta secção apresentam realmente vantagens, nomeadamente a de serem tão gerais que são aplicáveis numa larga quantidade de problemas, poderem ser usados em combinação com outros métodos, tornarem possíveis modelos muito

mais aproximados do problema real (por serem flexíveis e capazes de tratar funções objectivo e/ou restrições mais complexas) e serem de fácil realização computacional. Como foi dito é uma característica comum a de tenderem a convergir para uma solução que é um óptimo global no espaço do problema, embora isso não seja garantido para algum caso em particular, arriscando-se a encontrarem soluções sub-óptimas. Esta propriedade de convergência parece ser importante na medida em que permite, pelo menos, a melhoria, ou reparação, iterativa de uma solução sub-óptima. Além disso estes algoritmos dispõem de, e podem ser terminados com, uma solução ao fim de uma qualquer iteração (ainda que seja uma solução sub-óptima) e que seria melhorada na próxima iteração, uma propriedade que os torna apelidáveis de algoritmos *a-qualquer-tempo* (*anytime*), na literatura moderna. O facto de trabalharem sobre uma solução completa parece ter a vantagem de permitir uma avaliação (através da função objectivo) mais precisa da solução actual. Este problema de avaliação precisa põe-se quando o método usado funciona por *construção*, ao invés de por *reparação*, isto é, solução é construída por junção progressiva dos seus componentes (o problema põe-se no método de ramificar-e-limitar, na programação dinâmica, ou noutro método qualquer que em cada passo dispõe de uma solução parcial, ou de um conjunto de soluções parciais que é necessário avaliar para conduzir a procura nos passos subsequentes).

Um aspecto que, no entanto, penaliza fortemente estes algoritmos é o de não serem eficientes. Mesmo se se abandonar o objectivo de encontrar uma solução óptima e nos contentarmos com uma solução satisfatória para um problema, o que pode ser feito de um modo muito fácil (terminando o algoritmo quando a solução já ultrapassou um nível de qualidade considerado aceitável), a obtenção de uma solução satisfatória pode ser demorada. Este métodos não são, em geral, apropriados para *escalonamento reactivo*, isto é, escalonamento dinâmico em tempo real, em que a própria tomada de decisão de escalonamento é feita na presença de restrições temporais. Num contexto de escalonamento reactivo é necessário haver a capacidade de refazer um escalonamento previamente elaborado quando há que acomodar eventos inesperados (como falhas nos recursos, variações nas durações das tarefas, chegada de processos novos para execução, cancelamento de processos), com o mínimo de disrupções e prejuízo na qualidade no escalonamento. Um outro aspecto é o de estes métodos por si só, não proporem um modelo do problema que possa ser suficientemente próximo do problema real de modo a tirar partido da sua estrutura particular. Não preconizam um quadro geral onde se enquadrem os aspectos de modelação e representação explícita de conhecimento e de aspectos do domínio como os recursos, os processos e as tarefas, escalonamentos, ambientes de produção, restrições de vários tipos, resolução descentralizada do problema. Dadas as exigências colocadas ao escalonamento actual na produção, nos transportes, nos serviços, faz sentido uma modelação mais aproximada ao mundo real.

2.2.1.3 Escalonamento e Redes de Produção e Distribuição

As abordagens ao escalonamento descritas até aqui referem-se ao problema de escalonamento convencional. Neste problema assume-se normalmente um contexto que é o da instalação centralizada, onde estão concentrados todos os recursos físicos e em que o planeamento das actividades é feito de forma centralizada. Os recursos são processadores elementares ou grupos de processadores elementares (máquinas ou grupos de máquinas que podem realizar tipos de tarefas similares).

O contexto da produção e distribuição é tipicamente descentralizado e o problema de escalonamento põe-se a outra escala. Numa rede de produção e distribuição os recursos considerados são eles próprios instalações — *i.e.*, são *macro-recursos* como fábricas,

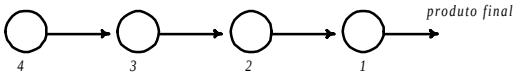
armazéns, frotas de transporte — que estão distribuídos por um conjunto de localizações geograficamente diversas. Cada nó de uma rede de produção e distribuição é um recurso com restrições de capacidade de natureza diferente, possivelmente capaz de realizar tarefas que permitem entregar produtos diversos e, possivelmente, gerido de forma independente. No entanto, estes nós estão interligados devido ao facto de as necessidades de consumo de produtos de uns serem satisfeitas por outros e levarem a cabo fases (de produção, de armazenamento, de transporte) complementares de um processo de produção e distribuição. Este tipo de sistemas com esta dependência são apelidados de sistemas multi-estágio, multi-nível ou ainda multi-escalão.

Ao nível operacional de planeamento o problema fundamental é aqui um problema de coordenação das tarefas dos nós da rede. Esta coordenação envolve várias dimensões como produto, quantidade, tempo e local. A actividade de escalonamento, no contexto da produção e distribuição, pode passar por ter de tomar em conta não só tempo e capacidade de processamento de recursos, mas também produto (já que existem produtos diferentes envolvidos), local (o nó da rede), quantidade (por exemplo, a dimensão do lote produzido ou armazenado), capacidade de armazenamento e controlo de quantidades armazenadas. Muitos trabalhos no contexto do planeamento coordenado da produção e distribuição podem ser vistos numa perspectiva alargada de escalonamento, em que se toma em conta simultaneamente produtos, quantidades, tempos e locais.

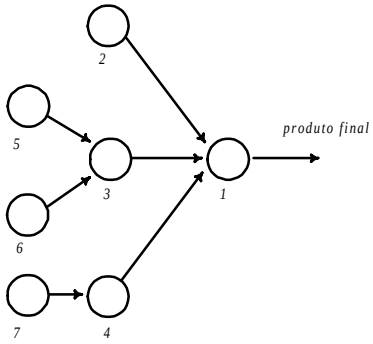
Trabalhos pioneiros no âmbito da logística de Forrester [Forrester 1958], [Forrester 1961] apontaram a necessidade de coordenação nos sistemas de produção e distribuição. Nestes trabalhos descreve-se o uso de modelos dinâmicos para, através de simulação em computador, estudar as oscilações dos fluxos de produtos originadas por variações na procura final em redes de produção e distribuição. Estes modelos incluem fluxos de produtos e fluxos de informação entre estágios e são apelidados de modelos de *dinâmica industrial*. Investigação baseada neste tipo de modelos de simulação continua a existir actualmente, veja-se por exemplo, [Towill 1982], [Wikner 1991] e [Towill 1992].

Nas décadas de 60 e 70 muitos trabalhos investiram consideravelmente em modelos matemáticos. Vejam-se, por exemplo, [Clark 1960], [Veinott 1965], [Veinott 1966], [Zangwill 1966], [Zangwill 1969], [Schwarz 1975], [Szendrovits 1975], [Folie 1976], [Schwarz 1978], [Szendrovits 1978], só para citar alguns. Tipicamente estes modelos conduzem a métodos de optimização em que se pretende minimizar um custo total, composto pela soma de um custo de preparação ou um custo de encomenda (fixo e a afectar a cada estágio sempre que um lote é produzido ou encomendado) e de um custo de posse (proporcional ao tempo de posse e à quantidade de produto possuída) em cada estágio. Frequentemente, os sistemas multi-estágio são apelidados de sistemas multi-escalão pelo facto de se fazer uso do conceito de custo de *stock de escalão* para os custos de posse. O *stock de escalão* de um estágio é definido pelo número de unidades de produto presentes no, ou que já transitaram pelo, estágio mas ainda não saíram do sistema. O uso dos custos de *stock de escalão* na função de custo permite simplificações matemáticas convenientes [Clark 1960], [Schwarz 1975].

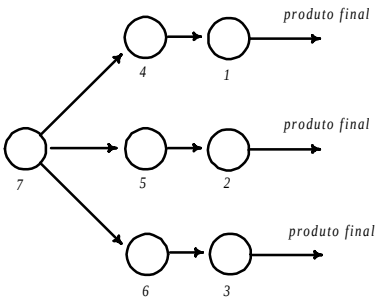
Vários tipos de estruturas de rede são possíveis como sistemas multi-estágio: estruturas em série (sequenciais ou cadeias propriamente ditas), estruturas convergentes (também apelidadas estruturas de montagem ou de produção), estruturas divergentes (também apelidadas de estruturas de distribuição) e combinações destas, incluindo estruturas convergente-divergente (também apelidadas de produção e distribuição). Ver exemplos de representação gráfica destes tipos de estruturas na Figura 2-10.



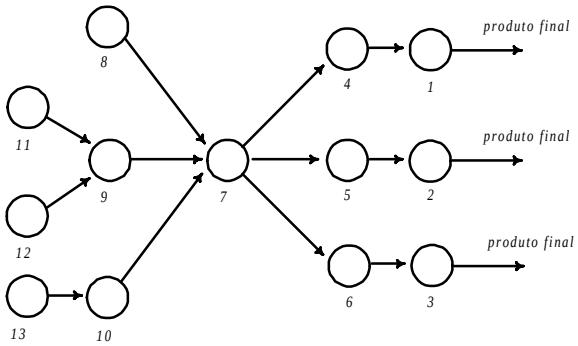
a) Estrutura multi-estágio em série.



b) Estrutura multi-estágio convergente.



c) Estrutura multi-estágio divergente.



d) Estrutura multi-estágio convergente-divergente.

Figura 2-10- Vários tipos de estruturas de rede multi-estágio.

Uma multiplicidade de estudos que se podem relacionar directa ou indirectamente com estruturas de rede de produção e distribuição foram desenvolvidos. Estes trabalhos focam aspectos vários, incluindo o controlo dos quantidades armazenadas (*stocks*), o dimensionamento de lotes, a determinação de tempos de ciclo e intervalos de reabastecimento em cada estágio do sistema multi-estágio. Vejam-se, por exemplo, [Williams 1981], [Blackburn 1982], [Graves 1982], [Tatsiopoulos 1983], [Williams 1983], [Afentakis 1984], [Bahl 1984], [Federgruen 1984], [Karmarkar 1985], [Maxwell 1985], [Billington 1986], [Moily 1986], [Bahl 1987], [Dobson 1987], [Karmarkar 1987a], [Karmarkar 1987b], [Muckstadt 1987], [Axsäter 1988], [Federgruen 1989], [Jacobs 1989], [Jackson 1990], [Roundy 1990], [Spearman 1990], [Badinelli 1992], [Spearman 1992], [Axsäter 1993], [Federgruen 1993a], [Federgruen 1993b], [Muckstadt 1993], [Bourland 1994], [Kuik 1994], [Tempelmeier 1994], [Diks 1996] ou [Hill 1996].[35] Em [Graves 1993] apresenta-se uma revisão extensa e abrangente do estado da arte na investigação em sistemas logísticos até 1993, incluindo vários capítulos dedicados a sistemas multi-estágio. Uma revisão do estado da arte mais recente, mas mais limitada, pode encontrar-se em [Thomas 1996].

Grande parte destes estudos, desenvolvem-se, de alguma forma, a partir da ideia da Quantidade Económica de Encomenda. Oferecem modelos matemáticos do problema do

[35] Alguns destes trabalhos são revisões do estado da arte, alguns outros tratam problemas de planeamento coordenado que se põem também nas redes de produção e distribuição.

dimensionamento de lote (económico) em versão multi-escalão e vêm o problema como um problema de optimização. O valor da função objectivo, que é usualmente a soma dos custos de posse e de preparação, ou de encomenda, de todos os nós da rede, deve ser minimizado para determinar a dimensão dos lotes (económicos) a serem transferidos entre cada par de nós da rede.

Muitos destes modelos são mais úteis numa abordagem partindo de um ponto de vista estratégico ou táctico, mas não tão úteis para um horizonte temporal de curta duração como é o do escalonamento das tarefas. Entre as suas limitações típicas incluem-se, frequentemente:

a) A produção é considerada instantânea e os tempos de ciclo são zero;

b) A procura do produto final é constante no tempo;

c) É suposto a rede poder seguir políticas predeterminadas como um todo. Por exemplo políticas estacionárias (cada nó entrega um lote de dimensão fixa) e políticas imbricadas (a dimensão do lote entregue por cada nó é um múltiplo inteiro da dimensão do lote entregue pelo seu nó sucessor na rede);

d) A rede está dedicada apenas a um produto final;

e) Eventos inesperados não são acomodados. Por exemplo, não se especifica o que deve acontecer se a procura se altera, ou os nós falham nas entregas previstas (o problema terá, em princípio, de ser reformulado e a solução recalculada a partir do início);

f) A rede é considerada um sistema centralizado, havendo apenas uma única função objectivo. É assumido que toda a informação (por exemplo, sobre os custos) é sempre conhecida;

g) É procurada uma solução óptima global. Uma solução óptima não pode, em geral, ser determinada *a tempo*, usando métodos matemáticos. Para além disso, num sistema com uma gestão não centralizada, haverá, por certo, perspectivas do problema e objectivos diversos e uma solução óptima global pode não corresponder a uma solução óptima local (*i.e.*, na perspectiva de alguns nós).

Em algumas abordagens não é empregue o conceito de *stock de escalão*. O uso deste conceito está normalmente associado a uma gestão centralizada dos sistemas multi-estágio, já que é necessário conhecer informação global do sistema (os níveis de armazenamento em cada escalão). Formas ditas descentralizadas de gestão envolvem, pelo contrário, a informação local a cada estágio, *i.e.*, os níveis de *stock de instalação* [Lee 1993a], [Axsäter 1993].

A maior parte destes temas, bem como o tema do escalonamento convencional, enquadram-se em áreas como a de gestão da produção e das operações, gestão de armazenamento e planeamento e controlo da produção. Ver, por exemplo, [Tersine 1988], [Schroeder 1993], [Anderson 1994], [Elsayed 1994], [Chase 1995], [Roldão 1995], [Browne 1996], [Courtois 1996], [Buffa 1997] ou [Vollmann 1997].

Gestão da cadeia de fornecimento (*supply chain management*, SCM) designa a gestão dos fluxos físicos (de produtos/materiais) em redes, ou cadeias,[36] abrangendo fornecedores, fábricas, centros de distribuição, retalhistas, para entrega de produtos finais no mercado, suportada por fluxos de informação [Christopher 1993], [Gattorna 1996], [Martin 1996], [Parnell 1996], [Thomas 1996], [Vollmann 1997] (no Capítulo 18, em Distribution Requirements Planning), [Camarinha-Matos 1999a]. O conceito de cadeia de fornecimento

[36] O termo "cadeia", frequentemente utilizado é, na realidade, uma simplificação já que, de um modo geral se trata de uma rede de empresas inter-actuantes [Ellram 1991].

complementa o de rede de produção e distribuição (ou outro sistema multi-estágio descentralizado qualquer) juntando aos fluxos físicos de produtos os fluxos de informação. Este conceito permite abranger sistemas de rede com dependências do tipo cliente-fornecedor, cujos nós são geridos por entidades autónomas, com capacidade de decisão independente (são, por exemplo, empresas diferentes) e com a possibilidade de coordenar decisões e cooperar, por meio de comunicação entre si, para melhor controlo dos fluxos físicos de produtos. A coordenação e a cooperação são aplicáveis não apenas à optimização dos fluxos físicos (diminuição do tempo total para entregar o produto final ao consumidor, redução de quantidades armazenadas de modo a reduzir custos de posse, resposta rápida a variações na procura de produto final) mas também ao projecto cooperativo de novos produtos e processos de produção e distribuição. As tecnologias de informação e das comunicações actuais, bem como os padrões actuais para troca de informação estruturada por via electrónica como, por exemplo EDI e STEP (protocolos para troca electrónica de informação comercial e informação técnica sobre produtos, respectivamente) fornecem o suporte a este cenário.

Modelos de planeamento coordenado que se enquadram no cenário da cadeia de fornecimento, são descritos em [Cohen 1988] e [Lee 1993a], onde é também apresentada uma revisão do estado da arte. Em [Lee 1992] discutem-se questões problemáticas e oportunidades para desenvolvimento de estratégias apropriadas no contexto da gestão integrada da cadeia de fornecimento. Em [Scott 1991] propõem-se três passos para integração da cadeia de fornecimento, nomeadamente: i) identificar tempos de escoamento e níveis de armazenamento médios em cada estágio, ii) identificar oportunidades para actividades colaborativas e estabelecer relações de cliente-fornecedor iii) identificar melhorias a nível operacional para aumentar a competitividade. [Thomas 1996] é um artigo onde se faz uma revisão abrangente do estado da arte de modelos de gestão integrada da cadeia de fornecimento. Em [Lee 1996] mostra-se como o projecto dos produtos e dos processos de produção e distribuição dos produtos pode influenciar a eficiência do planeamento e controlo. [Cohen 1990a], [Davis 1993], [Lee 1993b] e [Lee 1995] relatam essencialmente casos de aplicação prática. Estes modelos são modelos analíticos, a maioria dos quais são mais apropriados à tomada de decisão estratégica (*i.e.*, envolvendo decisões como instalação de uma nova fábrica ou de um novo centro de distribuição, afectação de equipamento a instalações fabris, escolha de localizações para produção de um novo produto, avaliação de alterações ao fluxo de um produto na cadeia). Em geral, na literatura de investigação nesta área, muito pouco é dedicado ao escalonamento em especial, a não ser mais recentemente, como se descreverá na secção seguinte.

Estes temas enquadram-se na área da logística. Obras modernas sobre logística são, por exemplo, [Farmer 1991], [Christopher 1992], [Cooper 1993], [Carvalho 1996] ou [Gattorna 1996].

Recentemente existe um interesse crescente pela gestão integrada de cadeias de fornecimento cooperativas (participantes em número limitado com competências específicas complementares e com maior interligação), mais dinâmicas e eficientes (resposta rápida à procura, quantidades armazenadas reduzidas) e mesmo de configuração flexível (participantes podem agrupar-se numa cadeia para o projecto, produção e distribuição de um produto, desfazendo-se o grupo no fim do ciclo de vida do produto).

Modelos como o da *Empresa Estendida* ou da *Empresa Virtual* pressupõem uma aliança temporária, de um grupo de empresas, tipicamente pequenas e médias empresas especializadas, com funcionalidades complementares, para a cooperação em pé de igualdade em actividades que podem estender-se a todo o ciclo de vida de um ou mais produtos [Camarinha-Matos

1999a], [Hunt 1997]. Esta cooperação temporária pode ser mais ou menos durável e mais ou menos flexível. A ideia da Empresa Virtual é mais abrangente e inclui a rede de empresas possivelmente mais volátil (menos durável) e de configuração mais mutável (participantes podem entrar e sair durante o tempo de vida da rede); a Empresa Estendida está mais associada à rede resultante da empresa produtora dominante que estende laços cooperativos com um número mais limitado de fornecedores e clientes, e a uma ligação possivelmente mais permanente. Em ambos os conceitos se põe a tónica na *cooperação* e na *ligação em rede electrónica*, pressupondo o recurso a tecnologias de informação e das comunicações [Camarinha-Matos 1999a].

Dado que, na sua realização, estes modelos conduzem a sistemas em que os recursos, as actividades, a informação e a tomada de decisão estão distribuídos por unidades autónomas, não é raro o recurso à tecnologia dos agentes da IAD. Alguns destes casos de realizações são revistos mais adiante (ver a secção 2.2.2.7).

2.2.2 Abordagens de Inteligência Artificial e de Inteligência Artificial Distribuída

Um dos campos da IA (ver o Apêndice B) é o Planeamento cuja investigação remonta à década de 70. Planear é entendido como um processo de selecção e sequenciação de acções, para formar um plano que atinja objectivos dados, satisfazendo um conjunto de restrições. Esta área tem aplicações, por exemplo, na robótica, no projecto de experiências científicas, no planeamento de observações de telescópios automáticos, no planeamento da produção ou da montagem na indústria. O problema do escalonamento pode ser entendido como uma forma de planeamento a um nível mais detalhado,[37] em que há que afectar recursos e associar tempos às actividades de um plano, dentro do horizonte temporal associado ao plano. Em especial a partir do meio da década de 80 iniciou-se uma fase de investigação dos problemas de escalonamento na área de IA.

Em [Le Pape 1994] refere-se que, dada a complexidade e diversidade dos problemas de escalonamento é natural esperar que procedimentos de escalonamento rígidos, orientados para produzir escalonamentos óptimos em circunstâncias particulares, não possam dar, em geral, resultados satisfatórios, noutras circunstâncias. Sendo assim, uma reacção natural será a de usar técnicas de IA no escalonamento, pelas seguintes razões:

- Representação e modularidade do conhecimento - As técnicas de IA facilitam expressão e aplicação do conhecimento do perito da área juntamente com o conhecimento empírico e teórico de outras áreas (por exemplo, a Matemática, a IO). Também a distinção entre conhecimento do domínio e o conhecimento de controlo na resolução do problema, torna mais fácil a adaptação a ambientes diferentes;
- Análise do problema - As técnicas de IA são úteis na análise e determinação de qual o conjunto de conhecimentos e quais os componentes do problema que são mais relevantes numa dada situação (ex.: ordens de produção e recursos críticos, efeitos de eventos não esperados, falhas na resolução do problema);

[37] Ver, a este respeito, na secção 2.1.2, as definições de [Fox 1994], de [Muscettola 1994a] e, principalmente, de [McDermott 1995].

- Interacção com utilizador - As técnicas de IA facilitam gestão das interacções com o utilizador na fábrica, permitindo que ele compreenda as decisões do sistema e possa aceitá-las ou não.

De um modo geral as abordagens de IA ao problema do escalonamento caracterizam-se por darem ênfase ao papel da *representação do conhecimento* específico do domínio do problema. Este conhecimento é utilizado para representar entidades do domínio de escalonamento como os recursos, os processos, as tarefas, as restrições e os objectivos. Segundo [Dorn 1994a] os sistemas baseados em conhecimento oferecem as seguintes vantagens na abordagem de problemas como o escalonamento:

- Preconizam e facilitam a utilização de heurísticas para reduzir a complexidade do problema;
- Permitem raciocínio com conhecimento incompleto, incerto, vago ou probabilístico;
- Representam o conhecimento explicitamente sendo mais fácil desenvolvê-lo e mantê-lo.

Resumos do estado da arte de escalonamento usando técnicas de IA podem ser encontrados em [Atabakhsh 1991], [Zweben 1994a], [Szelke 1994], [Bäckström 1994], [Brown 1995], [Kerr 1995]. Em [Friedman 1997] resumem-se várias técnicas aplicáveis ao escalonamento em *job shop* oriundas de várias áreas (IA incluída).

As abordagens ao escalonamento baseadas em IA podem classificar-se basicamente segundo as técnicas de usadas para representação do conhecimento e segundo metodologia [Atabakhsh 1991], [Blazewicz 1994]. Das técnicas de representação de conhecimento empregues podem distinguir-se os *enquadramentos* (*frames*) e as *redes semânticas* (*semantic networks*), a *lógica*, *regras*, ou *casos*.[38] A metodologia de escalonamento utilizada baseiam-se frequentemente em *procura heurística*, *raciocínio oportunista* (quer dizer, dependendo dos aspectos mais críticos durante a resolução do problema empregar diferentes estratégias e perspectivas de resolução do problema - como por exemplo perspectivas baseadas em recursos, ou baseadas em processos), *decomposição hierárquica* (resolução de subproblemas, abstracção e resolução distribuída de problemas), *emparelhamento de padrões* (por exemplo, uso de informação sobre o estado do sistema de produção e dos objectivos para aplicar regras de prioridade), *propagação de restrições*, com técnicas de *imposição* ou *relaxação de restrições*. Muitas abordagens recorrem à Inteligência Artificial Distribuída (IAD, ou *Distributed Artificial Intelligence*, DAI; ver o Apêndice D), uma sub-área da IA.

Dada a grande variedade e combinações de técnicas de representação e metodologias empregues nas abordagens de IA ao escalonamento optou-se, na secção 2.2.2, por centrar a revisão do estado da arte em casos exemplares de sistemas de escalonamento baseados em IA referidos na literatura. Nas duas secções que se seguem (as secções 2.2.2.1 e 2.2.2.2) e, em parte na terceira (a secção 2.2.2.3), foca-se principalmente o aspecto de como o conhecimento é representado nos sistemas de escalonamento, incluindo escalonamento fazendo uso de representação de conhecimento através de redes semânticas e enquadramentos, de regras e de lógica e de casos. A partir da terceira secção focam-se aspectos de metodologia geral de escalonamento. Incluem-se nestas últimas secções o escalonamento baseado em casos (na secção 2.2.2.3), o escalonamento baseado em procura e em satisfação de restrições (na secção 2.2.2.4), o escalonamento por construção (ou construtivo; na secção 2.2.2.5) e o escalonamento por reparação (na secção 2.2.2.6). O mesmo sistema de escalonamento pode

[38] *I.e.*, o conhecimento é representado numa base de conhecimento de casos e recorre-se ao *raciocínio baseado em casos*, ou *case based reasoning*, CBR.

aparecer incluído em mais de uma secção, se tiver características que o permitam assim enquadrar. A última secção diz respeito ao escalonamento empregando técnicas e métodos de IAD

2.2.2.1 Escalonamento com Representação do Conhecimento através de Redes Semânticas e Enquadramentos

Nas *redes semânticas* o conhecimento é representado por meio de uma rede, cujos nós representam objectos do domínio e cujos arcos etiquetados representam associações entre os objectos. Nos enquadramentos cada objecto é representado por uma colecção de *atributos*, frequentemente apelidados de *slots*, juntamente com os respectivos valores e possíveis restrições nos valores; a esta colecção de atributos chama-se *enquadramento*, ou *frame*. Nos sistemas baseados em enquadramentos os objectos do domínio são representados numa rede de enquadramentos, semelhante a uma rede semântica, entre os quais relações hierárquicas (classe, subclasse e instância) se estabelecem.

Exemplos de sistemas de escalonamento baseados em conhecimento que usam redes semânticas são o sistema de escalonamento para produção flexível descrito em [Yang 1988] e o sistema de escalonamento da produção OPAL [Bensana 1988].

No sistema o ISIS (*Intelligent Scheduling and Information System*) [Fox 1982], [Fox 1983], [Fox 1994], um dos primeiros sistemas de IA para escalonamento, que foi aplicado ao problema de escalonamento de processamento intermitente de uma fábrica de componentes para turbinas (Westinghouse, USA), é usada a linguagem baseada em enquadramentos SRL (*Schemata Representation Language*) para representar os objectos do domínio de escalonamento. O sistema ISIS foi um dos primeiros sistemas em que se aplicaram técnicas de *modelação* e de *procura heurística* de IA ao problema do escalonamento e em que, pela primeira vez, se tentou tratar toda a gama de restrições e objectivos do domínio da produção.

No sistema de escalonamento OPIS (*Opportunistic Intelligent Scheduler*) [Smith 1990], [Smith 1994], um sucessor do ISIS, para modelar processos, usam-se protótipos de processos (planos de produção não instanciados) que são representados por hierarquias de tarefas. Tarefas agregadas contêm sequências de tarefas mais detalhadas ou conjuntos de alternativas exclusivas de tarefas. A descrição de uma tarefa contém as relações de precedência (tarefas antecessora e sucessora), a duração e os requerimentos de capacidade e preparação. Para modelar os recursos, os recursos elementares são agrupados em conjuntos de recursos maiores para fornecer uma descrição das restrições de afectação de recursos (capacidade disponível, horas de operação) a cada nível de abstracção nos protótipos de processos. Há recursos unitários (afectáveis a um só processo, por exemplo, uma máquina), recursos de lote (afectáveis a vários processos num mesmo intervalo de tempo, por exemplo, um forno) e recursos conjuntivos ou disjuntivos (afectáveis a vários processos com ou sem sincronização temporal) de capacidade agregada.

2.2.2.2 Escalonamento com Representação do Conhecimento através de Regras e Lógica

Em muitas abordagens ao escalonamento, é representado conhecimento na forma regras de tipo condição-acção, ou empregando lógica.

O sistema Scheplan, descrito em [Numao 1994], é um sistema de apoio à decisão para escalonamento da produção no fabrico de aço. Neste ambiente de produção específico existem

restrições específicas relativamente ao tempo de espera entre tarefas, já que os processos têm de ser completados antes do aço solidificar. O Scheplan dispõe de uma componente interactiva em que o sistema e o utilizador cooperam na obtenção de um escalonamento. O processo de escalonamento consiste numa fase em que um escalonamento é gerado pelo sistema, com base em conhecimento heurístico de peritos, e uma fase iterativa em que o utilizador avalia e tenta refinar o escalonamento apoiado pelo sistema (para evitar violação de restrições) até obter um escalonamento satisfatório. Há dois critérios de avaliação considerados funções objectivo: o tempo de espera (a minimizar) e o número de lotes de fabrico (uma medida da produtividade, a maximizar). A prioridade atribuída a cada um varia de acordo com a situação, o tempo e quem avalia. A arquitectura do sistema, semelhante à de um *sistema pericial*, é a de um *sistema de produção*, baseado em *regras de produção* (ver o Apêndice B), sendo composta por três componentes: um motor de escalonamento, uma base de regras e uma interface para o utilizador. O motor de escalonamento recebe um escalonamento que, possivelmente, viola algumas restrições (de capacidade e de tempo de espera) e produz um escalonamento que os satisfaz, usando um algoritmo que aplica regras da base de regras. O sistema faz uso de regras que representam conhecimento dependente do domínio para resolver as restrições específicas do domínio, através de um mecanismo de emparelhamento de padrões e regras que representam o conhecimento heurístico de especialistas e que concorrem para uma maior eficiência do processo de escalonamento e uma melhor qualidade do escalonamento resultante. Existem outras regras para estabelecer processos mediante ordens de produção diárias recebidas, nomeadamente o estabelecimento de lotes de produção, a sequência de tarefas numa máquina e a duração de processo.

Uma abordagem semelhante é descrita em [Darby-Dowman 1995]. Trata-se de um sistema de apoio à decisão para escalonamento de transportes aplicado ao escalonamento de recursos (barcos e aeronaves de vários tipos) utilizados em missões críticas de guarda costeira nos Estados Unidos. É uma abordagem mista de sistema baseado em conhecimento e optimização (utiliza um modelo de programação inteira) e utiliza um sistema baseado em regras para diagnosticar (criticar) os escalonamentos gerados.

SONIA [Collinot 1988], [Le Pape 1994], é um sistema de escalonamento para processamento intermitente que dá especial ênfase às restrições e ao controlo inteligente da *propagação de restrições*. Esta é entendida como uma actividade dedutiva, executada por um sistema de propagação de restrições, em que são derivadas restrições novas a partir das já existentes e detectadas inconsistências entre restrições. A propagação de restrições é realizada por um sistema de inferência controlável, que faz uso de dois tipos de regras: regras de dedução, que determinam que inferências poderão fazer-se a partir de um conjunto de premissas; regras de controlo, que decidem que inferências devem ser feitas em cada passo do processo de resolução do problema. São usadas cláusulas de lógica para representar restrições. Por exemplo, uma restrição temporal pode ser representada pela cláusula `(time <= t1 t2 d)`, em que `t1` e `t2` são os pontos temporais de início e de fim de uma tarefa e `d` é uma duração, declarando que entre os instantes `t1` e `t2` devem decorrer, pelo menos `d` unidades de tempo; são possíveis igualmente, fórmulas disjuntivas, conjuntivas e negativas deste tipo de restrições. As restrições de reserva servem para a propagação de restrições manter horários e detectar conflitos de capacidade. Por exemplo, a restrição `(reserve res t1 t2 n lista-de-motivos)` declara que `n` recursos do grupo de recursos `res` estão indisponíveis durante o intervalo de tempo entre os instantes `t1` e `t2`, descrevendo-se na lista em `lista-de-motivos` as razões dessa indisponibilidade. No SONIA podem ainda representar-se relações de ordem em conjuntos infinitos totalmente ordenados (aceitando igualmente, cláusulas disjuntivas, conjuntivas e negativas), relações entre intervalos (do tipo

das descritas em [Allen 1983] e aceitando igualmente fórmulas disjuntivas, conjuntivas e negativas de relações de intervalos) e famílias várias de restrições de domínio das variáveis, incluindo igualdades e desigualdades.

O sistema DEVISER [Vere 1983] é um planeador de actividades que combina planeamento com simulação determinística, guiada por objectivos, para realizar escalonamento das actividades. Neste sistema as actividades são representadas por regras, apelidadas de produções relacionais, parecidas com as regras usadas no sistema de planeamento STRIPS [Fikes 1971] e são compostas por três componentes: o contexto, o antecedente e o consequente. O contexto contém pré-condições para a actividade, o antecedente pré-condições a serem retiradas do modelo do ambiente quando a actividade ocorre e o consequente literais a juntar ao modelo do ambiente quando a actividade ocorre.

Também o ISIS, referido na última secção, usa regras específicas do domínio que são aplicadas antes e depois da invocação do mecanismo de procura utilizado em vários níveis de construção de uma solução de escalonamento. As regras usadas antes da procura servem para construir um problema a ser resolvido com a procura como, por exemplo, determinar a direcção do escalonamento (escalonamento para a frente a partir da data de início ou para trás a partir da data limite), criação de restrições em falta (de data limite, relativos a trabalhos em curso), escolha do conjunto de operadores de procura. Após a procura aplicam-se regras para avaliar os resultados e, se não se obteve um escalonamento aceitável, para propor relaxação de restrições (por exemplo, relaxar a data limite) para fazer re-escalonamento.

Exemplos de outros sistemas de escalonamento baseados em regras são o OPAL, que utiliza regras juntamente com redes semânticas para representar conhecimento e o descrito em [Fischer 1994] que é realizado por um sistema multi-agente baseado em regras.

Muitos sistemas de IA representam conhecimento através de cláusulas de lógica, ou são realizados numa linguagem baseada em lógica. Por exemplo, REAKTION [Henseler 1993], um sistema para escalonamento reactivo, é realizado na linguagem PROLOG.

No sistema descrito em [Sauer 1993] o conhecimento de escalonamento é constituído por heurísticas representadas por regras do tipo *condição-acção*. A parte da condição especifica em que situações se aplica a regra, os objectivos que a regra atinge e que eventos resolve. A parte da acção é utilizada para invocar estratégias predeterminadas, para descrever completamente estratégias de escalonamento ou re-escalonamento ou ainda descrever planos não instanciados. Ambas as partes podem conter chamadas à linguagem PROLOG.

A *lógica difusa* (*fuzzy logic*), uma classe de lógicas de múltiplos valores em que as proposições podem ter valores de verdade difusos, não se limitando aos dois valores *verdadeiro* e *falso*, é por vezes utilizada representar o conhecimento vago dos peritos humanos. Estas lógicas baseiam-se na teoria dos *conjuntos difusos* (ver, por exemplo, [Kaufmann 1977]), em que a noção de membro de um conjunto é difusa, *i.e.*, pode variar gradualmente entre os extremos de membro e não membro. Numa versão aperfeiçoada do sistema OPAL descrita em [Bensana 1993], empregam-se critérios e regras de decisão difusos para focar a procura de uma solução. Durante a resolução do problema, um critério de avaliação é usado para escolher o conflito (par de tarefas competindo pelo mesmo recurso) mais interessante a resolver a seguir, entre os que existem. A seguir, regras de decisão difusas decidem a sequenciação das duas tarefas envolvidas no conflito. Estas regras exprimem uma avaliação baseada numa *preferência* na resolução de um conflito. Existem regras difusas de prioridade (análogas às regras heurísticas SPT, EDD, FIFO, etc.), de folga (para escolher a

sequência que economiza mais folga temporal) e regras difusas específicas que podem ser definidas dependendo da aplicação.

No sistema de escalonamento para fabrico de aço descrito em [Dorn 1994a] e [Dorn 1994b] a teoria dos conjuntos difusos é usada para representar restrições e comparar soluções mediante uma função de avaliação que determina se um escalonamento é melhor ou pior a respeitar as restrições. Valores difusos são usados para descrever a duração das tarefas, se os processos estão mais ou menos atrasados, se dois processos a executar sequencialmente no mesmo recurso são mais ou menos compatíveis do ponto de vista químico e se os processos são mais ou menos prioritários. Por exemplo, a compatibilidade de dois processos para execução sequencial no mesmo recurso, para cada elemento químico, é estabelecida por regras difusas que, a partir de uma medida de comparação entre as quantidades do elemento nos aços do primeiro processo e do seguinte fornecem uma medida de compatibilidade dos dois processos. A medida de comparação é expressa pelos valores difusos *menos*, *um-pouco-menos*, *igual*, *um-pouco-mais*, *muito-mais* e *no-limite*, associados, através de uma *função de pertença*, a gamas de valores numéricos obtidos da composição de cada aço; a conclusão é expressa pelos valores difusos de compatibilidade *muito-baixa*, *baixa*, *média*, *alta* e *muito-alta*. Para obter a compatibilidade para todos os elementos importantes nos dois aços é calculada uma média ponderada de cada um destes valores. Para avaliar em quanto as restrições de compatibilidade de um escalonamento são satisfeitas, as médias ponderadas para cada par de processos sequenciados em cada recurso, são depois combinadas através do operador de conjunção difusa.

O sistema descrito em [Custódio 1992] e [Custódio 1993], para planeamento e escalonamento da produção, baseia-se também em lógica difusa e usa uma estrutura hierárquica de controlo em que a resolução de problemas de produção diferentes é distribuída a vários níveis.

2.2.2.3 Escalonamento Baseado em Casos

Um sistema que resolve problemas por analogia com a resolução de outros é frequentemente designado por sistema de raciocínio baseado em casos (*case based reasoning* ou CBR). Esta classe de sistemas depende, em geral, de uma grande memória, ou colecção, de casos, vulgarmente chamada *base de casos*. Questões importantes que se lhes põem são: como estão os casos organizados na memória e como são acedidos (um acesso eficiente é, naturalmente, importante), como são os casos prévios adaptados para a resolução de novos problemas e como são inicialmente adquiridos os casos (aprendizagem de casos) para a colecção de casos [Rich 1991]. Em alguns sistemas de escalonamento é utilizada uma metodologia baseada em casos, idêntica à dos sistemas CBR. As vantagens da abordagens dos sistemas CBR têm a ver com a capacidade de conseguir uma procura de solução mais eficiente e adequada ao contexto e conseguir melhorar escalonamentos fazendo compromissos entre objectivos de escalonamento que são difíceis de exprimir numa simples função objectivo.

O sistema de escalonamento CABINS [Miyashita 1994], [Sycara 1994] opera por reparação de escalonamentos completos (melhoria iterativa), a partir de um escalonamento inicial e usa raciocínio baseado em casos para aquisição e reutilização de preferências de escalonamento e escolha de acções de reparação de escalonamentos. Um mecanismo de propagação de restrições serve para propagar as acções de modificação por todo o escalonamento. A base de casos contém exemplos aplicáveis em diversas situações de resolução de problemas, que são usados para obter uma parametrização apropriada de um procedimento de procura para um estado actual de resolução de um problema, avaliar resultados intermédios e retroceder em caso de falha da revisão do escalonamento. Cada caso é composto por atributos que

caracterizam a potencial flexibilidade de reparação para todo escalonamento e que indicam a eficácia esperada da aplicação de uma táctica de reparação particular. Uma táctica é uma acção de reparação que pode consistir, por exemplo, em avançar uma actividade dentro do horizonte temporal de reparação no mesmo recurso, ou num recurso alternativo, ou em trocar no tempo uma actividade com outra no mesmo recurso, ou num recurso alternativo, de modo a minimizar violações de restrições de precedência. Existe uma medida de semelhança entre casos, baseada nos valores dos atributos de casos diferentes, que permite extrair casos da base de casos semelhantes ao caso do problema a tratar. Além dos atributos referidos, cada caso contém também um registo da aplicação sucessiva de tácticas de reparação, com os seus efeitos (*i.e.*, o impacto de uma acção de reparação nos objectivos de escalonamento), avaliação (aceitável ou não aceitável) e uma explicação em caso de falha da reparação. Os atributos podem também ser utilizados como índices para aceder a casos na base de casos. Se o resultado da avaliação de uma reparação é inaceitável, os atributos do caso do problema actual são usados para aceder a casos do passado em que a reparação falhou de modo semelhante para poder recuperar da falha de reparação. Em casos de falhas sucessivas isto é repetido até que o resultado de uma reparação seja aceitável, ou até que se declare falha da reparação, por não haver mais tácticas a serem aplicadas na situação actual. Numa fase de treino, os casos são introduzidos na base pelo utilizador que escolhe uma táctica de reparação e avalia os resultados da sua aplicação, ou são propostos pelo sistema que escolhe tácticas de reparação e avalia o resultado havendo ou não, de seguida, a aceitação pelo utilizador. Em modo autónomo (sem intervenção do utilizador) a base de casos adquirida na fase de treino é utilizada para escolha de reparações e avaliação de resultados delas.

Ao contrário do CABINS, que repara escalonamentos, o sistema descrito em [Bezirgan 1993], também baseado em casos, é aplicado à geração de escalonamentos. Outros sistemas de escalonamento que usam CBR de um modo não muito diferente do CABINS são descritos em [Dorn 1995] e [Szelke 1995] (no primeiro, a semelhança de casos é definida por meio de conjuntos difusos). Em [Costello 1995] descreve-se uma ferramenta computacional baseada em casos para acumular conhecimento informal de peritos humanos em escalonamento.

2.2.2.4 Escalonamento Baseado em Procura e em Satisfação de Restrições

Um grande número de problemas pode ser visto como sendo casos especiais do *problema de satisfação de restrições* (PSR, *constraint satisfaction problem* ou CSP) e o escalonamento é um desses problemas. O PSR é o problema de encontrar valores para as variáveis de um dado conjunto de variáveis de modo a que certas restrições sejam satisfeitas; uma atribuição de valores a todas as variáveis que satisfaça todas as restrições é uma solução do PSR (ver o Apêndice C).

Visto que o escalonamento é um problema difícil, é muitas vezes impossível obter uma solução óptima para um problema de escalonamento em tempo útil, sendo preferível obter uma solução que seja satisfatória em tempo reduzido. Por esta razão, problemas de escalonamento são frequentemente formulados e resolvidos como problemas de satisfação de restrições [Fox 1982], [Prosser 1988], [Fox 1990], [Atabakhsh 1991], [Sadeh 1991], [Burke 1994], [Fox 1994], [Le Pape 1994], [Muscettola 1994a], [Sadeh 1994], [Smith 1994], [Sadeh 1995a]. Em vez de usar programação matemática e tratar o problema como um problema optimização é frequentemente preferível formular e resolver o problema como um PSR porque a representação na forma de PSR é muito mais próxima do problema original (as variáveis do problema correspondem directamente a entidades do problema e as restrições podem ser expressas sem terem de ser traduzidas para desigualdades lineares) e porque os algoritmos para

PSRs, sendo essencialmente muito simples, podem encontrar soluções mais rapidamente do que métodos de programação matemática [Smith 1995]. Também segundo [Fox 1990], ver o problema de escalonamento como um PSR pode ser útil, porque o PSR é um dos problemas que tem sido bastante investigado na área de IA, principalmente em termos de classificação de problemas e análise de complexidade; veja-se, por exemplo, [Mackworth 1977], [Haralick 1980], [Nudel 1983], [Purdom 1983], [Mackworth 1985], [Davis 1987], [Dechter 1988], [Dechter 1990] ou [Dechter 1991]. Se o problema de escalonamento é dinâmico, o conjunto de tarefas e de restrições podem alterar-se frequentemente e estas mudanças são mais bem acomodadas no caso do modelo de PSR, ao contrário de um modelo de optimização, para o qual será necessário reformular problema a recalcular a solução.

De um modo geral, para o domínio de escalonamento, as variáveis do PSR representam tarefas a escalonar, os valores possíveis são os tempos (de início, ou de fim) das tarefas e também os recursos a afectar, quando há tarefas com requerimentos de recurso para os quais há recursos alternativos; as restrições do PSR representam restrições temporais de precedência entre as tarefas, restrições temporais de data limite, restrições de capacidade sobre os recursos (por exemplo, um recurso não poder executar mais de uma tarefa de cada vez) e restrições que limitam o conjunto de recursos que podem ser usados para execução de cada tarefa, quando há recursos alternativos (ver, por exemplo [Sadeh 1991], ou [Zweben 1992]). Procura-se uma solução, *i.e.*, um conjunto de completo de atribuições de tempos e recursos para as tarefas, que satisfaça as restrições e pretende-se, em geral, que o processo de procura da solução seja o mais rápido possível.

Não é raro num PSR de escalonamento definirem-se, à partida, *restrições relaxáveis*, também apelidadas de *preferências*. Trata-se de um tipo de restrições que podem não ser respeitadas, não comprometendo, no entanto, só por isso, a exequibilidade de uma solução. Nesse caso distinguem-se as *restrições relaxáveis* das *restrições rígidas*,[39] designando este último tipo de restrições as que têm de ser sempre satisfeitas numa solução possível do problema de escalonamento. Num problema de PSR de escalonamento, ao encontrar uma solução não possível podem, então, usar-se técnicas de *relaxação de restrições*, em que se tentam valores alternativos para as variáveis sobre as quais incidem restrições relaxáveis [Fox 1994].[40] Tipicamente, as restrições de capacidade e as de precedência temporal impostas nas tarefas de um processo por razões de ordem tecnológica são restrições rígidas. Quando há recursos alternativos para as tarefas, as restrições que limitam o uso de um recurso a um recurso pertencente ao conjunto de recursos alternativos são consideradas restrições relaxáveis. Um tipo de restrições que podem ou não ser consideradas relaxáveis no escalonamento são as restrições de data limite.

2.2.2.5 Escalonamento Construtivo

No escalonamento construtivo, uma ou mais soluções parciais de escalonamento são mantidas e progressivamente ampliadas em cada passo, durante a resolução do problema.

No sistema ISIS [Fox 1982], [Fox 1983], [Fox 1994], aplicam-se técnicas procura heurística guiada por restrições (ver o Apêndice C) ao problema do escalonamento. Neste sistema é dado ênfase especial à representação e ao uso de conhecimento sobre restrições, que dá suporte à

---

[39] *Relaxable constraints*, *soft constraints* ou *preferences*, são designações frequentemente empregues na literatura, para o que aqui se designa por restrições relaxáveis, ou preferências; *hard constraints* ou *constraints*, designam as restrições rígidas, ou, simplesmente, restrições.

[40] Num caso extremo, a remoção da restrição.

resolução de conflitos e relaxação de restrições. O ISIS adopta uma perspectiva de escalonamento centrada no processo (*i.e.*, escalona um processo inteiro de cada vez) e segue uma abordagem construtiva (ver também o Apêndice C) hierarquizada em quatro fases, a seguir resumidamente descritas.

Na primeira fase o processo com maior prioridade é escolhido para ser escalonado. As prioridades são calculadas mediante a data limite e o tipo de ordem que desencadeou o processo (que varia desde a encomenda mais urgente para peças de substituição até à ordem de fabrico para armazenamento, menos urgente). Numa segunda fase, dados os limites temporais impostos para o processo (data de início e data limite), são gerados os tempos limite para cada tarefa do processo, para cada um dos recursos alternativos nos quais a tarefa pode ser executada. É usada programação dinâmica (análise de caminho crítico) para, tendo em conta a capacidade finita dos recursos, determinar tempos de início mais cedo e tempos de fim mais tarde de cada tarefa (a propagar para a fase seguinte na forma de restrições sobre cada tarefa), de forma rápida, ignorando restrições e recursos com pouco impacto em problemas de congestionamentos de recursos. Esta fase foi introduzida como aperfeiçoamento, após ter sido verificado que o desempenho do ISIS era fraco devido ao facto de haver um tempo de espera elevado em recursos congestionados e permite tentar tornear o problema do congestionamento de certos recursos por escolha de recursos alternativos, ou fazendo chegar o processo mais cedo ao recurso congestionado. Na terceira fase, que consiste na procura do escalonamento mais promissor, é usada uma estratégia de procura em faixa (ver o Apêndice B) num espaço de estados de escalonamentos parciais alternativos, para definir e ordenar um conjunto de escalonamentos considerados melhores. A propagação de restrições é progressiva, *i.e.*, em cada estado, as restrições existentes advêm das decisões feitas em estados anteriores na procura e servem para limitar o espaço de procura. Nesta fase os recursos são sequencialmente afectados às tarefas, escalonando para trás, a partir da última tarefa, ou para a frente, a partir da primeira tarefa do processo. Os limites temporais para cada tarefa determinados na fase anterior são refinados tendo em conta agora todas as restrições. Quando há conflitos, as restrições envolvidas são identificadas e relaxadas e o escalonamento reparado. Numa quarta fase estabelecem-se efectivamente as reservas de recursos, por estabelecimento dos tempos de execução das tarefas em cada recurso necessário, deslocando tarefas para mais cedo no tempo (avanço) ou para mais tarde (atraso), dentro dos limites temporais para elas determinados, de modo a minimizar o tempo de fluxo. Estas quatro fases são repetidas até que todos os processos tenham sido escalonados.

Na óptica do ISIS as restrições são classificadas em restrições de objectivos organizacionais (por exemplo: datas limite, níveis de trabalhos em curso, níveis de produção, níveis dos recursos como pessoal, matérias-primas, ferramentas), restrições físicas (limitações de funcionalidade dos recursos), restrições causais (precedências e requerimentos de recursos das tarefas), disponibilidade dos recursos (um recurso afectado a uma tarefa não estará disponível para outra) e preferências (preferências por recursos, tarefas, ou sequências de tarefas, por razões de qualidade ou de custo) [Fox 1994]. Na representação das restrições é-lhes associado conhecimento. Para relaxação de restrições são associados valores que exprimem uma preferência por cada relaxação possível, ditos valores de *utilidade*. Um outro aspecto é o da *importância* de uma restrição, um peso que indica a influência relativa da restrição (por exemplo, em processos de alta prioridade, o ISIS dá um peso elevado às restrições de data limite). Um terceiro aspecto é o da *relevância* de uma restrição, que define as condições em que ela é aplicável. A *interacção* entre restrições é outra forma de conhecimento, que indica como, e em quanto, a satisfação de uma restrição interfere com a satisfação de outra; estas interdependências assumem a forma de relações entre restrições, com medidas de sensibilidade

que indicam o graus e direcções de influência entre as restrições. Este conhecimento é usado para diagnosticar as causas de soluções propostas pelo sistema e sugerir relaxações para restrições relacionadas que poderão dar melhores resultados. A ideia básica da utilidade é a de permitir escolher, quando há que relaxar restrições, das alternativas ainda disponíveis, as que maximizam as preferências, de modo a tentar não comprometer muito a qualidade da solução (encontrar a solução menos "não satisfatória"). Também na fase de procura, o ISIS atribui, a cada estado de procura, um valor de avaliação resultante de uma média, ponderada pela importância, dos valores de utilidade das restrições relevantes para o estado actual e para os estados antecessores. Estes valores são usados para determinar se uma solução final é aceitável.

Mais do que um sistema único, identificou-se no projecto ISIS *uma família de sistemas* de escalonamento automático, baseados em *satisfação de restrições*, cujo objectivo era o de serem desenvolvidos e testados em fases sucessivas, com uma arquitectura progressivamente mais sofisticada em cada fase. Em princípio deveria vir a incorporar-se toda a gama de restrições de situações reais do problema, mas a continuação destes esforços parece ter sido abandonada, embora tenha sido depois, de alguma forma, continuada pelo sistema OPIS. Uma das razões para isto é o facto do desempenho do sistema ser negativamente afectado pela rigidez do procedimento de procura, que impunha uma *perspectiva centrada no processo*, isto é, um escalonamento dos processos um por um. O procedimento era eficiente no que respeita à redução das existências mas tinha dificuldade em optimizar a utilização dos recursos congestionados.

O sistema de escalonamento OPIS (*Opportunistic Intelligent Scheduler*) [Smith 1990], [Smith 1994] deu continuação aos esforços iniciados com o desenvolvimento do sistema ISIS e introduziu algumas técnicas novas de escalonamento baseado em restrições. Em primeiro lugar, identificou e demonstrou a utilidade de usar *múltiplas perspectivas de escalonamento*. O OPIS usa uma *perspectiva centrada nos processos*, idêntica à do ISIS, e uma *perspectiva centrada nos recursos*, que é usada para escalonar recursos congestionados. Distingue, portanto, recursos congestionados de recursos não congestionados e reconhece que podem surgir congestionamentos novos durante o processo de escalonamento. Em segundo lugar tem a capacidade de, durante o processo de escalonamento, mudar de perspectiva de escalonamento, o que é apelidado de estratégia de *escalonamento oportunista*. O escalonamento é um processo iterativo em que, no início de cada ciclo, é feita uma *análise de capacidade* para determinar se há um grau de competição elevado entre as tarefas por algum recurso particular. Se isso acontece esse recurso é considerado congestionado e é usada a perspectiva centrada no recurso para escalonar todas as tarefas que o requerem; caso contrário é usada a perspectiva centrada no processo da mesma forma que no ISIS (o processo com maior prioridade é escolhido e todas as tarefas desse processo são escalonadas). Em terceiro lugar, para além de *escalonamento predictivo, i.e.*, prescrição de soluções de escalonamento, como no ISIS, o OPIS prevê *escalonamento reactivo* (ver, por exemplo [Szelke 1994], [Kerr 1995]), de modo a poder adaptar dinamicamente escalonamentos estabelecidos predictivamente para acomodar eventos não previstos no ambiente (por exemplo, atrasos e falhas nos recursos, materiais que não chegam a tempo, alterações na prioridade dos processos, cancelamento de processos) em tempo real sem comprometer a continuidade da execução e a satisfação dos objectivos de escalonamento. No OPIS, o escalonamento reactivo é baseado na revisão/adaptação incremental do escalonamento e opera tanto para acomodar eventos não previstos no ambiente, como para responder a oportunidades de melhoria de uma solução que apareçam como resultado de acções anteriores de modificação do escalonamento

(quer dizer, pode operar tanto devido a razões externas como internas relativamente ao sistema de escalonamento).

O facto de repetir a análise de capacidade sempre que foi escalonado um recurso ou um processo permite ao OPIS detectar o aparecimento de novos congestionamentos durante a construção do escalonamento e rever a estratégia actual de escalonamento, se for necessário. Trata-se de um certo tipo de oportunismo, ou estratégia oportunista, de granulação grossa (é necessário escalonar completamente um recurso ou um processo, antes de focar a atenção noutro recurso ou processo) designado por *macro-oportunismo*. Embora com a vantagem de reduzir o número de análises de capacidade (que tem custo computacional elevado), o macro-oportunismo pode revelar-se limitado, pois nem sempre os congestionamentos se estendem por todo o horizonte temporal de escalonamento e podem deslocar-se antes de todas as tarefas associadas serem escalonados. Como resultado, ao fim de um ciclo de escalonamento, o sistema pode já ter enveredado por uma solução não tão boa quanto a que poderia ter se, após cada análise de capacidade, escalonasse um menor grupo de tarefas (e não todas as tarefas que usam um recurso, ou todas as tarefas de um processo inteiro). Uma abordagem que usa uma estratégia oportunista de granulação mais fina, em que é escalonado um grupo de tarefas em número relativamente pequeno (no limite, uma tarefa) após cada análise de capacidade, é designada por *estratégia micro-oportunista*, ou *micro-oportunismo*.

O sistema OPIS usa uma arquitectura baseada no conceito de *blackboard* que fornece a infra-estrutura para realização de métodos e estratégias de revisão de escalonamentos baseada em restrições e que facilita muito os aspectos positivos do OPIS acima salientados. A arquitectura de *blackboard* [Erman 1980], [Hayes-Roth 1988] pode ser considerada uma evolução da arquitectura apelidada na IA de *sistema de produção* (*production system*, ver o Apêndice B).[41] Nesta arquitectura, um conjunto de processos, designados por *fontes de conhecimento* (*knowledge sources*), usam uma memória partilhada denominada *quadro preto* (*blackboard*) para colocar estruturas simbólicas, apelidadas de *hipóteses*, que contribuem para a resolução de um mesmo problema. Cada fonte de conhecimento é um "especialista" numa área específica e pode encontrar uma hipótese na qual pode trabalhar e modificar ou criar novas hipóteses no quadro preto. Esta arquitectura tem a vantagem de permitir a coexistência e cooperação de várias fontes de conhecimento úteis na resolução de um problema no mesmo sistema, de uma forma modular.

No sistema OPIS existem fontes de conhecimento de métodos de escalonamento (executam tarefas de escalonamento e modificam a representação da solução actual), de actualização do modelo (que actua quando há indicação do exterior de modificações nas restrições do problema) e de análise (que fornecem informação necessária à formulação de tarefas de escalonamento). Adicionalmente, constituem a infra-estrutura oportunista e multi-perspectiva de escalonamento o subsistema de manutenção de escalonamentos (que mantém uma representação das restrições actuais da solução) e o gestor de topo (para coordenar o uso dos métodos de escalonamento, análise e actualização do modelo), que contém um ciclo dirigido por eventos onde é aplicado o conhecimento das estratégias de escalonamento com base em restrições.

O escalonamento das tarefas de um processo é obtido partindo da instanciação de um protótipo do processo (plano de produção). O subsistema de manutenção de escalonamentos mantém, em cada passo da resolução do problema, restrições de dois tipos: *limites temporais* (os tempos de início mais cedo e de fim mais tarde de cada tarefa) e *capacidade disponível dos*

[41] Ver em [Bond 1988], Capítulo I, página 5.

recursos (sequências de intervalos de tempo com a capacidades disponíveis associadas para todo o horizonte temporal para cada recurso). Estas restrições vão sendo actualizadas como resultado de decisões de escalonamento do sistema e de haver restrições originadas por eventos externos. A propagação de restrições resulta destas modificações no escalonamento, podendo originar dois tipos de conflitos: *conflitos temporais*, se há violação de restrições de precedência ou da data limite do processo e *conflitos de capacidade*, se os requerimentos de tarefas escalonadas excedem a capacidade disponível de um recurso durante algum intervalo de tempo.

As fontes de conhecimento de análise usam métricas para estimar, de forma resumida, graus de flexibilidade das restrições (folgas, atrasos, tempos de recurso parado) e de severidade dos conflitos (duração e número de tarefas envolvidas num conflito) existentes no escalonamento. Os valores fornecidos por estas métricas são usados para escolher as fontes de conhecimento que aplicam os métodos de escalonamento apropriados. Para além de métodos para escalonamento predictivo de recursos e de processos, existem uma série de outros métodos para revisão de escalonamentos usados em escalonamento reactivo, incluindo métodos para dar resposta a conflitos, métodos que realizam tarefas de escalonamento como deslocar tarefas no tempo para mais cedo, ou para mais tarde, ou trocar os recursos afectados a duas tarefas.

Tal como o OPIS, o sistema de escalonamento SONIA [Collinot 1988], [Le Pape 1994] para escalonamento predictivo e reactivo em processamento intermitente, é também baseado no conceito de *blackboard*. Esta arquitectura permite-lhe integrar algoritmos matemáticos, conhecimento de especialistas em escalonamento, conhecimento dependente da fábrica e capacidades interactivas. Em termos de modelação do domínio de escalonamento (representação de escalonamentos, de processos, hierarquias de recursos, tipos de restrições) o SONIA é muito semelhante ao OPIS. É um sistema de escalonamento macro-oportunista, pois cada fonte de conhecimento actua executando uma série de várias decisões de cada vez que é aplicada. É um sistema baseado em restrições e dá especial ênfase ao controlo inteligente da propagação de restrições e à selecção das heurísticas a empregar pelas fontes de conhecimento, que podem usar heurísticas diferentes conforme a situação em cada momento.

A arquitectura do SONIA incorpora fontes de conhecimento de escalonamento predictivo e reactivo, de análise de capacidade e de conflitos, de interface com o utilizador e com o ambiente exterior e duas fontes para controlo do comportamento do sistema, denominadas controlador de propagação de restrições e controlador de heurísticas. Adicionalmente o sistema incorpora um sistema de gestão de escalonamentos e um sistema de propagação de restrições. O primeiro recebe informação de decisões de escalonamento e de eventos inesperados (falhas de máquinas, atrasos, na fábrica) e mantém o escalonamento, criando restrições temporais e de reserva, que envia para o sistema de propagação. Este último trata o problema puramente na perspectiva do PSR.

Como já anteriormente foi dito, na óptica do SONIA, a propagação de restrições é entendida como uma actividade dedutiva, em que são derivados restrições novas a partir das já existentes e detectadas inconsistências entre restrições. Ao contrário dos sistemas ISIS e OPIS, que incluem um sistema de gestão de restrições próprio, SONIA é suportado por um conjunto de procedimentos e uma linguagem para exprimir restrições, baseados em técnicas gerais de propagação de restrições.[42] O componente gestor de restrições é um módulo independente do

[42] Existem linguagens orientadas para resolução de restrições. Por exemplo, *Programação em Lógica com Restrições*, ou *Constraint Logic Programming* (CLP), designa uma classe destas linguagens que são baseadas em lógica (ver, por exemplo, [Hentenryck 1992], [Hentenryck 1994], [Smith 1995] ou [Barahona 1997]).

sistema solucionador de problemas de escalonamento que se liga com este último, permitindo-lhe adicionar e remover restrições e permitindo o acesso à informação sobre as variáveis e as restrições do PSR. Aquele módulo pode registar dependências entre restrições (*i.e.*, memoriza de que outras restrições foi originada cada restrição), para poder fornecer a informação necessária ao sistema sobre conflitos existentes de modo a que ele possa realizar retrocesso selectivo sem recriar as mesmas condições de insucesso.[43]

O gestor de restrições contém regras de dedução e regras de controlo. As regras de dedução indicam que inferências poderão fazer-se a partir de um conjunto de premissas (*i.e.*, permitem deduzir restrições novas a partir de restrições existentes) e, ao serem aplicadas, executam funções de propagação de restrições. As regras de controlo especificam que inferências devem ser feitas em cada ponto do processo de resolução do problema e controlam, portanto, a aplicação das regras de dedução. Estas regras são associadas a cada função de propagação para especificar se, e em que condições, a função de propagação deve intervir no processo de propagação de restrições.

Existem fontes de conhecimento dedicadas a dois aspectos específicos de controlo do comportamento de escalonamento do sistema SONIA: o controlador de propagação de restrições e o controlador de heurísticas. O controlador de propagação de restrições pode alterar o conjunto de regras de controlo do gestor de restrições para regular a quantidade de propagação, de modo a permitir um compromisso entre a detecção de conflitos e o tempo computacional despendido na propagação. O controlador de heurísticas activa ou desactiva regras heurísticas do conjunto de regras heurísticas de escalonamento (*e.g.*, EDD, ou escalonar recursos congestionados primeiro, se os houver) usadas pelas fontes de conhecimento de escalonamento, mediante informação de contexto obtida através das fontes de conhecimento de análise de capacidade.

O Micro-Boss (*Micro-Bottleneck Scheduling System*) [Sadeh 1991], [Sadeh 1994] é um sistema de apoio à decisão para escalonamento de processamento intermitente desenvolvido para a produção *em-tempo* (*just-in-time*)[44] em fábricas, que integra componentes para escalonamento predictivo, reactivo e interactivo. O Micro-Boss tem em conta a capacidade finita dos recursos e dá ênfase à análise do espaço de soluções durante a procura, na qual emprega técnicas apelidadas de *olhar-para-a-frente* (*look-ahead*) para medir a procura e a competição das tarefas pelos recursos, usando heurísticas específicas para identificar e escalonar tarefas mais críticas. A abordagem ao escalonamento é micro-oportunista, pois o esforço para resolução do problema é continuamente direccionado para a tarefa mais crítica e para o recurso mais congestionado. O Micro-Boss tem uma perspectiva de escalonamento *centrada em tarefas*, *i.e.*, cada tarefa é considerada um ponto de decisão independente, podendo ser escalonada sem que outras tarefas, usando o mesmo recurso ou pertencendo ao mesmo processo, o sejam. Resultados experimentais relatados em [Sadeh 1991] e [Sadeh 1994] indicam que as técnicas usadas neste sistema de escalonamento produzem melhores soluções, e em menos tempo, do que outras propostas na literatura de IO e IA, incluindo várias regras de despacho e técnicas de procura de granulação grossa (*i.e.*, macro-oportunistas). Em particular, naqueles artigos defende-se que, em escalonamento da produção de processamento intermitente, com datas de lançamento e datas limite impostas, para os problemas maiores e

---

[43] Ou seja, permite técnicas de *retrocesso guiado por dependências* (ver os apêndices B e C).

[44] Sistema de gestão da produção guiado pela procura que, no que mais respeita ao escalonamento, dá especial atenção às datas limite e à minimização de atrasos e avanços. Baseia-se numa filosofia de produzir e entregar apenas o que é necessário, quando necessário e onde necessário e de sincronização dos fluxos de materiais e de redução das existências e tempos de fluxo [Roldão 1995].

mais difíceis, as heurísticas genéricas conhecidas para o PSR não são suficientes para guiar a procura, pois não têm em conta o grau de dificuldade em satisfazer restrições,[45] nem a conectividade dos grafos de restrições. Propõem-se, então, heurísticas específicas enquadradas na estratégia micro-oportunista do Micro-Boss.

Nos problemas de escalonamento da produção pretende-se tipicamente encontrar não só uma solução que seja satisfatória, mas também uma solução de boa qualidade (idealmente a solução de melhor qualidade, *i.e.*, pretende-se optimizar). No escalonamento da produção *just-in-time* pretende-se escalonar minimizando atrasos e avanços relativamente às datas limite impostas. Assim, este tipo de problemas, deve ser visto como um PSR com uma função objectivo a ser optimizada, *i.e.*, como um *problema de optimização com restrições*, (POR, *constrained optimization problem* ou COP) [Fox 1989], [Fox 1990], [Sadeh 1994]. Numa parte do trabalho de investigação exposto em [Sadeh 1991], é apresentado um modelo probabilístico do espaço de procura que permite a definição de heurísticas de ordenação de variáveis e de valores apropriadas para a resolução do PSR de escalonamento de processamento intermitente com datas de lançamento e datas limite, com o algoritmo de procura do Micro-Boss. Outra parte amplia este modelo para tratar o problema de optimização correspondente, *i.e.*, o POR de escalonamento. Aqui é usada uma função objectivo que tem em conta, de forma aproximada, a soma dos custos de atraso relativamente a datas limite e dos custos de posse trabalhos em curso e produtos acabados de cada processo.

O Micro-Boss emprega um algoritmo de procura em profundidade com retrocesso (ver o Apêndice B) que se inicia num estado de procura em que nenhuma tarefa está escalonada (ou, alternativamente, num estado com um escalonamento parcial que se pretende completar) com os seguintes passos [Sadeh 1994]:

1. Parar se todas as tarefas estiverem escalonadas;
2. Aplicar o procedimento de *imposição de consistência*;
3. Se se chegou a um estado sem saída executar retrocesso;
4. Se houver tarefas não escalonadas que ficaram com uma única reserva[46] possível, escalonar essas tarefas, criando um novo estado de procura para cada uma. Parar se todas as tarefas foram já escalonadas;
5. Aplicar o procedimento de análise *olhar-para-a-frente*;
6. Escolher a próxima tarefa a ser escalonada (usa heurística de ordenação das tarefas);
7. Escolher uma reserva para essa tarefa (usa heurística de ordenação das reservas);
8. Criar um novo estado de procura juntando a nova atribuição de reserva ao escalonamento parcial actual. Continuar em 1.

No passo 2, o procedimento de imposição de consistência actualiza as reservas possíveis para cada tarefa ainda não escalonada. Se a propagação de restrições resulta numa folga temporal negativa para uma tarefa (*i.e.*, tempo de início mais cedo maior que o tempo de início mais

[45] Referido em [Sadeh 1991] por *constraint tightness*, este grau de dificuldade é determinado, para uma restrição, pelo número de pares de valores permitidos por ela e por outras restrições que com ela inter-actuam (quanto menor o número de pares de valores permitidos maior o valor de *constraint tightness* de uma restrição).
[46] Uma reserva para uma tarefa é um par composto pelo recurso a afectar e pelo intervalo de tempo em que esse recurso é afectado à tarefa.

tarde) ou um recurso afectado acima da sua capacidade, detecta-se um conflito no passo 3. Nesse caso será, então, realizado retrocesso.[47]

O procedimento de análise *olhar-para-a-frente*, no passo 5, serve para ajudar o sistema a focar o esforço de resolução do problema nos conflitos que, em cada estado, parecem ser mais críticos (com maior impacto na qualidade de todo o escalonamento), para produzir escalonamentos de melhor qualidade. O sistema consegue assim ter o maior número de opções quanto possível para optimizar (pois, assim que os compromissos críticos foram resolvidos as tarefas por escalonar tendem a ser mais desacopladas e o seu escalonamento mais fácil de optimizar) e consegue limitar a quantidade de retrocesso (à medida que a competição de recursos diminui, diminui também a probabilidade de retrocesso). O procedimento de *olhar-para-a-frente* é levado a cabo em dois passos. O primeiro passo consiste na *optimização de atribuições de reservas*. As reservas possíveis de cada tarefa não escalonada são ordenadas de acordo com quanto minimizam os custos do processo a que pertencem (heurística da ordenação das reservas). Para medir a competição pelos recursos, para além dos melhores tempos de início possíveis para cada tarefa não escalonada é mantida uma aproximação de uma função `mincost(τ)` (a variável `τ` refere-se aos tempos de início de cada tarefa não escalonada) que indica o custo mínimo adicional em que incorre o processo respectivo se a tarefa começasse no tempo `τ`, em vez de num dos seus melhores tempos de início; por definição, se `τ` for um desses tempos então `mincost(τ)=0`. O valor de `mincost` advém de `2` parcelas, relativas ao custos de atraso e de posse. O segundo passo consiste na *construção de perfis de procura* para identificar pares de recurso/intervalo de tempo críticos. Pares recursos/intervalo de tempo pelos quais a competição é maior ajudam a identificar a tarefa crítica, a ser escalonada a seguir (heurística da ordenação das tarefas). A importância de um conflito depende do número de tarefas a competir pelo mesmo recurso, a extensão do intervalo de tempo de sobreposição entre os requerimentos de recurso destas tarefas, o número de reservas alternativas ainda disponíveis para cada uma destas tarefas em conflito e os seus custos (determinados pelas funções `mincost`). A cada tempo de início possível para cada tarefa não escalonada é associada uma *probabilidade subjectiva* `σ(τ)` de que cada tempo de início possível `τ` resulte para ela escolhido no escalonamento final (tempos de início possíveis com custo `mincost` mais baixos têm probabilidade mais alta, para reflectir a expectativa de que produzirão melhores escalonamentos). Com base em `σ(τ)` é calculada a *procura individual* da tarefa, `D(t)`, pelo recurso requerido, que é a probabilidade de que ela use o recurso no instante de tempo `t` (e que é, ao mesmo tempo, uma medida subjectiva da expectativa da tarefa na disponibilidade do recurso no tempo `t`). O *perfil de procura agregado* de um recurso para as tarefas não escalonadas é obtido adicionando a procura individual de cada uma dessas tarefas por esse recurso e é uma medida da competição delas por esse recurso.

O passo 6 consiste na escolha da tarefa a escalonar. O pico maior nos perfis de procura agregados permite identificar qual a tarefa mais crítica. A tarefa que mais contribui para aquele pico é a que mais depende da disponibilidade do correspondente recurso/intervalo de tempo e é, portanto, a mais crítica. É esta tarefa que deve ser escolhida para ser escalonada a seguir. O passo 7 escolhe a reserva para a tarefa mais crítica (escolhida para escalonamento no passo

[47] A técnica de resolução de conflitos usada pelo Micro-Boss é a do retrocesso cronológico (desfazer o escalonamento da última tarefa escalonada). Outros mecanismos de resolução de conflitos mais eficientes parecem ter sido, recentemente, desenvolvidos (ver em [Kjenstad 1998], páginas 21 e 22).

anterior). É escolhida a reserva que mais reduz os custos do processo a que a tarefa pertence e dos outros processos com os quais a tarefa compete.

O Micro-Boss dispõe de capacidades de escalonamento reactivo, para atender a factores como duração variável das tarefas, falhas de máquinas, atraso na chegada de matérias-primas, chegada de novas encomendas ou cancelamento de encomendas existentes. Pequenas disrupções são tratadas a um *nível de controlo* por meio de heurísticas de despacho (por exemplo, processar primeiro a tarefa com a data de início escalonada para mais cedo, ou quando uma máquina falhou, reconduzir os processos críticos para máquinas equivalentes, se as houver). Para desvios mais graves do escalonamento o nível de controlo invoca o módulo de escalonamento do Micro-Boss para reparar/re-optimizar o escalonamento a uma perspectiva mais global. Aqui o Micro-Boss procura ter uma visão mais global do problema e aproveitar as capacidades dos procedimentos de procura micro-oportunistas. A reparação é realizada a duas fases: 1) são identificadas as tarefas a serem re-escalonadas e desfaz-se o escalonamento das mesmas e 2) é resolvido do problema de escalonamento composto por aquelas tarefas (tendo em conta as restrições impostas nelas pelas já executadas e as que estão escalonadas) pelo módulo de escalonamento micro-oportunista. Em geral, o problema da fase 2) tem solução, mas se isso não acontece é necessário retroceder à fase 1) e desfazer o escalonamento de um maior número de tarefas. Este retrocesso pode ser evitado de antemão desfazendo, logo na fase 1), o escalonamento de mais tarefas do que o aparentemente necessário (o que poderá revelar-se vantajoso porque, apesar do espaço de procura resultante ser maior, pode conter melhores soluções de reparação).

O Micro-Boss dispõe também de capacidades de escalonamento interactivo, para manipulação do escalonamento, de modo a atender a restrições *ad hoc* e a preferências que ocorrem ocasionalmente, ou que variam ao longo do tempo, apoiar o utilizador na identificação de fontes de ineficiência (ordens atrasadas, recurso sobrecarregados, etc.) e opções possíveis para as corrigir (adicionar mais tempo/turnos nos recurso, reconduzir ordens por outro percurso, etc.), na exploração de cenários e alternativas possíveis (por exemplo, decidir se se devem deixar acabar processos atrasados excedendo datas limite ou introduzir trabalho/turnos suplementares). É permitido ao utilizador executar decisões de escalonamento, intercalando modos manual e automático (sob a supervisão do módulo de imposição de consistência) e analisar, editar, armazenar e comparar, usando métricas variadas como o custo total do escalonamento, atraso ponderado médio, avanço ponderado médio, trabalhos em curso e trabalhos no sistema, escalonamentos completos e parciais[48] através de uma interface em formato de gráfico de Gantt.

Investigação posterior ao desenvolvimento inicial levou a uma nova realização do Micro-Boss com generalização do enquadramento de modelação do domínio de escalonamento do sistema para dar suporte ao escalonamento no contexto da cadeia de fornecimento (*supply chain*) [Kjenstad 1998] e também à inclusão do Micro-Boss como fonte de conhecimento de um sistema semi-interactivo, baseado no conceito de *agente* e numa arquitectura de *blackboard*, para gestão do planeamento e escalonamento integrados, também no contexto da cadeia de fornecimento [Sadeh 1996], [Hildum 1997].[49]

---

[48] No caso de escalonamentos parciais, em que as métricas não podem ser calculadas de forma exacta, são usadas estimativas optimistas.

[49] Esta evolução no sentido de estender o escalonamento a um nível superior ao do escalonamento da produção tradicional do *chão-de-fábrica* (*shop-floor*), em particular ao nível logístico e inter-empresa não é, de modo nenhum, estranha, já que, dado o contexto actual de globalização dos mercados as empresas tendem cada vez mais a se organizarem em cadeias de fornecimento, com uma maior tendência para a integração e planeamento

O sistema HSTS (*Heuristic Scheduling Testbed System*) [Muscettola 1994a] é uma bancada de teste que representa incorpora quadro geral de representação e resolução de problemas abrangendo planeamento de acções e escalonamento, e que serve para desenvolvimento e teste experimental de sistemas de planeamento e escalonamento para domínios não convencionais.[50] No HSTS o problema a resolver é entendido como um conjunto de restrições e preferências nos valores de um *vector de estado*. Este vector contém as variáveis de estado que representam os recursos. A satisfação daquele conjunto de restrições e preferências identifica um plano/escalonamento no conjunto de todos os comportamentos possíveis. Funções de avaliação impõem preferências nos comportamentos possíveis do sistema e o sistema tenta obter comportamentos com um alto (globalmente máximo, se possível) nível de satisfação das preferências. As restrições têm a ver tanto com a execução de acções (como no escalonamento clássico), como com estados estacionários (como no planeamento de acções clássico de IA). De um modo geral, no processo de planeamento e escalonamento do HSTS, o que se faz é identificar um conjunto de comportamentos legais do sistema, *i.e.*, uma espécie de "envelope" de comportamento, mantendo flexibilidade temporal. Ao representar explicitamente alternativas possíveis ao curso de acções desejado, o sistema pode ter uma perspectiva clara dos impactos de desvios desse curso de acções e pode reagir, com um desempenho aceitável, mantendo-se dentro do "envelope" de comportamento. [51] É usada, para isto, uma metodologia de escalonamento heurística, denominada *escalonamento por partição de conflitos* (*conflict partition scheduling* ou CPS) [Muscettola 1994a], [Muscettola 1994b], que opera numa rede de restrições temporais flexíveis sob a orientação de estimativas estatísticas das propriedades da rede. Em [Muscettola 1994b] esta metodologia é descrita e são mostrados resultados de testes experimentais para várias possíveis configurações do CPS. Em [Muscettola 1994a] incluem-se também resultados de testes experimentais em que a heurística CPS é favoravelmente comparada a outras heurísticas que não seguem o mesmo princípio de flexibilidade temporal.

O HSTS é composto por dois componentes nucleares: o HSTS-DDL (HSTS *domain description language*) e o HSTS-TDB (HSTS *temporal database*). O primeiro componente, HSTS-DDL, serve para descrever modelos de domínio de planeamento/escalonamento e incorpora uma linguagem de descrição do domínio que permite, no contexto de um problema particular, a especificação da estrutura estática e dinâmica de um sistema. Dá suporte à expressão de um modelo como um conjunto modular de padrões de restrições a satisfazer por qualquer comportamento legal do sistema. Um *modelo de sistema* é organizado como um conjunto de *componentes de sistema*, cada um associado a um *conjunto de propriedades* sendo cada uma, uma entrada no vector de estado, que pode assumir um só *valor* em cada instante. Há *propriedades estáticas*, cujo valor é fixo no tempo (são tipicamente parâmetros do sistema) e *propriedades dinâmicas*, designadas por *variáveis de estado*, cujo valor é variável no tempo e que determinam o comportamento do sistema. A especificação de um modelo de

---

e controlo coordenado das suas actividades (ver, por exemplo, [Christopher 1993], [Thomas 1996] ou [Vollmann 1997] no Capítulo 19). [Morton 1993], em particular, entende que o escalonamento pode ser visto a vários níveis com horizontes temporais de diferentes extensões, que incluem, o nível de escalonamento reactivo, ou de controlo, o nível tradicional do chão-de-fábrica, e níveis de planeamento de curto, médio e longo prazo.

[50] Um destes domínios é o do escalonamento a curto prazo para o telescópio espacial Hubble (produção de comandos detalhados para o telescópio), outro é o do planeamento dos transportes.

[51] Esta filosofia, que se traduz por restringir minimamente o espaço de soluções possíveis e atrasar o máximo a definição de uma solução, assemelha-se à da estratégia usada em planeamento de acções em IA apelidada de *planeamento com mínimo comprometimento*, ou *least commitment planning* (ver, por exemplo, [Rich 1991] ou [Russell 1995]).

sistema pode ser decomposta em vários *níveis de abstracção* relacionados entre si, em que os componentes de sistema e variáveis de estado em níveis mais abstractos agregam os de níveis mais detalhados. As relações entre níveis são estabelecidas por descritores de refinamento que mapeiam alguns valores abstractos para uma rede de valores associada ao nível de maior detalhe seguinte (incluindo, por exemplo, correspondência entre tempos de início e fim).

O outro componente, HSTS-TDB, permite a construção dos comportamentos legais de um sistema através de uma base de dados temporal que usa representações do tipo *mapa temporal* (*time map*) [Dean 1987], [Schrag 1992] com extensões apropriadas. O *símbolo* (*token)* é a primitiva de descrição temporal e consiste num intervalo de tempo, identificado pelo *tempo de início* e o *tempo de fim*, no qual uma determinada condição é verdadeira. Representa um único segmento de evolução de uma única variável de estado. Uma *rede de símbolos* é o conjunto de *símbolos* juntamente com restrições entre eles, na base de dados temporal. É estabelecendo restrições entre pares de *símbolos* que se asseguram as condições necessárias na resolução de um problema. As restrições impostas nos *símbolos* advêm do próprio problema (por exemplo, uma data de lançamento na ocorrência de uma actividade) e do modelo de sistema em HSTS-DDL (por exemplo, especificações de compatibilidade que requerem a ocorrência de um padrão relacionado de actividades de suporte). Uma *rede de símbolos* é subdividida em *níveis* comunicantes, correspondentes no HSTS-DDL, a níveis de abstracção do modelo de sistema. Existem primitivas para criar e inserir símbolos e criar instâncias de *relações temporais*, bem como para removê-los. A consistência entre *símbolos* da rede é mantida através de uma rede auxiliar, do tipo *rede de pontos temporais* (ver [Dechter 1991]), em que os nós representam tempos de início ou de fim de um *símbolo* e os arcos representam restrições temporais que impõem distâncias temporais entre dois pontos; estas distâncias são derivadas das relações temporais colocadas na base de dados temporal. Uma *linha temporal* (*time line*) é uma sequência linear de *símbolos* que cobre completamente o horizonte de escalonamento para uma variável de estado. Num plano/escalonamento completamente especificado cada *linha temporal* contém uma sequência de *símbolos* todos com valores fixos.

Durante a resolução de um problema o HSTS vai colocando restrições de precedência que impõem sequência temporal nos *símbolos* que requerem o mesmo recurso. O objectivo é colocar restrições em número suficiente de modo a que a capacidade requerida de cada recurso não exceda a sua capacidade disponível. Isto é oposto ao método adoptado no escalonamento clássico, em que se associam valores (recursos e tempos de início e fim) exactos às variáveis (tarefas).[52] Este método tem vantagens sobre a atribuição de valores exactos, pois quando estabelece uma restrição apenas restringe o intervalo de valores possíveis das variáveis, sem limitar desnecessariamente a dimensionalidade do espaço de procura; isto permite manter um grande número de valores possíveis para as variáveis e diminuir o risco de o sistema se perder na procura.

O CPS [Muscettola 1994b] é um procedimento de colocação de restrições que segue o princípio de flexibilidade temporal. Partindo de um estado inicial com a uma rede de *símbolos* (cada um correspondendo a um pedido de capacidade e nenhum deles ainda inserido na correspondente *linha temporal* de recurso), este procedimento tem como objectivo juntar restrições à rede de *símbolos* de modo a que, a inserção de todos eles não origine conflitos de

[52] O método proposto, como referido em [Muscettola 1994a], pode contrapor-se ao método típico usado tanto nas abordagens construtivas como nas de reparação (ver a secção 2.2.2.6), *i.e.*, atribuição de valores (tempos e recursos) bem determinados às variáveis (tarefas) do problema. Na prática (e como se pode deduzir pela descrição que a seguir se faz do procedimento CPS), o método proposto corresponde a uma progressiva definição das sequências de tarefas nos recursos.

capacidade. Para isto, repetidamente, realiza uma análise de capacidade para identificar conjuntos de *símbolos* congestionados (os que têm alta probabilidade de competir pelo uso de um recurso) e impõe-lhes restrições de precedência para assegurar que não surge nenhum conflito; a identificação destes conjuntos e a determinação de quais as restrições mais favoráveis é feita através de uma metodologia de análise do espaço de procura baseada em simulação estocástica. Os passos do algoritmo deste procedimento são os seguintes:

1. Análise de capacidade - Estima a procura de *símbolos* e a competição pelos recursos. A procura de um *símbolo* mede em quanto as restrições e as preferências actuais fazem tender o tempo de execução de uma tarefa para um determinado tempo. A competição por um recurso mede em quanto as restrições e as preferências actuais gerarão congestionamentos de pedidos de capacidade (e uma inconsistência potencial, portanto) num determinado tempo;
2. Teste de terminação - Se a competição por cada recurso é zero, em todo o horizonte de escalonamento, termina com a rede de *símbolos* actual como solução;
3. Detecção de congestionamentos - Identifica o recurso e tempo com maior competição (porção da rede de pontos temporais onde há maior probabilidade de conflitos de capacidade);
4. Identificação de conflitos - Selecciona os *símbolos* que mais podem contribuir para o congestionamento (os que potencialmente estarão em conflito);
5. Partição de conflitos - Ordena o conjunto de *símbolos* (de acordo com a procura de *símbolos*) inserindo as restrições de precedência temporal apropriadas, de modo a diminuir a possibilidade de inconsistências na rede de *símbolos*. Procura-se que haja um equilíbrio entre introduzir uma simples restrição de precedência entre pares de *símbolos* do conjunto conflito (o que seria uma abordagem minimal, semelhante à do escalonamento micro-oportunista) e impor uma ordem total entre todos os *símbolos* do conjunto conflito (uma abordagem semelhante à do escalonamento macro-oportunista), para haver um equilíbrio entre a minimização da alteração da topologia da rede de *símbolos* e minimização do número de ciclos de resolução do problema.[53] Para que haja este equilíbrio, é empregue uma estratégia em que se particiona o conjunto conflito em dois subconjuntos e se restringe cada *símbolo* de um dos conjuntos de modo a ocorrer antes de qualquer *símbolo* no outro conjunto;
6. Propagação de restrições - Actualiza os limites temporais na rede de pontos temporais (como consequência da introdução dos novas restrições temporais);
7. Teste de consistência - No caso de a rede de pontos temporais ficar inconsistente assinala isso e termina;
8. Continua em 1.

Se o procedimento terminou saindo por 7 a rede de *símbolos* é reposta no estado inicial e o procedimento repetido. O caminho explorado será diferente dada a natureza estocástica da análise de capacidade. No entanto, o CPS terminará por falha se ao fim de um certo número de repetições fixo não se encontrar a solução.

Para concluir a presente secção, pode dizer-se que as abordagens construtivas ao escalonamento baseadas em satisfação restrições podem dividir-se nas que, atribuem valores à

[53] Introduzindo demasiadas restrições em cada ciclo podem-se gerar inconsistências e aumentar o retrocesso, mas introduzindo poucas, é necessário um maior número de passos de análise de capacidade.

variáveis desde o início da resolução do problema e as que procuram não se comprometer desde o início com atribuições [Liu 1993]. No primeiro tipo de abordagens emprega-se normalmente um tipo de estratégia de procura heurística oportunista e constrói-se o escalonamento focando, de modo oportunista, a atenção em partes promissoras do espaço de estados (por exemplo, recursos congestionados) e atribuindo um valor a uma variável de cada vez desde o início. Após uma atribuição de valor a uma variável propagam-se restrições para identificar possíveis violações de restrições, que são resolvidas ou por retrocesso ou por relaxação de restrições (o que implica, normalmente, custos computacionais elevados [Liu 1993]). No segundo tipo de abordagens, frequentemente apelidado de *colocação de restrições* (*constraint posting*), ciclicamente, analisa-se a configuração actual da solução e introduzem-se (colocam-se) restrições adicionais de modo a excluir conflitos de capacidade, deduzindo depois atribuições de valores a variáveis a partir da rede de restrições resultante. ISIS, OPIS e Micro-Boss são exemplos de sistema empregam o primeiro tipo de abordagem; a heurística de partição de conflitos do sistema HSTS é uma abordagem do segundo tipo. Outros exemplos de trabalhos em que se preconiza a colocação de restrições são [Cheng 1995], [Cesta 1998a] ou [Cesta 1998b], estes dois últimos com a particularidade de tratarem problemas de escalonamento envolvendo recursos de capacidade múltipla.

2.2.2.6 Escalonamento por Reparação

A generalidade dos sistemas de escalonamento descritos até aqui, na secção 2.2.2, usam um método construtivo de escalonamento, que se traduz na extensão incremental de uma solução parcial. Numa versão inicial o sistema de escalonamento GERRY empregava o mesmo tipo de abordagem, utilizando um método complementar baseado em retrocesso guiado por dependências. A aplicação deste sistema ao problema de escalonamento de tarefas em terra para o *vai-vém* (*space-shuttle*) da NASA (escalonamento de reparações e da manutenção entre voos) é descrita em [Zweben 1994b]. Segundo os autores, a versão inicial não era suficientemente poderosa para manipular restrições de estado, preempção e capacidades de re-escalonamento, essenciais para aquela aplicação da NASA. Deste modo, o sistema evoluiu para um método baseado em *reparação iterativa*, baseado no método de arrefecimento simulado, em que se opera por modificação iterativa de escalonamentos completos, com ênfase no re-escalonamento e no escalonamento preemptivo.

A reparação iterativa tem obviamente as desvantagens de o processo de procura da solução poder ficar preso em mínimos locais, perdendo-se em ciclos numa série de soluções não satisfatórias e de não garantir encontrar a melhor solução. No entanto, a opção pela reparação iterativa tem, no caso desta aplicação, várias vantagens. Uma destas é a do problema ser basicamente um problema de re-escalonamento. Para re-escalonar usando um método construtivo, o sistema teria de remover algumas tarefas do escalonamento e reiniciar o processo de escalonamento, pondo-se o problema de determinar que tarefas remover. Com o método de reparação não é necessário remover tarefas do escalonamento para re-escalonar (as tarefas permanecem sempre escalonadas). Outra vantagem tem a ver com oportunidades de re-escalonamento que podem perder-se se as tarefas não directamente afectadas pela necessidade de re-escalonamento não são consideradas; por exemplo, colocar uma tarefa não afectada mais tarde no escalonamento pode causar pouca perturbação e permitir que muitas outras, afectadas, fiquem bem colocadas. Na reparação iterativa isto não acontece porque todas as tarefas são envolvidas no processo de re-escalonamento. Outra vantagem tem a ver com o facto de ser normal os problemas de escalonamento do *vai-vém* serem super-constrangidos. Com um método construtivo de escalonamento o sistema deveria testar todas

as hipóteses, antes de se concluir que devem relaxar-se restrições. Os métodos baseados na reparação tentam iterativamente melhorar soluções e terminam com uma solução tão próxima do óptimo quanto seria possível produzir no tempo dado. Por último, havendo restrições globais[54] e critérios de optimização eles são melhor e mais eficientemente avaliados com métodos de reparação, pois a procura da solução ocorre no espaço de escalonamentos completos. Com um escalonamento parcial a avaliação de um critério global só pode ser aproximada. Para além destas características, a reparação iterativa segue um processo *a-qualquer-tempo* que, ao ser terminado ao fim de qualquer iteração, fornece sempre uma solução completa (mesmo que esta solução viole algumas restrições) e, além disso a solução actual é, em princípio, melhorada de iteração para iteração.

No GERRY há restrições de precedência temporal que ordenam as tarefas umas em relação às outras (são possíveis ordenações entre tempos de início, de fim, de início e de fim e de fim e de início, bem como atrasos positivos e negativos na ordem). É usado o algoritmo de Waltz para obter consistência temporal (consistência de arco, neste caso, ver o Apêndice C). Quando há que re-escalonar, ou quando há violação de restrições, deslocam-se tarefas no tempo até que as restrições sejam satisfeitas. Há também restrições temporais de datas limite (apelidadas *milestones*) que relacionam as tarefas com valores de tempo fixos, para que elas não sejam deslocadas para lá de certas datas. Para cada requerimento de recurso GERRY declara automaticamente uma restrição de capacidade, que impõe não dever existir sobre-afectação do recurso durante o intervalo de duração do requerimento. Os recursos são organizados em *classes de recursos*, representado cada classe um conjunto de *repositórios de recursos* (*resource pools*). Cada repositório de recursos é uma colecção de recursos individuais indistintos a que é associada uma *capacidade inicial* e uma *história* [Williams 1986], mantida para monitorar a disponibilidade do repositório ao longo do tempo. Para cada tarefa há um conjunto de *requerimentos de recursos*, cada um indicando o tipo (classe de recurso que satisfaz o requerimento) e quantidade de recurso. Podem modelar-se também *atributos de domínio*, cada um com estados (valores) possíveis. Cada tarefa tem um conjunto de *efeitos de estado*, que a tarefa impõe nos atributos, cada um com a duração do intervalo de tempo especificado pelo efeito (este intervalo é aberto em caso de persistência indefinida do efeito). Há *requerimentos de estado* que originam *restrições de estado* sobre as tarefas. Para cada atributo é mantida uma *história* (semelhante à história de um repositório de recursos) registando o valor actual do atributo, as tarefas modificadoras (aquelas cujos efeitos alteraram o valor do atributo até ao presente) e utilizadoras (as que têm, e fizeram uso de, um requerimento de estado para o atributo até ao presente) e a última tarefa modificadora do atributo, para cada intervalo de tempo do horizonte de escalonamento, formando uma rede de dependências.

Cada tarefa é associada a um calendário de períodos legais de trabalho e é dividida num conjunto de sub-tarefas. Normalmente, o escalonamento preemptivo impõe uma sobrecarga computacional, porque têm de se calcular os tempos de preempção e realizar a manipulação apropriada das restrições. As restrições de recurso e de estado podem ser impostas, ou durante cada sub-tarefa individual, sendo ignoradas durante os períodos de suspensão entre as sub-tarefas (é o caso de, por exemplo, mão-de-obra que está disponível durante os intervalos entre sub-tarefas), ou durante o intervalo de tempo que se estende da primeira à última sub-tarefa, caso em que deverão ser satisfeitas durante todo intervalo de tempo abrangendo a tarefa (é o caso de, por exemplo, maquinaria pesada, que sendo difícil de movimentar, quando

[54] Uma restrição global é uma restrição que envolve a solução completa. Por exemplo, manter a utilização de mão-de-obra aos fins de semana abaixo de um certo limite seria uma restrição global.

é posicionada num certo local, está indisponível para realizar tarefas noutros locais durante os intervalos entre sub-tarefas).

O GERRY usa como método de reparação iterativa o arrefecimento simulado (ver [Kirkpatrick 1983], [Van Laarhoven 1988], [Aarts 1989]). Um escalonamento inicial completo, de qualidade inferior, é modificado iterativamente, por reparação progressiva de restrições violadas, até obter uma qualidade satisfatória, ou até ser excedido um certo tempo limite para gerar a solução. Para medir a qualidade de uma solução `s`, é usada uma função de custo (avaliada em cada iteração) dada por:

$$cost(s) = \sum_{c_i \in \mathcal{C}} p_{c_i}(s) * w_{c_i}(s)$$

em que c_i é uma restrição (do conjunto de restrições $\mathcal{C}$), p_{c_i} é uma função que produz um número não negativo indicando o grau de violação de uma restrição na solução s e w_{c_i} é uma função que dá a importância, ou utilidade, de uma restrição.

Durante o processo de procura da solução são memorizadas a melhor solução (*i.e.*, de menor valor da função de custo) encontrada até ao momento e a solução actual. Em cada iteração do algoritmo de arrefecimento simulado, se a solução gerada é melhor que a gerada na iteração anterior, ela passa a ser a solução actual e é memorizada para a próxima iteração; em caso de a solução ser também a melhor encontrada até agora é também registada com tal. Em caso de a solução não ser melhor que a da iteração anterior poderá ser aceite ou rejeitada como solução actual, dependendo de um valor de probabilidade que decresce, num padrão predefinido, com o número de iterações (a possibilidade de aceitar soluções piores existe para permitir escapar de mínimos locais e ciclos, embora diminua com o número de passos já executados). Esta probabilidade é dada por uma função:

$$escape(s, s', T) = e^{-|cost(s)-cost(s')|/T}$$

Sendo `s` e `s'` duas soluções geradas em iterações seguidas e `T` o parâmetro da temperatura do arrefecimento simulado, que é gradualmente reduzido durante a procura. Quando uma solução obtida é pior, ela é aceite se um número aleatoriamente gerado, entre `0` e `1`, é menor que o valor da função `escape`.

A resolução/reparação de restrições violadas é feita através de reparações locais invocadas em cada iteração do ciclo de reparação iterativa, reparando-se sucessivamente um certo número de violações de cada tipo de restrição; isto permite reparar de forma focada num certo tipo de restrição. Há uma forma de reparação para cada tipo de restrição violada. Cada reparação é feita sem preocupações de interferência com outras restrições (podendo produzir estados globalmente não satisfatórios que, se aceites, são em geral, melhorados após algumas iterações). Em geral reparar uma violação de restrição envolve re-escalonamento de uma tarefa, sendo deslocadas, se necessário, as tarefas antecessora, ou sucessora, para respeitar restrições temporais.

Para cada tarefa envolvida numa sobre-afectação de recurso considera-se deslocá-la para um tempo mais cedo, ou mais tarde, mais próximos, em que o recurso esteja disponível. É atribuído um valor heurístico a cada deslocamento candidato que, convertido numa probabilidade, permite escolher o deslocamento de tarefa a realizar. O valor heurístico tem em conta os seguintes factores de preferência: a) deslocar a tarefa com requerimento de recurso mais parecido com a quantidade de sobre-afectação, b) deslocar a tarefa com o menor número de tarefas temporalmente dependentes, e c) deslocar a tarefa que leva a um deslocamento temporal menor.

Na reparação de restrições de estado, quando há conflito entre o valor do atributo requerido para uma tarefa e o valor presente, considera-se interpor antes da tarefa uma tarefa especial que estabelece o valor de atributo requerido, desde que isso não cause mais violações de restrições de estado, ou então deslocar a tarefa para um tempo mais cedo, ou mais tarde, para um valor de tempo em que o valor necessário para o atributo existe ou, em último caso, combinarem-se os dois métodos, desde que não sejam originadas violações de restrições de estado adicionais.

Em [Zweben 1994b] descrevem-se testes experimentais e conclui-se que a arquitectura de GERRY é a de um sistema de escalonamento poderoso para o tipo de problemas que se propõe resolver (os testes demonstram que se comporta bem mesmo para problemas grandes). Para além disso é um sistema relativamente independente do domínio pois, para incorporar uma restrição nova basta apenas o utilizador definir as funções componentes da função de custo, podendo ainda, opcionalmente, juntar heurísticas de focagem que permitem a ordenação das restrições para reparações.

Exemplos de outros sistemas que usam métodos de escalonamento por reparação iterativa são CABINS [Miyashita 1994], [Sycara 1994] e SPIKE [Johnston 1994b]. O primeiro usa CBR para aquisição e reutilização de preferências de escalonamento e escolha de acções de reparação de escalonamentos e foi já descrito noutra secção. O segundo foi desenvolvido para a NASA para escalonar pedidos de observações para o telescópio espacial Hubble. Uma heurística de reparação usada pelo SPIKE é a heurística designada por *mínimo-de-conflitos* (*min-conflicts*) [Johnston 1994a], que se baseia na atribuição de um valor que minimize o número de conflitos a cada variável envolvida num conflito (uma heurística de escolha de valor). Esta heurística é usada após a geração de uma solução inicial, num processo iterativo do tipo do subida do monte, em que a solução é progressivamente reparada. Tanto análises teóricas como resultados experimentais (ver [Johnston 1994a]) mostram que esta heurística pode ser muito eficaz na reparação de escalonamentos. Segundo [Johnston 1994b] melhorias adicionais obtêm-se combinando a heurística *mínimo-de-conflitos* com uma heurística de escolha de variável apelidada de *max-conflicts*, que se traduz, na prática, por escolher para reparação a tarefa envolvida no maior número de conflitos.

2.2.2.7 Escalonamento Distribuído e Multi-Agente

Os sistemas descritos nesta secção abordam o problema do escalonamento segundo o paradigma da IAD. Estes sistemas caracterizam-se, em geral, pelo facto de a informação associada aos objectos do domínio de escalonamento não estar centralizada numa única entidade, ou as decisões que a actividade de escalonamento envolve serem tomadas de forma descentralizada, ou então verificam-se ambas as situações.

O sistema DAS (*Distributed Asynchronous Scheduler*) [Burke 1991], [Burke 1994], opera num ambiente de produção industrial de chapa de alumínio para os mercados aeroespacial e de defesa (Alcan Plate Ltd., U.K.). Este sistema dá ênfase ao aspecto de modelação e representação do ambiente fabril e baseia-se em enquadramentos para representar os recursos, as tarefas e os planos dos processos de produção. O enquadramento de cada plano contém o conjunto de tarefas do plano e a sequência para o lançamento das tarefas no escalonamento. O enquadramento de cada tarefa representa restrições temporais (basicamente as relações de precedência com outras tarefas, a data de lançamento e a data limite) e tecnológicas (lista de recursos capazes de executar a tarefa), bem como atributos para conter valores resultantes do escalonamento (tempo de início e recurso atribuído). Os recursos são representados em três níveis hierárquicos: o nível operacional, em que se representam recursos individuais

(máquinas), o nível táctico, em que se representam recursos agregados (conjuntos de máquinas com idênticas funcionalidades) e o nível estratégico, que tem a ver essencialmente com planos de processo e está no topo da hierarquia. Cada objecto em cada um dos níveis é representado por um enquadramento com valores particulares de atributos, e cada um destes enquadramentos está associado a um *agente inteligente* que tem a seu cargo a tomada de decisões específicas do respectivo nível de hierarquia. Quando há uma encomenda nova, o agente do nível estratégico (existe um único agente, neste nível) escolhe o plano de processo apropriado, instancia-o e delega em agentes do nível táctico a afectação dos recursos apropriados às tarefas do processo; cada um destes agentes, por sua vez, delega o escalonamento de cada tarefa num recurso nos agentes do nível operacional, que lhe são subordinados. A hierarquia de representação dos recursos serve, em boa parte, para subdividir progressivamente o problema de escalonar as tarefas para a produção de uma encomenda em subproblemas mais simples.

O esforço de resolução do problema é distribuído por *agentes inteligentes* cooperantes, associados a entidades no ambiente fabril, hierarquizados, em três níveis — os níveis *estratégico*, *táctico* e *operacional* — de forma a reflectir a organização onde se inserem e a estrutura do problema tratado. Como procura manter um escalonamento satisfatório num ambiente aberto em tempo real, o DAS pode ser visto como um sistema do tipo *a-qualquer-tempo*. O problema de escalonamento é visto na perspectiva do PSR e os aspectos de escalonamento predictivo e reactivo estão integrados. O objectivo maior é satisfazer as restrições temporais e só em caso extremo de impossibilidade se opta pela relaxação destas restrições.

O agentes comunicam entre si através de mensagens. A maior parte da comunicação passa através de um subsistema denominado CMS (*constraint management system*), o sistema de gestão de restrições, que mantém o domínio temporal de cada tarefa do problema e informa os agentes apropriados quando os seus problemas locais sofreram alteração. Através das mensagens recebidas os agentes mantêm uma visão do mundo no qual podem impor mudança por meio das mensagens que enviam. O CMS é o meio principal de comunicação através de mensagens, que endereça apenas aos agentes a que é adequado. Pode haver mensagens para notificar um agente de que o domínio de uma tarefa do seu problema foi alterado, que uma tarefa foi adicionada ou removida, ou que uma decisão tomada relativamente a uma tarefa (atribuição de tempo de início).

Sempre que há uma encomenda nova o agente do nível estratégico, apelidado *S-Agent*, instancia um plano de processo adequado e delega o escalonamento de cada tarefa do processo num agente de nível táctico, ou *T-Agent*, apropriado (o *T-Agent* que gere o grupo de recursos adequados para execução da tarefa). Os *T-Agent*s tentam satisfazer restrições tecnológicas de tarefas associando-as a recursos e vêm uma tarefa como uma variável com um domínio que é o conjunto dos recursos possíveis para executar a tarefa. Cada *T-Agent* tem associado um assistente táctico, ou *T-Assistant*, que é um *sistema de manutenção de verdade baseado em assumpções* (*assumption based truth maintenance system*), o qual representa as tarefas do *T-Agent* respectivo na forma de variáveis cujos valores possíveis são recursos. Cada *T-Agent* obtém do *T-Assistant* associado o conjunto de recursos em que a tarefa a escalonar pode ser executada sem gerar conflitos a nível operacional e informa o *T-Assistant* da escolha do recurso, delegando a definição do intervalo de tempo de execução da tarefa num agente operacional, ou *O-Agent*, adequado (o *O-Agent* que gere o recurso individual escolhido). O que cada *O-Agent* faz é tentar colocar a nova tarefa no seu *escalonamento local*, tentando manter um escalonamento satisfatório no seu recurso. Assume-se sempre que está lidando com conhecimento incerto/incompleto, devido à natureza assíncrona do sistema DAS e à dinâmica

do ambiente. Cada *O-Agent* representa o seu problema através de um grafo de restrições cujos nós representam tarefas, sendo o domínio de cada uma um conjunto de intervalos de tempo (o conjunto de tempos de início possíveis da tarefa), e cujos arcos representam restrições binárias. O escalonamento das tarefas num recurso é entendido como um PSR dinâmico, em que se assume uma topologia mutável do grafo de restrições. Os *O-Agent*s não se preocupam com a origem das mudanças, *i.e.*, se elas são devidas a efeitos exógenos ou endógenos. O *O-Agent* usa um algoritmo híbrido [Prosser 1991], que combina *verificação para a frente* (*forward checking*) [Haralick 1980] com *aprendizagem superficial* (*shallow learning*) [Dechter 1990] e *retrocesso guiado por dependências* (*dependency directed backtracking*) [Stallman 1977] (ver apêndices B e C), ampliado para reagir a mudanças impostas ao agente. Um passo de procura no espaço de estados envolve: a) escolha de uma tarefa não escalonada, b) atribuição de um tempo de início do domínio temporal da tarefa escolhida, c) aplicação de *verificação para a frente* a partir da tarefa para todas as tarefas não escalonadas e adjacentes à primeira no grafo de restrições, com a consequente remoção de todos os valores do domínio de cada uma destas últimas tarefas que são inconsistentes com o valor atribuído à primeira (os valores removidos ficam explicitamente representados juntamente com uma justificação, o que inclui aquela tarefa e o valor atribuído). Em caso de sucesso o *O-Agent* devolve ao *T-Agent* superior um tempo de início para a tarefa, desencadeando assim a propagação de restrições através do plano do processo e originando mensagens enviadas a outros agentes. Nesta altura não há nenhuma ordenação das decisões sobre o processo e as decisões podem ser feitas assincronamente. Em caso de insucesso (*i.e.*, se o domínio de uma tarefa ficar vazio) o *O-Agent* devolve um *conjunto conflito* (tarefas que crê não poderem ser escalonadas de forma consistente no seu recurso) ao *T-Agent* superior. Este procura então equilibrar a carga dos recursos do grupo de recursos por ele geridos. Conjuntos conflito são representados nos *T-Assistant*s como conjunções ilegais de assumpções (apelidadas de *nogoods*). Cada *T-Assistant* dá também apoio ao *T-Agent* associado na resolução de conflitos. Em caso de não restar opção alguma para equilibrar a carga e resolver o conflito, o *T-Agent* conclui que o problema está super-constrangido e devolve ao *S-Agent* um conjunto conflito que envolve tarefas em vários recursos, com os conjuntos alternativos de tarefas que devem ser simultaneamente relaxadas para eliminar o conflito. O *S-Agent* analisa os conjuntos conflito dos *T-Agents* subordinados e decide-se por retrocesso inter-agente (retrocesso guiado por dependências entre *T-Agent*s) ou, em último caso, se isso não for suficiente, relaxação das datas limite. No caso de retrocesso considera várias hipóteses de anulação do escalonamento de outras tarefas, de modo a alargar o domínio temporal da tarefa a relaxar, para que seja possível escaloná-la. Para evitar ciclos infinitos na ordem de tomada de decisão neste processo de retrocesso o *S-Agent* regista a sequência de decisões tomadas. No decurso destes acontecimentos é obtido novo conhecimento do problema ao nível operacional, táctico e estratégico. O retrocesso significa uma oportunidade para aprender acerca do problema e quando acontece, o sistema extrai informação sobre os caminhos de procura sem saída. Cada agente (a qualquer nível da arquitectura) tem capacidades reactivas, de retrocesso guiado por dependência e de aprendizagem, sendo a capacidade de aprendizagem a que permite reter conhecimento sobre o espaço de procura. A representação usada para recursos e tarefas permite ao sistema representar todas as oportunidades que restam no escalonamento. O sistema não diferencia alterações induzidas pelo exterior, ou pelo interior, bem como não diferencia escalonamento predictivo de escalonamento reactivo (a reacção é entendida como uma continuação da procura da solução). Quando o retrocesso inter-agente falha (*i.e.*, não há escalonamentos de tarefas para anular) o *S-Agent* recorre à relaxação de datas limite. O escalonamento é desfeito para todas as tarefas do processo e a seguir elas são re-escalonadas do início para o fim do plano do processo.

No CMS, a propagação de restrições é apoiada por uma rede de restrições em que se representam restrições unárias (como tempo de início, recurso, data de lançamento) e restrições binárias (relações de precedência) que, combinados, dão origem ao *domínio temporal* (tempos de início válidos) de cada tarefa. Cada restrição temporal é composta por uma colecção de restrições componentes (de cuja intersecção se obtém uma representação resultante), cada um com a forma `(Source (S E))`, em que `Source` identifica a origem da restrição componente e `S` e `E` os tempos de início e fim, respectivamente, do intervalo de tempo permitido. Quando uma decisão é desfeita, as restrições componentes a retrair podem ser identificadas através do conteúdo de `Source`. Também, na resolução de conflitos, a representação das restrições temporais componentes permite a identificação de restrições unárias em conflito pelo *S-Agent*. A propagação de restrições processa-se através das restrições binárias e é originada por uma mudança no domínio temporal de uma tarefa, resultante da alteração de uma restrição unária. Devido à natureza assíncrona da resolução do problema e à dinâmica do ambiente a *retracção de restrições* é usual em DAS. Este processo de retracção é incremental, ao contrário de muitos algoritmos existentes (como, por exemplo, o proposto por [Mackworth 1985]), que recalculam a rede do princípio quando há uma retracção.

O micro-oportunismo de DAS é conseguido através de um mecanismo de prioridades de tarefas. Cada tarefa tem um atributo denominado *prioridade operacional*, com valor inicial de 0, que é uma medida da dificuldade de satisfação das restrições temporais e tecnológicas associadas à tarefa (pode ser vista como uma representação do esforço despendido para chegar a uma decisão de escalonamento da tarefa, durante a resolução do problema). Cada agente foca a sua atenção, em primeiro lugar, nas decisões de escalonamento de tarefas com maior prioridade. Sempre que um conflito é comunicado para um nível acima na hierarquia de agentes a prioridade das tarefas do conflito é incrementada.

Para a sincronização dos níveis táctico e operacional pode haver dois tipos de regimes. No *regime disciplinado*, cada *T-Agent* espera até que os *O-Agents* subordinados tenham resolvido o problema delegado neles (ou chegado a uma situação de conflito), para então delegar neles mais trabalho. Neste regime não há assincronia mas o *T-Agent* obtém a informação completa vinda dos subordinados, sendo melhor para problemas super-constrangidos. No *regime indisciplinado*, o *T-Agent* mistura resolução de conflitos com delegação. Há neste regime um maior grau de assincronia, mas também uma maior possibilidade de perda de oportunidades na tomada de decisão; é um regime melhor para problemas pouco constrangidos. A sincronização dos níveis estratégico e táctico e a sincronização dos níveis táctico e operacional são semelhantes. Nos níveis táctico e operacional, o regime indisciplinado resulta, em princípio, num estilo de gestão super-reactivo originando perda de informação estratégica, enquanto que no regime disciplinado haverá perda de assincronia. Na actual realização de DAS encontrou-se um equilíbrio entre os regimes disciplinado e indisciplinado, originando a um trabalho do *S-Agent* com conhecimento "quase" completo. Para se conseguir este equilíbrio, só se permite que o *S-Agent* introduza novo trabalho no sistema quando todos os *T-Agents* já não tiverem trabalho para delegar nos *O-Agents* subordinados; isto não acontece necessariamente apenas quando os *T-Agents* e os *O-Agents* tenham atingido um estado final na resolução do seu problema, nem implica que o *S-Agent* trabalhe com informação completa.

Em [Custódio 1998] discutem-se formas de organização de uma sociedade de agentes para o controlo da produção. Estruturas hierárquicas, em que os agentes estão organizados em diferentes níveis de, são contrapostas a estruturas não hierárquicas em que a responsabilidade é descentralizada.

Em [Sycara 1991a] é descrito um sistema descentralizado baseado em agentes para escalonamento da produção, no contexto de um projecto de investigação denominado CORTES. A natureza inerentemente descentralizada das actividades a coordenar numa fábrica requer uma correspondente descentralização dos mecanismos de planeamento e controlo. No plano teórico, o interesse maior deste projecto é a investigação em técnicas de coordenação quando a informação disponível se degrada ou se torna incompleta e quando a necessidade de reagir de cada agente aumenta devido a conflitos com decisões de escalonamento de outros agentes. Os agentes do CORTES supervisionam a execução de diferentes processos de diferentes encomendas (neste aspecto, difere bastante do sistema DAS, descrito anteriormente). Adicionalmente, cada agente monitora a capacidade disponível de recursos (máquinas) que têm a seu cargo. Os agentes são programas que correm em computadores separados e podem comunicar através de mensagens via rede electrónica. Tanto em [Sycara 1991a] como em [Sycara 1991b] é descrito o protocolo de comunicação entre agentes. Os recursos da fábrica são partilhados pelos agentes: estes competem pela utilização dos recursos para execução das tarefas dos seus processos. Como é usual existem basicamente dois tipos de restrições: restrições temporais, incluindo de precedência entre tarefas e datas de lançamento e limite, e restrições de capacidade. A capacidade limitada dos recursos (máquinas com capacidade unitária) induz interacções entre processos diferentes, do mesmo agente e de agentes diferentes, que competem simultaneamente pelo mesmo recurso.

Neste ambiente de escalonamento distribuído cada agente tem um conhecimento limitado do ambiente, das restrições e das intenções de outros agentes (devido ao facto de a quantidade de informação que os agentes podem comunicar entre si ser limitada e também de tender a ficar rapidamente desactualizada com as decisões que os agentes tomam de forma assíncrona). Os agentes tomam decisões locais de afectação de recursos a tarefas específicas em intervalos de tempo específicos e o sistema como um todo deve chegar a soluções globais intercalando computação local com comunicação entre agentes. O objectivo é determinar escalonamentos não apenas possíveis mas também de boa qualidade, isto é, que optimizam um objectivo global (como, por exemplo, minimizar atraso nos processos, ou minimizar volume de trabalhos em curso) e ainda de uma forma eficiente (*i.e.*, minimizar o retrocesso na procura da solução). No projecto CORTES as duas preocupações, da qualidade da solução e da eficiência no processo de procura da solução, coexistem.

A metodologia para abordar o problema de escalonamento distribuído empregue no CORTES baseia-se num modelo de resolução distribuída de problemas, apelidado *procura heurística com restrições distribuída* (*distributed constrained heuristic search* ou CHS) formalizado em [Sycara 1991b], que evoluiu a partir de um modelo correspondente para o caso da resolução de problemas centralizado (ver, por exemplo, [Fox 1989], [Fox 1990]). Essencialmente, estes mesmos conceitos foram aplicados para o caso de escalonamento centralizado no sistema Micro-Boss, já anteriormente descrito.

Tal como acontecia no Micro-Boss, a cada tarefa é associada uma função de preferência que associa, a cada possível reserva para a tarefa[55] um valor de preferência. Estas preferências advêm de objectivos organizacionais globais como minimização de atrasos (para cumprir datas limite), minimização de avanços (para reduzir existências de produtos acabados), minimização de tempos de fluxo de processos (para reduzir o volume de trabalhos em curso) e outras

[55] Tal como acontecia no Micro-Boss, cada tarefa é vista como uma variável agregada, cujo valor é uma reserva possível isto é, um tempo de início e um recurso para cada requerimento de recurso da tarefa. Em [Sycara 1991a] considera-se o caso simples de cada tarefa requerer um único recurso (para o qual poderá haver várias alternativas).

preferências, como preferências de uma tarefa por uma certa máquina, ou por ser executada num certo turno. Num escalonamento, a soma dos valores das funções de preferência para todas as tarefas escalonadas por todos os agentes do sistema define o valor da função objectivo global. A soma dos valores das preferências das tarefas escalonadas por cada agente constitui a *visão local* do agente da função objectivo global (os agentes não conhecem a função objectivo global). Como os agentes competem pela utilização dos recursos partilhados, para cooperarem não é suficiente optimizarem apenas as suas preferências locais, sendo necessário terem também em conta as preferências dos outros agentes quando escalonam as suas tarefas.

Inicialmente, existem encomendas que originam processos. São então atribuídos processos a cada agente para escalonamento das respectivas tarefas. Para determinação de uma solução, cada agente, iterativamente, escolhe uma tarefa para escalonar e uma reserva para essa tarefa, construindo incrementalmente, e de forma assíncrona, um escalonamento para os processos que lhe foram atribuídos. Sempre que um agente escalona uma tarefa são criadas novas restrições no sistema que reflectem a nova reserva. Estas restrições são propagadas localmente para o agente e para toda a rede de agentes, *i.e.*, comunicadas aos outros agentes e propagadas localmente neles. Em caso de a propagação determinar que existe violação de restrições, o sistema executará retrocesso; no caso contrário, o agente prossegue escolhendo a próxima tarefa a escalonar e uma reserva para essa tarefa. Por uma questão de eficiência, o mecanismo de propagação de restrições procura apenas determinar dois tipos de inconsistência específicos: violação de restrições de precedência num processo e violação de restrições de capacidade entre um grupo de tarefas escalonadas e uma tarefa a escalonar. No caso específico das restrições de capacidade, as inconsistências envolvendo tarefas ainda não escalonadas só serão detectadas quando sistema escalonou todas as actividades em conflito menos uma. Pode assim acontecer que uma certa reserva decidida por um agente num certo momento venha a forçar os agentes a retroceder mais tarde.

O problema do retrocesso é aqui ainda mais crítico do que na versão centralizada do modelo por causa do custo adicional da comunicação entre agentes. Por essa razão, é importante que o sistema foque a procura de modo a reduzir o número de retrocessos e de decisões a serem desfeitas quando o retrocesso tem de ocorrer. Isto é conseguido à custa de técnicas heurísticas, nomeadamente heurísticas de ordenação de variáveis (para as tarefas), heurísticas de ordenação de valores (para as reservas) e um mecanismo de retrocesso (não cronológico) denominado *salto-para-trás* (*backjumping*, ver Apêndice C).

No que respeita às heurísticas de ordenação o processo é semelhante ao empregue no Micro-Boss e inclui, adicionalmente, passos de comunicação entre agentes. Para ordenação das tarefas, a cada tarefa não escalonada é atribuído um valor de criticalidade que mede, de forma aproximada, a probabilidade de a tarefa vir a estar envolvida num conflito (conflito este que não pode ser detectado de forma imediata pelo mecanismo de propagação de restrições). Ao escolher como próxima tarefa a escalonar a tarefa que tem maior valor de criticalidade, os agentes minimizam a probabilidade de estarem construindo escalonamentos parciais que não poderão completar e, portanto, reduzem a probabilidade de retrocesso.

Para obter os valores de criticalidade cada agente constrói um *perfil de procura agregado* que reflecte, para cada recurso que as suas tarefas requerem, o grau de necessidade dessas tarefas agente relativamente ao recurso ao longo do tempo. Os agentes comunicam entre si estes

perfis de procura agregados para cada recurso partilhado[56] e podem, então, calcular o perfil de procura agregado global do sistema para cada recurso partilhado. A criticalidade de uma tarefa é dada pela contribuição da tarefa para este perfil global de procura para um dado recurso. O recurso com o maior pico de procura é identificado como o recurso então congestionado e das suas tarefas, cada agente escolherá para próxima tarefa a escalonar a tarefa que mais contribui para esse pico.

Para escolha da reserva atribuir à tarefa seleccionada para escalonamento o sistema dispõe de um conjunto de estratégias que podem variar entre a estratégia do *valor menos limitativo* (*least constraining value* ou LCV), e uma estratégia *glutona*, ou *greedy value* (GV). No caso extremo em que a estratégia se baseia numa heurística do tipo LCV, cada agente tem um comportamento, mais altruísta e menos míope, em que tenta escolher a reserva que menos impede as outras tarefas, dele ou de outros agentes, de serem escalonadas. No outro caso extremo, baseado numa heurística do tipo GV, cada agente pode escolher reservas baseado apenas nas suas preferências locais e sem ter em conta necessidades futuras, dele ou de outros agentes. Segundo [Sycara 1991a] a experiência com sistemas centralizados indica que o uso de heurísticas do tipo LCV minimiza a quantidade de procura necessária, mas resulta em escalonamentos de fraca qualidade, pois a escolha das reservas é feita independentemente da sua contribuição para a função objectivo. Pelo contrário, o uso de heurísticas do tipo GV produz, em geral, melhores escalonamentos, mas à custa de mais retrocessos sendo, por isso, a procura mais demorada. A escolha da estratégia intermédia adequada dependerá de factores como o tempo disponível para encontrar uma solução, a carga nos agentes e a competição pelos recursos. Como é referido em [Sycara 1991a], é possível combinar a heurística de ordenação de valor tipo GV com a heurística de ordenação de variável de forma compensatória de modo a reduzir consideravelmente o retrocesso. No caso descentralizado, no entanto, os agentes actuam assincronamente, sendo então expectável que o efeito da ordenação das variáveis seja mais fraco. Também, o custo do retrocesso num sistema distribuído é maior, já que há uma sobrecarga adicional para coordenação entre agentes cujas decisões tomadas são dependentes umas das outras. Sendo assim, no caso do sistema distribuído haverá, em princípio, maior tendência para uma heurística do tipo LCV.

No que respeita ao retrocesso há que ter em conta que, num sistema distribuído, o comportamento assíncrono dos agentes pode invalidar a informação que eles trocam, por mais completa que ela pudesse ser. Mesmo usando uma estratégia LCV, esta estratégia é sempre heurística e não poderá impedir sempre que certos agentes estabeleçam reservas que venham a originar violações de restrições para outros agentes. Num sistema centralizado pode parecer mais natural atribuir-se a causa de uma violação de restrição à reserva efectuada mais recentemente, já que o sistema validou reservas previamente feitas e daí ser natural empregar o retrocesso cronológico (*chronological backtracking*). Num sistema multi-agente, como o do CORTES, os outros agentes podem efectuar reservas durante a procura de um agente, sendo mais difícil determinar que conjunto de reservas previamente efectuadas, pelo próprio ou por outros agentes, é responsável por uma violação de restrições detectada. Nestes casos é possível, e será frequente, que reservas feitas recentemente por outros agentes originem, juntamente com reservas feitas anteriormente pelo agente, um conflito. Para sair desta situação usando retrocesso cronológico pode ser necessário retroceder em várias decisões. Em vez de tentar todas as alternativas possíveis de reservas para a tarefa a escalonar actualmente e só depois, em caso de falha, retroceder para a última tarefa escalonada (como é feito no

[56] Os agentes apenas trocam informação da procura agregada de cada agente por cada recurso partilhado e não da procura de cada tarefa individual por cada recurso.

retrocesso cronológico) adopta-se um mecanismo específico de *salto-para-trás*, adequado à incerteza deste ambiente multi-agente. Neste mecanismo, o agente desfaz a reserva da tarefa actual e testa se a reserva que se efectuou imediatamente antes, juntamente com as dos outros agentes, se mantém possível. Em caso afirmativo, testa uma reserva alternativa para a tarefa actual; no caso contrário o agente segue um processo que intercala o desfazer de reservas de tarefas previamente escalonadas com teste de consistência até que encontre uma combinação de reservas dele e de outros agentes consistente [Sycara 1991b]. Este processo procura evitar testar reservas alternativas para a tarefa actual quando são as reservas das tarefas anteriormente escalonadas que contribuem para as violações de restrições.

Em [Liu 1993] e [Liu 1994] propõe-se uma metodologia apelidada de *partição de restrições e reacção coordenada* (*constraint partition and coordinated reaction*, CP&CR) no sistema CORA para escalonamento baseado em satisfação de restrições distribuída. O conjunto de restrições do problema de escalonamento (restrições de capacidade e restrições temporais de precedência e de datas limite) é dividido em subconjuntos de restrições de tipos diferentes. O sistema multi-agente é composto por conjuntos de agentes reactivos especializados em determinados tipos de restrições. Cada agente tem a responsabilidade de impor as restrições em que é especializado num conjunto de variáveis (tarefas) sob o seu controlo. Existem agentes de encomenda e agentes de recurso. Um agente de encomenda é responsável pelas restrições temporais das tarefas do processo que satisfaz uma determinada encomenda. Um agente de recurso é responsável por restrições de capacidade num recurso.

A resolução do problema começa com a criação dos agentes por um agente supervisor, a decomposição do problema e a distribuição das variáveis e restrições pelos agentes apropriados. A data de lançamento e a data limite para cada processo são, então, estabelecidas pelos agentes de encomenda e uma solução inicial é construída, a seguir, pelos agentes de recurso. Os agentes passam então a actuar num ciclo, controlado por um agente supervisor, em que cada agente revê a sua solução local actual (atribuições de tempos a tarefas) e reage, alterando-a, se detecta restrições que são violadas. O ciclo termina quando os agentes concordam numa solução, *i.e.*, nenhum agente reage à solução actual por não haver restrições que são violadas.

Cada agente desconhece a existência dos outros agentes e desconhece as restrições pelas quais não é responsável. Como uma tarefa pode estar sujeita a mais de uma restrição, o tempo associado à tarefa pode ser alterado por mais de um agente e uma alteração por um agente pode ocasionar violação de restrições de outros agentes. Por esta razão o sistema depende de informação de coordenação que deve ser trocada entre os agentes. É definida uma medida de intensidade de procura por recurso (baseada numa razão entre a soma das durações da tarefas no recurso e a duração do horizonte temporal de escalonamento no recurso) que permite aos agentes de recurso anunciarem-se como agentes de recurso congestionado e assinalarem as tarefas sob o seu controlo como tarefas de recurso congestionado.[57] Esta medida, juntamente com uma medida de folga temporal no recurso e indicações de alteração do tempo de cada tarefa por um agente de recurso, constituem informação de coordenação dos agentes de recurso para os agentes de encomenda. O intervalo entre o tempo de início mais cedo e o tempo de fim mais tarde de uma tarefa, a folga temporal entre uma tarefa e a tarefa seguinte de um processo e medidas que ponderam a possibilidade de uma tarefa originar violação de restrições se for re-escalonada (por exemplo, medidas de proximidade de uma tarefa

[57] Neste sistema multi-agente os agentes solucionadores do problema não comunicam, mas apenas interagem através do ambiente, associando informação às tarefas.

relativamente às tarefas de recurso congestionado mais próximas) constituem informação de coordenação dos agentes de encomenda para os agentes de recurso. Agentes de encomenda, agentes de recurso e agentes de recurso congestionado usam heurísticas para reagir de forma diferente de acordo com a informação de coordenação.

Em [Kjenstad 1998] descreve-se um enquadramento para coordenação operacional do escalonamento na cadeia de fornecimento baseado na tecnologia de IAD, adequado a um ambiente de produção por encomenda. Propõem-se também mecanismos e políticas que permitem uma melhor coordenação da cadeia, com base no escalonamento de recursos de capacidade finita.

Neste enquadramento vários agentes autónomos, responsáveis por diversos elementos da cadeia, cooperam para manter escalonamentos consistentes entre si. O enquadramento é um subconjunto de uma arquitectura mais abrangente, e ainda em desenvolvimento, apelidada de MASCOT (de *Multi-Agent Supply Chain Coordination Tool*; proposta pelo Intelligent Coordination and Logistics Laboratory da universidade de Carnegie Mellon), para coordenação em cadeias de fornecimento. Na arquitectura MASCOT os agentes estão organizados em vários níveis de abstracção, dentro de cada organização participante na cadeia. Agentes de níveis baixos são responsáveis por decisões de nível operacional, envolvendo instalações individuais num horizonte temporal de curto-médio prazo, e agentes de níveis altos são responsáveis por decisões de nível estratégico e táctico, envolvendo várias instalações de uma mesma organização. A coordenação na cadeia ocorre tanto na direcção horizontal, com interacção entre agentes similares, como na direcção vertical, com interacção entre agentes de níveis adjacentes. Tarefas são anunciadas aos participantes da cadeia de modo que eles possam fazer as suas propostas. Os agentes de alto nível recebem estes anúncios e negociam sub-tarefas como fornecedores, podendo, se necessário, também comunicar com os agentes de baixo nível da sua organização para uma validação mais precisa das decisões.

As políticas descritas em [Kjenstad 1998] dizem respeito à coordenação horizontal entre agentes do nível mais baixo, apelidados de *agentes de coordenação*. A coordenação horizontal ocorre por partilha de informação de restrições, tanto entre agentes de coordenação da mesma organização como de organizações adjacentes na cadeia. Os agentes de coordenação têm capacidades integradas de planeamento (selecção de planos de produção) e de escalonamento adequadas à produção ágil. Basicamente, um agente de coordenação encapsula uma nova realização computacional do sistema de escalonamento Micro-Boss [Sadeh 1991], [Sadeh 1994] (anteriormente referido).

Existe um mecanismo, dito *mecanismo de contexto*, para cada agente de coordenação, que permite manter estabilidade dos escalonamentos em vias de execução face a eventos inesperados. Um contexto, o *contexto lançado* (*released context*), representa o escalonamento que foi inicialmente enviado para execução, enquanto que o *contexto de trabalho* (*working context*) reflecte o resultado da coordenação em tempo real entre os agentes e da informação de retorno do chão-de-fábrica. Outro mecanismo empregue é o de *rede de produto*, que integra informação de estrutura de um produto e do plano de processo de produção do produto (tradicionalmente estes elementos estão separados). Por fim o mecanismo do *conector* permite ligar redes de produtos (instanciadas) entre agentes de coordenação ao longo de toda a cadeia de fornecimento. Este mecanismo pode ser usado para rastrear um lote ao longo do trajecto na cadeia, ou para identificar qual o lote de um agente que é afectado por um re-escalonamento realizado por outro agente, entre agentes de coordenação de organizações adjacentes na cadeia (cria portanto uma relação de um para um, entre cliente e fornecedor, relativamente a uma encomenda). É sugerida uma ampliação possível do sistema para

ambientes de produção para armazenamento, ou de montagem por encomenda, através de um *conector com atraso* (um conector temporário para uma possível encomenda futura).

Para além da interacção com entidades externas à cadeia de fornecimento é, em especial, definida a interacção entre agentes de coordenação, dentro da cadeia. Esta interacção ocorre por troca assíncrona de mensagens entre agentes de coordenação, cuja informação de conteúdo é incorporada nos contextos de trabalho dos agentes na forma de *resultados não resolvidos* (*unresolved issues*; indicam inconsistências em aspectos particulares da solução).

São descritas várias políticas de coordenação para a cadeia de fornecimento, tanto com agentes cooperativos como com agentes não cooperativos, ou auto-interessados. As políticas especificam quando, o que, e com que outro agente, um agente deve comunicar e como o agente pode usar a informação que recebe. Várias políticas de *just-in-time* (ordens de produção são escalonadas em *just-in-time* em toda a cadeia sem armazenamento de segurança e sem tempo de segurança), políticas de tempo de segurança (folgas temporais de segurança são introduzidas no escalonamento), políticas de recusa de oferta (assume-se que um participante só emite uma oferta se tiver benefícios superiores aos custos), políticas de negociação de data prometida (prevêm negociação de datas limite entre cliente e fornecedor e que o fornecedor faça ofertas mesmo tendo por certo que vai ter atrasos).

Por fim, são apresentados resultados de simulações para várias configurações de cadeia de fornecimento, incluindo um estudo da sensibilidade das várias políticas a mudanças nas condições do exterior.

Em [Rabelo 1997], e também [Rabelo 1994a], [Rabelo 1994b], [Rabelo 1996a], [Rabelo 1996b] e [Rabelo 1998a], propõe-se um enquadramento integrado do escalonamento com as restantes actividades de empresa, no contexto da produção industrial, como são o planeamento de processos, o planeamento da produção e a supervisão da execução. Este enquadramento, apelidado de HOLOS, baseia-se na modelação e integração da informação e da empresa, numa arquitectura multi-agente genérica de referência e numa metodologia e num sistema interactivo de suporte à geração de sistemas de escalonamento.

No que respeita ao escalonamento, o HOLOS dá suporte à geração de sistemas particulares de escalonamento para produção ágil, *i.e.*, sistemas integrados e flexíveis, capazes de acomodarem eventos inesperados (escalonamento dinâmico e reactivo) e adaptarem flexivelmente os recursos ao plano de produção. A arquitectura proposta para os sistemas de escalonamento gerados é baseada em quatro classes de agentes autónomos cooperantes e pressupõe integração com sistemas legados na empresa.

Cada tarefa é representada por uma árvore de processo de negócio (*business process*, BP) contendo, no nível mais baixo, tarefas denominadas actividades de empresa (*enterprise activities*, EA), juntamente com um conjunto de regras de precedência (é usado o modelo descrito em [AMICE 1993]).

A arquitectura de um agente HOLOS individual comporta dois módulos: o módulo de conhecimento e o módulo de cooperação. O módulo de conhecimento comporta componentes de conhecimento estático, dinâmico e de raciocínio (responsável pelo controlo do agente); o módulo de cooperação apoia a interacção do agente com o seu exterior (genericamente, outros agentes HOLOS, o sistema de informação, ou um utilizador, existindo um protocolo para cada possibilidade). Os agentes são realizados por *enquadramentos* (*frames*) com *atributos* (*slots*) e métodos associados.

As quatro classes de agentes são o agente *supervisor do escalonamento* (SE), o agente *centro local de distribuição* (CLD), o agente *actividade de empresa* (AE) e o agente *consórcio*. Os agentes, organizados hierarquicamente, usam uma metodologia de resolução distribuída de problemas; a coordenação baseia-se na comunicação por mensagens, com um protocolo tipo rede de contratos (ou *contract net*; ver o Apêndice D). Cada agente SE supervisiona globalmente um escalonamento, decompondo tarefas em sub-tarefas e construindo e distribuindo anúncios de sub-tarefas. A distribuição descentralizada de anúncios de tarefas (BPs e EAs) por agentes AE é delegada em agentes CLD, que têm também a seu cargo aceitar e avaliar ofertas e estabelecer contratos para execução de tarefas. Agentes AE são responsáveis pela execução propriamente dita de tarefas (EAs) e representam recursos físicos.

Os agentes SE, CLD e AE formam uma hierarquia de entidades permanentes (com o agente SE no topo), através da qual, qualquer BP que chegue à fábrica, é decomposto e distribuído pelas entidades apropriadas para o executarem. A interacção entre estes agentes ocorre apenas no direcção vertical da hierarquia. Os agentes de consórcio são entidades temporárias, criadas pelos agentes SE, com um papel de supervisão paralelo na hierarquia (em cada agente de consórcio é delegada, pelo agente SE, a supervisão da execução de um BP), que apenas existem quando há BPs. Para além da interacção indicada, um agente SE interage com o sistema de informação (para obter informação sobre BPs) e o utilizador; o agente AE pode interagir também com o sistema de informação (para troca de informação complementar sobre uma EA).

O conceito de consórcio permite agrupar dinamicamente recursos de produção de uma organização numa associação lógica temporária a um BP, independentemente da sua localização física ou agrupamento funcional (utiliza o conceito de *produção virtual*). Um escalonamento global corresponde, em cada instante, a um conjunto de consórcios. Conflitos são primeiro tratados localmente, *i.e.*, dentro de cada consórcio. No caso de impossibilidade de resolução local de um conflito, o agente SE é chamado para intervir na gestão do re-escalonamento inter-consórcios.

A metodologia proposta baseia-se em cinco fases: i) analisar e preparar (incluindo a análise da empresa e a preparação de informação), ii) instanciar (que inclui a definição do conjunto de agentes para o sistema particular de escalonamento), iii) validar (realizar ajustamentos finais nos agentes do sistema), iv) executar (implantação dos agentes) e vi) avaliar (incluindo a análise do desempenho do sistema e reconfigurações já durante o tempo de vida do sistema). É, adicionalmente, descrito o sistema de derivação de sistemas particulares de escalonamento.

Os trabalhos em [Rabelo 1998b], [Rabelo 1998c], [Camarinha-Matos 1999c], [Klen 1999] e [Rabelo 1999] são desenvolvimentos complementares posteriores aos anteriormente referidos. Estes trabalhos enquadram-se num âmbito mais vasto de projectos de aplicação prática de tecnologias de agentes, tecnologias de informação e de comunicações, para criação de sistemas de apoio à gestão de empresas virtuais, nomeadamente para produção e produção e distribuição, ver [Camarinha-Matos 1997a], [Camarinha-Matos 1997b], [Camarinha-Matos 1997c], [Klen 1998], [Spinosa 1998a], [Spinosa 1998b].

Exemplos de outros trabalhos com âmbito similar (*i.e.*, gestão de empresa virtual/empresa estendida/cadeia de fornecimento) podem encontrar-se em [Arnold 1997], [Dudenhausen 1997a], [Dudenhausen 1997b], [Leach 1997a], [Leach 1997b] e [Teixeira 1997] (aplicação à cadeia de fornecimento na produção de semicondutores), em [Fox 1993], [Barbuceanu 1994], [Beck 1994a], [Beck 1994b], [Barbuceanu 1995a] e [Barbuceanu 1995b] (gestão integrada da cadeia de fornecimento), em [Beck 1995], [Tate 1995] e [Tate 1996] (produção virtual e

logística) e em [Parunak 1995], [Parunak 1996c], [Parunak 1998c] e [Parunak 1998d] (modelação e simulação da cadeia de fornecimento e produção ágil).

3 Um Modelo para Escalonamento Multi-Agente — Nível Físico

Neste capítulo inicia-se a descrição do modelo de escalonamento multi-agente. O modelo está organizado em dois níveis: o *nível físico*, exposto no presente capítulo, onde se definem componentes do modelo como nós (recursos), redes e tarefas no contexto de redes de produção e distribuição; e o *nível virtual*, exposto no Capítulo 4, onde o assunto principal são os agentes de rede e a interacção entre agentes necessária à coordenação da actividade de escalonamento.

O trabalho descrito neste capítulo e no Capítulo 4 estabelece as entidades básicas para a construção de um ambiente computacional no qual se possam simular a actividade de escalonamento num contexto multi-agente das redes cooperativas de produção e distribuição e testar mecanismos de coordenação no que respeita a essa actividade.

3.1 Introdução

A actividade de escalonamento de redes de produção e distribuição insere-se no contexto mais geral de gestão da cadeia de fornecimento (s*upply chain*). Em particular, o modelo aqui desenvolvido tem aplicabilidade nas *redes cooperativas de produção e distribuição*, cujos nós são, em geral, pequenas e médias empresas especializadas, que cooperam no desempenho de vários tipos de tarefas bem definidas no processo de produção e distribuição. Assume-se o caso geral de uma *rede multi-produto*, ou seja, que entrega vários produtos finais para o exterior. Adopta-se o paradigma das redes de *Empresa Estendida* [Sackett 1994], [O'Neill 1994], [O'Neill 1996] ou [Browne 1996], onde existe uma perspectiva de conjunto da rede e os nós cooperam em pé de igualdade procurando conjugar objectivos individuais com objectivos globais da rede.

Numa rede de produção e distribuição pode-se considerar que existem *macro-recursos*, que são os nós de uma rede logística, que produzem, armazenam e transportam um conjunto de produtos. Estes macro-recursos podem ser fábricas, armazéns ou transportadoras, ligados por dependências em cadeia originadas pelo facto de as necessidades de produtos de uns serem satisfeitas por outros. Os nós da rede estão, portanto, ligados por *relações de cliente-fornecedor*. Vários tipos de fluxos ocorrem neste contexto. No que toca ao escalonamento das tarefas de cada nó da rede interessa considerar, em particular, dois tipos de fluxos: os *fluxos físicos* e os *fluxos de informação*. Um fluxo físico é um fluxo de produtos de montante para jusante na rede, de cada cliente para cada fornecedor, até ao mercado. Neste fluxo, fornecedores de matérias-primas abastecem produtores, produtores produzem produtos, distribuidores levam os produtos até ao cliente final (lojas, consumidor) e poderá haver vários estágios de produção e de distribuição. O *fluxo de informação* é um fluxo não físico necessário ao funcionamento da actividade na rede e no qual é, essencialmente, trocada informação sobre as necessidades de produtos. Este fluxo ocorre tanto no sentido jusante-montante, de cada cliente para cada fornecedor (encomendas) como no sentido inverso (aceitação ou rejeição de encomendas) e inclui informação necessária à sincronização das actividades (quantidades, datas). De acordo com [Forrester 1958] ou [Forrester 1961], para além dos fluxos físicos e dos fluxos de informação, outros tipos de fluxos ocorrem em redes de produção distribuição (*e.g.*, fluxos de dinheiro, fluxos de mão-de-obra). No entanto, no presente trabalho, não se considera que esses fluxos sejam relevantes para a actividade de escalonamento.

A actividade de escalonamento das tarefas logísticas na rede de EE é entendida na perspectiva do paradigma Multi-Agente da Inteligência Artificial Distribuída (ver, por exemplo, [Bond 1988], [O'Hare 1996] ou [ICMAS 1996]), como um problema de *coordenação multi-agente*. Os agentes cooperam de modo a poderem respeitar certas restrições, nomeadamente capacidades limitadas dos nós da rede física, precedências temporais das tarefas de cada nó e datas limite para satisfação dos pedidos à rede. No entanto, os agentes poderão competir para satisfazer preferências individuais de escalonamento, como escalonar uma tarefa de modo a ser realizada o mais cedo possível, ou o mais tarde possível, ou de uma forma intermédia a estas duas extremas. Nesta perspectiva, o ambiente em que os agentes se inserem é *semi-cooperativo, i.e.*, o problema de escalonamento multi-agente é *parcialmente adversario/parcialmente cooperativo*, segundo as ideias expressas em [Yokoo 1991] e [Yokoo 1990].

O modelo aqui apresentado é dividido em dois níveis, o *nível físico* e o *nível virtual*. No nível físico existe a *rede física* de produção e distribuição com *nós físicos* (os macro-recursos), ou

simplesmente *nós*, entre os quais se estabelecem fluxos físicos de produtos. O nível virtual é um nível de decisão e troca de informação onde existem agentes (as empresas), denominados *agentes gestores*, que afectam a capacidade dos nós a tarefas para satisfazer pedidos que eles trocam entre si, de modo a satisfazer pedidos (encomendas) do exterior à rede. Os agentes são ligados por canais de comunicação através dos quais trocam informação. Cada agente gere a capacidade de um nó físico e cada nó físico é gerido por um agente.

Tabela 3-1- Classes de objectos do nível físico e sublcasses.

Classe	**Subclasses**		
dimensão	*produto* *quantidade* *tempo* *local*		
evento	*evento simples*		
nó	*nó de capacidade*	*acumulador (*S*)*	
		processador	*produtor (*P*)* *transportador (*T*)*
	*nó de retalho (*R*)*		
	*nó de matéria-prima (*M*)*		
arco			
rede			
tarefa	*tarefa de capacidade*	*acumulação (*S*)*	
		processamento	*produção (*P*)* *transporte (*T*)*
	*tarefa de retalho (*R*)*		
	*tarefa de matéria-prima (*M*)*		
fluxo			
processo			

Relativamente ao problema de escalonamento, os agentes gestores têm apenas uma perspectiva local do problema. Adicionalmente, e de acordo com o paradigma da EE, considera-se existir um agente especial, denominado *agente supervisor*, o qual não gere a capacidade de nenhum nó físico, mas tem uma função de integração e está associado a actividades de decisão ligadas a uma perspectiva global (abrangendo toda a rede de produção e distribuição) e num horizonte

temporal de médio-longo prazo. Actividades de decisão ao nível táctico e estratégico,[1] como o planeamento a médio e longo prazo, a previsão da procura ou a configuração da rede dizem respeito, portanto, ao agente supervisor. No modelo de escalonamento aqui desenvolvido considera-se que a intervenção do agente supervisor ocorre apenas para introdução de trabalho no sistema, isto é, para transmitir aos agentes localizados em pontos extremos da rede (apelidados agentes de retalho e agentes matéria-prima) pedidos do exterior à rede. O agente supervisor actua, deste modo, apenas como interface da rede com o exterior. No que respeita à actividade de escalonamento, considera-se que ela pertence estritamente a um nível operacional e que, portanto, não sofre qualquer influência do agente supervisor, apenas dizendo respeito aos agentes gestores. O envolvimento do agente supervisor na previsão da procura e no planeamento de médio e longo prazo e a influência destas actividades na actividade de escalonamento dos agentes gestores será objecto de trabalho futuro.

Na Tabela 3-1 identificam-se as principais classes de objectos definidas para o nível físico do modelo, que são descritas a seguir. As classes de objectos consideradas acessórias, porque apenas ajudam à definição das principais, são referidas e descritas apenas no contexto apropriado. No Apêndice A expõe-se um conjunto de diagramas de classes para o modelo desenvolvido que inclui as classes descritas no presente capítulo.

3.1.1 Convenções para Descrição dos Objectos do Modelo

Para a descrição das classes de objectos do modelo desenvolvido no presente trabalho adoptaram-se convenções que a seguir se expõem. Estas convenções aplicam-se ao presente capítulo e também ao Capítulo 4.

Símbolos e Variáveis

Por regra é usado um símbolo principal diferente para denotar objectos de classes diferentes. Quando há necessidade de distinguir objectos diferentes da mesma classe usam-se símbolos acessórios como sufixos (índices) do símbolo principal. Símbolos podem ser usados como variáveis que representam objectos.

Atributos

Em primeiro lugar, uma classe de objectos é descrita em termos dos seus atributos, que são, para todos os efeitos, tratados como componentes dos objectos da classe. Estes são denotados por símbolos e, apelando a conceitos matemáticos como *tuplo* (*par*, *triplo*, *terno*, etc.), *conjunto*, *número* ou *valor lógico*, são descritos textualmente. Embora estes conceitos sejam abstractos, com eles pretende-se sugerir para uma realização computacional, o emprego de certos tipos de dados apropriados à representação de informação do modelo. Por exemplo, tuplos sugerem o emprego de tipos *ficha* (*estrutura* ou *record*); conjuntos sugerem o emprego de tipos *lista*, ou *vector*. Estes tipos de dados, juntamente com vários tipos numéricos e o tipo lógico estão disponíveis na maior parte das linguagens de programação, em geral.

---

[1] No âmbito do Planeamento da Produção é frequente as decisões serem hierarquizadas em três níveis: nível estratégico, nível táctico e nível operacional, cada nível envolvendo decisões que afectam horizontes temporais sucessivamente mais curtos e que constrangem progressivamente as decisões de níveis abaixo [Bitran 1993].

Operações

Uma classe de objectos é, adicionalmente, descrita através da definição das operações associadas à classe, quer dizer, as operações que permitem manipular a informação associada aos objectos da classe. Estas operações podem descrever atributos adicionais para, ou descrever o modo de operar em, objectos de certa classe e são traduzidas por funções parametrizadas e com valor associado, dependendo dos valores dos parâmetros. Na definição de função é, implicitamente, incluída a assinatura, o que abrange o identificador, a classe, o número e a ordem dos parâmetros; resumindo, funções com o mesmo identificador mas com número e/ou classes de parâmetros diferentes correspondem a operações diferentes. Deste modo, pretende-se sugerir para uma realização computacional a adopção dos *paradigmas de programação funcional* — operações são funções — e *orientado por objectos* — funções definem *métodos* para as classes de objectos e sugerem-se a organização das classes em hierarquias e características de herança e polimorfismo — (ver, por exemplo [Watt 1990]).[2]

Em geral define-se, para cada classe de objectos, um conjunto de operações primitivas constituído tipicamente por operações selectoras (para seleccionar, ou aceder a, componentes de um objecto da classe) e operações construtoras (para gerar objectos da classe). Outros tipos de operações podem ser definidas, como predicados (operações de teste), que produzem um valor lógico (`VERDADEIRO` ou `FALSO`), operações modificadoras, que descrevem a modificação do estado de um objecto em termos da alteração dos seus componentes de uma forma funcional,[3] ou outras que se considerem apropriadas. Regra geral, a descrição dos objectos de cada classe é realizada por meio de um núcleo de operações associadas à classe (constituído pelas operações construtoras e selectoras) de forma abstracta como *tipos de dados abstractos* (ver, por exemplo, [Cardelli 1985], [Liskov 1986] ou [Morgado 1989]).

A sintaxe da definição de uma operação é:

```
IDENTIFICADOR-DE-FUNÇÃO(...,parâmetro,...) ::= expressão .
```

Em que `IDENTIFICADOR-DE-FUNÇÃO` denota o identificador da operação definida, `...,parâmetro,...` denota zero ou mais ocorrências de `parâmetro`, em que `parâmetro` representa um valor inteiro, um valor real, um valor lógico ou um símbolo, um tuplo ou um conjunto, denotando um objecto de determinada classe. `expressão` pode conter um valor inteiro, um valor real, um valor lógico, um símbolo, um tuplo, um conjunto ou valores inteiros, valores reais, valores lógicos, símbolos, tuplos, ou conjuntos, combinados com operações. O valor da função é dado pelo valor de `expressão`. Símbolos no contexto

---

[2] Para a realização computacional do modelo descrito no presente trabalho foi utilizada a linguagem de programação CLOS (Common Lisp Object System [Steele 1990]). CLOS é uma extensão de Common Lisp, uma linguagem de programação de alto nível flexível e extensível que permite várias formas de abstracção e a construção rápida de programas. Uma das razões da opção pelo uso de CLOS é o facto de ser uma linguagem funcional e orientada por objectos, que permite a modularização de programas, a reutilização de componentes de programa, herança e polimorfismo e que retém todas as vantagens de Common Lisp. Como é referido nos capítulos 1 e 6, até à data de terminação do presente trabalho uma boa parte do modelo descrito foi realizada computacionalmente. Apesar de incompleta, esta realização computacional é referida por ter ajudado a refinar, testar e consolidar ideias durante o desenvolvimento do modelo.

[3] Quer dizer, não é empregue qualquer tipo de operação de atribuição de valor a variável para exprimir a alteração do estado de um objecto: em geral, a alteração do estado de um objecto é expressa através de uma função que recebe como parâmetro o objecto e produz um valor que é um objecto idêntico, à excepção de que o valor de algum atributo foi modificado.

de uma definição de função podem ser empregues como variáveis e o domínio lexical dessas variáveis é o da definição da função.

A sintaxe da invocação de uma função é:

`IDENTIFICADOR-DE-FUNÇÃO(...,expressão,...)`

Em que `IDENTIFICADOR-DE-FUNÇÃO` denota o identificador da operação sendo invocada e `...,expressão,...` denota zero ou mais ocorrências de `expressão` (em número igual ao número de parâmetros presentes na definição da função) e `expressão` representa uma expressão (como descrito acima, na sintaxe da definição de uma operação) com um valor associado que tem de ser da mesma classe do parâmetro que lhe corresponde na definição da função.

Comentários poderão ser colocados nas linhas de texto de uma definição. Comentários iniciam-se pelo símbolo `;` numa linha e abrangem todo o texto até ao fim dessa. Não têm qualquer efeito no significado da função definida e apenas servem para melhorar a legibilidade.

Condicionais

Assume-se como predefinida a operação `SE`, que aceita uma expressão lógica e duas expressões e produz o valor da primeira ou da segunda expressão, conforme o valor da expressão lógica seja `VERDADEIRO` ou `FALSO`, respectivamente. Isto é, sendo $\texttt{expressão}_1$ e $\texttt{expressão}_2$ expressões:

$\texttt{SE(VERDADEIRO,expressão}_1\texttt{,expressão}_2\texttt{) ::= expressão}_1\texttt{.}$

$\texttt{SE(FALSO,expressão}_1\texttt{,expressão}_2\texttt{) ::= expressão}_2\texttt{.}$

Valores Numéricos e Lógicos

Assumem-se como predefinidos o conjunto dos números inteiros e o conjunto dos números reais, bem como as operações aritméticas sobre números inteiros e números reais `+` (adição), `-` (subtracção), `*` (multiplicação) e `/` (divisão) e, adicionalmente, a operação `CEILING`, uma função de um parâmetro que devolve o menor número inteiro que é maior ou igual ao número dado como parâmetro. O símbolo de somatório $\sum$ pode ser usado para exprimir somas em série. Por exemplo, as expressões:

$$\sum_{i=1,\ldots,n} c_i \qquad \text{ou} \qquad \sum_{i=1}^{n} c_i$$

exprimem ambas a soma $c_1+\ldots+c_i+\ldots+c_n$,; a expressão:

$$\sum_{c \in \mathcal{C}} c$$

exprime a soma de todos os elementos do conjunto $\mathcal{C}$.

Podem ser usados no contexto de expressões lógicas os valores lógicos `VERDADEIRO` e `FALSO`, bem como as operações lógicas $\neg$ (negação lógica), $\wedge$ (conjunção lógica) e $\vee$ (disjunção lógica) e as operações lógicas de comparação de dois valores do mesmo conjunto, ou dois objectos da mesma classe, `=` (igual a) $\neq$ e (diferente de). O quantificador universal $\forall$ e

o quantificador existencial ∃ podem também ser usados no contexto de expressões lógicas, para exprimir, respectivamente, a universalidade e a contingência de uma expressão lógica. Por exemplo, a expressão lógica:

$\forall_{x}$ (x∈ℂ) : TESTE?(x)

exprime que, para todos os elementos do conjunto ℂ, se verifica TESTE?(x) (*i.e.*, se verifica TESTE?(x)=VERDADEIRO, em que o predicado TESTE? é definido algures); as expressões lógicas:

$\exists_{x}$: (x∈ℂ) ∧ TESTE?(x) ou $\exists_{x}$ (x∈ℂ) : TESTE?(x)

ambas exprimem que existe pelo menos um elemento do conjunto ℂ para o qual se verifica TESTE?(x).

As operações < (menor que), ≤ (menor que ou igual a), > (maior que) e ≥ (maior que ou igual a) são usadas para comparação de valores sobre os quais existe uma ordem predefinida (por exemplo, números inteiros e reais).

As operações MAX e MIN são usadas para denotar, respectivamente, o valor máximo e o valor mínimo para uma variável numérica, dada como parâmetro, que satisfaça uma restrição dada por uma expressão lógica. Por exemplo:

MAX(x) : (x∈ℂ) ∧ TESTE?(x) ou $\underset{x\in\mathbb{C}}{MAX}$: TESTE?(x)

ambas denotam o valor máximo que está presente no conjunto ℂ e que satisfaz o predicado TESTE?.

Tuplos e Conjuntos

Um tuplo é uma colecção de elementos, possivelmente de tipo diferente, em número fixo e numa ordem fixa. Literalmente, um tuplo é denotado pelo par de símbolos < e > (este par de símbolos pode ser interpretado como uma operação construtora universal de tuplos), entre os quais se listam os elementos do tuplo na ordem apropriada, separados por vírgulas.

Um conjunto é uma colecção de zero ou mais elementos do mesmo tipo, denotado pelo par de símbolos { e }, entre os quais se enumeram os elementos do conjunto, separados por vírgulas. O par de símbolos { e } pode ser usado como operação construtora universal de conjuntos para definir um conjunto em compreensão. Por exemplo, a expressão:

{ x : TESTE?(x) }

exprime o conjunto contendo todos os elementos x que satisfazem o predicado TESTE?(x).

Um conjunto pode ser, ou não, ordenado. Num conjunto ordenado a ordem dos elementos importa. Assume-se que os elementos de um conjunto podem ser enumerados, que para um conjunto ordenado os elementos podem ser enumerados pela ordem que o conjunto estabelece para esses elementos e que é possível seleccionar o i-ésimo elemento do conjunto, dado o índice i para o elemento do conjunto. As operações ∩ (intersecção de conjuntos), ∪ (reunião de conjuntos), \ (diferença de conjuntos), ⊂ (teste de inclusão de um conjunto noutro), ⊄ (negação do teste de inclusão), ∈ (teste de membro de um conjunto) e ∉ (teste de não membro) assumem-se como predefinidas, bem como a cardinalidade (número de elementos) de um conjunto que é dada por |ℂ|, para qualquer conjunto ℂ. As operações ∪ e ∩ podem exprimir reuniões ou intersecções de conjuntos em série. Por exemplo, as expressões:

$\bigcup_{i=1,\ldots,n} \mathcal{C}_i$ ou $\bigcup_{i=1}^{n} \mathcal{C}_i$

exprimem ambas a reunião de conjuntos $\mathcal{C}_1 \cup, \ldots, \cup \mathcal{C}_i \cup, \ldots, \cup \mathcal{C}_n$; a expressão:

$\bigcup_{\text{TESTE?}(\mathcal{C}_i)} \mathcal{C}_i$

exprime a reunião de todos os conjuntos $\mathcal{C}_i$ para os quais se verifica TESTE?($\mathcal{C}_i$).

Entre dois tuplos ou entre dois conjuntos, as operações de igualdade e desigualdade são denotadas por = e ≠, respectivamente. Dois conjuntos ou dois tuplos são iguais se têm elementos iguais e em igual número (e na mesma ordem, caso se trate de conjuntos ordenados ou tuplos). Adicionalmente, definem-se para conjuntos, as operações a seguir descritas.

Para conjuntos ordenados usam-se as operações genéricas (que operam sobre conjuntos de quaisquer tipos de elementos) NÉSIMO, POSIÇÃO, INSERE-NÉSIMO e RETIRA-NÉSIMO. A operação NÉSIMO, dados um conjunto ordenado $\mathcal{C}$, não vazio, tal que $\mathcal{C}=\{\ldots,e_n,\ldots\}$, em que e_n denota o elemento na posição n (n=0 se e_n é o primeiro elemento de $\mathcal{C}$) e um índice n de $\mathcal{C}$, um inteiro tal que n≥0 e n<|$\mathcal{C}$|, produz o elemento de $\mathcal{C}$ na posição de índice n e é definida como se segue:

NÉSIMO({...,e_n,...},n) ::= e_n.

A operação POSIÇÃO, dados um conjunto ordenado $\mathcal{C}$ e um elemento e_n tal que $e_n \in \mathcal{C}$ produz o índice de e_n em $\mathcal{C}$ e é definida como se segue:

POSIÇÃO($\mathcal{C}$,e_n) ::= n : (NÉSIMO($\mathcal{C}$,n)=e_n).

A operação INSERE-NÉSIMO, dados um conjunto ordenado $\mathcal{C}$, um elemento e_x e um índice n tal que n≥0 e n≤|$\mathcal{C}$|, produz um conjunto ordenado $\mathcal{C}'$, igual a $\mathcal{C}$ à excepção de que o elemento e_x foi inserido na posição n. Esta operação define-se do seguinte modo:

INSERE-NÉSIMO($\mathcal{C}$,e_x,n) ::=
$\mathcal{C}'$: ($\mathcal{C}'\setminus\{e_x\}=\mathcal{C}$) ∧ (NÉSIMO($\mathcal{C}'$,n)=$e_x$) ∧
($\forall_i$ (i≥0,i<n) : (NÉSIMO($\mathcal{C}'$,i)=NÉSIMO($\mathcal{C}$,i))) ∧
($\forall_j$ (j>n,j≤|$\mathcal{C}$|) : (NÉSIMO($\mathcal{C}'$,j)=NÉSIMO($\mathcal{C}$,j-1))).

A operação RETIRA-NÉSIMO, dados um conjunto ordenado $\mathcal{C}$ e um índice n tal que n≥0 e n<|$\mathcal{C}$|, produz um conjunto ordenado $\mathcal{C}'$, igual a $\mathcal{C}$ à excepção de que o elemento na posição n foi retirado. Esta operação define-se do seguinte modo:

RETIRA-NÉSIMO($\mathcal{C}$,n) ::=
$\mathcal{C}'$: (INSERE-NÉSIMO($\mathcal{C}'$,NÉSIMO($\mathcal{C}$,n),n)=$\mathcal{C}$).

Por comodidade definem-se também as seguintes operações genéricas com conjuntos ordenados não vazios ($\mathcal{C}$ denota um conjunto ordenado):

PRIMEIRO($\mathcal{C}$) ::= NÉSIMO($\mathcal{C}$,0).

ÚLTIMO($\mathcal{C}$) ::= NÉSIMO($\mathcal{C}$,|$\mathcal{C}$|-1).

RETIRA-PRIMEIRO($\mathcal{C}$) ::= RETIRA-NÉSIMO($\mathcal{C}$,0).

A operação JUNTA-NO-FIM, dados um conjunto ordenado qualquer $\mathcal{C}$, e um elemento e, produz um conjunto ordenado $\mathcal{C}'$, igual a $\mathcal{C}$ à excepção de que contém mais um elemento, o elemento e, na última posição de $\mathcal{C}'$:

JUNTA-NO-FIM($\mathcal{C}$,e) ::= $\mathcal{C}'$: $\mathcal{C}'$=INSERE-NÉSIMO($\mathcal{C}$,e,|$\mathcal{C}$|).

A operação genérica SUBSTITUI, dados um conjunto e dois elementos da mesma classe dos elementos do conjunto produz um novo conjunto de acordo com o que se segue: no caso de o primeiro elemento existir no conjunto, SUBSTITUI produz um conjunto idêntico à excepção de que o primeiro elemento dado foi substituído pelo segundo (se o conjunto dado é um conjunto ordenado, no conjunto produzido o elemento novo ocupa a mesma posição que o elemento retirado ocupava); no caso de aquele elemento não existir no conjunto dado, SUBSTITUI produz um conjunto igual. Denotando o conjunto, o primeiro elemento e o segundo elemento dados por, respectivamente $\mathcal{C}$, e_1 e e_2, para o primeiro caso (o caso em que $e_1 \in \mathcal{C}$), a operação SUBSTITUI é definida do seguinte modo:

SUBSTITUI($\mathcal{C}$,e_1,e_2) ::= ($\mathcal{C}$\{e_1})$\cup${e_2}.

Para o caso particular de conjuntos ordenados define-se ($\mathcal{C}$ denota um conjunto ordenado):

SUBSTITUI($\mathcal{C}$,e_1,e_2) ::=
INSERE-NÉSIMO(RETIRA-NÉSIMO($\mathcal{C}$,POSIÇÃO($\mathcal{C}$,e_1)),
e_2,
POSIÇÃO($\mathcal{C}$,e_1)).

à excepção de quando $e_1 \notin \mathcal{C}$, caso em que a operação SUBSTITUI se define do seguinte modo:

SUBSTITUI($\mathcal{C}$,e_1,e_2) ::= $\mathcal{C}$.

3.2 Dimensão

Colocar o produto certo, na quantidade certa, no momento certo, no local certo é o objectivo básico de um sistema de produção e distribuição. *Produto, quantidade, tempo* e *local* são *dimensões* de um espaço onde ocorre o processo de produção e distribuição; podem considerar-se os graus de liberdade com os quais é possível jogar quando há que tornar disponível um produto ao cliente final.

As dimensões de *quantidade* e *tempo* são, obviamente, omnipresentes numa rede de produção e distribuição. Questões de quantidade e de tempo põem-se sempre que há que satisfazer uma procura (variável ao longo do tempo) no mercado, ou quando há que considerar custos (qual a quantidade económica? Quanto custa mais produzir/entregar em menos tempo?). Obviamente que questões de *local* afectarão também os produtores (proximidade da fonte/fornecedor de matérias-primas, materiais ou componentes, proximidade em relação a clientes imediatos, a outros produtores ou distribuidores). Também as questões de *produto* não deixarão de estar presentes nas decisões dos distribuidores (que tipo e variedade de produtos levar até ao mercado?).

Para um domínio de produção e distribuição estas dimensões são conjuntos de valores possíveis de produto, quantidade, tempo e local. Valores particulares de produto, quantidade, tempo e local são usados para referenciar e definir cada tarefa no contexto de uma rede de produção e distribuição envolvendo múltiplos intervenientes e múltiplos produtos. Segue-se uma descrição do significado de cada dimensão.

- *Dimensão produto* - É o conjunto de *identificadores de produtos* (incluindo matérias-primas, produtos intermédios e produtos finais da rede) de produtos diferentes, que podem ser trocados entre os nós da rede, ou entre os nós da rede e o exterior desta. Este conjunto é finito e enumerável e os seus elementos são símbolos. Não se estabelece qualquer relação entre eles para além da relação de igualdade e desigualdade. Esta dimensão é formalmente representada pelo conjunto:

  $\mathbb{P}=\{p_1,...,p_i,...,p_{Np}\}$

  em que os Np elementos p_i ($i=1,...,Np$) identificam todos os produtos de uma instância de modelo (todos os produtos que podem ser trocados entre os nós da rede de produção e distribuição têm um identificador p_i que é um elemento do conjunto $\mathbb{P}$).

- *Dimensão quantidade* - É o conjunto de valores possíveis de *quantidades*. Um valor de quantidade é um número real positivo (que pode significar unidades, quilogramas, metros cúbicos, etc.), que representa uma quantidade a armazenar ou a processar. Existem relações de ordem, igualdade e desigualdade entre os valores deste conjunto que correspondem às mesmas relações entre números reais. Simbolicamente, valores de quantidade são designados por q, ou q_i, em que i é um índice usado para distinguir diferentes quantidades num mesmo contexto.

- *Dimensão tempo* - É o conjunto de valores possíveis de *instantes de tempo*, ou *instantes*. Representado pelo conjunto dos números inteiros com as relações de ordem, igualdade e desigualdade. Simbolicamente, instantes de tempo são designados por t, ou t_i, em que i é um índice usado para distinguir diferentes instantes num mesmo contexto.

- *Dimensão local* - É o conjunto de valores possíveis de identificadores dos *nós* de rede de produção e distribuição.[4] Os valores do conjunto de valores possíveis são símbolos e o conjunto é finito e enumerável. Entre os valores deste conjunto estabelecem-se relações igualdade, ou desigualdade. A dimensão local é formalmente representada pelo conjunto:

  $\mathcal{v}=\{v_1,...,v_{Nn}\}$

  em que os Nn elementos v_x ($x=1,...,Nn$) identificam todos os nós da rede de produção e distribuição de uma instância de modelo (todos os nós têm um identificador v_i que é um elemento do conjunto $\mathcal{v}$).

3.2.1 Intervalo de Tempo

No contexto da dimensão tempo, define-se a seguinte classe acessória de objectos:

- *Intervalo-de-tempo* - Um objecto desta classe, designado por *intervalo de tempo*, ou simplesmente *intervalo*, simbolicamente $\mathbb{h}$, é um par de instantes $<t_s,t_e>$ tais que $t_e>t_s$, sendo t_s e t_e designados por *instante inicial* e *instante final*, respectivamente. Um intervalo de tempo $<t_s,t_e>$ representa o conjunto ordenado de todos os instantes t tais que $t\geq t_s$ e $t<t_e$ (*i.e.*, representa o conjunto ordenado $\{t_s,t_s+1,...,t_e-1\}$)

A *duração* é um atributo de um intervalo. Para um intervalo $<t_s,t_e>$ a duração é igual ao valor de t_e-t_s e é um inteiro positivo. Se $t_e=t_s+1$ diz-se que $<t_s,t_e>$ tem a duração de uma *unidade de tempo* (o intervalo de tempo mais pequeno representável é aquele que tem a duração de uma unidade de tempo). O significado de uma unidade de tempo depende da escala temporal adoptada. Esta escala é uma convenção: 1 unidade de tempo poderá corresponder a um segundo, um minuto, uma hora, uma semana, etc., dependendo da granulação temporal adequada a cada modelo.

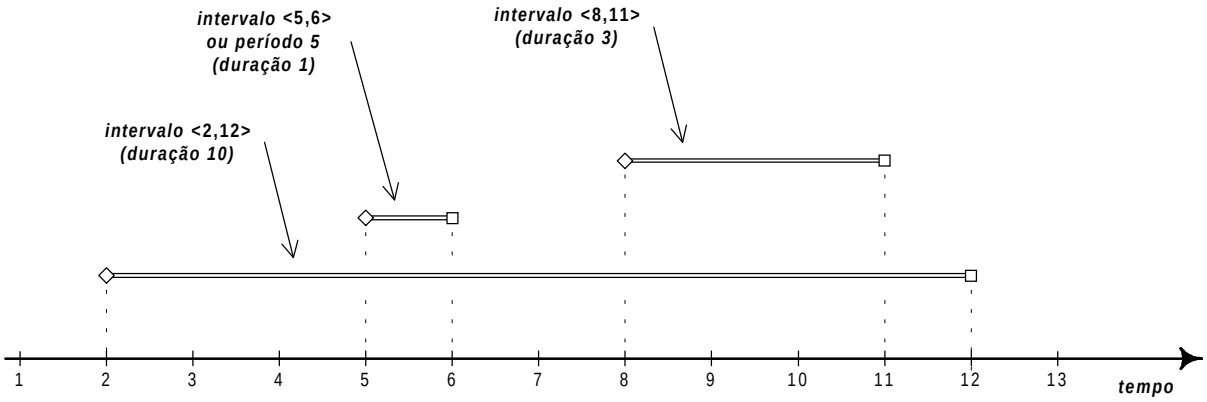


Figura 3-1- Exemplos de intervalos de tempo. Intervalos são representados por barras horizontais com os instantes extremos assinalados por um losango (início) e um quadrado (fim).

[4] Que não estão, necessariamente, associados a localizações espaciais reais mas servem para distinguir nós diferentes da rede.

O intervalo de tempo mais pequeno considerado em termos de afectação de capacidade para escalonamento de tarefas é denominado *período temporal*, ou simplesmente *período* e tem a duração de uma unidade de tempo. O período temporal com o instante de início t_x é referido por "o período temporal t_x" ou "o período t_x". Um *horizonte temporal* é um intervalo de tempo abrangendo um número inteiro de períodos consecutivos em que, por exemplo, uma ou mais tarefas devem ser escalonadas.

Na Figura 3-1 representam-se esquematicamente exemplos de intervalos.

3.2.1.1 Operações com Intervalos de Tempo

As operações primitivas sobre intervalos de tempo são descritas a seguir.

$TS(<t_s,t_e>) ::= t_s.$

$TE(<t_s,t_e>) ::= t_e.$

$INTERVALO(t_s,t_e) ::= <t_s,t_e>.$

A operação DURAÇÃO produz a duração de um intervalo de tempo:

$DURAÇÃO(\mathbb{h}) ::= TE(\mathbb{h})-TS(\mathbb{h}).$

Os seguintes predicados testam, respectivamente, se um instante t_k, está dentro de (ou ocorre durante) um intervalo de tempo $\mathbb{h}$ e se todos os instantes de um intervalo $\mathbb{h}_1$ estão dentro de outro intervalo $\mathbb{h}_2$.

$DURANTE?(t_k,\mathbb{h}) ::= (t_k \geq TS(\mathbb{h})) \wedge (t_k < TE(\mathbb{h})).$

$DURANTE?(\mathbb{h}_1,\mathbb{h}_2) ::= (TS(\mathbb{h}_1) \geq TS(\mathbb{h}_2)) \wedge (TE(\mathbb{h}_1) \leq TE(\mathbb{h}_2)).$

3.3 Evento

Um *evento* é um acontecimento pontual e corresponde a um tuplo de valores particulares contendo informação de produto, quantidade, tempo e local. Como a informação relativa à quantidade só tem significado quando se refere a um produto, quantidades e produtos são, nos eventos, associados na forma de conjuntos de pares produto-quantidade.

3.3.1 Par-Produto-Quantidade, Conjunto-De-Pares-Produto-Quantidade

Antes de definir evento introduzem-se as duas classes de objectos acessórias:

- *Par-produto-quantidade* - Par $<p,q>$ em que p é um produto e q uma quantidade. Objectos desta classe, designados por *pares produto-quantidade*, são simbolicamente

referidos por $\mathbb{p}q$ ou $\mathbb{p}q_i$, em que `i` é um índice usado para distinguir pares produto-quantidade diferentes.

- *Conjunto-de-pares-produto-quantidade* - Conjunto `{...,`$\mathbb{p}q_j$`,...}`, em que os $\mathbb{p}q_j$ são pares produto-quantidade tais que, para quaisquer dois pares diferentes $\mathbb{p}q_j=<p_j,q_j>$ e $\mathbb{p}q_k=<p_k,q_k>$ do conjunto se tem $p_j \neq p_k$. Simbolicamente referido por $\mathcal{pq}$ ou $\mathcal{pq}_i$, em que `i` é um índice para distinguir diferentes conjuntos de pares produto-quantidade.

3.3.1.1 Operações com Pares Produto-Quantidade e Conjuntos de Pares Produto-quantidade

As operações primitivas sobre pares produto-quantidade são:

`PRODUTO(<p,q>) ::= p.`

`QUANTIDADE(<p,q>) ::= q.`

`PAR-PQ(p,q) ::= <p,q>.`

A operação `CONTÉM-PAR-PQ?` é um predicado que testa se existe, num conjunto de pares produto-quantidade dado, um par produto-quantidade com um dado produto associado.

`CONTÉM-PAR-PQ?(`p_i`,`$\mathcal{pq}$`)` $::= \exists \mathbb{p}q_i : (\mathbb{p}q_i \in \mathcal{pq}) \wedge$ `(PRODUTO(`$\mathbb{p}q_i$`)=`p_i`).`

A operação seguinte produz o par produto-quantidade de um dado conjunto de pares produto-quantidade $\mathcal{pq}$, que tem associado um dado produto p_i. Esta operação apenas é aplicável quando se verifica `CONTÉM-PAR-PQ?(`p_i`,`$\mathcal{pq}$`)`.

`PAR-PQ(`p_i`,`$\mathcal{pq}$`)` $::= \mathbb{p}q_i : (\mathbb{p}q_i \in \mathcal{pq}) \wedge$ `(PRODUTO(`$\mathbb{p}q_i$`)=`p_i`).`

3.3.2 Evento e Evento Simples

Formalmente, um evento é descrito por um triplo:

`<`$\mathcal{pq}$`,t,v>`

em que `t` é o instante associado ao evento, `v` identifica o local (nó de rede) onde ele ocorre e $\mathcal{pq}=\{...,<p_i,q_i>,...\}$ ($p_i \in \mathcal{P}$) é o conjunto de pares produto-quantidade associados ao evento. Simbolicamente, eventos são designados por e ou e_i, em que `i` é um índice usado para distinguir diferentes eventos num mesmo contexto.

Um *evento simples* é um evento cujo conjunto de pares produto-quantidade é singular, isto é, um evento simples é da forma:

`<{<p,q>},t,v>`

3.3.2.1 Operações com Eventos

As operações primitivas sobre eventos são descritas a seguir.

PQ-CONJ(<$p$$q$,t,v>) ::= $p$$q$.

TEMPO(<$p$$q$,t,v>) ::= t.

LOCAL(<$p$$q$,t,v>) ::= v.

EVENTO($p$$q$,t,v) ::= <$p$$q$,t,v>.

EVENTO?(<$p$$q$,t,v>) ::= VERDADEIRO.

A operação a seguir definida é um predicado que testa se um evento e ocorre durante um intervalo de tempo h:

DURANTE?(e,h) ::= DURANTE?(TEMPO(e),h).

Só para eventos simples definem-se, adicionalmente, as seguintes operações (e denota um evento simples):

PRODUTO(e) ::= PRODUTO(PRIMEIRO(PQ-CONJ(e))).

QUANTIDADE(e) ::= QUANTIDADE(PRIMEIRO(PQ-CONJ(e))).

O seguinte predicado permite testar se um dado evento é um evento simples:

EVENTO-SIMPLES?(e) ::= EVENTO?(e) $\wedge$ (|PQ-CONJ(e)|=1).

Um evento pode conter um ou mais pares produto-quantidade. Neste último caso, cada par diferente tem um produto diferente. Por exemplo um evento de início de uma tarefa de produção[5] pode conter mais de um par produto-quantidade, em que cada produto de cada par corresponde a um material, ou componente, que tem de estar presente e disponível no início da execução da tarefa e cada quantidade indica a quantidade necessária do produto respectivo.

Exemplos:

a) <{<p_1,q_1>,<p_2,q_2>,<p_3,q_3>},t_3,v_2> - Designa o evento a que estão associados três produtos, p_1, p_2 e p_3, respectivamente nas quantidades q_1, q_2, e q_3, no instante t_3 e no nó v_2;

[5] Ver a secção 3.7.

b) `<{<parafusoM5,3000>},320,ArmazemX30>` - Pode ser, por exemplo, o evento no qual `3000` unidades do produto `parafusoM5` entraram no local `ArmazemX30`, no instante `320`.

3.4 Nó

Os recursos estão sujeitos a restrições que têm a ver com limitações de capacidade: caso dos recursos ditos *renováveis*. Ou então estão associados a uma quantidade de valor, ou energia, neles incorporada e à possibilidade de serem produzidos ou consumidos: os recursos ditos *não renováveis*. Segundo [Blazewicz 1994] os recursos renováveis estão sujeitos a restrições que dizem respeito a uma limitação ao seu uso total em cada momento: quando um recurso é afectado a uma tarefa, a capacidade disponível do recurso é reduzida numa quantidade igual à necessária à execução da tarefa durante o intervalo de tempo desta; a partir do tempo de fim da tarefa aquela quantidade de capacidade fica novamente disponível. Os recursos não renováveis estão sujeitos a restrições que têm a ver com limitações ao seu consumo, ou produção, até dado momento: se um recurso não renovável é consumido numa tarefa, não pode voltar a sê-lo por outra tarefa; se é produzido por uma tarefa a sua quantidade é sempre limitada até um certo momento no tempo.[6]

Neste modelo, os produtos representam os recursos não renováveis, *i.e.*, representam materiais, componentes, produtos intermédios, produtos finais; os nós de uma rede de produção e distribuição representam os recursos renováveis, *i.e.*, os macro-recursos da rede de produção e distribuição, como fábricas, linhas de produção, máquinas, armazéns, frotas ou veículos de transporte. A classe de nós que representa estes macro-recursos é a classe *nó de capacidade* (ver Tabela 3-1), já que aqueles objectos têm um atributo denominado *capacidade*. Dependendo do tipo de capacidade, o nó poderá desempenhar tarefas de processamento (produção ou transporte) ou tarefas de acumulação (armazenamento) de produtos. No primeiro caso o nó pertence à classe *processador*, no último caso à classe *acumulador*; na classe processador podem ainda distinguir-se as classes *produtor* e *transportador*. Há ainda duas classes adicionais de nós, denominadas *nó de retalho* e *nó de matéria-prima* (ver Tabela 3-1) cujos objectos são casos singulares, pois não dispõem de capacidade (consideram-se, por isso, nós fictícios) e servem para representar a fronteira da rede com o exterior. Nós de retalho são nós apenas clientes, isto é, não são fornecedores de nó algum de rede (são pontos da fronteira de saída de produtos da rede para o exterior) e numa rede estão sempre posicionados nos extremos jusante da rede. Nós de matéria-prima são nós apenas fornecedores, isto é, não são clientes de nó algum de rede (são pontos da fronteira de entrada de produtos do exterior para a rede) e estão sempre posicionados nos extremos montante da rede. A existência destas classes adicionais de nós é justificada pela singularidade na fronteira com o exterior da rede, que não é explicitamente representado no modelo.[7]

---

[6] Uma designação alternativa para "recurso renovável" é "recurso reutilizável". Para "recurso não renovável" alternativas são "recurso não reutilizável" ou "recurso consumível" ou "recurso produzível" (ver, por exemplo [Beck 1995]).

[7] Como resultado tem-se uma classificação mais sistemática dos nós, no sentido de que cada classe tem um conjunto de atributos similares. Ver as secções 3.4.1 e 3.4.2. Uma outra alternativa seria ter, no conjunto de nós de uma rede, apenas nós de capacidade, mas existirem dois subconjuntos desse conjunto contendo nós de capacidade distintos dos restantes e que desempenhariam o papel de fronteira de saída e de fronteira de entrada da rede.

3.4.1 Nó de Capacidade

Nós de capacidade são nós que têm capacidade e têm entradas e saídas de produtos. Produtos de entrada de um nó provêm de *nós fornecedores* e correspondem a materiais, ou componentes, para consumo e incorporação nos produtos de saída do nó. Produtos de saída de um nó destinam-se aos seus *nós clientes*. Os produtos de saída são disponíveis à custa de consumo de produtos de entrada e à custa de um consumo temporário de capacidade empregue pelo nó em tarefas[8] que permitem essa disponibilidade. Nesta secção caracteriza-se a capacidade e depois descrevem-se as classes de nós acumulador e processador.

3.4.1.1 Capacidade

A capacidade, genericamente referida pelo símbolo C, serve para representar a natureza limitada dos nós de capacidade como recursos e é caracterizada em duas dimensões:

- *Tipo de capacidade*, ou *classe de capacidade* - Que distingue duas naturezas diferentes de capacidade, referidas por *capacidade de acumulação* ou *capacidade de tipo* A e *capacidade de processamento* ou *capacidade de tipo* W;
- Para cada tipo de capacidade podem ter-se: a *capacidade máxima*, a *capacidade utilizada* e a *capacidade disponível*.

A seguir clarificam-se progressivamente os conceitos introduzidos sobre capacidade.

3.4.1.2 Capacidade Máxima, Utilizada e Disponível. Restrições de Capacidade

Para afectar capacidade a tarefas de um nó de capacidade é necessário saber se existe capacidade não utilizada suficiente para a afectação. Define-se então:

- *Capacidade máxima* - Significa a maior quantidade de capacidade que um nó pode afectar a tarefas. Um valor de capacidade máxima está sempre associado a um instante e corresponde a uma quantidade de produto inteira e não negativa. É denotada genericamente por C_{max}, ou por A_{max} ou W_{max} para cada tipo de capacidade. De um modo geral, como a capacidade máxima pode variar ao longo do tempo, dentro de um horizonte temporal de escalonamento, usa-se a notação $C_{max}(t)$ (ou $A_{max}(t)$ ou $W_{max}(t)$), que denota a função inteira que mapeia cada período t do horizonte temporal para um valor de capacidade máxima. Esta função é apelidada de *perfil de capacidade máxima* e mantém-se inalterada durante a actividade de escalonamento;
- *Capacidade utilizada* - Significa a quantidade de capacidade que um nó tem afectada a tarefas escalonadas. Um valor de capacidade utilizada está sempre associado a um instante e corresponde a uma quantidade de produto inteira e não negativa. É denotada genericamente por c, ou por a ou w para cada tipo de capacidade. Como varia ao longo do tempo, dentro de um horizonte temporal de escalonamento, usa-se a notação $c(t)$ (ou $a(t)$ ou $w(t)$), que denota a função inteira que mapeia cada período t do horizonte temporal para um valor de capacidade utilizada. Esta função é apelidada de *perfil de capacidade utilizada* e é alterável com a actividade de escalonamento;

[8] Ver a secção 3.7.

- *Capacidade disponível* - Significa a parte da capacidade máxima que não está afectada a tarefas escalonadas (portanto, está disponível para ser afectada a tarefas). Um valor de capacidade disponível está sempre associado a um instante e corresponde a uma quantidade de produto inteira não negativa. É denotada genericamente por C, ou por A e W para cada tipo de capacidade. Como varia ao longo do tempo, usa-se a notação $C(t)$ (ou $A(t)$ ou $W(t)$), que denota uma função que mapeia cada período t de um dado horizonte temporal para um valor de capacidade disponível. Esta função é apelidada de *perfil de capacidade disponível* e é alterável com a actividade de escalonamento.

A capacidade máxima $C_{max}(t)$ significa o valor máximo de quantidade capacidade que um nó pode aplicar em tarefas para entregar produtos *no contexto de uma rede*. Para um nó particular este valor máximo poderá variar ao longo dos períodos de um mesmo horizonte temporal, tanto por a capacidade do nó poder ser ampliada ou reduzida de forma planeada, como por a capacidade que o nó planeou ter disponível *para a rede* poder variar ao longo do tempo. Este último caso acomoda a possibilidade de o nó poder repartir a sua capacidade por tarefas no contexto da rede (que entregam produtos *para a rede*) e tarefas fora do contexto da rede (que entregam produtos para entidades *fora da rede*).

A capacidade disponível, $C(t)$, para cada período t é dada por:

$$C(t)=C_{max}(t)-\sum_{j=1}^{n}c_j(t)$$

em que n é o número de produtos de saída do nó (sendo $p_1,\ldots,p_n$, esses produtos).

Restrições estruturais do modelo designadas por *restrições de capacidade* impõem que $C(t)\geq 0$, ou seja, impõem que:

$$C_{max}(t)-\sum_{j=1}^{n}c_j(t)\geq 0$$

para qualquer t, no horizonte temporal de escalonamento considerado, em qualquer solução de escalonamento dita possível. Cada restrição $C(t)\geq 0$, para um determinado valor t é uma restrição de capacidade. Qualquer estado de um nó de capacidade no qual há restrições de capacidade que são violadas é um estado que contém *conflitos de capacidade*. Nesse caso, a cada valor particular t tal que $C(t)<0$ corresponde um conflito de capacidade.

3.4.1.3 Tipo de Capacidade

No que respeita ao tipo de capacidade pode ter-se:

- *Capacidade de acumulação*, ou *capacidade de tipo* A - É um tipo de capacidade limitada em quantidade e está disponível nos nós de capacidade da classe *acumulador*.[9] $A_{max}(t)$ significa a quantidade máxima de um produto que um nó acumulador poderia acumular em cada período t, se o nó estivesse dedicado apenas a esse produto, medida em *unidades de capacidade de acumulação* do nó. $a(t)$ significa a parte de $A_{max}(t)$ utilizada (ocupada) em cada período t. Quando a capacidade de acumulação é partilhada por vários produtos, $a_j(t)$ denota a capacidade de acumulação utilizada no produto p_j em cada período t e a

[9] Ver a secção 3.4.1.13.

parte de $A_{max}(t)$ utilizada é dada por $\sum_{j=1}^{n} a_j(t)$, em que n é o número de produtos de saída do nó;

- *Capacidade de processamento*, ou *capacidade de tipo* W - É um tipo de capacidade limitada em quantidade e tempo e está disponível nos nós de capacidade da classe *processador*.[9] $W_{max}(t)$ significa a quantidade máxima de um produto *por unidade de tempo*[10] que um nó processador poderia processar (produzir ou transportar) em cada período t se o nó estivesse dedicado apenas a esse produto, medida em *unidades de capacidade de processamento* do nó. $w(t)$ significa a parte de $W_{max}(t)$ utilizada (ocupada) em cada período t. Quando a capacidade de processamento é partilhada por vários produtos, $w_j(t)$ denota a capacidade de processamento utilizada no produto p_j em cada período t e a parte de $W_{max}(t)$ utilizada é dada por $\sum_{j=1}^{n} w_j(t)$, em que n é o número de produtos de saída do nó.

A unidade de capacidade é sempre relativa a um nó e é referida por *unidade de produto padrão de acumulação* (abreviado para *uppa*), para uma capacidade de acumulação, ou por *unidade de produto padrão de processamento* (abreviado para *uppp*), para uma capacidade de processamento. Estas unidades são convenções internas ao nó que servem para medir a capacidade consumida por quantidades dos produtos de saída associadas a tarefas escalonadas, ou a escalonar, e têm um interesse particular quando a capacidade do nó é partilhada por produtos diferentes. Para unidade de medida interna de capacidade, cada nó convenciona a unidade de um produto de saída real, ou fictício, que é tomado como *produto padrão*. A capacidade consumida por cada unidade entregue de um produto de saída por uma tarefa escalonada no nó pode, então, ser medida em termos do número destas unidades de produto padrão.[11] A convenção da *uppa* ou da *uppp*, é estabelecida para um nó, através de um conjunto de factores de conversão não negativos r_j,[9] em que cada r_j indica quantas unidades produto padrão de capacidade cada tarefa requer para poder entregar uma unidade de produto de saída p_j do nó. Se uma tarefa, a ser escalonada num nó, deve entregar uma quantidade de q_j unidades do produto p_j do nó, o número de unidades de produto padrão de capacidade (*uppa* ou *uppp*) do nó que a tarefa requer é de $q_j * r_j$.

A medida da capacidade de acumulação é uma medida de quantidade de espaço (ou área, ou peso ou outra grandeza considerada relevante no armazenamento) utilizado no armazenamento de produtos. Por exemplo, suponha-se que um nó acumulador tem dois produtos de saída p_x e p_y e que cada unidade de p_y ocupa duas vezes o espaço de armazenamento que cada unidade de p_x ocupa. Se o nó estivesse dedicado a acumular apenas um dos dois produtos, no máximo acumularia para p_x o dobro do número de unidades de p_y. Se internamente p_x é tomado como produto padrão, uma *uppa* do nó equivale a uma unidade de p_x; cada unidade de p_x consome

[10] Trata-se, portanto, de uma *taxa de processamento* (*i.e.*, uma quantidade de produto por unidade de tempo).

[11] Em certa literatura sobre redes de produção e distribuição (ver, por exemplo, [Williams 1981]) a capacidade de cada nó é medida numa unidade, igual para todos os nós da rede, que é a quantidade necessária para satisfazer a procura do produto final em todos os nós de retalho da rede durante um intervalo de tempo fixo. Esta opção é demasiado limitativa pois pressupõe uma procura constante e não serve para redes com vários produtos finais em que o mesmo produto intermédio pode ser incorporado em mais de um produto final em quantidades diferentes.

então uma unidade de capacidade ($r_x=1$) e cada unidade de p_y consome duas unidades de capacidade do nó ($r_y=2$).

A medida da capacidade de processamento é uma medida de quantidade por unidade de tempo, ou taxa de processamento. Por exemplo, suponha-se que um nó processador tem dois produtos de saída p_x e p_y e que cada unidade de p_y é processada no dobro do tempo de cada unidade de p_x. Se o nó estivesse dedicado a processar apenas um dos dois produtos, no máximo e num mesmo período processaria para p_x o dobro do número de unidades de p_y. Se internamente p_x é tomado como produto padrão, uma *uppp* do nó equivale a uma unidade de p_x; cada unidade de p_x consome então uma unidade de capacidade ($r_x=1$) e cada unidade de p_y consome duas unidades de capacidade do nó ($r_y=2$).

3.4.1.4 Perfil de Capacidade

Um perfil de capacidade indica os valores de capacidade em cada período de um horizonte temporal (os valores são constantes dentro de cada período). Representa uma evolução da capacidade ao longo do tempo, isto é, uma história possível da evolução temporal da capacidade.[12] Um perfil de capacidade é uma função de inteiros para inteiros que mapeia períodos temporais para valores de capacidade.

Formalmente, define-se perfil de capacidade num horizonte temporal $<t_x, t_y>$ como sendo um conjunto ordenado:

$\{s_x, \ldots, s_k, \ldots, s_{y-1}\}$

com um valor s_k associado a cada período temporal t_k tal que $t_k \geq t_x$ e $t_k < t_y$. Cada s_k é denominado o *segmento de perfil de capacidade* para o período t_k. Cada segmento s_k é um triplo:

$<t_k, c_k, \mathcal{O}_k>$

quer dizer o perfil de capacidade é da forma:

$\{<t_x, c_x, \mathcal{O}_x>, \ldots, <t_k, c_k, \mathcal{O}_k>, \ldots, <t_{y-1}, c_{y-1}, \mathcal{O}_{y-1}>\}$

em que, para cada segmento s_k:

- t_k é o período do segmento;
- c_k é o valor de capacidade associado ao segmento no período t_k, mais adiante relacionado com $\mathcal{O}_k$;
- $\mathcal{O}_k$ é o conjunto de tarefas[13] associado ao segmento no período t_k, que contém todas as tarefas com capacidade afectada no período t_k. Para o caso de um perfil de capacidade máxima, $\mathcal{O}_k$ é um conjunto vazio, para qualquer k (neste caso, os conjuntos $\mathcal{O}_k$ não são utilizados).

A relação que existe entre os valores c_k e $\mathcal{O}_k$ de um perfil de capacidade utilizada para um dado produto é descrita como se segue.

Seja $<t_k, c_k, \mathcal{O}_k>$ um qualquer segmento de um perfil de capacidade utilizada de um produto p num nó v para o período t_k e seja $\mathcal{O}_k=\{\mathbb{O}_1, \ldots, \mathbb{O}_i, \ldots, \mathbb{O}_n\}$, tendo as tarefas $\mathbb{O}_i$

[12] Ver por exemplo [Zweben 1994b] ou [Williams 1986].
[13] Ver a secção 3.7.

eventos de saída definidos por $<\{<p_i, q_i>\}, t_k, v>$.[13] Então, para a capacidade de acumulação, tem-se:

$$c_k = \sum_{i=1}^{n} q_i * r_i$$

em que r_i é o factor de conversão de unidades de quantidade do produto p_i para *uppa* para o nó v. Para capacidade de processamento, para cada tarefa $\mathcal{O}_i$, é estipulada uma quantidade de capacidade qw_i (em *uppp*), constante durante o intervalo da tarefa e a afectar a esta[14] e tem-se:

$$c_k = \sum_{i=1}^{n} qw_i * r_i$$

Os valores c_k e $\mathcal{O}_k$ de um perfil de capacidade utilizada, ou de um perfil de capacidade disponível, podem ser alterados com o escalonamento de tarefas novas, ou cancelamento de tarefas escalonadas.

Um conjunto de tarefas escalonadas num nó de capacidade que originam um conflito de capacidade num mesmo instante de tempo é denominado um *conjunto conflito*.

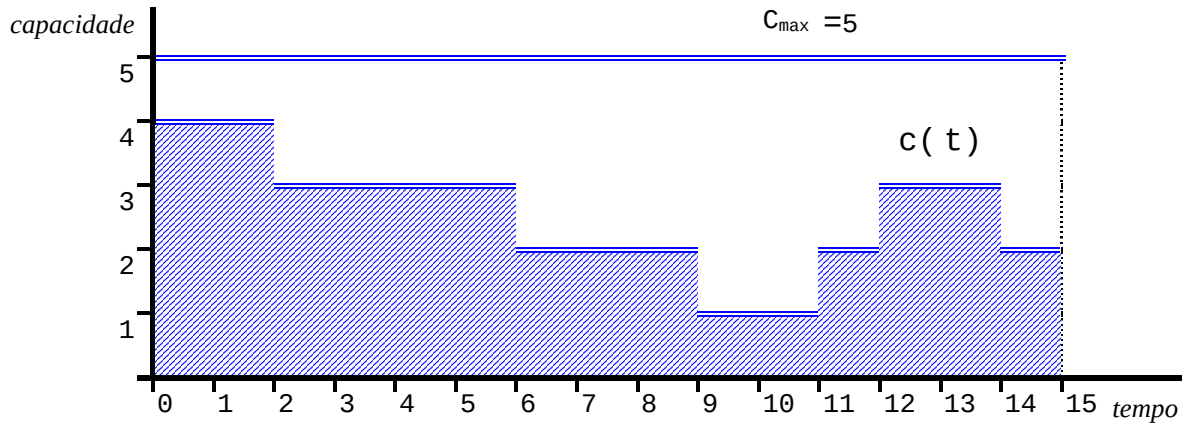


Figura 3-2- Perfil de capacidade máxima e perfil de capacidade utilizada (primeiro exemplo).

Uma forma de visualizar os valores de capacidade num certo horizonte temporal e de visualizar como eles se alteram mediante as decisões de afectação de capacidade no escalonamento é a que emprega uma representação gráfica dos perfis de capacidade. A título de exemplo, representam-se na Figura 3-2 um perfil de capacidade máxima $C_{max}(t)=5$, constante (área entre o eixo do tempo e a linha horizontal a 5 unidades de capacidade) e um perfil de capacidade utilizada, $c(t)$ (área a cheio), no horizonte temporal $<0,15>$ (que abrange os períodos de 0 a 14). O perfil de capacidade disponível pode ser visto como sendo representado pela área a branco. Nesta figura pode-se ver que a capacidade utilizada, por exemplo, nos intervalos $<2,6>$, $<6,9>$ e $<12,14>$ é, respectivamente de 3, 2 e 3 unidades de capacidade (sendo a capacidade disponível de 2, 3, e 2 unidades de capacidade nos mesmos intervalos, respectivamente).

[14] Ver a secção 3.4.1.8.

Para uma capacidade de acumulação estes valores significam quantidades de um produto acumuladas nos intervalos acima referidos. Para uma capacidade de processamento estes valores significam quantidades de um produto produzidas ou transportadas em cada período abrangido por cada intervalo referido. No intervalo <2,6> (períodos 2, 3, 4 e 5), por exemplo, c(t)=3 o que significa que 3 unidades de capacidade estão sendo utilizadas e 2 estão livres. Para uma capacidade particular isto significaria também que:

– Para uma capacidade do tipo A, 3 unidades de produto padrão de acumulação estão armazenadas e há ainda espaço para mais 2.

– Para uma capacidade do tipo W, 3 unidades de produto padrão de processamento estão sendo processadas em cada período (totalizando 4*3=12 unidades processadas entre t=2 e t=6), podendo ainda processar-se adicionalmente 2 unidades em cada período.

3.4.1.5 Operações com Segmentos de um Perfil de Capacidade

As operações primitivas sobre segmentos de perfil de capacidade são:

TEMPO(<t,c,$\mathcal{O}$>) ::= t.

VAL(<t,c,$\mathcal{O}$>) ::= c.

TAREFAS(<t,c,$\mathcal{O}$>) ::= $\mathcal{O}$.

SEGMENTO(t,c,$\mathcal{O}$) ::= <t,c,$\mathcal{O}$>.

3.4.1.6 Operações com Perfis de Capacidade

As operações primitivas sobre perfis de capacidade são (c(t) denota um perfil de capacidade):

TS(c(t)) ::= TEMPO(PRIMEIRO(c(t))).

TE(c(t)) ::= TEMPO(ÚLTIMO(c(t))) + 1.

SEGMENTO(t_k,c(t)) ::= NÉSIMO(c(t),t_k-TS(c(t))).

PERFIL-C(t_x,t_y) ::=
c(t) : (|c(t)|=t_y-t_x) $\wedge$
($\forall_{s_k}$ ($s_k \in$ c(t)) :
(s_k=SEGMENTO(t_x+POSIÇÃO(c(t),s_k),0,{}))).

TS, TE e SEGMENTO são operações selectoras. TS e TE produzem, respectivamente, o tempo de início e o tempo de fim do horizonte temporal de um perfil de capacidade dado; SEGMENTO produz o segmento do perfil associado a um instante dado. PERFIL-C é uma operação construtora.

A operação ALTERA-SEGMENTO descreve a modificação de um perfil de capacidade $c(t)$ por modificação do segmento no período t_k, dados os novos valores para o valor de capacidade (c_k') e o conjunto de tarefas ($\mathcal{O}_k'$) a associar ao segmento. Esta operação é definida como se segue:

$$\text{ALTERA-SEGMENTO}(t_k, c(t), c_k', \mathcal{O}_k') ::= \text{SUBSTITUI}(c(t), \text{SEGMENTO}(t_k, c(t)), \text{SEGMENTO}(t_k, c_k', \mathcal{O}_k')).$$

Esta operação vai servir para descrever a modificação de um perfil de capacidade utilizada por afectação e desafectação de capacidade a tarefas e a modificação do perfil de capacidade máxima (ver, em secções a seguir, para o primeiro caso as operações CARREGA-UTIL e DESCARREGA-UTIL e, para o segundo caso, a operação ALTERA-MAX).

3.4.1.7 Operação de Geração de um Conjunto de Perfis de Capacidade

Para gerar o conjunto dos perfis de capacidade utilizada de um nó de capacidade, define-se a operação PERFIS-UTIL que, dados os limites do horizonte temporal, t_x e t_y ($t_x<t_y$) e o número de perfis n ($n>0$), produz um conjunto de n perfis de capacidade, cada um com horizonte temporal $<t_x, t_y>$ e sem tarefas afectadas. A operação PERFIS-UTIL é definida como se segue:

$$\text{PERFIS-UTIL}(t_x, t_y, n) ::= \mathcal{C}(t) : (|\mathcal{C}(t)|=n) \wedge (\forall_{c(t)} (c(t)\in\mathcal{C}(t)) : (c(t)=\text{PERFIL-C}(t_x, t_y))).$$

3.4.1.8 Afectação de Capacidade

O escalonamento de uma tarefa num nó é efectivado fazendo uma *afectação de capacidade* do nó à tarefa. A operação inversa desfaz o escalonamento de uma tarefa a que previamente foi feita uma afectação de capacidade do nó e é levada a cabo fazendo uma *desafectação de capacidade* do nó à tarefa.

Para afectar capacidade de acumulação a uma tarefa $\mathbb{O}$, a realizar por um nó acumulador, é necessário dispor do número de unidades de produto padrão de acumulação q do produto a acumular[15] e dos instantes inicial e final da tarefa, t_s e t_e ($t_e>t_s$). A afectação é dita possível se em todos os períodos de t_s a t_e-1 (incluindo estes) houver capacidade disponível igual ou superior ao menor inteiro que é maior ou igual a q, isto é, se:

$C(t_k) \geq \text{CEILING}(q)$ para $t_k \geq t_s$ e $t_k < t_e$

Ao ser realizada a afectação de capacidade, o perfil de capacidade utilizada $c(t)$, para o produto em questão, é modificado, resultando dessa modificação um perfil $c'(t)$, tal que:

$c_k' = c_k$ e $\mathcal{O}_k' = \mathcal{O}_k$ para $t_k < t_s$ ou $t_k \geq t_e$ e

$c_k' = c_k + \text{CEILING}(q)$ e $\mathcal{O}_k' = \mathcal{O}_k \cup \{\mathbb{O}\}$ para $t_k \geq t_s$ e $t_k < t_e$

---

[15] *I.e.*, $q=q_j * r_j$, em que q_j é a quantidade do produto a entregar pela tarefa e r_j o factor de conversão de unidades de capacidade para esse produto para o nó em questão.

Por consequência, também a capacidade disponível resultará modificada no intervalo $<t_s, t_e>$.

Por exemplo, assumindo capacidade de acumulação, para o estado de capacidade representado na Figura 3-2, a afectação de capacidade com $q=1$, $t_s=3$ e $t_e=11$ (duração 8) é possível e, após realizada altera os perfis para o estado representado na Figura 3-3.

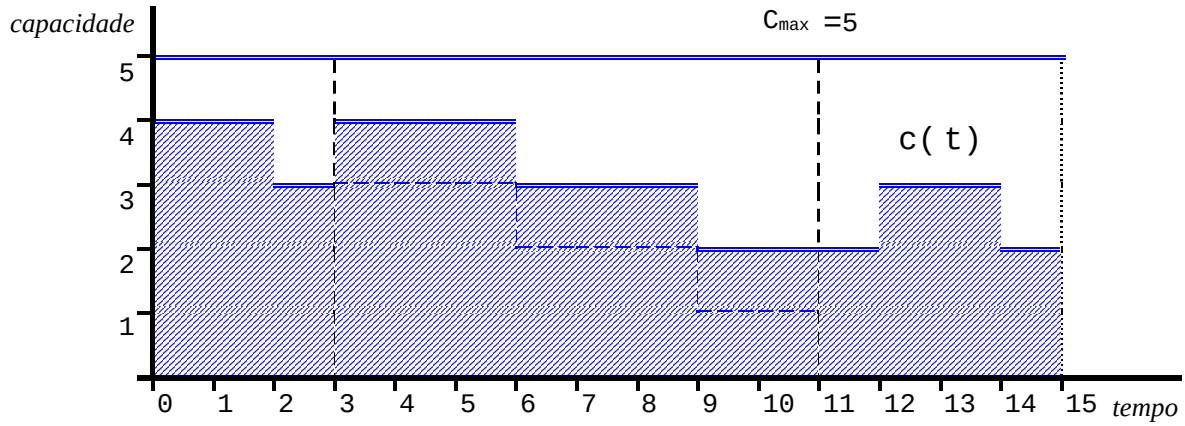


Figura 3-3- Perfil de capacidade máxima e perfil de capacidade utilizada (segundo exemplo).

Para afectar capacidade de processamento a uma tarefa $\mathcal{O}$, a realizar por um nó processador, é necessário dispor do número de unidades de produto padrão de processamento q do produto a processar,[15] do instante inicial requerido para a tarefa t_s e da quantidade de capacidade a afectar à tarefa qw (um inteiro positivo tal que $qw \leq q$, também em unidades de produto padrão de processamento), constante durante o intervalo da tarefa. A afectação é dita possível se existirem períodos consecutivos a partir de t_s (incluindo o período t_s), com capacidade disponível igual ou superior a qw, em número necessário para processar toda a quantidade q, isto é, se:

$C(t_k) \geq qw$ para $t_k \geq t_s$ e $t_k < t_s + CEILING(q/qw)$

sendo a duração da tarefa dada por $CEILING(q/qw)$ e t_e dado por $t_e = t_s + CEILING(q/qw)$. Alternativamente, em vez do instante inicial pode ser dado o instante final para a tarefa t_e, sendo o instante inicial calculado a partir de $t_s = t_e - CEILING(q/qw)$.

Ao ser realizada a afectação de capacidade, o perfil de capacidade utilizada $c(t)$, para o produto em questão, é modificado, resultando dessa modificação um perfil $c'(t)$, tal que:

$c_k' = c_k$ e $\mathcal{O}_k' = \mathcal{O}_k$ para $t_k < t_s$ ou $t_k \geq t_e$ e

$c_k' = c_k + qw$ e $\mathcal{O}_k' = \mathcal{O}_k \cup \{\mathcal{O}\}$ para $t_k \geq t_s$ e $t_k < t_e$

Por consequência também a capacidade disponível resultará modificada no intervalo $<t_s, t_e>$.

Por exemplo, assumindo uma capacidade de processamento, para o estado de capacidade representado na Figura 3-2, a afectação de capacidade com $q=8$, $t_s=3$ e $qw=1$ é possível e,

após realizada altera os perfis para o estado representado na Figura 3-3, resultando numa duração de tarefa de 8 (no intervalo <3,11>).

Note-se que, enquanto que a duração de uma tarefa que requer capacidade de tipo A tem de ser imposta exogenamente (a capacidade de tipo A é estática e serve para actuar como tampão), para uma tarefa que requer capacidade de tipo W, a duração depende da quantidade de capacidade qw que se coloca disponível para a tarefa.

3.4.1.9 Operações de Carga e Descarga com Perfis de Capacidade Utilizada

A operação que se define a seguir, modifica um perfil de capacidade utilizada dado c(t), afectando a quantidade de capacidade cval (um inteiro tal que cval>0) a uma tarefa $\mathbb{O}$,[16] desde um período t_m até um período t_n-1 (ambos incluídos e com $t_n > t_m$). Esta operação é aplicável só se $t_m \geq TS(c(t))$ e $t_n \leq TE(c(t))$.

$$
\begin{aligned}
&CARREGA\text{-}UTIL(c(t), t_m, t_n, cval, \mathbb{O}) ::= \\
&c'(t) : (|c'(t)| = |c(t)|) \wedge \\
&\quad (TS(c'(t)) = TS(c(t))) \wedge (TE(c'(t)) = TE(c(t))) \wedge \\
&\quad (\forall_{t_i}\, (t_i \geq TS(c'(t)), t_i < t_m) : \\
&\qquad (SEGMENTO(t_i, c'(t)) = SEGMENTO(t_i, c(t)))) \wedge \\
&\quad (\forall_{t_j}\, (t_j \geq t_n, t_j < TE(c'(t))) : \\
&\qquad (SEGMENTO(t_j, c'(t)) = SEGMENTO(t_j, c(t)))) \wedge \\
&\quad (\forall_{t_k}\, (t_k \geq t_m, t_k < t_n) : \\
&\qquad (SEGMENTO(t_k, c'(t)) = \\
&\qquad\quad SEGMENTO(\, t_k, \\
&\qquad\qquad\quad VAL(SEGMENTO(t_k, c(t))) + cval, \\
&\qquad\qquad\quad TAREFAS(SEGMENTO(t_k, c(t))) \cup \{\mathbb{O}\}))).
\end{aligned}
$$

O efeito produzido pela operação $CARREGA\text{-}UTIL(c(t), t_m, t_n, cval, \mathbb{O})$ é equivalente a iterar a operação:

$$
\begin{aligned}
&ALTERA\text{-}SEGMENTO(\; t_k, \\
&\qquad\qquad\quad c(t), \\
&\qquad\qquad\quad VAL(SEGMENTO(t_k, c(t))) + cval, \\
&\qquad\qquad\quad TAREFAS(SEGMENTO(t_k, c(t))) \cup \{\mathbb{O}\})
\end{aligned}
$$

sobre o perfil c(t), para cada um dos períodos t_k tal que $t_k \geq t_m$ e $t_k < t_n$.

A operação DESCARREGA-UTIL, a inversa de CARREGA-UTIL, modifica um perfil de capacidade utilizada por desafectação, desde um período t_m até um período t_n-1 (ambos

[16] cval deve ser igual à quantidade de capacidade a afectar a $\mathbb{O}$ em cada período t_k (CEILING(q), no caso de capacidade de acumulação e qw, no caso de capacidade de processamento, antes descritas) e t_m e t_n devem ser, respectivamente o tempo de início e de fim de $\mathbb{O}$.

incluídos e com $t_n>t_m$), da quantidade de capacidade cval (um inteiro tal que cval>0) a uma tarefa $\mathbb{O}$ [17] e é definida como se segue:

DESCARREGA-UTIL(c(t),t_m,t_n,cval,$\mathbb{O}$) ::=
c'(t) : (|c'(t)|=|c(t)|) $\wedge$
(TS(c'(t))=TS(c(t))) $\wedge$ (TE(c'(t))=TE(c(t))) $\wedge$
($\forall_{t_i}$ ($t_i \geq$TS(c'(t)),$t_i<t_m$) :
(SEGMENTO(t_i,c'(t))=SEGMENTO(t_i,c(t)))) $\wedge$
($\forall_{t_j}$ ($t_j \geq t_n$,$t_j<$TE(c'(t))) :
(SEGMENTO(t_j,c'(t))=SEGMENTO(t_j,c(t)))) $\wedge$
($\forall_{t_k}$ ($t_k \geq t_m$,$t_k<t_n$) :
(SEGMENTO(t_k,c'(t))=
SEGMENTO(t_k,
VAL(SEGMENTO(t_k,c(t)))-cval,
TAREFAS(SEGMENTO(t_k,c(t)))\{$\mathbb{O}$}))).

O efeito produzido pela operação DESCARREGA-UTIL(c(t),t_m,t_n,cval,$\mathbb{O}$) é equivalente a iterar a operação:

ALTERA-SEGMENTO(t_k,
c(t),
VAL(SEGMENTO(t_k,c(t)))-cval,
TAREFAS(SEGMENTO(t_k,c(t)))\{$\mathbb{O}$})

sobre o perfil c(t), para cada um dos períodos t_k tal que $t_k \geq t_m$ e $t_k<t_n$.

As operações CARREGA-UTIL e DESCARREGA-UTIL apresentam a seguinte propriedade:

DESCARREGA-UTIL(CARREGA-UTIL(c(t),t_m,t_n,cval,$\mathbb{O}$),t_m,t_n,cval,$\mathbb{O}$)=
c(t)

e:

CARREGA-UTIL(DESCARREGA-UTIL(c(t),t_m,t_n,cval,$\mathbb{O}$),t_m,t_n,cval,$\mathbb{O}$)=
c(t)

isto é, uma afectação (desafectação) a uma tarefa imediatamente seguida da desafectação (afectação) da mesma tarefa deixa o perfil de capacidade utilizada no mesmo estado.

3.4.1.10 Operações de Modificação de um Perfil de Capacidade Máxima

Esta operação modifica um perfil de capacidade máxima dado C_{max}(t), estabelecendo, para um intervalo de tempo abrangendo os períodos de t_m a t_n-1 (com $t_n>t_m$), a quantidade de capacidade máxima cval (um inteiro tal que cval≥0). Esta operação é aplicável só se $t_m \geq$TS(c(t)) e $t_n \leq$TE(c(t)).

ALTERA-MAX(C_{max}(t),t_m,t_n,cval) ::=

[17] À qual se assume que foi anteriormente afectada a quantidade de capacidade cval de t_m a t_n-1, no perfil.

$$C_{max}'(t) : (|C_{max}'(t)| = |C_{max}(t)|) \wedge$$
$$(TS(C_{max}'(t)) = TS(C_{max}(t))) \wedge (TE(C_{max}'(t)) = TE(C_{max}(t))) \wedge$$
$$(\forall_{t_i} (t_i \geq TS(C_{max}'(t)), t_i < t_m) :$$
$$(SEGMENTO(t_i, C_{max}'(t)) = SEGMENTO(t_i, C_{max}(t)))) \wedge$$
$$(\forall_{t_j} (t_j \geq t_n, t_j < TE(C_{max}'(t))) :$$
$$(SEGMENTO(t_j, C_{max}'(t)) = SEGMENTO(t_j, C_{max}(t)))) \wedge$$
$$(\forall_{t_k} (t_k \geq t_m, t_k < t_n) :$$
$$(SEGMENTO(t_k, C_{max}'(t)) = SEGMENTO(t_k, cval, \{}))).$$

O efeito produzido pela operação ALTERA-MAX($C_{max}(t), t_m, t_n$, cval) é equivalente a iterar a operação:

ALTERA-SEGMENTO($t_k, C_{max}(t)$, cval, {})

sobre o perfil $C_{max}(t)$, para cada um dos períodos t_k tal que $t_k \geq t_m$ e $t_k < t_n$.

3.4.1.11 Reserva de Capacidade

Previamente à afectação de capacidade de um nó para uma tarefa, quando o intervalo de tempo de escalonamento da tarefa ainda não é conhecido, mas é sabido que deve estar contido num dado horizonte temporal, pode fazer-se uma *reserva de capacidade* para a tarefa para o horizonte temporal. Este, é uma janela temporal na qual o intervalo de tempo da tarefa terá de estar temporalmente contido. Isto é, se o horizonte temporal de reserva é um intervalo $<t_a, t_b>$, o intervalo de tempo de escalonamento da tarefa $<t_s, t_e>$ terá de ser tal que $t_s \geq t_a$ e $t_e \leq t_b$.

Uma reserva de capacidade de acumulação de q unidades, de duração Δt, para um dado horizonte temporal entre $<t_a, t_b>$ é possível se existe pelo menos um intervalo $<t_s, t_e>$ tal que $t_s \geq t_a$, $t_e \leq t_b$ e $t_e - t_s = \Delta t$, em que a afectação de capacidade de acumulação de q unidades é possível.

Uma reserva de capacidade de processamento de q unidades utilizando qw unidades por período, para um dado horizonte temporal entre t_a e t_b é possível se existe pelo menos um intervalo $<t_s, t_e>$ tal que $t_e = t_s + CEILING(q/qw)$ e $t_s \geq t_a$, $t_e \leq t_b$, em que a afectação de capacidade de processamento de q unidades, utilizando qw unidades por período, é possível.

Podem conceber-se, à semelhança dos perfis de capacidade máxima e utilizada, *perfis de capacidade reservada*, que indicam os valores de capacidade reservada em conjuntos de períodos consecutivos de um dado horizonte temporal. No presente trabalho, o uso de reservas de capacidade é, no entanto, relegado para trabalho futuro.

Juntamente com os perfis de capacidade utilizada e o perfil de capacidade máxima, os perfis de capacidade reservada permitiriam saber se a soma da quantidade de capacidade afectada a tarefas com a capacidade reservada excede a capacidade disponível em cada período e, se não excede, qual a quantidade de capacidade que é ainda possível reservar. Deste modo, poderia basear-se a decisão de aceitação, ou não aceitação, de mais uma tarefa para escalonar na existência, ou não existência, de capacidade reservável suficiente para a necessidade da tarefa.

Uma outra utilização possível dos perfis de capacidade reservada é a de servir de suporte a um mecanismo de prioridades de escalonamento e re-escalonamento. Estas prioridades permitem

decidir, em cada momento, qual a tarefa a escalonar a seguir, de um conjunto de tarefas a escalonar e qual o intervalo de tempo mais apropriado para a tarefa.

Por exemplo, o método heurístico de escalonamento micro-oportunista descrito em [Sadeh 1995b] e [Sadeh 1994] baseia-se numa heurística que escolhe a tarefa a escalonar no recurso mais constrangido para a qual as alternativas de escalonamento, apelidadas *reservas possíveis*, parecem ser mais restritas. Neste método usam-se prioridades dinâmicas baseadas em valores de um *perfil de procura* das tarefas por um recurso que permitem identificar (a tarefa a considerar prioritariamente para escalonamento num recurso é a tarefa que mais contribui para valores de pico do perfil de procura).[18] Um perfil de capacidade reservada pode servir para representar um perfil de procura para apoio a um mecanismo de prioridades de escalonamento ou de re-escalonamento baseado na capacidade.

3.4.1.12 Realização Computacional de Perfis de Capacidade

Numa realização computacional há, por razões de economia de memória ocupada, vantagem em representar um perfil capacidade de forma compacta. Uma solução para este problema é, em vez de representar explicitamente para cada período temporal t_k um triplo $<t_k, c_k, \mathcal{O}_k>$, adoptar uma representação em que cada segmento s_k é um triplo $<h_k, c_k, \mathcal{O}_k>$, sendo h_k um intervalo de tempo associado ao segmento. O intervalo h_k inclui o número máximo de períodos consecutivos, na vizinhança temporal de qualquer instante t tal que $\mathrm{DURANTE?}(t, h_k)$, em que o valor de capacidade do perfil é igual a c_k. Isto quer dizer que, se o perfil abrange temporalmente o período $\mathrm{TS}(h_k)-1$, este período é representado por um segmento s_{k-1} (antecessor imediato de s_k) e é sempre $c_{k-1} \neq c_k$; se o perfil abrange temporalmente o período $\mathrm{TE}(h_k)$, este período é representado por um segmento s_{k+1} (sucessor imediato de s_k) e é sempre $c_{k+1} \neq c_k$.

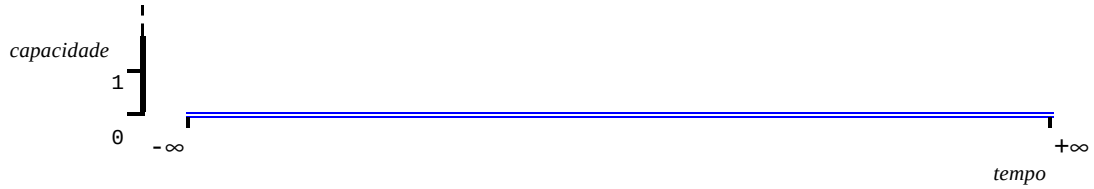
Esta opção de representação terá, em geral, a vantagem de reduzir o número de segmentos de um perfil representados na memória. No entanto, operações de carga e descarga sucessivas (ver a secção 3.4.1.9) podem causar fragmentação no perfil de capacidade, originando segmentos consecutivos com iguais valores de capacidade. Para manter o pressuposto desta forma de representação (períodos consecutivos com igual valor de capacidade são representados através do mesmo segmento de perfil) há então que prever, para aquelas operações, procedimentos adicionais que evitem a fragmentação. Estes procedimentos devem garantir a fusão de segmentos consecutivos com iguais valores de capacidade, resultantes de uma operação de carga ou descarga com o perfil.

Adicionalmente, pode ser conveniente também não limitar os extremos do horizonte temporal de escalonamento e representar perfis de capacidade que abrangem um número de períodos temporais infinitos, com a vantagem de se poderem efectuar operações de carga e descarga num perfil em quaisquer períodos de tempo previamente desconhecidos. É possível adoptar uma forma de representação coerente que emprega um número de segmentos finito, quer o horizonte temporal seja finito ou não, se se assumir a opção acima descrita (segmentos de um perfil podem abranger mais de um período).

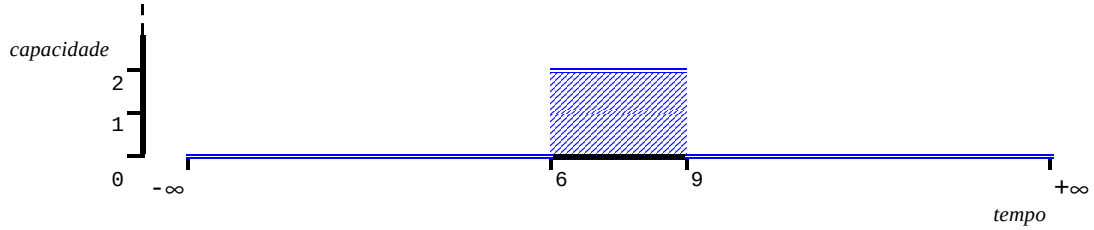
---

[18] Conforme resultados de testes exibidos em [Sadeh 1995b] e [Sadeh 1994] este método parece ser eficaz na determinação de boas soluções de escalonamento. Uma versão distribuída é descrita em [Sycara 1991a] e [Sycara 1991b].

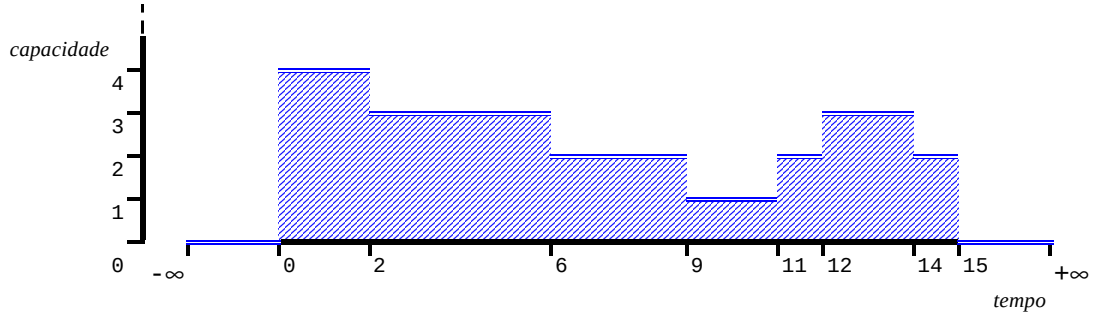
Com a forma de representação que se pretende sugerir terão de ser possíveis intervalos de tempo abertos à esquerda e/ou abertos à direita (*i.e.*, com extremos infinitos) e com duração infinita (nos casos em que pelo menos um dos extremos é infinito). Mais concretamente, para qualquer intervalo $\mathbb{h}$ deverá ser possível ter-se TS($\mathbb{h}$)=-∞ e/ou TE($\mathbb{h}$)=+∞ e, ainda, DURAÇÃO($\mathbb{h}$)=+∞ para os casos em que TS($\mathbb{h}$)=-∞ e/ou TE($\mathbb{h}$)=+∞. Estes aspectos exigem uma extensão da representação da dimensão tempo, que use, para além de números inteiros, dois valores de tempo (simbólicos) especiais +∞ e -∞. Estes têm o significado de um instante "infinitamente futuro" — o maior valor de tempo — e um instante "infinitamente passado" — o menor valor de tempo — respectivamente. A relação de ordem entre valores de tempo deve ser ampliada de modo a acomodar estes valores especiais e uma extensão das operações aritméticas de modo a se poder operar com tempos e durações infinitos é também necessária. Para além disso, deverá permitir-se que os segmentos extremos de um perfil tenham intervalos abertos à esquerda, para o caso do primeiro segmento, e à direita, para o caso do último segmento, mas apenas nesses casos.



a) Imediatamente após a criação.



b) Após carga de 2 unidades de capacidade no intervalo <6,9>.



c) Após várias operações de carga/descarga.

Figura 3-4- Perfil de capacidade com horizonte temporal aberto à esquerda e à direita.

Adoptando estas modificações podem representar-se perfis de capacidade com horizontes temporais abertos à esquerda, à direita ou à esquerda e à direita. Para o primeiro caso, o primeiro segmento do perfil tem um intervalo de tempo aberto à esquerda; para o segundo caso, o último segmento tem um intervalo de tempo aberto à direita; para o último caso, o primeiro segmento tem um intervalo de tempo aberto à esquerda e o último um intervalo de tempo aberto à direita.

A realização computacional das soluções apresentadas na presente secção é descrita em detalhe em [Reis 1998]. A representação de tempo baseada em pontos temporais e em intervalos possivelmente abertos num horizonte temporal infinito é inspirada no mapa temporal (*time map*, uma estrutura adequada a gerir asserções que correspondem à ocorrência de eventos e à persistência dos seus efeitos ao longo do tempo) de [Dean 1987].

A Figura 3-4 ilustra estados de um perfil de capacidade utilizada com horizonte temporal aberto à esquerda e à direita, assumindo que se adoptaram as opções descritas na presente secção. Imediatamente após a criação o perfil contém apenas um segmento, como representa a Figura 3-4-a). Este segmento único tem um intervalo aberto à esquerda e à direita (<-∞,+∞>, com duração +∞ [19]) e um valor de capacidade 0. Após a carga de 2 unidades de capacidade no intervalo <6,9>, na Figura 3-4-b), o perfil passa a conter três segmentos, com intervalos <-∞,6>, <6,9> e <9,+∞> com valores de capacidade 0, 2 e 0, respectivamente. A Figura 3-4-c) representa o estado do perfil após várias outras operações de carga e/ou descarga subsequentes.

3.4.1.13 Acumulador e Processador

A Figura 3-5 contém a representação esquemática de um nó de capacidade. As setas representam possíveis entradas de produtos de entrada e possíveis saídas de produtos de saída. No contexto de uma representação de rede física estas setas correspondem a arcos da rede (no máximo um arco por cada par de nós da rede). TN designa o tipo de nó de capacidade, $C_{max}(t)$ e $c_1(t),...c_i(t),...c_n(t)$ representam, respectivamente, a capacidade máxima do nó e a parte dessa capacidade utilizada para cada produto de saída $p_1,...p_i,...p_n$, em cada instante t. O tipo de nó, TN, depende do tipo de capacidade e quanto ao tipo, um nó pode ser classificado como:

- *Acumulador*, *nó acumulador*, ou nó de tipo S - Dispõe de capacidade de acumulação, isto é, capacidade de tipo A. O nó realiza tarefas de acumulação (tarefas do tipo S);[20]
- *Processador*, *nó processador*, ou nó de tipo P ou T - Dispõe de capacidade de processamento, isto é, capacidade de tipo W. Um *produtor*, ou *nó produtor*, ou nó de tipo P, realiza tarefas de produção (tarefas do tipo P);[20] um *transportador*, ou *nó transportador*, ou nó de tipo T, realiza tarefas de transporte (tarefas do tipo T).[20]

Um nó acumulador armazena produtos. No armazenamento, os produtos à entrada e à saída de um acumulador são os mesmos e existe conservação da sua quantidade, diferindo na coordenada tempo.[21] Um nó acumulador permite modelar um armazém ou espaço de armazenamento de materiais, produtos intermédios, ou produtos acabados de uma rede de produção distribuição.

---

[19] A representação deste estado inicial é incoerente se não se adoptar a primeira solução descrita nesta secção. Se a solução referida não fosse adoptada cada segmento representaria a capacidade num único período temporal que, por definição tem duração unitária; mas então, no estado inicial, o período do único segmento (o período $t_{-\infty}$) deveria "durar" de -∞ a +∞.

[20] Ver a secção 3.7.

[21] Há conservação da quantidade no sentido em que a diferença entre as quantidades que entram e as que saem do nó num intervalo de tempo diferem na quantidade acumulada no intervalo. Há diferença na coordenada tempo no sentido em que uma quantidade de produto poderá ficar armazenado mais ou menos tempo.

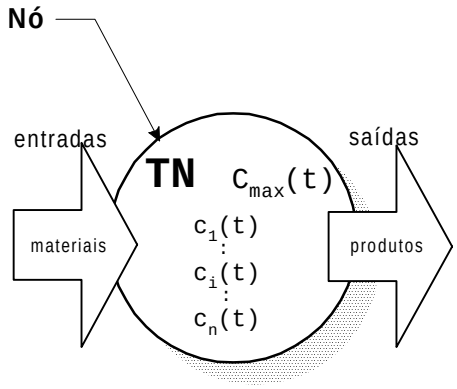
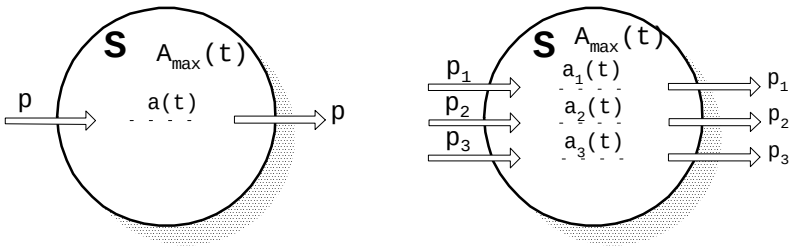


Figura 3-5- Um nó.

Um nó processador processa produtos de entrada para obter produtos de saída, dependendo a duração de uma tarefa de processamento da taxa de processamento dedicada ao produto da tarefa e também da quantidade de produto. No caso de um nó produtor, os produtos de entrada são transformados para obter produtos de saída. Produtos de entrada e de saída são diferentes e para a produção de cada produto de saída serão, em geral, necessários mais de um produto de entrada. Um nó produtor permite modelar uma fábrica. Um nó transportador toma produtos de saída de um outro nó e (por execução de tarefas de transporte) torna-os disponíveis a outro nó. Neste caso, os produtos à entrada e à saída são os mesmos, havendo conservação da sua quantidade (semelhante ao nó acumulador). Um nó transportador permite modelar uma frota de veículos de transporte (camiões, barcos, aviões, etc.).

Quanto ao número de produtos de saída, um nó pode ser classificado como:

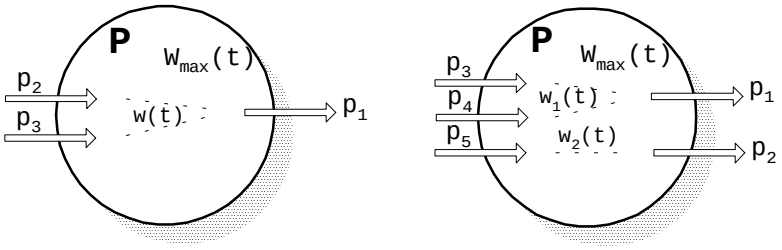
- *Nó dedicado*, ou *nó mono-produto* - Tem um único produto de saída (a capacidade do nó é dedicada a tarefas para entregar um único produto);
- *Nó partilhado*, ou *nó multi-produto* - Tem mais de um produto de saída (a capacidade do nó é partilhada por tarefas que entregam produtos diferentes).



a) Acumulador dedicado (mono-produto). b) Acumulador partilhado (multi-produto).

Figura 3-6- Nós acumuladores.

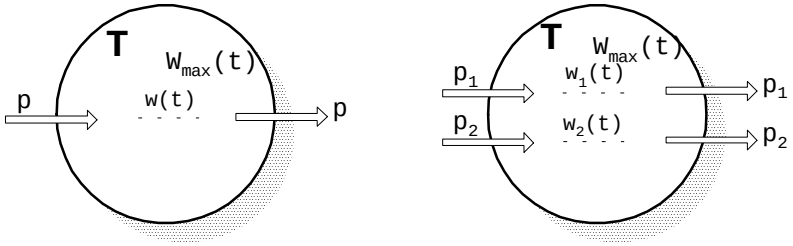
Representam-se esquematicamente na Figura 3-6 um nó acumulador dedicado e um nó acumulador partilhado (neste último caso, com três produtos de saída: p_1, p_2 e p_3). As setas indicando os produtos de saída e de entrada de um nó são usadas mais no contexto de uma representação de rede física e correspondem a arcos de rede.



a) Produtor dedicado (mono-produto). b) Produtor partilhado (multi-produto).

Figura 3-7- Nós produtores.

Um processador é esquematicamente representado na Figura 3-7 e na Figura 3-8. A Figura 3-7-a) representa um produtor dedicado com o produto de saída p_1 e dois produtos de entrada p_2 e p_3; a Figura 3-7-b) representa um produtor partilhado com dois produtos de saída p_1 e p_2 e com os produtos de entrada p_3, p_4 e p_5. A Figura 3-8-a) representa um transportador dedicado e a Figura 3-8-b) um transportador partilhado.



a) Transportador dedicado (mono-produto). b) Transportador partilhado (multi-produto).

Figura 3-8- Nós transportadores.

Formalmente, um nó da classe *nó de capacidade* $\mathbb{V}$ é descrito por um tuplo de oito elementos:

$<v, TN, p, m, b, r, C_{max}(t), c(t)>$

em que os componentes v, TN, p, m e b constituem a interface do nó e os componentes r, $C_{max}(t)$ e $c(t)$ o núcleo do nó. A interface relaciona o nó como seu exterior, sendo:

– $v \in \mathbb{v}$, em que v é um símbolo que identifica univocamente o nó no contexto de uma rede;

– $TN \in \{S, P, T\}$, sendo S, P ou T para nós acumuladores, processadores ou transportadores, respectivamente;

– $\mathcal{p}=\{p_1,\ldots,p_j,\ldots,p_n\}$ (com $\mathcal{p}\subset\mathcal{P}$), em que $\mathcal{p}$ é o conjunto ordenado de produtos de saída do nó. Se $|\mathcal{p}|=1$ ($n=1$) trata-se de um nó dedicado;

– $\mathcal{m}=\{m_1,\ldots,m_i,\ldots,m_m\}$ (com $\mathcal{m}\subset\mathcal{P}$), em que $\mathcal{m}$ é o conjunto ordenado de produtos de entrada do nó;

– $\mathcal{b}=\{b_{1,1},\ldots,b_{1,n},b_{2,1},\ldots,b_{i,j},\ldots,b_{m,n}\}$, a lista de materiais do nó,[22] um conjunto ordenado de números não negativos que corresponde a uma matriz, em que cada elemento $b_{i,j}$ significa a quantidade de produto de entrada m_i ($m_i\in\mathcal{m}$) necessária para entregar uma unidade de produto de saída p_j ($p_j\in\mathcal{p}$). Em nós de tipo S e T tem-se $\mathcal{p}=\mathcal{m}$, correspondendo $\mathcal{b}$ a uma matriz identidade.

O núcleo regista informação relativa à capacidade, incluindo o estado actual de capacidade, sendo:

– $\mathcal{r}=\{r_1,\ldots,r_j,\ldots,r_n\}$ (com $|\mathcal{r}|=|\mathcal{p}|$), um conjunto ordenado de factores de conversão (números racionais não negativos) r_j, um por cada elemento de $\mathcal{p}$, que indica quantas unidades produto padrão de capacidade do nó cada tarefa requer para poder entregar uma unidade de produto de saída p_j ($p_j\in\mathcal{p}$);

– $C_{max}(t)$ é o perfil de capacidade máxima,[23] que regista a capacidade máxima do nó em unidades de produto padrão, em cada período t;

– $\mathcal{c}(t)=\{c_1(t),\ldots,c_j(t),\ldots,c_n(t)\}$ (com $|\mathcal{c}(t)|=|\mathcal{p}|$), um conjunto ordenado de perfis de capacidade utilizada, um para cada produto de saída, em que cada $c_j(t)$ indica a quantidade de capacidade, em unidades de produto padrão, utilizada em tarefas para entregar o produto de saída p_j ($p_j\in\mathcal{p}$) em cada período t.[24]

3.4.1.14 Operações com Nós de Capacidade (Acumuladores ou Processadores)

As operações primitivas sobre nós de capacidade são descritas a seguir. As operações selectoras são:

LOCAL(<v,TN,$\mathcal{p}$,$\mathcal{m}$,$\mathcal{b}$,$\mathcal{r}$,$C_{max}(t)$,$\mathcal{c}(t)$>) ::= v.

TIPO(<v,TN,$\mathcal{p}$,$\mathcal{m}$,$\mathcal{b}$,$\mathcal{r}$,$C_{max}(t)$,$\mathcal{c}(t)$>) ::= TN.

PRODUTOS(<v,TN,$\mathcal{p}$,$\mathcal{m}$,$\mathcal{b}$,$\mathcal{r}$,$C_{max}(t)$,$\mathcal{c}(t)$>) ::= $\mathcal{p}$.

MATERIAIS(<v,TN,$\mathcal{p}$,$\mathcal{m}$,$\mathcal{b}$,$\mathcal{r}$,$C_{max}(t)$,$\mathcal{c}(t)$>) ::= $\mathcal{m}$.

---

[22] $\mathcal{b}$ corresponde a uma lista de materiais (ou *bill-of-materials*).

[23] $A_{max}(t)$ para o caso de capacidade de acumulação e $W_{max}(t)$ para o caso de capacidade de processamento.

[24] $\mathcal{c}(t)=\mathcal{a}(t)$, com $\mathcal{a}(t)=\{\ldots,a_j(t),\ldots\}$ para o caso de capacidade de acumulação e $\mathcal{c}(t)=\mathcal{w}(t)$, com $\mathcal{w}(t)=\{\ldots,w_j(t),\ldots\}$ para o caso de capacidade de processamento. Os limites temporais do horizonte temporal do perfil de capacidade $C_{max}(t)$ e dos perfis de capacidade em $\mathcal{c}(t)$ devem ser iguais.

LISTAS-DE-MATERIAIS(<v,TN,p,m,b,r,$C_{max}(t)$,$c(t)$>) ::= b.

FACTORES-C(<v,TN,p,m,b,r,$C_{max}(t)$,$c(t)$>) ::= r.

PERFIL-C-MAX(<v,TN,p,m,b,r,$C_{max}(t)$,$c(t)$>) ::= $C_{max}(t)$.

PERFIS-C-UTIL(<v,TN,p,m,b,r,$C_{max}(t)$,$c(t)$>) ::= $c(t)$.

A operação construtora é definida por:

NÓ(v,TN,p,m,b,r,$C_{max}(t)$,$c(t)$) ::= <v,TN,p,m,b,r,$C_{max}(t)$,$c(t)$>.

(em que TN deve ser P, T ou S, com os parâmetros p, m, b, r e $c(t)$ de acordo com o tipo de nó e entre si no que respeita ao número e ordem dos seus elementos, e sendo $TS(C_{max}(t))=TS(c_j(t))$ e $TE(C_{max}(t))=TE(c_j(t))$ para qualquer $c_j(t)$ tal que $c_j(t)\in c(t)$).

Definem-se também as seguintes operações ($\mathbb{V}$ denota um nó):

TODOS-PRODUTOS($\mathbb{V}$) ::= PRODUTOS($\mathbb{V}$)$\cup$MATERIAIS($\mathbb{V}$).

(produz o conjunto contendo todos os produtos de entrada e de saída de um nó dado)

RAZÃO-MATERIAL-PRODUTO($\mathbb{V}$,p_j,m_i) ::=
$b_{i,j}$: ($b_{i,j}$=
NÉSIMO(LISTAS-DE-MATERIAIS($\mathbb{V}$),
POSIÇÃO(MATERIAIS($\mathbb{V}$),m_i) * |PRODUTOS($\mathbb{V}$)| +
POSIÇÃO(PRODUTOS($\mathbb{V}$),p_j))).

(produz a quantidade do produto de entrada m_i necessária para que o nó $\mathbb{V}$ torne disponível uma unidade do produto de saída p_j, sendo $m_i\in$MATERIAIS($\mathbb{V}$) e $p_j\in$PRODUTOS($\mathbb{V}$))

FACTOR-C($\mathbb{V}$,p_j) ::=
r_j : (r_j=NÉSIMO(FACTORES-C($\mathbb{V}$),POSIÇÃO(PRODUTOS($\mathbb{V}$),p_j))).

(produz a quantidade de capacidade necessária em cada unidade de tempo para que o nó $\mathbb{V}$ torne disponível uma unidade do produto de saída p_j, sendo $p_j\in$PRODUTOS($\mathbb{V}$))

PERFIL-C-UTIL($\mathbb{V}$,p_j) ::=
$c_j(t)$: ($c_j(t)$=
NÉSIMO(PERFIS-C-UTIL($\mathbb{V}$),POSIÇÃO(PRODUTOS($\mathbb{V}$),p_j))).

(produz o perfil de capacidade utilizada para o produto de saída p_j do nó $\mathbb{V}$, sendo $p_j \in$ PRODUTOS($\mathbb{V}$))

MATERIAIS-PARA-PRODUTO($\mathbb{V}$,p_j) ::=
{ m_i :($m_i \in$ MATERIAIS($\mathbb{V}$)) $\wedge$
(RAZÃO-MATERIAL-PRODUTO($\mathbb{V}$,p_j,m_i)≠0) }.

(produz o conjunto de produtos de entrada necessários ao nó $\mathbb{V}$ para que torne disponível o produto de saída p_j, sendo $p_j \in$ PRODUTOS($\mathbb{V}$))

PRODUTOS-DE-MATERIAL($\mathbb{V}$,m_i) ::=
{ p_j :($p_j \in$ PRODUTOS($\mathbb{V}$)) $\wedge$
(RAZÃO-MATERIAL-PRODUTO($\mathbb{V}$,p_j,m_i)≠0) }.

(produz o conjunto de produtos de saída para cuja entrega, pelo nó $\mathbb{V}$, o produto de entrada m_i é necessário, sendo $m_i \in$ MATERIAIS($\mathbb{V}$))

Definem-se a seguir operações que permitem calcular valores de capacidade (em *uppa*, para nós acumuladores e em *uppp*, para nós processadores) de um nó num dado instante, ou durante um intervalo dados os instantes extremos. As quatro operações que se seguem permitem calcular, para um dado nó $\mathbb{V}$, a capacidade máxima num instante t_k, a capacidade utilizada num instante t_k, a capacidade utilizada num produto p_j (com $p_j \in$ PRODUTOS($\mathbb{V}$)) e num instante t_k e a capacidade disponível num instante t_k, respectivamente.

C-MAX($\mathbb{V}$,t_k) ::= VAL(SEGMENTO(t_k,PERFIL-C-MAX($\mathbb{V}$))).

$$\text{C-UTIL}(\mathbb{V},t_k) ::= \sum_{j=1}^{|\text{PRODUTOS}(\mathbb{V})|} \text{VAL}(\text{SEGMENTO}(t_k,\text{PERFIL-C-UTIL}(\mathbb{V},p_j))).$$

C-UTIL($\mathbb{V}$,p_j,t_k) ::= VAL(SEGMENTO(t_k,PERFIL-C-UTIL($\mathbb{V}$,p_j))).

C-DISP($\mathbb{V}$,t_k) ::= C-MAX($\mathbb{V}$,t_k) - C-UTIL($\mathbb{V}$,t_k).

A seguir define-se a operação C-DISP-MIN, que calcula o valor mínimo da capacidade disponível de um nó dado $\mathbb{V}$, num intervalo de tempo dado pelos instantes extremos t_s e t_e (com $t_s<t_e$). Esta operação vai servir para determinar se há ou não capacidade disponível para escalonar uma tarefa no nó dentro de um dado horizonte temporal, dada a capacidade necessária para a tarefa.

$$\text{C-DISP-MIN}(\mathbb{V},t_s,t_e) ::= \min_{t_k=t_s,\ldots t_e-1} (\text{C-DISP}(\mathbb{V},t_k)).$$

As quatro operações que se seguem permitem calcular valores que correspondem a um total de unidades de capacidade (*uppa*, ou *uppp*) de um nó dado $\mathbb{V}$, durante um intervalo de tempo

dado pelos instantes extremos t_s e t_e (com $t_s<t_e$). Estas operações calculam, respectivamente, o total de capacidade máxima no intervalo, o total de capacidade utilizada no intervalo e o total de capacidade utilizada num produto p_j no intervalo.

$$\texttt{C-MAX-TOTAL}(\mathbb{V},t_s,t_e) ::= \sum_{t_k=t_s}^{t_e-1} \texttt{C-MAX}(\mathbb{V},t_k).$$

$$\texttt{C-UTIL-TOTAL}(\mathbb{V},t_s,t_e) ::= \sum_{t_k=t_s}^{t_e-1} \texttt{C-UTIL}(\mathbb{V},t_k).$$

$$\texttt{C-UTIL-TOTAL}(\mathbb{V},p_j,t_s,t_e) ::= \sum_{t_k=t_s}^{t_e-1} \texttt{C-UTIL}(\mathbb{V},p_j,t_k).$$

As duas operações definidas a seguir permitem calcular, para um dado nó e durante um intervalo dado pelos instantes extremos, a razão entre o total de capacidade utilizada e o total de capacidade máxima e a razão entre o total de capacidade utilizada num produto e o total de capacidade máxima.

```
C-UTIL-FRACÇÃO(𝕍,ts,te) ::=
C-UTIL-TOTAL(𝕍,ts,te) / C-MAX-TOTAL(𝕍,ts,te).

C-UTIL-FRACÇÃO(𝕍,pj,ts,te) ::=
C-UTIL-TOTAL(𝕍,pj,ts,te) / C-MAX-TOTAL(𝕍,ts,te).
```

Para criação de um nó com os perfis de capacidade devidamente iniciados define-se a operação NÓ-INICIAL:

```
NÓ-INICIAL(v,TN,𝒑,𝒎,𝒃,𝒓,tx,ty) ::=
NÓ(v,TN,𝒑,𝒎,𝒃,𝒓,PERFIL-C(tx,ty),PERFIS-UTIL(tx,ty,|𝒑|)).
```

A operação MODIFICA-C-MAX, a seguir definida, modifica o perfil de capacidade máxima de um nó dado $\mathbb{V}$, colocando como valor de capacidade máxima num intervalo, dado pelos instantes extremos t_x e t_y, o valor cval (com $\texttt{cval}\geq 0$). Esta operação é aplicável só se $t_x \geq \texttt{TS(PERFIL-C-MAX}(\mathbb{V}))$ e $t_y \leq \texttt{TE(PERFIL-C-MAX}(\mathbb{V}))$.

```
MODIFICA-C-MAX(𝕍,tm,tn,cval) ::=
NÓ( LOCAL(𝕍),
    TIPO(𝕍),
    PRODUTOS(𝕍),
    MATERIAIS(𝕍),
    LISTAS-DE-MATERIAIS(𝕍),
    FACTORES-C(𝕍),
    ALTERA-MAX(PERFIL-C-MAX(𝕍),tm,tn,cval),
```

```
    PERFIS-C-UTIL(𝕍)).
```

As duas operações que se definem a seguir, CARREGA e DESCARREGA, permitem respectivamente, afectar e desafectar cval unidades de capacidade de um nó para entrega de um produto de saída desse nó p_j, por um tarefa 𝕆, num intervalo dado pelos instantes extremos t_s e t_e, por modificação do perfil de capacidade utilizada do produto p_j no nó.

```
CARREGA(𝕍,pⱼ,tₛ,tₑ,cval,𝕆) ::=
NÓ( LOCAL(𝕍),
    TIPO(𝕍),
    PRODUTOS(𝕍),
    MATERIAIS(𝕍),
    LISTAS-DE-MATERIAIS(𝕍),
    FACTORES-C(𝕍),
    PERFIL-C-MAX(𝕍),
    SUBSTITUI(  PERFIS-C-UTIL(𝕍),
                PERFIL-C-UTIL(𝕍,pⱼ),
                CARREGA-UTIL( PERFIL-C-UTIL(𝕍,pⱼ),
                              tₛ,
                              tₑ,
                              cval,
                              𝕆))).

DESCARREGA(𝕍,pⱼ,tₛ,tₑ,cval,𝕆) ::=
NÓ( LOCAL(𝕍),
    TIPO(𝕍),
    PRODUTOS(𝕍),
    MATERIAIS(𝕍),
    LISTAS-DE-MATERIAIS(𝕍),
    FACTORES-C(𝕍),
    PERFIL-C-MAX(𝕍),
    SUBSTITUI(  PERFIS-C-UTIL(𝕍),
                PERFIL-C-UTIL(𝕍,pⱼ),
                DESCARREGA-UTIL(  PERFIL-C-UTIL(𝕍,pⱼ),
                                  tₛ,
                                  tₑ,
                                  cval,
                                  𝕆))).
```

As operações CARREGA e DESCARREGA têm a seguinte propriedade:

DESCARREGA(CARREGA(𝕍,p_j,t_s,t_e,cval,𝕆),p_j,t_s,t_e,cval,𝕆)=𝕍

e:

CARREGA(DESCARREGA(𝕍,p_j,t_s,t_e,cval,𝕆),p_j,t_s,t_e,cval,𝕆)=𝕍

isto é, uma afectação (desafectação) imediatamente seguida da correspondente operação inversa de desafectação (afectação), deixa o nó no mesmo estado.

A operação ESCALONADA? é um predicado que testa se para um dado nó de capacidade $\mathbb{V}$, uma tarefa $\mathbb{O}$ foi escalonada no nó. Esta operação é aplicável se PRODUTO($\mathbb{O}$)=PRODUTOS($\mathbb{V}$).

ESCALONADA?($\mathbb{V}$,$\mathbb{O}$) ::=
$\exists_{\mathbb{s}_k}$: ($\mathbb{s}_k \in$ PERFIL-C-UTIL($\mathbb{V}$,PRODUTO($\mathbb{O}$))) $\wedge$ ($\mathbb{O} \in$ TAREFAS($\mathbb{s}_k$)).

As duas operações a seguir definidas, ambas identificadas por TAREFAS-ESCALONADAS, produzem para um nó de capacidade $\mathbb{V}$, o conjunto de tarefas escalonadas no instante t_k e o conjunto de tarefas escalonadas durante um intervalo dado pelos instantes limite t_s e t_e.[25]

TAREFAS-ESCALONADAS($\mathbb{V}$,t_k) ::=
{ $\mathbb{O}_j$: ESCALONADA?($\mathbb{V}$,$\mathbb{O}_j$) $\wedge$ DURANTE?(t_k,$\mathbb{O}_j$) }.

$$\text{TAREFAS-ESCALONADAS}(\mathbb{V},t_s,t_e) ::= \bigcup_{t_k=t_s}^{t_e-1} \text{TAREFAS-ESCALONADAS}(\mathbb{V},t_k).$$

A operação ESCALONA, escalona efectivamente uma tarefa $\mathbb{O}$ num nó de capacidade $\mathbb{V}$. O escalonamento efectuado não originará uma violação de restrições de capacidade nos casos em que se verifica ESCALONÁVEL?($\mathbb{V}$,$\mathbb{O}$).[26]

ESCALONA($\mathbb{V}$,$\mathbb{O}$) ::=
CARREGA($\mathbb{V}$,PRODUTO($\mathbb{O}$),TS($\mathbb{O}$),TE($\mathbb{O}$),CVAL($\mathbb{O}$),$\mathbb{O}$).

A operação DES-ESCALONA desfaz um escalonamento antes realizado para a tarefa $\mathbb{O}$, num nó de capacidade $\mathbb{V}$ (apenas válida se previamente se verifica ESCALONADA?($\mathbb{V}$,$\mathbb{O}$)).

DES-ESCALONA($\mathbb{V}$,$\mathbb{O}$) ::=
DESCARREGA($\mathbb{V}$,PRODUTO($\mathbb{O}$),TS($\mathbb{O}$),TE($\mathbb{O}$),CVAL($\mathbb{O}$),$\mathbb{O}$).

As operações ESCALONA e DES-ESCALONA têm a seguinte propriedade:

DES-ESCALONA(ESCALONA($\mathbb{V}$,$\mathbb{O}$),$\mathbb{O}$)=$\mathbb{V}$

e:

ESCALONA(DES-ESCALONA($\mathbb{V}$,$\mathbb{O}$),$\mathbb{O}$)=$\mathbb{V}$

isto é, o escalonamento de uma tarefa imediatamente seguido da operação que o desfaz deixa o nó no mesmo estado; o mesmo se diz para a sequência inversa.

---

[25] A operação de tarefas DURANTE? é um predicado que testa se um dado instante temporal ocorre dentro do intervalo de uma dada tarefa, ver a secção 3.7.1.4.

[26] Ver operação ESCALONÁVEL? para nós, nas secções 3.4.1.15 e 3.4.1.16. As operações de tarefas de capacidade PRODUTO, TS, TE, e CVAL produzem, respectivamente, o produto de saída, o tempo de início, o tempo de fim e a quantidade de capacidade que deve ser afectada à tarefa, ver a secção 3.7.1.

As operações a seguir definidas para nós de capacidade têm a ver com a identificação de conflitos de capacidade. As duas operações identificadas por C-CONFLITO? são predicados que testam, para um nó $\mathbb{V}$, se há um conflito de capacidade no instante t_k e se há algum conflito de capacidade durante um intervalo dado pelos instantes limite t_s e t_e.

$$\text{C-CONFLITO?}(\mathbb{V},t_k) ::= (t_k \geq \text{TS}(\text{PERFIL-C-MAX}(\mathbb{V}))) \wedge (t_k < \text{TE}(\text{PERFIL-C-MAX}(\mathbb{V}))) \wedge (\text{C-DISP}(\mathbb{V},t_k) < 0).$$

$$\text{C-CONFLITO?}(\mathbb{V},t_s,t_e) ::= \exists_{t_k} : (t_k \geq t_s) \wedge (t_k < t_e) \wedge \text{C-CONFLITO?}(\mathbb{V},t_k).$$

As duas operações identificadas por C-CONFLITO-CONJ a seguir definidas, produzem, para um dado nó de capacidade $\mathbb{V}$, o conjunto de tarefas que estão envolvidas num conflito de capacidade num dado instante temporal t_k (*i.e.*, o conjunto conflito, eventualmente vazio, de tarefas escalonadas em $\mathbb{V}$ no instante t_k) e o conjunto de tarefas que estão envolvidas em conflitos de capacidade durante um intervalo cujos instantes limite t_s e t_e, são dados (no caso desta última operação, se existir mais de um conflito de capacidade entre t_s e t_e, o conjunto produzido reúne, possivelmente, tarefas de diferentes conjuntos conflito).

$$\text{C-CONFLITO-CONJ}(\mathbb{V},t_k) ::= \{ \mathbb{O}_j : \text{ESCALONADA?}(\mathbb{V},\mathbb{O}_j) \wedge \text{DURANTE?}(t_k,\mathbb{O}_j) \wedge \text{C-CONFLITO?}(\mathbb{V},t_k) \}.$$

$$\text{C-CONFLITO-CONJ}(\mathbb{V},t_s,t_e) ::= \bigcup_{t_k=t_s}^{t_e-1} \text{C-CONFLITO-CONJ}(\mathbb{V},t_k).$$

A operação C-CONFLITO-TAREFA? é um predicado que testa se, para um nó de capacidade $\mathbb{V}$, uma tarefa $\mathbb{O}$ está envolvida num conflito de capacidade em algum instante.

$$\text{C-CONFLITO-TAREFA?}(\mathbb{V},\mathbb{O}) ::= |\text{C-CONFLITO-CONJ}(\mathbb{V},\text{TS}(\mathbb{O}),\text{TE}(\mathbb{O}))| > 0.$$

A operação C-CONFLITO-TAREFA-CONJ produz um conjunto de tarefas que, num dado nó $\mathbb{V}$, estão envolvidas em conflitos de capacidade com uma tarefa $\mathbb{O}$ (caso este conjunto conflito não esteja vazio, ele inclui, naturalmente, a tarefa $\mathbb{O}$).

$$\text{C-CONFLITO-TAREFA-CONJ}(\mathbb{V},\mathbb{O}) ::= \text{C-CONFLITO-CONJ}(\mathbb{V},\text{TS}(\mathbb{O}),\text{TE}(\mathbb{O})).$$

3.4.1.15 Operações com Nós Acumuladores

As operações que se descrevem nesta secção são aplicáveis apenas a nós acumuladores.

A operação C-NEC serve para, dado um nó acumulador $\mathbb{V}$, um produto p_j tal que $p_j \in$ PRODUTOS($\mathbb{V}$) e uma quantidade de produto q_j, calcular o número de unidades de produto padrão de acumulação (*uppa*) de $\mathbb{V}$ que é necessário para entregar a quantidade q_j do produto p_j (*i.e.*, calcula a capacidade necessária). Esta operação é definida por:

C-NEC($\mathbb{V}$,p_j,q_j) ::= CEILING(q_j*FACTOR-C($\mathbb{V}$,p_j)).

A seguir definem-se operações que permitem testar se existem opções de escalonamento para trás e para a frente, obter estas opções, testar se uma tarefa dada é escalonável num nó acumulador dado, testar se existem opções de re-escalonamento para trás e para a frente para uma tarefa escalonada e obter estas opções, respectivamente.

ESCALONA-PTRÁS?($\mathbb{V}$,p_j,q_j,t_x,t_y,duração) ::=
($p_j \in$ PRODUTOS($\mathbb{V}$)) $\wedge$
($\exists_{t_e}$: DURANTE?(INTERVALO(t_e-duração,t_e),
INTERVALO(TS(PERFIL-C-UTIL($\mathbb{V}$,p_j)),
TE(PERFIL-C-UTIL($\mathbb{V}$,p_j)))) $\wedge$
DURANTE?(INTERVALO(t_e-duração,t_e),INTERVALO(t_x,t_y)) $\wedge$
(C-DISP-MIN($\mathbb{V}$,t_e-duração,t_e) $\geq$ C-NEC($\mathbb{V}$,p_j,q_j))).

O predicado ESCALONA-PTRÁS?, acima definido, testa se é possível escalonar a partir do instante t_y e para trás no tempo num horizonte temporal $<t_x, t_y>$, uma hipotética tarefa com a quantidade q_j de produto de saída p_j e duração de duração períodos. O predicado ESCALONA-PFRENTE?, a seguir definido, é semelhante mas testa a possibilidade de escalonamento para a frente (a partir de t_x).

ESCALONA-PFRENTE?($\mathbb{V}$,p_j,q_j,t_x,t_y,duração) ::=
($p_j \in$ PRODUTOS($\mathbb{V}$)) $\wedge$
($\exists_{t_s}$: DURANTE?(INTERVALO(t_s,t_s+duração),
INTERVALO(TS(PERFIL-C-UTIL($\mathbb{V}$,p_j)),
TE(PERFIL-C-UTIL($\mathbb{V}$,p_j)))) $\wedge$
DURANTE?(INTERVALO(t_s,t_s+duração),INTERVALO(t_x,t_y)) $\wedge$
(C-DISP-MIN($\mathbb{V}$,t_s,t_s+duração) $\geq$ C-NEC($\mathbb{V}$,p_j,q_j))).

A operação OPÇÃO-ESCALONA-PTRÁS, a seguir definida, deve ser usada só se a operação ESCALONA-PTRÁS?, quando usada imediatamente antes, produziu o valor VERDADEIRO.

OPÇÃO-ESCALONA-PTRÁS($\mathbb{V}$,p_j,q_j,t_x,t_y,duração) ::=
MAX(t_e) : DURANTE?(INTERVALO(t_e-duração,t_e),
INTERVALO(TS(PERFIL-C-UTIL($\mathbb{V}$,p_j)),
TE(PERFIL-C-UTIL($\mathbb{V}$,p_j)))) $\wedge$
DURANTE?(INTERVALO(t_e-duração,t_e),INTERVALO(t_x,t_y)) $\wedge$
(C-DISP-MIN($\mathbb{V}$,t_e-duração,t_e) $\geq$ C-NEC($\mathbb{V}$,p_j,q_j)).

A operação OPÇÃO-ESCALONA-PTRÁS produz o maior tempo de fim possível t_e, para o escalonamento de uma hipotética tarefa com a quantidade q_j de produto de saída p_j e duração de duração períodos, a escalonar a partir do instante t_y e para trás no tempo num horizonte temporal $<t_x, t_y>$. A operação OPÇÃO-ESCALONA-PFRENTE, a seguir definida, deve ser usada só se a operação ESCALONA-PFRENTE?, quando usada imediatamente antes, produziu o valor VERDADEIRO.

OPÇÃO-ESCALONA-PFRENTE($\mathbb{V}$,p_j,q_j,t_x,t_y,duração) ::=
MIN(t_s) : DURANTE?(INTERVALO(t_s,t_s+duração),
INTERVALO(TS(PERFIL-C-UTIL($\mathbb{V}$,p_j)),
TE(PERFIL-C-UTIL($\mathbb{V}$,p_j)))) ∧
DURANTE?(INTERVALO(t_s,t_s+duração),INTERVALO(t_x,t_y)) ∧
(C-DISP-MIN($\mathbb{V}$,t_s,t_s+duração) ≥ C-NEC($\mathbb{V}$,p_j,q_j)).

A operação OPÇÃO-ESCALONA-PFRENTE produz o menor tempo de início possível t_s, para o escalonamento de uma hipotética tarefa com a quantidade q_j de produto de saída p_j e duração de duração períodos, a escalonar a partir do instante t_x e para a frente no tempo num horizonte temporal $<t_x, t_y>$. O predicado ESCALONÁVEL?, que se define a seguir, testa se uma tarefa é escalonável.[27]

ESCALONÁVEL?($\mathbb{V}$,$\mathbb{O}$) ::=
(LOCAL($\mathbb{V}$)=LOCAL($\mathbb{O}$)) ∧ (PRODUTO($\mathbb{O}$)∈PRODUTOS($\mathbb{V}$)) ∧
(TAREFA-ELEMENTAR?($\mathbb{O}$)) ∧
(MATERIAIS-PARA-PRODUTO($\mathbb{V}$,PRODUTO($\mathbb{O}$))=MATERIAIS($\mathbb{O}$)) ∧
($\forall_{m_i}$ (m_i∈MATERIAIS($\mathbb{O}$)) :
(RAZÃO-MATERIAL-PRODUTO($\mathbb{V}$,PRODUTO($\mathbb{O}$),m_i)=RAZÃO($\mathbb{O}$,m_i))) ∧
ESCALONA-PTRÁS?($\mathbb{V}$,
PRODUTO($\mathbb{O}$),
QUANTIDADE($\mathbb{O}$),
TS($\mathbb{O}$),
TE($\mathbb{O}$),
DURAÇÃO($\mathbb{O}$)).

Para re-escalonamento, definem-se outras operações que permitem testar se existem opções de re-escalonamento para trás e para a frente e obter essas opções.

---

[27] As operações QUANTIDADE, DURAÇÃO, MATERIAIS e RAZÃO produzem, respectivamente, a quantidade de produto de saída da tarefa, a duração, os produtos a consumir (materiais) e a razão entre a quantidade de um produto a consumir e o produto de saída de uma tarefa de capacidade. A operação TAREFA-ELEMENTAR?, é um predicado que testa se uma dada tarefa de capacidade é uma tarefa elementar (apenas tarefas elementares são escalonáveis). *A priori*, sem entrar em considerações relativas à capacidade disponível, uma tarefa é escalonável num nó se foi construída para esse nó. A operação LOCAL de tarefas de capacidade produz o identificador do nó de capacidade para o qual uma tarefa dada foi construída. Ver a secção 3.7.1.
Alternativamente, também se poderia definir ESCALONÁVEL? usando, em vez de ESCALONA-PTRÁS?, a operação ESCALONA-PFRENTE? (com os mesmos parâmetros).

```
RE-ESCALONA-PTRÁS?(𝕍,𝕆,tx,ty,duração) ::=
ESCALONADA?(𝕍,𝕆) ∧
(∃te :
  DURANTE?( INTERVALO(te-duração,te),
            INTERVALO(  TS(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))),
                        TE(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))))) ∧
  DURANTE?(INTERVALO(te-duração,te),INTERVALO(tx,ty)) ∧
  (C-DISP-MIN(DES-ESCALONA(𝕍,𝕆),te-duração,te) ≥
                      C-NEC(𝕍,PRODUTO(𝕆),QUANTIDADE(𝕆)))).
```

O predicado RE-ESCALONA-PTRÁS?, acima definido, testa se é possível re-escalonar a partir do instante t_y e para trás no tempo num horizonte temporal $<t_x, t_y>$, a tarefa 𝕆 com a duração de duração períodos. O predicado RE-ESCALONA-PFRENTE?, a seguir definido, é semelhante mas testa a possibilidade de re-escalonamento para a frente (a partir de t_x).

```
RE-ESCALONA-PFRENTE?(𝕍,𝕆,tx,ty,duração) ::=
ESCALONADA?(𝕍,𝕆) ∧
(∃ts :
  DURANTE?( INTERVALO(ts,ts+duração),
            INTERVALO(  TS(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))),
                        TE(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))))) ∧
  DURANTE?(INTERVALO(ts,ts+duração),INTERVALO(tx,ty)) ∧
  (C-DISP-MIN(DES-ESCALONA(𝕍,𝕆),ts,ts+duração) ≥
                      C-NEC(𝕍,PRODUTO(𝕆),QUANTIDADE(𝕆)))).
```

A operação OPÇÃO-RE-ESCALONA-PTRÁS, a seguir definida, deve ser usada só se a operação RE-ESCALONA-PTRÁS?, quando usada imediatamente antes, produziu o valor VERDADEIRO.

```
OPÇÃO-RE-ESCALONA-PTRÁS(𝕍,𝕆,tx,ty,duração) ::=
MAX(te) :
  DURANTE?( INTERVALO(te-duração,te),
            INTERVALO(  TS(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))),
                        TE(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))))) ∧
  DURANTE?(INTERVALO(te-duração,te),INTERVALO(tx,ty)) ∧
  (C-DISP-MIN(DES-ESCALONA(𝕍,𝕆),te-duração,te) ≥
                      C-NEC(𝕍,PRODUTO(𝕆),QUANTIDADE(𝕆))).
```

A operação OPÇÃO-RE-ESCALONA-PTRÁS produz o maior tempo de fim possível t_e, para o re-escalonamento de uma tarefa com a duração de duração períodos, a re-escalonar a partir do instante t_y e para trás no tempo num horizonte temporal $<t_x, t_y>$. A operação OPÇÃO-RE-ESCALONA-PFRENTE, a seguir definida, deve ser usada só se a operação RE-ESCALONA-PFRENTE?, quando usada imediatamente antes, produziu o valor VERDADEIRO.

```
OPÇÃO-RE-ESCALONA-PFRENTE(𝕍,𝕆,tx,ty,duração) ::=
MIN(ts) :
  DURANTE?( INTERVALO(ts,ts+duração),
            INTERVALO(  TS(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))),
                        TE(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))))) ∧
  DURANTE?(INTERVALO(ts,ts+duração),INTERVALO(tx,ty)) ∧
  (C-DISP-MIN(DES-ESCALONA(𝕍,𝕆),ts,ts+duração) ≥
                      C-NEC(𝕍,PRODUTO(𝕆),QUANTIDADE(𝕆))).
```

A operação OPÇÃO-RE-ESCALONA-PFRENTE produz o menor tempo de início possível t_s, para o re-escalonamento de hipotética tarefa com a duração de duração períodos, a re-escalonar a partir do instante t_x e para a frente no tempo num horizonte temporal $<t_x, t_y>$.

3.4.1.16 Operações com Nós Processadores

As operações que se descrevem nesta secção são aplicáveis apenas a nós processadores.

A operação D-NEC serve para, dado um nó processador $\mathbb{V}$, um produto p_j tal que $p_j \in$ PRODUTOS($\mathbb{V}$), uma quantidade de produto q_j e uma quantidade de capacidade cval a afectar constantemente em cada período temporal (número de unidades de produto padrão de processamento, *uppp*), calcular a duração de uma hipotética tarefa a escalonar em $\mathbb{V}$ de modo a entregar a quantidade q_j do produto p_j (*i.e.*, calcula a duração necessária). Esta operação é definida por:

```
D-NEC(𝕍,pj,qj,cval) ::= CEILING(qj*FACTOR-C(𝕍,pj)/cval).
```

A seguir definem-se operações que permitem testar se existem opções de escalonamento para trás e para a frente, obter estas opções, testar se uma tarefa dada é escalonável num nó processador dado, testar se existem opções de re-escalonamento para trás e para a frente para uma tarefa escalonada e obter estas opções, respectivamente.

```
ESCALONA-PTRÁS?(𝕍,pj,qj,tx,ty,cval) ::=
(pj∈PRODUTOS(𝕍)) ∧
(∃te : DURANTE?( INTERVALO(te - D-NEC(𝕍,pj,qj,cval),te),
                 INTERVALO(  TS(PERFIL-C-UTIL(𝕍,pj)),
                             TE(PERFIL-C-UTIL(𝕍,pj)))) ∧
       DURANTE?( INTERVALO(te - D-NEC(𝕍,pj,qj,cval),te),
                 INTERVALO(tx,ty)) ∧
       (C-DISP-MIN(𝕍,te - D-NEC(𝕍,pj,qj,cval),te) ≥ cval)).
```

O predicado ESCALONA-PTRÁS?, acima definido, testa se é possível escalonar a partir do instante t_y, para trás no tempo num horizonte temporal $<t_x, t_y>$ e afectando cval unidades de capacidade de processamento do nó em cada período temporal de afectação, uma hipotética tarefa com a quantidade q_j de produto de saída p_j. O predicado ESCALONA-PFRENTE?, a

seguir definido, é semelhante mas testa a possibilidade de escalonamento para a frente (a partir de t_x).

```
ESCALONA-PFRENTE?(𝕍,pj,qj,tx,ty,cval) ::=
(pj∈PRODUTOS(𝕍)) ∧
(∃ts : DURANTE?( INTERVALO(ts,ts + D-NEC(𝕍,pj,qj,cval)),
                 INTERVALO(  TS(PERFIL-C-UTIL(𝕍,pj)),
                             TE(PERFIL-C-UTIL(𝕍,pj)))) ∧
       DURANTE?( INTERVALO(ts,ts + D-NEC(𝕍,pj,qj,cval)),
                 INTERVALO(tx,ty)) ∧
       (C-DISP-MIN(𝕍,ts,ts + D-NEC(𝕍,pj,qj,cval)) ≥ cval)).
```

A operação OPÇÃO-ESCALONA-PTRÁS, a seguir definida, deve ser usada só se a operação ESCALONA-PTRÁS?, quando usada imediatamente antes, produziu o valor VERDADEIRO.

```
OPÇÃO-ESCALONA-PTRÁS(𝕍,pj,qj,tx,ty,cval) ::=
MAX(te) : DURANTE?( INTERVALO(te - D-NEC(𝕍,pj,qj,cval),te),
                    INTERVALO(  TS(PERFIL-C-UTIL(𝕍,pj)),
                                TE(PERFIL-C-UTIL(𝕍,pj)))) ∧
          DURANTE?( INTERVALO(te - D-NEC(𝕍,pj,qj,cval),te),
                    INTERVALO(tx,ty)) ∧
          (C-DISP-MIN(𝕍,te - D-NEC(𝕍,pj,qj,cval),te) ≥ cval).
```

A operação OPÇÃO-ESCALONA-PTRÁS produz o maior tempo de fim possível t_e, para o escalonamento de uma hipotética tarefa com a quantidade q_j de produto de saída p_j e com uma afectação de uma quantidade de capacidade de processamento constante de cval unidades em cada período temporal de afectação, a escalonar a partir do instante t_y e para trás no tempo num horizonte temporal $<t_x,t_y>$. A operação OPÇÃO-ESCALONA-PFRENTE, a seguir definida, deve ser usada só se a operação ESCALONA-PFRENTE?, quando usada imediatamente antes, produziu o valor VERDADEIRO.

```
OPÇÃO-ESCALONA-PFRENTE(𝕍,pj,qj,tx,ty,cval) ::=
MIN(ts) : DURANTE?( INTERVALO(ts,ts + D-NEC(𝕍,pj,qj,cval)),
                    INTERVALO(  TS(PERFIL-C-UTIL(𝕍,pj)),
                                TE(PERFIL-C-UTIL(𝕍,pj)))) ∧
          DURANTE?( INTERVALO(ts,ts + D-NEC(𝕍,pj,qj,cval)),
                    INTERVALO(tx,ty)) ∧
          (C-DISP-MIN(𝕍,ts,ts + D-NEC(𝕍,pj,qj,cval)) ≥ cval).
```

A operação OPÇÃO-ESCALONA-PFRENTE produz o menor tempo de início possível t_s, para o escalonamento de uma hipotética tarefa com a quantidade q_j de produto de saída p_j e com uma afectação de uma quantidade de capacidade de processamento constante de cval unidades em cada período temporal de afectação, a escalonar a partir do instante t_x e para a

frente no tempo num horizonte temporal $<t_x, t_y>$. O predicado ESCALONÁVEL?, que se define a seguir, testa se uma tarefa é escalonável.[28]

```
ESCALONÁVEL?(𝕍,𝕆) ::=
(LOCAL(𝕍)=LOCAL(𝕆)) ∧ (PRODUTO(𝕆)∈PRODUTOS(𝕍)) ∧
(TAREFA-ELEMENTAR?(𝕆)) ∧
(MATERIAIS-PARA-PRODUTO(𝕍,PRODUTO(𝕆))=MATERIAIS(𝕆)) ∧
(∀m_i (m_i∈MATERIAIS(𝕆)) :
    (RAZÃO-MATERIAL-PRODUTO(𝕍,PRODUTO(𝕆),m_i)=RAZÃO(𝕆,m_i))) ∧
ESCALONA-PTRÁS?(  𝕍,
                  PRODUTO(𝕆),
                  QUANTIDADE(𝕆),
                  TS(𝕆),
                  TE(𝕆),
                  CVAL(𝕆)).
```

Para re-escalonamento, definem-se outras operações que permitem testar se existem opções de re-escalonamento para trás e para a frente e obter essas opções.

```
RE-ESCALONA-PTRÁS?(𝕍,𝕆,t_x,t_y,cval) ::=
ESCALONADA?(𝕍,𝕆) ∧
(∃t_e :
  DURANTE?( INTERVALO(  t_e - D-NEC( 𝕍,
                                     PRODUTO(𝕆),
                                     QUANTIDADE(𝕆),
                                     cval),
                        t_e),
            INTERVALO(  TS(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))),
                        TE(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))))) ∧
  DURANTE?( INTERVALO(  t_e - D-NEC( 𝕍,
                                     PRODUTO(𝕆),
                                     QUANTIDADE(𝕆),
                                     cval),
                        t_e),
            INTERVALO(t_x,t_y)) ∧
  (C-DISP-MIN(  DES-ESCALONA(𝕍,𝕆),
                t_e - D-NEC(𝕍,PRODUTO(𝕆),QUANTIDADE(𝕆),cval),
                t_e) ≥ cval)).
```

O predicado RE-ESCALONA-PTRÁS?, acima definido, testa se é possível re-escalonar a partir do instante t_y, para trás no tempo num horizonte temporal $<t_x, t_y>$ e afectando cval unidades de capacidade de processamento do nó em cada período temporal de afectação, uma

[28] Alternativamente, também se poderia definir ESCALONÁVEL? usando, em vez de ESCALONA-PTRÁS?, a operação ESCALONA-PFRENTE? (com os mesmos parâmetros).

tarefa dada. O predicado RE-ESCALONA-PFRENTE?, a seguir definido, é semelhante mas testa a possibilidade de re-escalonamento para a frente (a partir de t_x).

```
RE-ESCALONA-PFRENTE?(𝕍,𝕆,tx,ty,cval) ::=
ESCALONADA?(𝕍,𝕆) ∧
(∃ts :
  DURANTE?( INTERVALO(  ts,
                        ts + D-NEC( 𝕍,
                                    PRODUTO(𝕆),
                                    QUANTIDADE(𝕆),
                                    cval)),
            INTERVALO(  TS(PERFIL-C-UTIL(𝕍, PRODUTO(𝕆))),
                        TE(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))))) ∧
  DURANTE?( INTERVALO(  ts,
                        ts + D-NEC( 𝕍,
                                    PRODUTO(𝕆),
                                    QUANTIDADE(𝕆),
                                    cval)),
            INTERVALO(tx,ty)) ∧
  (C-DISP-MIN(  DES-ESCALONA(𝕍,𝕆),
                ts,
                ts + D-NEC(𝕍,PRODUTO(𝕆),QUANTIDADE(𝕆),cval)) ≥
                                    cval)).
```

A operação OPÇÃO-RE-ESCALONA-PTRÁS, a seguir definida, deve ser usada só se a operação RE-ESCALONA-PTRÁS?, quando usada imediatamente antes, produziu o valor VERDADEIRO.

```
OPÇÃO-RE-ESCALONA-PTRÁS(𝕍,𝕆,tx,ty,cval) ::=
MAX(te) :
  DURANTE?( INTERVALO(  te - D-NEC( 𝕍,
                                    PRODUTO(𝕆),
                                    QUANTIDADE(𝕆),
                                    cval),
                        te),
            INTERVALO(  TS(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))),
                        TE(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))))) ∧
  DURANTE?( INTERVALO(  te - D-NEC( 𝕍,
                                    PRODUTO(𝕆),
                                    QUANTIDADE(𝕆),
                                    cval),
                        te),
            INTERVALO(tx,ty)) ∧
  (C-DISP-MIN(  DES-ESCALONA(𝕍,𝕆),
                te - D-NEC(𝕍,PRODUTO(𝕆),QUANTIDADE(𝕆),cval),
                te) ≥ cval).
```

A operação OPÇÃO-RE-ESCALONA-PTRÁS produz o maior tempo de fim possível t_e, para o re-escalonamento de uma tarefa com uma afectação de uma quantidade de capacidade de processamento constante de cval unidades em cada período temporal de afectação, a re-escalonar a partir do instante t_y e para trás no tempo num horizonte temporal $<t_x, t_y>$. A operação OPÇÃO-RE-ESCALONA-PFRENTE, a seguir definida, deve ser usada só se a operação RE-ESCALONA-PFRENTE?, quando usada imediatamente antes, produziu o valor VERDADEIRO.

```
OPÇÃO-RE-ESCALONA-PFRENTE(𝕍,𝕆,tx,ty,cval) ::=
MIN(ts) :
  DURANTE?( INTERVALO(  ts,
                        ts + D-NEC( 𝕍,
                                    PRODUTO(𝕆),
                                    QUANTIDADE(𝕆),
                                    cval)),
            INTERVALO(  TS(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))),
                        TE(PERFIL-C-UTIL(𝕍,PRODUTO(𝕆))))) ∧
  DURANTE?( INTERVALO(  ts,
                        ts + D-NEC( 𝕍,
                                    PRODUTO(𝕆),
                                    QUANTIDADE(𝕆),
                                    cval)),
            INTERVALO(tx,ty)) ∧
  (C-DISP-MIN(  DES-ESCALONA(𝕍,𝕆),
                ts,
                ts + D-NEC(𝕍,PRODUTO(𝕆),QUANTIDADE(𝕆),cval)) ≥
                                    cval).
```

A operação OPÇÃO-RE-ESCALONA-PFRENTE produz o menor tempo de início possível t_s, para o re-escalonamento de uma tarefa com uma afectação de uma quantidade de capacidade de processamento constante de cval unidades em cada período temporal de afectação, a re-escalonar a partir do instante t_x e para a frente no tempo num horizonte temporal $<t_x, t_y>$.

3.4.2 Nó de Retalho e Nó de Matéria-Prima

Os nós de retalho e os nós de matéria-prima são desprovidos de capacidade e, portanto não necessitam de representar um estado interno de capacidade. No entanto, como estes nós servem de fronteira com o exterior, é necessário que mantenham um estado relativo a eventos que advêm de pedidos *do exterior à rede*, no caso dos nós de retalho e de pedidos *da rede ao exterior*, no caso de nós de matéria-prima. Estes *eventos de pedido* são eventos simples que servem para representar tarefas fictícias que, no primeiro caso, apenas consomem um produto final da rede (significam o consumo do exterior) e no último caso apenas entregam um produto de entrada da rede (significam entrega do exterior). Seguidamente, descreve-se como são representados os eventos de pedido nos nós de retalho e de matéria-prima e depois descrevem-se as classes nó de retalho e nó de matéria-prima.

3.4.2.1 Perfil de Eventos de Pedido

Um *perfil de eventos de pedido* ed(t), é uma função de inteiros para reais que mapeia períodos do horizonte temporal para quantidades de produto. Para um produto dado p e para um horizonte temporal $<t_x, t_y>$, ed(t) é o conjunto ordenado:

$\{s_x, \ldots, s_k, \ldots s_{y-1}\}$

com um valor s_k, para cada período temporal t_k tal que $t_k \geq t_x$ e $t_k < t_y$. Cada s_k é denominado o *segmento de perfil de eventos de pedido* para o período k. Cada segmento s_k é um triplo:

$<t_k, q_k, e_k>$

ou seja, o perfil de eventos é da forma:

$\{<t_x, q_x, e_x>, \ldots, <t_k, q_k, e_k>, \ldots, <t_{y-1}, q_{y-1}, e_{y-1}>\}$

em que, para cada segmento s_k:

- t_k é o período do segmento;
- cada q_k é a quantidade total de produto associada ao período t_k, mais adiante relacionado com e_k;
- cada e_k é um conjunto de eventos associado ao período t_k, contendo todos os eventos de pedido com valor de tempo t, tal que $t=t_k$.

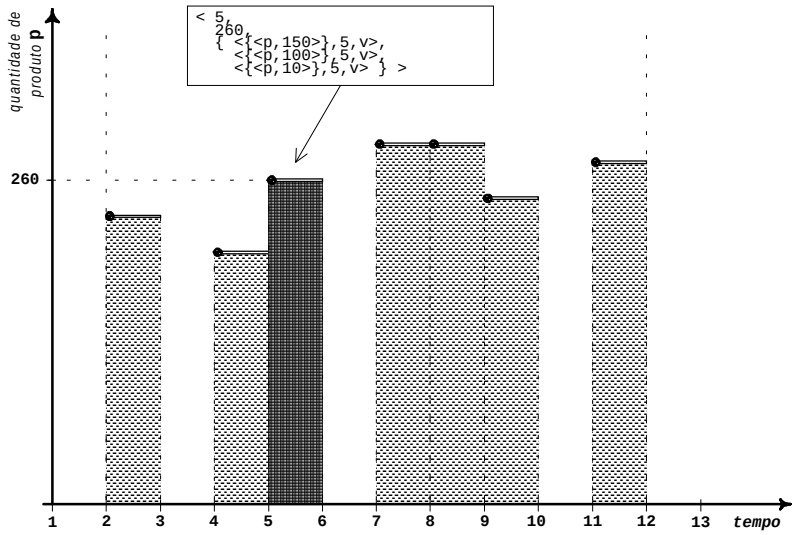


Figura 3-9- Exemplo de perfil de eventos de pedido de um produto p num nó v (de retalho ou de matéria-prima), para o horizonte temporal <2,12>. A faixa assinalada corresponde ao elemento de perfil associado ao período 5 (o triplo <5,260,{<{<p,150>},5,v>, <{<p,100>},5,v>, <{<p,10>},5,v>}>).

A relação entre os valores q_k e e_k de um perfil de eventos de pedido para um dado produto é descrita como se segue. Seja $<t_k, q_k, e_k>$ o segmento de um perfil de eventos de pedido de um produto p, num nó de retalho ou de matéria-prima v, para o período t_k e seja

$e_k=\{e_1,...,e_i,...,e_n\}$. Se os eventos de pedidos e_i são definidos por $<\{<p,q_i>\},t_i,v>$, tem-se:

$t_i=t_k$ (para qualquer i tal que $i\geq 1$ e $i\leq n$) e

$$q_k=\sum_{i=1}^{n} q_i$$

Os valores q_k e e_k de um perfil de procura podem ser alterados com o estabelecimento de novos eventos de pedido.

A título de exemplo, representa-se esquematicamente na Figura 3-9 um perfil de eventos de pedido de um produto.

3.4.2.2 Operações com Perfis de Eventos de Pedido

As operações primitivas sobre segmentos de perfil de eventos de pedido são:

```
TEMPO(<t,q,e>) ::= t.

VAL(<t,q,e>) ::= q.

EVENTOS(<t,q,e>) ::= e.

SEGMENTO(t,q,e) ::= <t,q,e>.
```

As operações primitivas sobre perfis de eventos são (TS, TE e SEGMENTO são operações selectoras e PERFIL-E é uma operação construtora):

```
TS(ed(t)) ::= TEMPO(PRIMEIRO(c(t))).

TE(ed(t)) ::= TEMPO(ÚLTIMO(c(t)))+1.

SEGMENTO(tk,ed(t)) ::= NÉSIMO(ed(t),tk-TS(ed(t))).

PERFIL-E(tx,ty) ::=
ed(t) : (|ed(t)|=ty-tx) ∧
        (∀sk (sk∈ed(t)) :
            (sk=SEGMENTO(tx+POSIÇÃO(ed(t),sk),0,{}))).
```

A operação ALTERA-SEGMENTO modifica um perfil de eventos por modificação de um dos seus segmentos.

```
ALTERA-SEGMENTO(tk,ed(t),qk',ek') ::=
SUBSTITUI(ed(t),SEGMENTO(tk,ed(t)),SEGMENTO(tk,qk',ek')).
```

A operação CARREGA-EVENTO modifica um perfil de eventos por colocação de um evento adicional no perfil:

```
CARREGA-EVENTO(ed(t),e) ::=
ALTERA-SEGMENTO(  TEMPO(e),
                  ed(t),
                  VAL(SEGMENTO(TEMPO(e),ed(t)))+QUANTIDADE(e),
                  EVENTOS(SEGMENTO(TEMPO(e),ed(t)))∪{e}).
```

A operação DESCARREGA-EVENTO modifica um perfil de eventos retirando do perfil um evento:

```
DESCARREGA-EVENTO(ed(t),e) ::=
ALTERA-SEGMENTO(  TEMPO(e),
                  ed(t),
                  VAL(SEGMENTO(TEMPO(e),ed(t)))-QUANTIDADE(e),
                  EVENTOS(SEGMENTO(TEMPO(e),ed(t)))\{e}).
```

As operações CARREGA-EVENTO e DESCARREGA-EVENTO têm a seguinte propriedade:

DESCARREGA-EVENTO(CARREGA-EVENTO(ed(t),e),e)=ed(t)

e:

CARREGA-EVENTO(DESCARREGA-EVENTO(ed(t),e),e)=ed(t)

isto é, a colocação de um evento no perfil imediatamente seguida da retirada do mesmo evento do perfil deixa o perfil no mesmo estado; o mesmo se diz para a sequência inversa.

3.4.2.3 Nó de Retalho

Os nós de retalho, para além de serem desprovidos de capacidade, não têm produtos de saída e não têm nós clientes na rede física. No contexto de uma rede, funcionam como sumidouros, ou "poços", de produtos finais da rede.

Formalmente, um nó da classe *nó de retalho*, ou nó de tipo R, é descrito por um tuplo de quatro valores:

<v,TN,m,ed(t)>

em que:

- v∈ v, em que v é um símbolo que identifica univocamente o nó no contexto de uma rede;
- TN=R sempre, indicando que se trata de um nó de retalho;
- m={...,m_i,...} (com i>0 e $m \subset P$), em que m é o conjunto ordenado de produtos de entrada do nó de retalho;

– $\mathcal{ed}$(t)={...,ed_i(t),...} (com i>0 e |$\mathcal{ed}$(t)|=|$\mathcal{m}$|) é o conjunto ordenado em que cada elemento ed_i(t) é o perfil de eventos de pedido do exterior à rede para o produto m_i [29]

a) Nó de retalho dedicado. b) Nó de retalho partilhado.

Figura 3-10- Nós de retalho.

Nós de retalho são representados por círculos, mais pequenos que os nós de capacidade, a cinzento. Na Figura 3-10 representam-se esquematicamente um nó de retalho com um único produto de entrada p (nó de retalho dedicado, ou mono-produto) e um nó de retalho com os produtos de entrada p_1, p_2 e p_3 (nó de retalho partilhado, ou multi-produto). As setas indicando os produtos de entrada do nó de retalho são usadas mais no contexto de uma representação de rede física e correspondem a arcos de rede partindo de outros nós, de capacidade.

3.4.2.4 Operações com Nós de Retalho

As operações primitivas sobre nós de retalho são:

LOCAL(<v,R,$\mathcal{m}$,$\mathcal{ed}$(t)>) ::= v.

TIPO(<v,R,$\mathcal{m}$,$\mathcal{ed}$(t)>) ::= R.

MATERIAIS(<v,R,$\mathcal{m}$,$\mathcal{ed}$(t)>) ::= $\mathcal{m}$.

PERFIS-E(<v,R,$\mathcal{m}$,$\mathcal{ed}$(t)>) ::= $\mathcal{ed}$(t).

NÓ(v,R,$\mathcal{m}$,$\mathcal{ed}$(t)) ::= <v,R,$\mathcal{m}$,$\mathcal{ed}$(t)>.

(em que os parâmetros $\mathcal{m}$ e $\mathcal{ed}$(t) devem estar de acordo com o tipo de nó e entre si no que respeita ao número e ordem dos seus elementos, e sendo $TS(ed_i(t))=TS(ed_j(t))$ e $TE(ed_i(t))=TE(ed_j(t))$ para quaisquer $ed_i(t)$ e $ed_j(t)$ tais que $ed_i(t)\in\mathcal{ed}$(t) e $ed_j(t)\in\mathcal{ed}$(t)).

Definem-se também as seguintes operações:

[29] Os limites temporais do horizonte temporal dos perfis de eventos em $\mathcal{ed}$(t) devem ser iguais.

```
TODOS-PRODUTOS(<v,R,m,ed(t)>) ::= m.

PERFIL-E(𝕍,m_i) ::=
ed_i(t) : (ed_i(t)=NÉSIMO(PERFIS-E(𝕍),POSIÇÃO(MATERIAIS(𝕍),m_i))).
```

O predicado ESCALONÁVEL? testa se uma tarefa de retalho é escalonável num nó de retalho.[30]

```
ESCALONÁVEL?(𝕍,𝕆) ::=
(LOCAL(𝕍)=LOCAL(𝕆)) ∧ (PRODUTO(𝕆)∈MATERIAIS(𝕍)) ∧
DURANTE?( EVENTO(𝕆),
          INTERVALO(  TS(PERFIL-E(𝕍,PRODUTO(𝕆)),
                      TE(PERFIL-E(𝕍,PRODUTO(𝕆))))).
```

A operação seguinte escalona efectivamente uma tarefa de retalho $\mathbb{O}$ num nó de retalho $\mathbb{V}$ (só aplicável caso se verifique ESCALONÁVEL?($\mathbb{V}$,$\mathbb{O}$)).

```
ESCALONA(𝕍,𝕆) ::=
NÓ( LOCAL(𝕍),
    TIPO(𝕍),
    MATERIAIS(𝕍),
    SUBSTITUI(  PERFIS-E(𝕍),
                PERFIL-E(𝕍,PRODUTO(𝕆)),
                CARREGA-EVENTO( PERFIL-E(𝕍,PRODUTO(𝕆)),
                                EVENTO(𝕆)))).
```

A operação seguinte desfaz o escalonamento antes realizado para a tarefa de retalho $\mathbb{O}$ num nó de retalho $\mathbb{V}$.

```
DES-ESCALONA(𝕍,𝕆) ::=
NÓ( LOCAL(𝕍),
    TIPO(𝕍),
    MATERIAIS(𝕍),
    SUBSTITUI(  PERFIS-E(𝕍),
                PERFIL-E(𝕍,PRODUTO(𝕆)),
                DESCARREGA-EVENTO(  PERFIL-E(𝕍,PRODUTO(𝕆)),
                                    EVENTO(𝕆)))).
```

As operações ESCALONA e DES-ESCALONA têm a seguinte propriedade:

```
DES-ESCALONA(ESCALONA(𝕍,𝕆),𝕆)=𝕍
```

---

[30] As operações PRODUTO, LOCAL e EVENTO produzem, respectivamente, o produto de uma tarefa de retalho, o identificador do nó de retalho para o qual uma tarefa de retalho foi construída e o evento de uma tarefa de retalho dada, ver a secção 3.7.2.1. Uma tarefa é escalonável num nó de retalho se foi construída para esse nó.

e:

ESCALONA(DES-ESCALONA($\mathbb{V}$,$\mathbb{O}$),$\mathbb{O}$)=$\mathbb{V}$

isto é, um escalonamento de uma tarefa imediatamente seguido da operação que o desfaz deixa o nó no mesmo estado; o mesmo se diz para a sequência inversa. A operação seguinte é um predicado que testa se, para um dado nó de retalho $\mathbb{V}$, uma tarefa de retalho $\mathbb{O}$ foi escalonada no nó.

ESCALONADA?($\mathbb{V}$,$\mathbb{O}$) ::=
$\exists_{s_k}$: ($s_k \in$ PERFIL-E($\mathbb{V}$,PRODUTO($\mathbb{O}$))) $\wedge$ (EVENTO($\mathbb{O}$)$\in$EVENTOS(s_k)).

As duas operações seguintes, ambas identificadas por TAREFAS-ESCALONADAS, produzem, para um nó de retalho dado $\mathbb{V}$, o conjunto de tarefas escalonadas num dado instante t_k e o conjunto de tarefas escalonadas durante um intervalo dado pelos instantes limite t_s e t_e, respectivamente.[31]

TAREFAS-ESCALONADAS($\mathbb{V}$,t_k) ::=
{ $\mathbb{O}_j$: ESCALONADA?($\mathbb{V}$,$\mathbb{O}_j$) $\wedge$ (TEMPO($\mathbb{O}_j$)=t_k) }.

TAREFAS-ESCALONADAS($\mathbb{V}$,t_s,t_e) ::=
{ $\mathbb{O}_j$: ESCALONADA?($\mathbb{V}$,$\mathbb{O}_j$) $\wedge$ (TEMPO($\mathbb{O}_j$)$\geq t_s$) $\wedge$ (TEMPO($\mathbb{O}_j$)$<t_e$) }.

3.4.2.5 Nó de Matéria-Prima

Os nós de matéria-prima, para além de serem desprovidos de capacidade, não têm produtos de entrada e não têm nós fornecedores na rede física. No contexto de uma rede funcionam como fontes de produtos de entrada da rede.

Formalmente, um nó da classe *nó de matéria-prima*, ou nó de tipo M, é descrito por tuplo de quatro valores:

<v,TN,$\mathbb{p}$,$\mathbb{ed}$(t)>

em que:

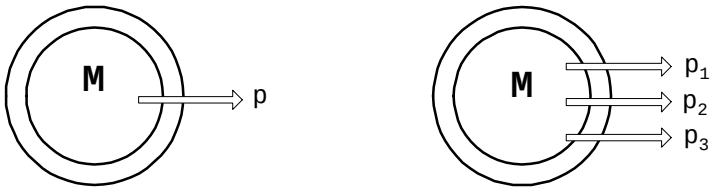
- v$\in \mathbb{v}$, em que v é um símbolo que identifica univocamente o nó no contexto de uma rede;
- TN=M sempre, indicando que se trata de um nó de matéria-prima;
- $\mathbb{p}$={...,p_j,...} (com j>0 e $\mathbb{p} \subset \mathbb{P}$), em que $\mathbb{p}$ é o conjunto ordenado de produtos de saída do nó de matéria-prima;
- $\mathbb{ed}$(t)={...,ed_j(t),...} (com j>0 e |$\mathbb{ed}$(t)|=|$\mathbb{p}$|) é o conjunto ordenado em que cada elemento ed_j(t) é o perfil de eventos de pedido da rede ao exterior para o produto p_j.[32]

---

[31] A operação TEMPO produz o valor de tempo do evento que representa uma tarefa de retalho (ou uma tarefa de matéria-prima), ver a secção 3.7.2.3.

[32] Os limites temporais do horizonte temporal dos perfis de eventos em $\mathbb{ed}$(t) devem ser iguais.

Nós de matéria-prima são representados por dois círculos concêntricos, mais pequenos que os nós de capacidade. Na Figura 3-11 representam-se esquematicamente um nó de matéria-prima com um único produto de saída p (nó de matéria-prima dedicado, ou mono-produto) e um nó de matéria-prima com os produtos de saída p_1, p_2 e p_3 (nó de matéria-prima partilhado, ou multi-produto). As setas indicando os produtos de saída do nó de matéria-prima são usadas mais no contexto de uma representação de rede física e correspondem a arcos de rede terminando noutros nós, de capacidade.



a) Nó de matéria-prima dedicado. b) Nó de matéria-prima partilhado.

Figura 3-11- Nós de matéria-prima.

3.4.2.6 Operações com Nós de Matéria-Prima

As operações primitivas sobre nós de matéria-prima são:

LOCAL(<v,M,$\mathcal{p}$,$\mathcal{ed}$(t)>) ::= v.

TIPO(<v,M, $\mathcal{p}$,$\mathcal{ed}$(t)>) ::= M.

PRODUTOS(<v,M, $\mathcal{p}$,$\mathcal{ed}$(t)>) ::= $\mathcal{p}$.

PERFIS-E(<v,M, $\mathcal{p}$,$\mathcal{ed}$(t)>) ::= $\mathcal{ed}$(t).

NÓ(v,M, $\mathcal{p}$,$\mathcal{ed}$(t)) ::= <v,M, $\mathcal{p}$,$\mathcal{ed}$(t)>.

(em que os parâmetros $\mathcal{p}$ e $\mathcal{ed}$(t) devem estar de acordo com o tipo de nó e entre si no que respeita ao número e ordem dos seus elementos, e sendo $TS(ed_i(t))=TS(ed_j(t))$ e $TE(ed_i(t))=TE(ed_j(t))$ para quaisquer $ed_i(t)$ e $ed_j(t)$ tais que $ed_i(t)\in\mathcal{ed}(t)$ e $ed_j(t)\in\mathcal{ed}(t)$).

Definem-se também as seguintes operações:

TODOS-PRODUTOS(<v,M,$\mathcal{p}$,$\mathcal{ed}$(t)>) ::= $\mathcal{p}$.

PERFIL-E($\mathbb{V}$,p_j) ::=
$ed_j(t)$: ($ed_j(t)$=NÉSIMO(PERFIS-E($\mathbb{V}$),POSIÇÃO(PRODUTOS($\mathbb{V}$),p_j))).

O predicado ESCALONÁVEL? testa se uma tarefa tarefa de matéria-prima é escalonável num nó de matéria-prima.[33]

```
ESCALONÁVEL?(𝕍,𝕆) ::=
(LOCAL(𝕍)=LOCAL(𝕆)) ∧ (PRODUTO(𝕆)∈PRODUTOS(𝕍)) ∧
DURANTE?( EVENTO(𝕆),
          INTERVALO(  TS(PERFIL-E(𝕍,PRODUTO(𝕆)),
                      TE(PERFIL-E(𝕍,PRODUTO(𝕆))))).
```

A operação seguinte escalona efectivamente uma tarefa de matéria-prima 𝕆 num nó de matéria-prima 𝕍 (só aplicável caso se verifique ESCALONÁVEL?(𝕍,𝕆)).

```
ESCALONA(𝕍,𝕆) ::=
NÓ( LOCAL(𝕍),
    TIPO(𝕍),
    PRODUTOS(𝕍),
    SUBSTITUI(  PERFIS-E(𝕍),
                PERFIL-E(𝕍,PRODUTO(𝕆)),
                CARREGA-EVENTO( PERFIL-E(𝕍,PRODUTO(𝕆)),
                                EVENTO(𝕆)))).
```

A operação seguinte desfaz o escalonamento antes realizado para a tarefa 𝕆, num nó de matéria-prima 𝕍.

```
DES-ESCALONA(𝕍,𝕆) ::=
NÓ( LOCAL(𝕍),
    TIPO(𝕍),
    PRODUTOS(𝕍),
    SUBSTITUI(  PERFIS-E(𝕍),
                PERFIL-E(𝕍,PRODUTO(𝕆)),
                DESCARREGA-EVENTO(  PERFIL-E(𝕍,PRODUTO(𝕆)),
                                    EVENTO(𝕆)))).
```

As operações ESCALONA e DES-ESCALONA têm a seguinte propriedade:

DES-ESCALONA(ESCALONA(𝕍,𝕆),𝕆)=𝕍

e:

ESCALONA(DES-ESCALONA(𝕍,𝕆),𝕆)=𝕍

isto é, um escalonamento de uma tarefa imediatamente seguido da operação que o desfaz deixa o nó no mesmo estado; o mesmo se diz para a sequência inversa. Adicionalmente, a operação

[33] As operações PRODUTO, LOCAL e EVENTO produzem, respectivamente, o produto de uma tarefa de matéria-prima, o identificador do nó de matéria-prima para o qual uma tarefa de matéria-prima foi construída e o evento de uma tarefa de matéria-prima dada, ver a secção 3.7.2.2. Uma tarefa é escalonável num nó de matéria-prima se foi construída para esse nó.

ESCALONADA? e as operações identificadas por TAREFAS-ESCALONADAS, anteriormente definidas para nós de retalho, são definidas do mesmo modo para nós de matéria-prima.

3.5 Arco

Um *arco* é uma ligação direccionada entre quaisquer dois nós, que indica troca possível de produtos entre os dois nós. Os arcos representam relações *fornecedor-cliente* entre pares de nós. O sentido do arco é de um nó fornecedor, no extremo montante do arco, para um nó cliente, no extremo jusante do arco. Não existem arcos partindo de nós de retalho de uma rede física e não existem arcos terminando em nós de matéria-prima de uma rede física.

Formalmente, um arco ligando o nó $\mathbb{V}_x$ ao nó $\mathbb{V}_y$, simbolicamente por $\mathbb{E}_{x,y}$, é descrito por um triplo:

$<\mathbb{V}_x, \mathbb{V}_y, pm>$

sendo:

- $\mathbb{V}_x \in \mathbb{V}$ o *nó fornecedor* (o nó no extremo montante do arco), qualquer nó de uma rede física que não seja um nó de retalho da rede;
- $\mathbb{V}_y \in \mathbb{V}$ o *nó cliente* (o nó no extremo jusante do arco), qualquer nó de uma rede física que não seja um nó de matéria-prima;
- $pm = \{..., p_k, ...\}$ (com $|pm| > 0$ e $pm \subset \mathbb{P}$), é o conjunto não vazio de produtos de que o nó $\mathbb{V}_x$ é fornecedor e o nó $\mathbb{V}_y$ é cliente, isto é, $pm \subset (\text{PRODUTOS}(\mathbb{V}_x) \cap \text{MATERIAIS}(\mathbb{V}_y))$.

Os arcos entre pares de nós de uma rede são esquematicamente representados por setas cujo sentido indica o sentido possível dos fluxos de produtos de um nó para o outro. Na Figura 3-12-a) representa-se, como exemplo, um segmento de rede com quatro nós $\mathbb{V}_1$, $\mathbb{V}_2$, $\mathbb{V}_3$ e $\mathbb{V}_4$, (etiquetados pelos identificadores v_1, v_2, v_3, e v_4, respectivamente) ligados pelos arcos $\mathbb{E}_{1,3}$, $\mathbb{E}_{2,3}$ e $\mathbb{E}_{3,4}$. Neste exemplo, os nós $\mathbb{V}_1$ e $\mathbb{V}_2$ são nós fornecedores do nó $\mathbb{V}_3$ e este é nó cliente de $\mathbb{V}_1$ e $\mathbb{V}_2$. Por sua vez, o nó $\mathbb{V}_3$ é nó fornecedor do nó $\mathbb{V}_4$ e este é cliente do nó $\mathbb{V}_3$. Em vez de etiquetar graficamente cada arco de rede com um símbolo denotando o arco, como é feito na Figura 3-12-a), pode-se fazê-lo indicando o conjunto pm associado, como é feito na Figura 3-12-b).

3.5.1.1 Operações com Arcos

As operações primitivas selectoras para arcos são:

NÓ-FORNECEDOR($<\mathbb{V}_x, \mathbb{V}_y, pm>$) ::= $\mathbb{V}_x$.

NÓ-CLIENTE($<\mathbb{V}_x, \mathbb{V}_y, pm>$) ::= $\mathbb{V}_y$.

PRODUTOS($<\mathbb{V}_x, \mathbb{V}_y, pm>$) ::= pm.

A operação ARCO-VÁLIDO?, que se define a seguir, testa se é possível haver um arco de rede com um conjunto de produtos associados pm, partindo de um nó $\mathbb{V}_x$ e terminando num nó $\mathbb{V}_y$.

ARCO-VÁLIDO?($\mathbb{V}_x$,$\mathbb{V}_y$,pm) ::=
($\mathbb{V}_x \neq \mathbb{V}_y$) $\wedge$ (|pm|>0) $\wedge$ ($pm \subset$(PRODUTOS($\mathbb{V}_x$)$\cap$MATERIAIS($\mathbb{V}_y$))).

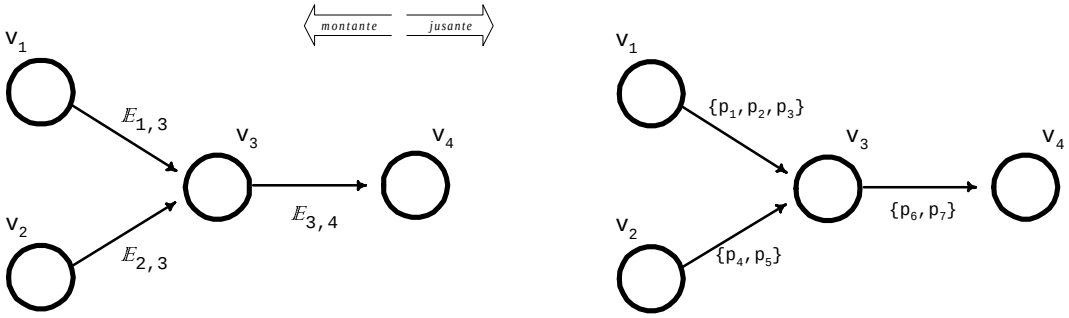


Figura 3-12- Arcos entre nós de uma rede.

A operação primitiva construtora que se descreve a seguir constrói um arco dados os nós extremos do arco $\mathbb{V}_x$ e $\mathbb{V}_y$ e só deve ser usada no caso de verificar ARCO-VÁLIDO?($\mathbb{V}_x$,$\mathbb{V}_y$,pm).

ARCO($\mathbb{V}_x$,$\mathbb{V}_y$,pm) ::= <$\mathbb{V}_x$,$\mathbb{V}_y$,pm>.

3.6 Rede

Uma *rede física*, ou simplesmente, *rede* é um conjunto de nós ligados por *arcos* formando um grafo acíclico e conexo. Produtos são acumulados e processados em nós de uma rede e transferidos de uns nós para outros, a partir dos *nós de matéria-prima* até aos *nós de retalho*, no sentido de montante para jusante (no sentido dos arcos da rede), para obter produtos finais da rede. Os arcos da rede significam um conjunto de "canais" de troca de produtos entre grupos de recursos. Numa rede, os nós de matéria-prima estão posicionados nos extremos a montante da rede e os nós de retalho no extremo a jusante.

Uma rede $\mathbb{RF}$, é formalmente descrita por um par:

<$\mathbb{V}$, $\mathbb{E}$>

sendo:

– $\mathbb{V}=\{\mathbb{V}_1,\ldots,\mathbb{V}_{Nn}\}$, o conjunto dos nós da rede contendo, pelo menos, um nó de capacidade, um nó de retalho e um nó de matéria-prima (Nn nós, com $Nn>2$);[34]

– $\mathbb{E}=\{\ldots,\mathbb{E}_{x,y},\ldots\}$, o conjunto de arcos da rede contendo, por cada par de nós no máximo um arco; os nós em $\mathbb{V}$ e os arcos em $\mathbb{E}$ devem formar um grafo acíclico e conexo.

Os produtos consumidos pelos nós de retalho de uma rede física são apelidados de *produtos finais* ou *produtos de saída da rede*. Os produtos entregues pelos nós de matéria-prima de uma rede física são apelidados *produtos de entrada* ou *matérias-primas da rede*.

Redes permitem modelar redes de produção e distribuição, em que os recursos (fábricas, armazéns, unidades de transporte) são representadas pelos nós e as ligações fornecedor-cliente entre os recursos são representadas pelos arcos.

Uma rede é esquematicamente representada por um *grafo* acíclico direccionado onde se incluem os nós e os arcos. Nós de matéria-prima e nós de retalho são distinguidos dos nós de capacidade representando-os mais pequenos e através de dois círculos concêntricos, no primeiro caso e círculos a cinzento, no último caso.

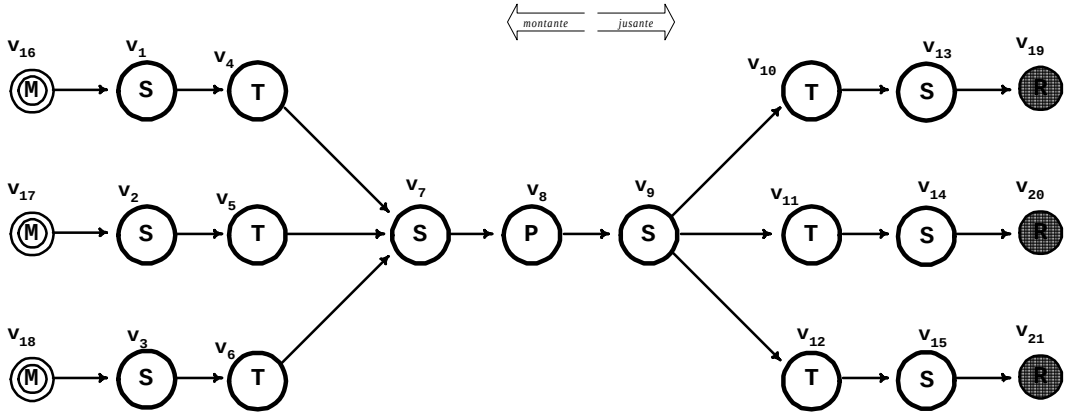


Figura 3-13- Uma rede física.

Como exemplo, na Figura 3-13 é esquematicamente representada uma rede física com 15 nós de capacidade, rotulados v_1 a v_{15}, dos quais v_1, v_2, v_3, v_7, v_9, v_{13}, v_{14} e v_{15} são acumuladores e v_4, v_5, v_6, v_8, v_{10}, v_{11} e v_{12} processadores (v_4, v_5, v_6, v_{10}, v_{11} e v_{12} são transportadores e v_8 é um produtor). Os nós rotulados v_{16}, v_{17} e v_{18} são os nós de matéria-prima e os rotulados v_{19}, v_{20} e v_{21} os nós de retalho desta rede.

3.6.1.1 Operações com Redes

As operações primitivas selectoras para redes físicas são definidas a seguir:

NÓS(<$\mathbb{V}$,$\mathbb{E}$>) ::= $\mathbb{V}$.

ARCOS(<$\mathbb{V}$,$\mathbb{E}$>) ::= $\mathbb{E}$.

[34] Os limites temporais do horizonte temporal dos perfis de capacidade e dos perfis de eventos dos nós em $\mathbb{V}$ devem ser todos iguais.

O predicado A-MONTANTE?, a seguir definido (de forma recursiva), testa se, dados o conjunto de nós $\mathbb{V}$ e o conjunto de arcos $\mathbb{E}$ de uma hipotética rede física, um nó $\mathbb{V}_i$ está ligado por um caminho de rede (sequência de arcos de rede) a um outro nó $\mathbb{V}_j$, isto é, se $\mathbb{V}_i$ é fornecedor, directo ou indirecto, de $\mathbb{V}_j$, atendendo às relações fornecedor-cliente (atendendo apenas aos arcos e não aos produtos associados aos arcos) em $\mathbb{E}$.[35]

A-MONTANTE?($\mathbb{V}$,$\mathbb{E}$,$\mathbb{V}_i$,$\mathbb{V}_j$) ::=
($\mathbb{V}_i \in \mathbb{V}$) $\wedge$ ($\mathbb{V}_j \in \mathbb{V}$) $\wedge$ ($\mathbb{V}_i \neq \mathbb{V}_j$) $\wedge$ (TIPO($\mathbb{V}_j$)≠M) $\wedge$ (TIPO($\mathbb{V}_i$)≠R) $\wedge$
(($\exists_{\mathbb{E}_k}$: ($\mathbb{E}_k \in \mathbb{E}$) $\wedge$ (NÓ-FORNECEDOR($\mathbb{E}_k$)=$\mathbb{V}_i$) $\wedge$ (NÓ-CLIENTE($\mathbb{E}_k$)=$\mathbb{V}_j$) $\wedge$
ARCO-VÁLIDO?($\mathbb{V}_i$,$\mathbb{V}_j$,PRODUTOS($\mathbb{E}_k$))) $\vee$
($\exists_{\mathbb{E}_k,\mathbb{V}_n}$: ($\mathbb{E}_k \in \mathbb{E}$) $\wedge$ ($\mathbb{V}_n \in \mathbb{V}$) $\wedge$ ($\mathbb{V}_n \neq \mathbb{V}_i$) $\wedge$ ($\mathbb{V}_n \neq \mathbb{V}_j$) $\wedge$
(NÓ-FORNECEDOR($\mathbb{E}_k$)=$\mathbb{V}_n$) $\wedge$ (NÓ-CLIENTE($\mathbb{E}_k$)=$\mathbb{V}_j$) $\wedge$
ARCO-VÁLIDO?($\mathbb{V}_n$,$\mathbb{V}_j$,PRODUTOS($\mathbb{E}_k$)) $\wedge$
A-MONTANTE?($\mathbb{V}$,$\mathbb{E}$,$\mathbb{V}_i$,$\mathbb{V}_n$))).

O predicado a seguir definido é também identificado por A-MONTANTE?, e é semelhante ao anterior mas atende à relações fornecedor-cliente para cada produto. Este predicado testa se, dados o conjunto de nós $\mathbb{V}$ e o conjunto de arcos $\mathbb{E}$ de uma hipotética rede, um nó $\mathbb{V}_i$ está ligado por um caminho de rede a um outro nó $\mathbb{V}_j$ para um dado produto p de $\mathbb{V}_j$, isto é, se $\mathbb{V}_i$ é fornecedor, directo ou indirecto, de produtos de entrada de $\mathbb{V}_j$ para o produto p.

A-MONTANTE?($\mathbb{V}$,$\mathbb{E}$,$\mathbb{V}_i$,$\mathbb{V}_j$,p) ::=
(p∈PRODUTOS($\mathbb{V}_j$) $\wedge$ ($\mathbb{V}_i \in \mathbb{V}$) $\wedge$ ($\mathbb{V}_j \in \mathbb{V}$) $\wedge$ ($\mathbb{V}_i \neq \mathbb{V}_j$) $\wedge$ (TIPO($\mathbb{V}_j$)≠M) $\wedge$
(TIPO($\mathbb{V}_i$)≠R) $\wedge$
(($\exists_{\mathbb{E}_k,m}$: ($\mathbb{E}_k \in \mathbb{E}$) $\wedge$ (m∈MATERIAIS-PARA-PRODUTO($\mathbb{V}_j$,p)) $\wedge$
(NÓ-FORNECEDOR($\mathbb{E}_k$)=$\mathbb{V}_i$) $\wedge$ (NÓ-CLIENTE($\mathbb{E}_k$)=$\mathbb{V}_j$) $\wedge$
ARCO-VÁLIDO?($\mathbb{V}_i$,$\mathbb{V}_j$,PRODUTOS($\mathbb{E}_k$)) $\wedge$
(m∈(PRODUTOS($\mathbb{E}_k$)∩PRODUTOS($\mathbb{V}_i$)))) $\vee$
($\exists_{\mathbb{E}_k,\mathbb{V}_n,m}$: ($\mathbb{E}_k \in \mathbb{E}$) $\wedge$ ($\mathbb{V}_n \in \mathbb{V}$) $\wedge$ ($\mathbb{V}_n \neq \mathbb{V}_i$) $\wedge$ ($\mathbb{V}_n \neq \mathbb{V}_j$) $\wedge$
(m∈MATERIAIS-PARA-PRODUTO($\mathbb{V}_j$,p)) $\wedge$
(NÓ-FORNECEDOR($\mathbb{E}_k$)=$\mathbb{V}_n$) $\wedge$ (NÓ-CLIENTE($\mathbb{E}_k$)=$\mathbb{V}_j$) $\wedge$
ARCO-VÁLIDO?($\mathbb{V}_n$,$\mathbb{V}_j$,PRODUTOS($\mathbb{E}_k$)) $\wedge$
(m∈(PRODUTOS($\mathbb{E}_k$)∩PRODUTOS($\mathbb{V}_n$))) $\wedge$
A-MONTANTE?($\mathbb{V}$,$\mathbb{E}$,$\mathbb{V}_i$,$\mathbb{V}_n$,m))).

O predicado REDE-VÁLIDA?, que se define a seguir, testa se um conjunto de nós $\mathbb{V}$ e um conjunto de arcos $\mathbb{E}$ dados podem constituir uma rede física.

REDE-VÁLIDA?($\mathbb{V}$,$\mathbb{E}$) ::=

[35] Esta operação é definida de forma recursiva.

```
(∃V_r  : (V_r∈V) ∧ (TIPO(V_r)=R)) ∧        ; há em V pelo menos 1
(∃V_m  : (V_m∈V) ∧ (TIPO(V_m)=M)) ∧        ; nó R, 1 nó M e 1
(∃V_k  : (V_k∈V) ∧                         ; 1 nó S, P, ou T
         ((TIPO(V_k)=S) ∨ (TIPO(V_k)=P) ∨ (TIPO(V_k)=T))) ∧

(∀E_k  (E_k∈E)  :                          ; todos arcos em E unem nós em V
       (∃V_i,V_j : (V_i∈V) ∧ (V_j∈V) ∧
                   (NÓ-FORNECEDOR(E_k)=V_i) ∧
                   (NÓ-CLIENTE(E_k)=V_j) ∧
                   ARCO-VÁLIDO?(V_i,V_j,PRODUTOS(E_k)))) ∧

                                           ; todos nós R, S, P ou T em V são
(∀V_j  (V_j∈V) :                           ; clientes de algum nó em V
       ((TIPO(V_j)=R) ∨ (TIPO(V_i)=S) ∨ (TIPO(V_i)=P) ∨
        (TIPO(V_i)=T)) ∧
       (∃E_k,V_i  : (E_k∈E) ∧ (V_i∈V) ∧
                   (NÓ-FORNECEDOR(E_k)=V_i) ∧
                   (NÓ-CLIENTE(E_k)=V_j) ∧
                   ARCO-VÁLIDO?(V_i,V_j,PRODUTOS(E_k)))) ∧

                                           ; todos nós M, S, P ou T de V são
(∀V_i  (V_i∈V) :                           ; fornecedores de algum nó de V
       ((TIPO(V_i)=M) ∨ (TIPO(V_i)=S) ∨ (TIPO(V_i)=P) ∨
        (TIPO(V_i)=T)) ∧
       (∃E_k,V_j  : (E_k∈E) ∧ (V_j∈V) ∧
                   (NÓ-FORNECEDOR(E_k)=V_i) ∧
                   (NÓ-CLIENTE(E_k)=V_j) ∧
                   ARCO-VÁLIDO?(V_i,V_j,PRODUTOS(E_k)))) ∧

                                           ; cada par de nós em V é unido por,
(∀V_i,V_j (V_i∈V) ∧ (V_j∈V) ∧ (V_i≠V_j)  :     ; no máximo, 1 arco
       (MAX( |{ E_k : (E_k∈E) ∧            ; de E
                   (NÓ-FORNECEDOR(E_k)=V_i) ∧
                   (NÓ-CLIENTE(E_k)=V_j) }| ) = 1)) ∧

                                           ; para quaisquer dois nós de V, ou
                                           ; um está a montante do outro, ou
                                           ; nenhum deles está a montante do
(∀V_i,V_j (V_i∈V) ∧ (V_j∈V)  :             ; outro
       (A-MONTANTE?(V,E,V_i,V_j) ∧
                   (¬ A-MONTANTE?(V,E,V_j,V_i))) ∨
```

$$((\neg\ \text{A-MONTANTE?}(\mathcal{V},\mathcal{E},\mathbb{V}_i,\mathbb{V}_j)) \wedge \text{A-MONTANTE?}(\mathcal{V},\mathcal{E},\mathbb{V}_j,\mathbb{V}_i)) \vee ((\neg\ \text{A-MONTANTE?}(\mathcal{V},\mathcal{E},\mathbb{V}_i,\mathbb{V}_j)) \wedge (\neg\ \text{A-MONTANTE?}(\mathcal{V},\mathcal{E},\mathbb{V}_j,\mathbb{V}_i))).$$

Note-se que a última expressão lógica na definição do predicado REDE-VÁLIDA? impede que um conjunto de nós e um conjunto de arcos formando um grafo contendo ciclos possa ser considerado válido para construir uma rede física. A expressão referida exprime que, se um nó $\mathbb{V}_i$ está a montante de outro nó $\mathbb{V}_j$, então $\mathbb{V}_j$ não pode estar a montante de $\mathbb{V}_i$, e vice-versa. Note-se que é perfeitamente possível terem-se casos em que nem $\mathbb{V}_i$ está a montante de $\mathbb{V}_j$ nem $\mathbb{V}_j$ está a montante de $\mathbb{V}_i$, designadamente se são o mesmo nó, se $\mathbb{V}_i$ e/ou $\mathbb{V}_j$ são nós de retalho ou de matéria-prima, ou se são nós que estão localizados em ramos diferentes de uma rede convergente ou divergente.

A operação primitiva construtora para redes é:

$$\text{REDE}(\mathcal{V},\mathcal{E}) ::= <\mathcal{V},\mathcal{E}>.$$

A operação REDE é aplicável só se o conjunto de nós e o conjunto de arcos dados $\mathcal{V}$ e $\mathcal{E}$, são tais que se verifica REDE-VÁLIDA?($\mathcal{V}$, $\mathcal{E}$). O predicado NÓ?, a seguir definido, testa se, dados uma rede e um nó, este é um nó da rede.

$$\text{NÓ?}(\mathbb{RF},\mathbb{V}_k) ::= (\mathbb{V}_k \in \text{NÓS}(\mathbb{RF})).$$

O predicado ARCO? testa se, dados uma rede e dois nós estes são nós da rede e o primeiro é fornecedor do segundo.

$$\text{ARCO?}(\mathbb{RF},\mathbb{V}_x,\mathbb{V}_y) ::= \text{NÓ?}(\mathbb{RF},\mathbb{V}_x) \wedge \text{NÓ?}(\mathbb{RF},\mathbb{V}_y) \wedge (\exists_{\mathbb{E}_{x,y}} : (\mathbb{E}_{x,y} \in \text{ARCOS}(\mathbb{RF})) \wedge (\text{NÓ-FORNECEDOR}(\mathbb{E}_{x,y})=\mathbb{V}_x) \wedge (\text{NÓ-CLIENTE}(\mathbb{E}_{x,y})=\mathbb{V}_y)).$$

A operação NÓ produz um nó de uma rede dada, dado o identificador do nó:

$$\text{NÓ}(\mathbb{RF},v_k) ::= \mathbb{V}_k : \text{NÓ?}(\mathbb{RF},\mathbb{V}_k) \wedge (\text{LOCAL}(\mathbb{V}_k)=v_k).$$

A operação ARCO produz um arco de uma rede dada, dados os nós fornecedor e cliente:

$$\text{ARCO}(\mathbb{RF},\mathbb{V}_x,\mathbb{V}_y) ::= \mathbb{E}_{x,y} : \text{ARCO?}(\mathbb{RF},\mathbb{V}_x,\mathbb{V}_y).$$

As seguintes operações são também definidas:

$$\text{TODOS-PRODUTOS}(\mathbb{RF}) ::= \quad ; \textit{todos os produtos de } \mathbb{RF}$$

$\{ p_j : (\exists_{\mathbb{V}_k} : \text{NÓ?}(\mathbb{RF},\mathbb{V}_k) \wedge (p_j \in \text{TODOS-PRODUTOS}(\mathbb{V}_k))) \}.$

(o conjunto $\mathcal{P}$).

$\text{NÓS-CLIENTES}(\mathbb{RF},\mathbb{V}_k) ::=$ *; nós clientes de* $\mathbb{V}_k$
$\{ \mathbb{V}_i : \text{NÓ?}(\mathbb{RF},\mathbb{V}_i) \wedge (\text{TIPO}(\mathbb{V}_i) \neq \text{M}) \wedge \text{ARCO?}(\mathbb{RF},\mathbb{V}_k,\mathbb{V}_i) \}.$

$\text{NÓS-CLIENTES}(\mathbb{RF},\mathbb{V}_k,p_j) ::=$ *; nós clientes de* $\mathbb{V}_k$ *para* p_j
$\{ \mathbb{V}_i : (\mathbb{V}_i \in \text{NÓS-CLIENTES}(\mathbb{RF},\mathbb{V}_k)) \wedge (p_j \in \text{TODOS-PRODUTOS}(\mathbb{RF})) \wedge$
$(\exists_{\mathbb{E}_{k,i}} : (\mathbb{E}_{k,i} = \text{ARCO}(\mathbb{RF},\mathbb{V}_k,\mathbb{V}_i)) \wedge (p_j \in \text{PRODUTOS}(\mathbb{E}_{k,i}))) \}.$

$\text{NÓS-CLIENTES}(\mathbb{RF},p_j) ::=$ *; nós que consomem* p_j
$\{ \mathbb{V}_i : \text{NÓ?}(\mathbb{RF},\mathbb{V}_i) \wedge (\text{TIPO}(\mathbb{V}_i) \neq \text{M}) \wedge$
$(p_j \in \text{TODOS-PRODUTOS}(\mathbb{RF})) \wedge (p_j \in \text{MATERIAIS}(\mathbb{V}_i)) \}.$

$\text{NÓS-FORNECEDORES}(\mathbb{RF},\mathbb{V}_k) ::=$ *; nós fornecedores de* $\mathbb{V}_k$
$\{ \mathbb{V}_i : \text{NÓ?}(\mathbb{RF},\mathbb{V}_i) \wedge (\text{TIPO}(\mathbb{V}_i) \neq \text{R}) \wedge \text{ARCO?}(\mathbb{RF},\mathbb{V}_i,\mathbb{V}_k) \}.$

$\text{NÓS-FORNECEDORES}(\mathbb{RF},\mathbb{V}_k,p_j) ::=$ *; fornecedores de* $\mathbb{V}_k$ *para* p_j
$\{ \mathbb{V}_i : (\mathbb{V}_i \in \text{NÓS-FORNECEDORES}(\mathbb{RF},\mathbb{V}_k)) \wedge$
$(p_j \in \text{TODOS-PRODUTOS}(\mathbb{RF})) \wedge$
$(\exists_{\mathbb{E}_{i,k}} : (\mathbb{E}_{i,k} = \text{ARCO}(\mathbb{RF},\mathbb{V}_i,\mathbb{V}_k)) \wedge (p_j \in \text{PRODUTOS}(\mathbb{E}_{i,k}))) \}.$

$\text{NÓS-FORNECEDORES}(\mathbb{RF},p_j) ::=$ *; nós que fornecem* p_j
$\{ \mathbb{V}_i : \text{NÓ?}(\mathbb{RF},\mathbb{V}_i) \wedge (\text{TIPO}(\mathbb{V}_i) \neq \text{R}) \wedge$
$(p_j \in \text{TODOS-PRODUTOS}(\mathbb{RF})) \wedge (p_j \in \text{PRODUTOS}(\mathbb{V}_i)) \}.$

$\text{NÓS-DE-RETALHO}(\mathbb{RF}) ::= \{ \mathbb{V}_i : \text{NÓ?}(\mathbb{RF},\mathbb{V}_i) \wedge (\text{TIPO}(\mathbb{V}_i) = \text{R}) \}.$

$\text{NÓS-DE-RETALHO}(\mathbb{RF},p_j) ::=$ *; nós* R *que consomem* p_j
$\{ \mathbb{V}_i : (\mathbb{V}_i \in \text{NÓS-DE-RETALHO}(\mathbb{RF})) \wedge (p_j \in \text{TODOS-PRODUTOS}(\mathbb{RF})) \wedge$
$(p_j \in \text{MATERIAIS}(\mathbb{V}_i)) \}.$

$\text{NÓS-DE-MATÉRIA-PRIMA}(\mathbb{RF}) ::=$
$\{ \mathbb{V}_i : \text{NÓ?}(\mathbb{RF},\mathbb{V}_i) \wedge (\text{TIPO}(\mathbb{V}_i) = \text{M}) \}.$

$\text{NÓS-DE-MATÉRIA-PRIMA}(\mathbb{RF},p_j) ::=$ *; nós* M *que fornecem* p_j
$\{ \mathbb{V}_i : (\mathbb{V}_i \in \text{NÓS-DE-MATÉRIA-PRIMA}(\mathbb{RF})) \wedge$
$(p_j \in \text{TODOS-PRODUTOS}(\mathbb{RF})) \wedge (p_j \in \text{PRODUTOS}(\mathbb{V}_i)) \}.$

$\text{PRODUTOS}(\mathbb{RF}) ::=$ *; produtos de saída da rede*

$\{ p_j : (p_j \in$ TODOS-PRODUTOS($\mathbb{RF}$)) $\wedge$ (|NÓS-DE-RETALHO($\mathbb{RF}, p_j$)|>0) $\}$.

MATERIAIS($\mathbb{RF}$) ::= *; produtos de entrada da rede*
$\{ p_j : (p_j \in$ TODOS-PRODUTOS($\mathbb{RF}$)) $\wedge$
(|NÓS-DE-MATÉRIA-PRIMA($\mathbb{RF}, p_j$)|>0) $\}$.

A operação COMPONENTES-DE-PRODUTO produz o conjunto de produtos que compõem um dado produto de saída p, de um dado nó $\mathbb{V}_k$, dada a rede $\mathbb{RF}$ a que $\mathbb{V}_k$ pertence. Esta operação é definida da seguinte forma (recursiva):

COMPONENTES-DE-PRODUTO($\mathbb{RF}, \mathbb{V}_k$, p) ::=
SE(($\mathbb{V}_k \in$ NÓS-DE-MATÉRIA-PRIMA($\mathbb{RF}$, p)) $\vee$
($\mathbb{V}_k \notin$ (NÓS-FORNECEDORES($\mathbb{RF}$, p) \
NÓS-DE-MATÉRIA-PRIMA($\mathbb{RF}$, p))),
{ },
MATERIAIS-PARA-PRODUTO($\mathbb{V}_k$, p) $\cup$
($\bigcup_{m \in \text{MATERIAIS-PARA-PRODUTO}(\mathbb{V}_k, p),\ \mathbb{V}_i \in \text{NÓS-FORNECEDORES}(\mathbb{RF}, \mathbb{V}_k, m)}$ COMPONENTES-DE-PRODUTO($\mathbb{RF}, \mathbb{V}_i$, m))).

A operação DERIVADOS-DE-PRODUTO produz o conjunto de produtos dos quais um dado produto de entrada m, de um dado nó $\mathbb{V}_k$ é componente, dada a rede $\mathbb{RF}$ a que $\mathbb{V}_k$ pertence. Esta operação é definida da seguinte forma (recursiva):

DERIVADOS-DE-PRODUTO($\mathbb{RF}, \mathbb{V}_k$, m) ::=
SE(($\mathbb{V}_k \in$ NÓS-DE-RETALHO($\mathbb{RF}$, m)) $\vee$
($\mathbb{V}_k \notin$ (NÓS-CLIENTES($\mathbb{RF}$, m) \ NÓS-DE-RETALHO($\mathbb{RF}$, m))),
{ },
PRODUTOS-DE-MATERIAL($\mathbb{V}_k$, m) $\cup$
($\bigcup_{p \in \text{PRODUTOS-DE-MATERIAL}(\mathbb{V}_k, m),\ \mathbb{V}_i \in \text{NÓS-CLIENTES}(\mathbb{RF}, \mathbb{V}_k, p)}$ DERIVADOS-DE-PRODUTO($\mathbb{RF}, \mathbb{V}_i$, p))).

A operação MATERIAIS-PARA-PRODUTO produz o conjunto de produtos de entrada da rede física (e apenas esses) que fazem parte de um dado produto de saída p, de um dado nó $\mathbb{V}_k$, dada a rede $\mathbb{RF}$ a que $\mathbb{V}_k$ pertence. Esta operação é definida da seguinte forma (recursiva):

MATERIAIS-PARA-PRODUTO($\mathbb{RF}, \mathbb{V}_k$, p) ::=
SE(($\mathbb{V}_k \in$ NÓS-DE-MATÉRIA-PRIMA($\mathbb{RF}$, p)) $\vee$
($\mathbb{V}_k \notin$ (NÓS-FORNECEDORES($\mathbb{RF}$, p) \
NÓS-DE-MATÉRIA-PRIMA($\mathbb{RF}$, p))),
{ p },
$\bigcup_{m \in \text{MATERIAIS-PARA-PRODUTO}(\mathbb{V}_k, p),\ \mathbb{V}_i \in \text{NÓS-FORNECEDORES}(\mathbb{RF}, \mathbb{V}_k, m)}$ MATERIAIS-PARA-PRODUTO($\mathbb{RF}, \mathbb{V}_i$, m)).

A operação PRODUTOS-DE-MATERIAL produz o conjunto de produtos finais da rede (e apenas esses) dos quais um dado produto de entrada m, de um dado nó $\mathbb{V}_k$ é componente, dada a rede $\mathbb{RF}$ a que $\mathbb{V}_k$ pertence. Esta operação é definida da seguinte forma (recursiva):

$$\begin{aligned}&\text{PRODUTOS-DE-MATERIAL}(\mathbb{RF},\mathbb{V}_k,m) ::= \\ &\text{SE}(\ (\mathbb{V}_k \in \text{NÓS-DE-RETALHO}(\mathbb{RF},m)) \vee \\ &\qquad (\mathbb{V}_k \notin (\text{NÓS-CLIENTES}(\mathbb{RF},m) \setminus \text{NÓS-DE-RETALHO}(\mathbb{RF},m))), \\ &\quad \{\ m\ \}, \\ &\qquad \bigcup_{\substack{p \in \text{PRODUTOS-DE-MATERIAL}(\mathbb{V}_k,m), \\ \mathbb{V}_i \in \text{NÓS\_CLIENTES}(\mathbb{RF},\mathbb{V}_k,p)}} \text{DERIVADOS-DE-PRODUTO}(\mathbb{RF},\mathbb{V}_i,p)).\end{aligned}$$

3.7 Tarefa

Uma *tarefa* é a unidade de escalonamento no ambiente da produção e distribuição. Uma tarefa é criada para ser escalonada num nó de rede física, *i.e.*, a tarefa *pertence* a um nó.[36]

Existem três classes, ou tipos, de tarefas: *tarefa de capacidade*, *tarefa de retalho* e *tarefa de matéria-prima*. Uma tarefa de capacidade consome *produtos de entrada* e torna disponível um único *produto de saída* (o produto da tarefa) e requer, para isso, a utilização temporária da capacidade de um nó de capacidade. Uma tarefa de capacidade pode ser *elementar* (ou *atómica*) ou *não elementar* (ou *não atómica*), conforme não seja, ou seja, decomposta em duas ou mais tarefas mais simples, apelidadas de *sub-tarefas*. No caso de *decomposição* a tarefa original é apelidada de *tarefa mãe* das sub-tarefas; as sub-tarefas são tarefas elementares. Tarefas de retalho e tarefas de matéria-prima são *tarefas fictícias* que representam, respectivamente, o consumo de produtos e a entrega de produtos pelo exterior de uma rede física.

3.7.1 Tarefa de Capacidade

Tarefas de capacidade podem ser elementares ou não elementares. Formalmente, uma tarefa elementar $\mathbb{O}$ é definida por um triplo:

$<e_s, e_e, \text{cval}>$

sendo:

- e_s o evento extremo da tarefa denominado *evento de início*, ou *evento de entrada*, da tarefa;
- e_e o evento extremo da tarefa denominado *evento de fim*, ou *evento de saída*, da tarefa. É um evento simples;

[36] Como se verá no Capítulo 4, as tarefas não são criadas por um agente central e depois distribuídas por, e atribuídas a, agentes que as escalonam nos nós por eles geridos. Neste modelo, uma tarefa está sempre associada a um nó, é um objecto privado e só conhecido do agente que gere esse nó e é criada para ser escalonada nesse nó, para satisfação de um pedido (encomenda) de outro agente.

– `cval`, a quantidade de capacidade afectada, ou a afectar, à tarefa, quando do escalonamento da tarefa no nó a que pertence, em cada instante durante o intervalo da tarefa.[37]

Os eventos e_s e e_e descrevem os valores das dimensões produto, quantidade, tempo e local, à entrada e à saída da tarefa (no início e no fim do intervalo de tempo da tarefa, respectivamente). Sendo $e_s=<pq_s,t_s,v>$ e $e_e=<pq_e,t_e,v>$, o intervalo $<t_s,t_e>$ é o *intervalo da tarefa*, isto é, o intervalo de tempo em que tarefa é escalonada e `v` identifica o nó em que a tarefa é escalonada. Sendo $pq_s=\{<p_1,q_1>,...,<p_n,q_n>\}$ e $pq_e=\{<p,q>\}$, p_i e q_i (com $i\geq1$ e $i\leq n$) são, respectivamente, os produtos de entrada da tarefa e as quantidades consumidas; `p` e `q` são, respectivamente, o produto de saída da tarefa e a quantidade entregue.

Uma tarefa de capacidade tem, relativamente ao nó em que é escalonada, consumo de *produtos de entrada* em determinadas quantidades no instante de início e entrega de um único *produto de saída*, em determinada quantidade no instante de fim. Os eventos de início e fim de uma tarefa correspondem a alterações no estado de capacidade do nó onde a tarefa é escalonada, já que a sua realização origina um decremento da capacidade disponível respectiva durante o intervalo de tempo da tarefa.

Formalmente, uma tarefa não elementar O é definida por um triplo:

$<e_s,e_e,\mathcal{O}>$

em que:

– e_s e e_e são os eventos extremos, de início e de fim, respectivamente, da tarefa mãe;

– $\mathcal{O}=\{O_1,...,O_k,...,O_m\}$, com $m>1$, é o conjunto das sub-tarefas de O. As sub-tarefas O_k são tarefas elementares.

Cada sub-tarefa O_k satisfaz as seguintes restrições:

– A tarefa mãe O é única para cada sub-tarefa O_k;

– Cada sub-tarefa O_k requer a capacidade de um nó de tipo idêntico (`S`, `P` ou `T`) ao que a tarefa mãe O requer;

– Cada sub-tarefa O_k tem os mesmos locais de entrada e saída que O;

– Cada sub-tarefa O_k tem os mesmos produtos de entrada e saída que O; para cada produto de entrada e para o produto de saída, o grupo de sub-tarefas O_k totaliza as quantidades correspondentes para a tarefa mãe, O; as proporções entre as quantidades de entrada e de saída são idênticas para todas as sub-tarefas e são iguais à mesma proporção para a tarefa mãe (restrições de continuidade internas do modelo[38]);

---

[37] Para nós acumuladores `cval` é um valor inteiro calculável pela operação `C-NEC`, dados o nó a que pertence a tarefa, o produto e a quantidade da tarefa, ver a secção 3.4.1.15. Para nós processadores `cval` é um parâmetro cujo valor deve ser escolhido no momento da criação da tarefa e que condiciona a sua duração; a duração é, então, calculável pela operação `D-NEC`, dados o nó a que pertence a tarefa, o produto, a quantidade e o valor de `cval` escolhido, ver a secção 3.4.1.16.

[38] Ver a secção 3.7.1.2.

– O intervalo $<t_s^k, t_e^k>$ de cada $\mathbb{O}_k$ está temporalmente contido no intervalo $<t_s, t_e>$ de $\mathbb{O}$ e é $\underset{k}{MIN}(t_s^k)=t_s$ e $\underset{k}{MAX}(t_e^k)=t_e$ (restrições temporais internas do modelo[39]). Para efeitos de afectação de capacidade e, portanto, escalonamento, os tempos t_s^k e t_e^k de cada uma das sub-tarefas são tomados como os *tempos* de início e *tempos* de fim efectivos da tarefa mãe $\mathbb{O}$.

A decomposição de tarefas é usada principalmente por razões de capacidade. Por exemplo, quando não há capacidade disponível suficiente para afectar a uma tarefa em períodos consecutivos (no número necessário), ou quando um nível baixo de trabalhos em curso é preferido, ou quando a tarefa deve ser coordenada com tarefas sucessoras e/ou antecessoras, também decompostas, de modo a permitir um menor tempo de fluxo da sequência de tarefas.[40]

3.7.1.1 Acumulação e Processamento

Conforme o tipo de capacidade de nó requerida, uma tarefa pode ser apelidada de:

- *Acumulação, tarefa de acumulação,* ou tarefa de tipo S - Requer capacidade de acumulação e é criada para ser escalonada num determinado nó acumulador com o mesmo produto de saída da tarefa. Os eventos de início e de fim diferem apenas no valor de tempo. A duração de uma tarefa de acumulação é imposta exogenamente, tipicamente por tarefas antecessoras (tarefas fornecedoras) e sucessoras (tarefas clientes).
- *Processamento, tarefa de processamento,* ou tarefa de tipo P ou T - Requer capacidade de processamento. A duração é imposta pelo processador executante, dada uma quantidade de produto a entregar. Uma tarefa de tipo P, ou *produção*, ou *tarefa de produção*, é criada para ser escalonada num nó produtor com o mesmo produto de saída e os mesmos produtos de entrada da tarefa. Neste caso, os eventos de início e de fim de uma tarefa de tipo P diferem nos valores de produto (havendo, em geral, mais de um produto de entrada), quantidade e tempo. Uma tarefa de tipo T, ou *transporte*, ou *tarefa de transporte*, é criada para ser escalonada num nó transportador com o mesmo produto de saída da tarefa. Neste caso, os eventos de início e de fim de uma tarefa de tipo T diferem apenas nos valores de tempo.

Uma tarefa de capacidade diz-se *escalonada* quando lhe foi afectada capacidade de um determinado nó de capacidade de tipo idêntico ao da tarefa, com o mesmo produto de saída e com os mesmos produtos de entrada (para aquele produto de saída) da tarefa. Uma tarefa de capacidade deve respeitar as restrições estruturais do modelo, *i.e.*, no caso de haver decomposição da tarefa, as restrições internas e, no contexto de um processo, as restrições externas.[41]

Esquematicamente, as tarefas de capacidade representam-se como barras rectangulares, como na Figura 3-14. No contexto da representação de um processo, ou de um fragmento de processo, as tarefas são representadas juntamente com setas indicando fluxos[42] de produtos de entrada e saída (adicionalmente podem também ser anotadas com quantidades e tempos).

---

[39] Ver a secção 3.7.1.3.

[40] Este aspecto é usualmente referido por *subdivisão de lote* (ou *lot-splitting*, ver, por exemplo, [Jacobs 1989] ou [Moily 1986]).

[41] Se houver decomposição de tarefas, apenas às tarefas elementares é efectivamente afectada capacidade. Ver as secções 3.7.1.2, 3.7.1.3, 3.8.1 e 3.8.2.

[42] Ver a secção 3.8.

Assumindo que se trata de tarefas elementares, os seguintes eventos poderiam ser empregues para descrever as tarefas da Figura 3-14:

<<{<p,q>},t_s,v>,<{<p,q>},t_e,v>,cval>

para as tarefas de tipo S e T, e

<<{<p_1,q_1>,...,<p_n,q_n>},t_s,v>,<{<p,q>},t_e,v>,cval>

para a tarefa de tipo P. [43]

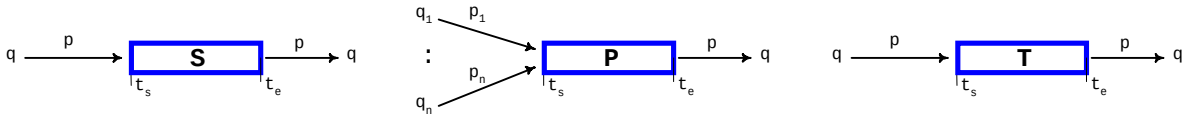


Figura 3-14- Exemplos de tarefas de tipo S, P e T.

3.7.1.2 Restrições de Continuidade Internas

Restrições de continuidade internas são restrições estruturais do modelo que exprimem a conservação das quantidades de produtos para *uma* tarefa de capacidade não elementar. Relacionam quantidades do mesmo produto, entregue ou consumido, pela tarefa mãe e pelas suas sub-tarefas.

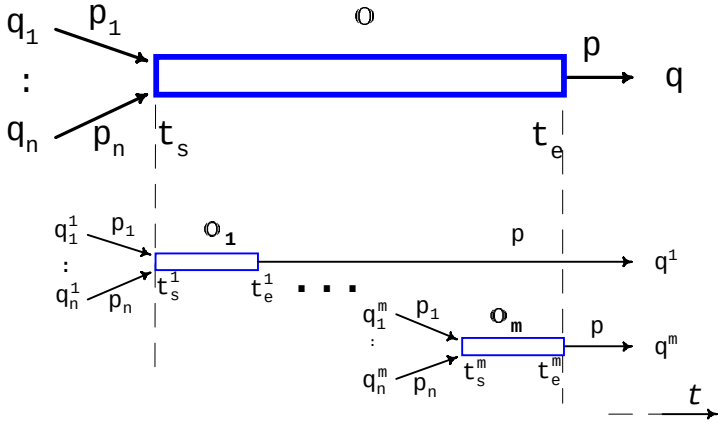


Figura 3-15- Decomposição de uma tarefa O em sub-tarefas $O_1, \ldots, O_m$ (m>1).

Seja O uma tarefa de capacidade com n produtos de entrada, $p_1, \ldots, p_n$ e com o produto de saída p,[44] decomposta em m tarefas elementares, $O_1, \ldots, O_m$ (m>1, ver Figura 3-15). Nestas circunstâncias, O é descrita por:

[43] v é o identificador do nó afectado à tarefa.

$<\mathcal{e}_s, \mathcal{e}_e, \{<\mathcal{e}_s^1, \mathcal{e}_e^1, cval^1>, \ldots, <\mathcal{e}_s^k, \mathcal{e}_e^k, cval^k>, \ldots, <\mathcal{e}_s^m, \mathcal{e}_e^m, cval^m>\}>$

em que:

$\mathcal{e}_s = <\{<p_1, q_1>, \ldots, <p_n, q_n>\}, t_s, v>$

$\mathcal{e}_e = <\{<p, q>\}, t_e, v>$

$\mathcal{e}_s^k = <\{<p_1, q_1^k>, \ldots, <p_n, q_n^k>\}, t_s^k, v> \quad (k=1, \ldots, m)$

$\mathcal{e}_e^k = <\{<p, q^k>\}, t_e^k, v> \quad (k=1, \ldots, m)$

As *restrições de continuidade internas* impõem que seja, para cada sub-tarefa $\mathcal{O}_k$:

$\sum_{k=1}^{m} q_i^k = q_i \quad (i=1, \ldots, n)$,

$\sum_{k=1}^{m} q^k = q$ e

$q_i^k / q^k = q_i / q \quad (k=1, \ldots, m$ e $i=1, \ldots, n)$.

3.7.1.3 Restrições Temporais Internas

Restrições temporais internas são restrições estruturais do modelo que constrangem os valores de tempo dos eventos de tarefas de capacidade e sub-tarefas correspondentes.

Seja $\mathcal{O}$ uma tarefa de capacidade com n produtos de entrada, $p_1, \ldots, p_n$ e com o produto de saída p, decomposta em m tarefas elementares, $\mathcal{O}_1, \ldots, \mathcal{O}_m$ ($m>1$, ver Figura 3-15), como descrito na secção 3.7.1.2.

As *restrições temporais internas* impõem que seja, para cada sub-tarefa $\mathcal{O}_k$:

$\underset{k=1,\ldots,m}{MIN}(t_s^k) = t_s$ e

$\underset{k=1,\ldots,m}{MAX}(t_e^k) = t_e$.

3.7.1.4 Operações com Tarefas de Capacidade

As operações primitivas para tarefas elementares de capacidade são:

$ES(<\mathcal{e}_s, \mathcal{e}_e, cval>) ::= \mathcal{e}_s.$

$EE(<\mathcal{e}_s, \mathcal{e}_e, cval>) ::= \mathcal{e}_e.$

$CVAL(<\mathcal{e}_s, \mathcal{e}_e, cval>) ::= cval.$

$TAREFA(\mathcal{e}_s, \mathcal{e}_e, cval) ::= <\mathcal{e}_s, \mathcal{e}_e, cval>.$

---

[44] Se a tarefa $\mathcal{O}$ é do tipo S ou T, será $n=1$ e $p_1=p$.

TAREFA?(<e_s,e_e,cval>) ::= VERDADEIRO.

Adicionalmente, são definidas as seguintes operações para tarefas elementares:

PRODUTO($\mathbb{O}$) ::= PRODUTO(EE($\mathbb{O}$)).

QUANTIDADE($\mathbb{O}$) ::= QUANTIDADE(EE($\mathbb{O}$)).

TE($\mathbb{O}$) ::= TE(EE($\mathbb{O}$)).

LOCAL($\mathbb{O}$) ::= LOCAL(EE($\mathbb{O}$)).

MATERIAIS($\mathbb{O}$) ::=
{ m_i : ($\exists_{\mathbb{p}\mathbb{q}_i}$: ($\mathbb{p}\mathbb{q}_i \in$ PQ-CONJ(ES($\mathbb{O}$)) $\wedge$ (PRODUTO($\mathbb{p}\mathbb{q}_i$)=m_i)) }.

QUANTIDADE($\mathbb{O}$,m_i) ::=
SE($m_i \in$MATERIAIS($\mathbb{O}$),QUANTIDADE(PAR-PQ(m_i,(PQ-CONJ(ES($\mathbb{O}$)))),0).

TS($\mathbb{O}$) ::= TS(ES($\mathbb{O}$)).

DURAÇÃO($\mathbb{O}$) ::= TE($\mathbb{O}$)-TS($\mathbb{O}$).

RAZÃO($\mathbb{O}$,m_i) ::= QUANTIDADE($\mathbb{O}$,m_i)/QUANTIDADE($\mathbb{O}$).

DURANTE?(t_k,$\mathbb{O}$) ::= ($t_k \geq$TS($\mathbb{O}$)) $\wedge$ (t_k<TE($\mathbb{O}$)).

A operação TAREFA é uma operação construtora adicional de tarefas de capacidade, apropriada à geração de uma tarefa elementar escalonável num dado nó de capacidade. Para nós acumuladores, dado um nó $\mathbb{V}$, um produto p tal que p$\in$PRODUTOS($\mathbb{V}$), uma quantidade q (positiva) e tempos de início e fim t_s e t_e, tais que $t_e > t_s$, esta operação constrói uma tarefa elementar de acumulação para o nó $\mathbb{V}$:

```
TAREFA(𝕍,p,q,tₛ,tₑ) ::=
TAREFA( EVENTO( { PAR-PQ(mᵢ,qᵢ) :
                  (mᵢ∈MATERIAIS-PARA-PRODUTO(𝕍,p)) ∧
                  (qᵢ=RAZÃO-MATERIAL-PRODUTO(𝕍,p,mᵢ)*q) },
                tₛ,
                LOCAL(𝕍)),
        EVENTO({ PAR-PQ(p,q) },tₑ,LOCAL(𝕍)),
        C-NEC(𝕍,p,q)).
```

Para nós processadores, dado um nó $\mathbb{V}$, um produto p tal que p$\in$PRODUTOS($\mathbb{V}$), uma quantidade q (positiva), o tempo de início t_s e a quantidade de capacidade de $\mathbb{V}$ a afectar à

tarefa em cada instante cval, a operação TAREFA constrói uma tarefa elementar de processamento para o nó $\mathbb{V}$:

```
TAREFA(𝕍,p,q,tₛ,cval) ::=
TAREFA( EVENTO( { PAR-PQ(mᵢ,qᵢ) :
                  (mᵢ∈MATERIAIS-PARA-PRODUTO(𝕍,p)) ∧
                  (qᵢ=RAZÃO-MATERIAL-PRODUTO(𝕍,p,mᵢ)*q) },
                tₛ,
                LOCAL(𝕍)),
        EVENTO( { PAR-PQ(p,q) },
                tₛ + D-NEC(𝕍,p,q,cval),
                LOCAL(𝕍)),
        cval).
```

As operações para tarefas não elementares de capacidade são:

```
ES(<𝕖ₛ,𝕖ₑ,𝒪>) ::= 𝕖ₛ.

EE(<𝕖ₛ,𝕖ₑ,𝒪>) ::= 𝕖ₑ.

SUB-TAREFAS(<𝕖ₛ,𝕖ₑ,𝒪>) ::= 𝒪.

TAREFA?(<𝕖ₛ,𝕖ₑ,𝒪>) ::= VERDADEIRO.
```

Todas as operações identificadas por PRODUTO, QUANTIDADE, TE, LOCAL, MATERIAIS, TS, DURAÇÃO e RAZÃO, que foram acima definidas para tarefas elementares são definidas, do mesmo modo, para tarefas não elementares. O predicado DURANTE? é definido do seguinte modo para tarefas não elementares ($\mathbb{O}$ denota uma tarefa não elementar):

```
DURANTE?(tₖ,𝕆) ::=
(∃𝕆ⱼ  : (𝕆ⱼ∈SUB-TAREFAS(𝕆)) ∧ (tₖ≥TS(𝕆ⱼ)) ∧ (tₖ<TE(𝕆ⱼ))).
```

Os predicados que permitem distinguir tarefas elementares de tarefas não elementares são:

```
TAREFA-ELEMENTAR?(<𝕖ₛ,𝕖ₑ,cval>) ::= VERDADEIRO.

TAREFA-ELEMENTAR?(<𝕖ₛ,𝕖ₑ,𝒪>) ::= FALSO.
```

A operação DECOMPONÍVEL? é um predicado que testa se uma tarefa dada $\mathbb{O}$, pode decompor-se nas sub-tarefas de um conjunto de sub-tarefas elementares dado $\mathcal{O}$:

```
DECOMPONÍVEL?(𝕆,𝒪) ::=
TAREFA-ELEMENTAR?(𝕆) ∧
(∀𝕆ₖ  (𝕆ₖ∈𝒪)  :
```

$$\begin{aligned}
&\text{TAREFA-ELEMENTAR?}(\mathcal{O}_k) \wedge \\
&(\text{LOCAL}(\mathcal{O})=\text{LOCAL}(\mathcal{O}_k)) \wedge (\text{PRODUTO}(\mathcal{O})=\text{PRODUTO}(\mathcal{O}_k)) \wedge \\
&(\text{MATERIAIS}(\mathcal{O})=\text{MATERIAIS}(\mathcal{O}_k)) \wedge \\
&(\forall_{m_i} : (m_i \in \text{MATERIAIS}(\mathcal{O})) \wedge \\
&\qquad (\text{RAZÃO}(\mathcal{O},m_i)=\text{RAZÃO}(\mathcal{O}_k,m_i)))) \wedge \\
&(\text{QUANTIDADE}(\mathcal{O})=\sum_{\mathcal{O}_k \in \mathcal{O}} \text{QUANTIDADE}(\mathcal{O}_k)) \wedge \\
&(\underset{\mathcal{O}_k \in \mathcal{O}}{\text{MAX}}(\text{TE}(\mathcal{O}_k))=\text{TE}(\mathcal{O})) \wedge (\underset{\mathcal{O}_k \in \mathcal{O}}{\text{MIN}}(\text{TS}(\mathcal{O}_k))=\text{TS}(\mathcal{O})).
\end{aligned}$$

A operação DECOMPOSIÇÃO produz uma tarefa não elementar com base numa tarefa elementar dada $\mathcal{O}$ e um conjunto de sub-tarefas tarefas elementares dado $\mathcal{O}$, tais que se verifica DECOMPONÍVEL?($\mathcal{O}$,$\mathcal{O}$). Esta operação é definida por:

$$\text{DECOMPOSIÇÃO}(\mathcal{O},\mathcal{O}) ::= <\text{ES}(\mathcal{O}),\text{EE}(\mathcal{O}),\mathcal{O}>.$$

3.7.2 Tarefa de Retalho e Tarefa de Matéria-Prima

As tarefas de retalho e as tarefas de matéria-prima não requerem capacidade; por isso se consideram tarefas fictícias. Tarefas de retalho representam o consumo de um produto final pelo exterior e estão associadas a eventos de pedido do exterior à rede física. Tarefas de matéria-prima representam a entrega de um produto pelo exterior e estão associadas a eventos de pedido da rede física ao exterior. Tarefas de retalho e tarefas de matéria-prima não são decomponíveis.

3.7.2.1 Tarefa de Retalho

As tarefas de retalho, para além de não requererem capacidade, não têm produtos de saída, não têm tarefas clientes no processo a que pertencem e têm apenas um produto de entrada. Num processo, uma tarefa de retalho funciona como uma actividade fictícia consumidora de toda a quantidade de um produto final da rede física.

Formalmente, uma tarefa $\mathcal{O}_y$ da classe *tarefa de retalho*, ou tarefa de tipo R, é descrita por um único evento de pedido do exterior à rede e_y, na forma:

$<\{<p_y,q_y>\},t_y,v_y>$

em que:

- v_y identifica o nó de retalho da rede física no qual o evento de pedido do exterior é estabelecido;
- p_y é o produto pedido à rede física pelo exterior (o produto de entrada da tarefa);
- q_y é a quantidade de p_y pedida (a quantidade consumida pela tarefa);
- t_y é o valor de tempo do evento de pedido (o tempo de início da tarefa).

Uma tarefa de retalho considera-se *escalonada* quando lhe foi afectado um nó de retalho com o mesmo produto de entrada da tarefa, estando a tarefa completamente instanciada (*i.e.*, com o

evento de pedido do exterior à rede estabelecido no nó). No contexto de um processo, a instanciação deve respeitar as restrições estruturais do modelo denominadas restrições externas.[45]

Tarefas de retalho são representadas por pequenas barras rectangulares, mais pequenas que as tarefas capacidade, a cinzento. Na Figura 3-16-a) representa-se esquematicamente uma tarefa de retalho. No contexto da representação de um processo, ou de um fragmento de processo, as tarefas de retalho são representadas juntamente com uma seta indicando o fluxo do produto de entrada (adicionalmente podem também ser anotadas com quantidades e tempos).

3.7.2.2 Tarefa de Matéria-Prima

As tarefas de matéria-prima, para além de não requererem capacidade, não têm produtos de entrada e não têm tarefas fornecedoras no processo a que pertencem. Num processo, uma tarefa de matéria-prima funciona como uma actividade fictícia que entrega um produto de entrada da rede física em que o processo é escalonado.

Figura 3-16- Tarefa de retalho, em a), e tarefa de matéria-prima, em b).

Formalmente, uma tarefa O_x da classe *tarefa de matéria-prima*, ou tarefa de tipo `M`, é descrita por um único evento de pedido da rede ao exterior e_x, na forma:

$<\{<p_x, q_x>\}, t_x, v_x>$

em que:

- v_x identifica o nó de matéria-prima da rede física no qual o evento de pedido ao exterior é estabelecido;
- p_x é o produto pedido pela rede física ao exterior (o produto de saída da tarefa);
- q_x é a quantidade de p_x pedida (a quantidade entregue pela tarefa);
- t_x é o valor de tempo do evento de pedido (o tempo de fim da tarefa).

Uma tarefa de matéria-prima considera-se *escalonada* quando lhe é afectado um nó de matéria-prima com o mesmo produto de saída da tarefa, estando a tarefa completamente instanciada. No contexto de um processo, a instanciação deve respeitar as restrições estruturais do modelo denominadas restrições externas.[45]

Tarefas de matéria-prima são representadas por pequenas barras rectangulares, mais pequenas que as tarefas capacidade. Na Figura 3-16-b) representa-se esquematicamente uma tarefa de matéria-prima. No contexto da representação de um processo, ou de um fragmento de processo, as tarefas de matéria-prima são representadas juntamente com uma seta indicando o

[45] Ver as secções 3.8.1 e 3.8.2.

fluxo do produto de saída (adicionalmente podem também ser anotadas com quantidades e tempos).

3.7.2.3 Operações com Tarefas de Retalho e Tarefas de Matéria-Prima

Como tarefas de retalho e tarefas de matéria-prima são representadas por eventos, definem-se primeiro duas operações conversoras entre tarefas e eventos:

```
EVENTO(<{<p,q>},t,v>) ::= <{<p,q>},t,v>.

TAREFA(<{<p,q>},t,v>) ::= <{<p,q>},t,v>.
```

As operações para tarefas de retalho e de matéria-prima são, então, definidas como se segue:

```
PRODUTO(𝕆) ::= PRODUTO(EVENTO(𝕆)).

QUANTIDADE(𝕆) ::= QUANTIDADE(EVENTO(𝕆)).

TEMPO(𝕆) ::= TEMPO(EVENTO(𝕆)).

LOCAL(𝕆) ::= LOCAL(EVENTO(𝕆)).
```

A operação TAREFA-EVENTO? é um predicado que permite distinguir uma tarefa de retalho ou de matéria-prima de uma tarefa de capacidade:

```
TAREFA-EVENTO?(<{<p,q>},t,v>) ::= VERDADEIRO.

TAREFA-EVENTO?(<𝕖s,𝕖e>) ::= FALSO.

TAREFA-EVENTO?(<𝕖s,𝕖e,𝒪>) ::= FALSO.
```

3.8 Fluxo

Um *fluxo* é um arco direccionado entre duas tarefas de um processo que representa uma troca de um produto e tem associados o produto, a quantidade do produto trocado e o instante de tempo da troca. A tarefa da qual parte o fluxo é apelidada *tarefa fornecedora* e a tarefa onde ele termina a *tarefa cliente*. A quantidade de produto associada ao fluxo é a que é entregue pela tarefa fornecedora e consumida pela tarefa cliente. Quanto ao tipo de tarefas ligadas por um fluxo, apenas pode haver um dos três casos seguintes:

a) As tarefas fornecedora e cliente são ambas tarefas de capacidade;

b) A tarefa fornecedora é uma tarefa de capacidade e a tarefa cliente é uma tarefa de retalho;

c) A tarefa fornecedora é uma tarefa de matéria-prima e a tarefa cliente é uma tarefa de capacidade.

Formalmente, um fluxo ligando uma tarefa $\mathbb{O}_x$ a uma tarefa $\mathbb{O}_y$, designado por fluxo $\mathbb{F}_{x,y}$, é descrito por um par:

$<\mathbb{O}_x, \mathbb{O}_y>$

sendo que:

– $\mathbb{O}_x$ é a tarefa fornecedora, posicionada no extremo montante do fluxo e $\mathbb{O}_y$ a tarefa cliente, posicionada no extremo jusante do fluxo, ambas elementares, ou ambas não elementares;

– O *produto associado ao fluxo* p_e^x é o produto de saída da tarefa fornecedora $\mathbb{O}_x$ e é consumido pela tarefa cliente $\mathbb{O}_y$ (é um produto de entrada desta última);

– O fluxo respeita as *restrições de continuidade externas* e as *restrições temporais externas* do modelo. Estas são restrições estruturais do modelo, que impõem valores particulares aos eventos de tarefas ligadas por fluxos e que são descritas mais adiante;[45]

– O fluxo corresponde a um único arco de rede existente entre os dois nós afectados às tarefas $\mathbb{O}_x$ e $\mathbb{O}_y$, e o produto associado ao fluxo p_e^x é um elemento do conjunto de produtos desse arco. Isto quer dizer que, se $\mathbb{RF}$, $\mathbb{V}_x$ e $\mathbb{V}_y$ forem a rede física, o nó afectado à tarefa $\mathbb{O}_x$ e o nó afectado à tarefa $\mathbb{O}_y$, respectivamente, se verifica $\mathbb{V}_y \in \texttt{NÓS-CLIENTES}(\mathbb{RF}, \mathbb{V}_x, p_e^x)$.

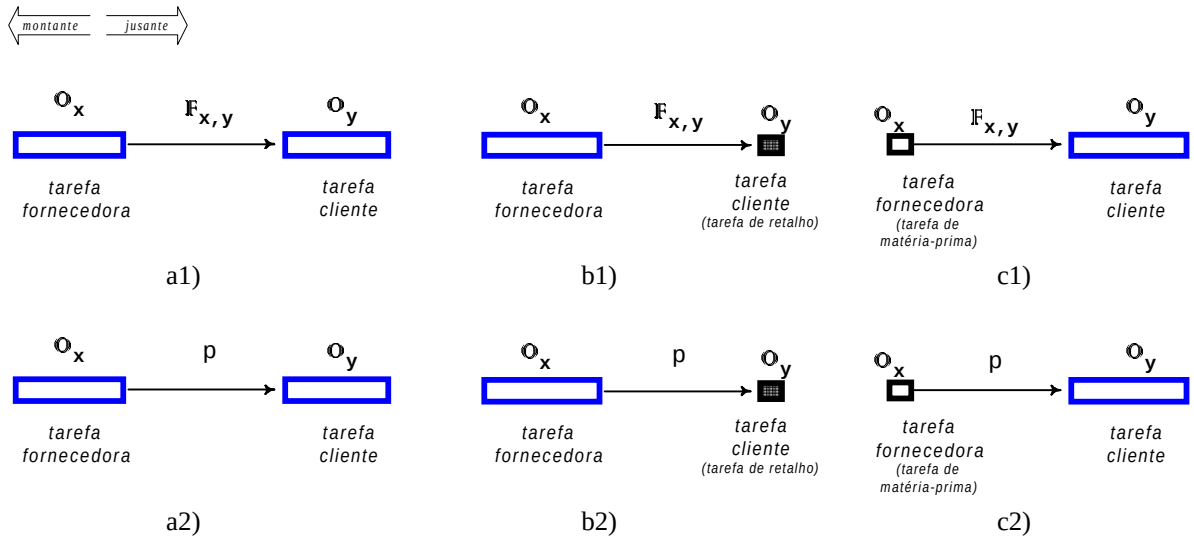


Figura 3-17- Fluxos.

Fluxos são esquematicamente representados por setas ligando o par fornecedor-cliente de tarefas. Na Figura 3-17 representam-se esquematicamente os três tipos de casos acima mencionados. Em vez de etiquetar graficamente cada seta de fluxo com um símbolo denotando o fluxo, como é feito na Figura 3-17- a1), b1) e c1) pode-se fazê-lo indicando o produto associado ao fluxo, como é feito na Figura 3-17- a2), b2) e c2).

Note-se que os fluxos que ligam as tarefas podem não corresponder exactamente às relações de precedência temporal que poderão sugerir. Se fosse esse o caso, a tarefa cliente deveria sempre iniciar-se após a tarefa fornecedora terminar. Ora isto pode não acontecer se as tarefas

forem decompostas e se estiverem coordenadas, por exemplo, de modo a tirar partido da subdivisão de lote (*lot-splitting*), com a decomposição da tarefa em sub-tarefas, para redução do tempo de fluxo total de ambas. Neste caso, a tarefa mãe fornecedora pode terminar após a tarefa mãe cliente se iniciar.

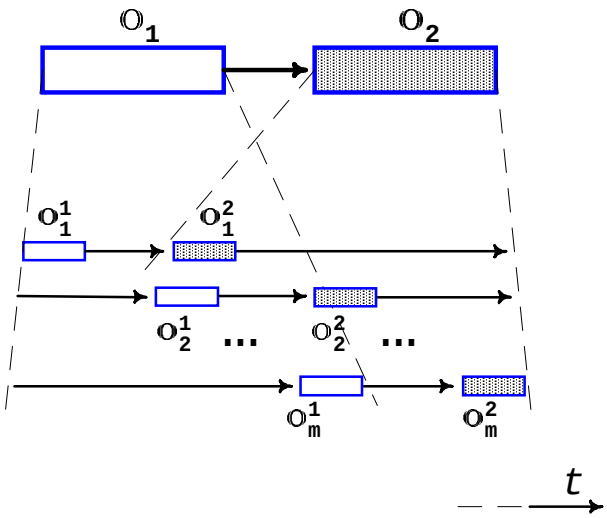


Figura 3-18- Fluxos entre tarefas e sub-tarefas.

Veja-se o exemplo da Figura 3-18, onde as tarefas O_1 e O_2 são decompostas em tarefas elementares O_1^1, O_2^1,..., O_m^1, e O_1^2, O_2^2,..., O_m^2), permitindo o início da primeira sub-tarefa de O_2 (O_1^2) antes de a tarefa O_1 terminar. Apenas no caso de um fluxo entre tarefas elementares o fluxo corresponderá uma relação de precedência entre o evento de fim da tarefa fornecedora e o evento de início da tarefa cliente.

3.8.1 Restrições de Continuidade Externas

Estas restrições exprimem a conservação das quantidades de produtos e dizem respeito aos fluxos entre pares de tarefas *fornecedor-cliente*. As restrições de continuidade externas impõem entrega e consumo do mesmo produto nas mesmas quantidades em ambos os extremos dos fluxos. Existe uma restrição de continuidade externa para cada produto trocado entre cada tarefa de um par de tarefas fornecedor-cliente. Este tipo de restrições podem ser descritas analisando os fluxos à entrada (ver Figura 3-19) e à saída (ver Figura 3-20) de uma tarefa genérica O.

Para cada produto de entrada p_i, de uma tarefa O,[46] sejam $O_1, \ldots, O_m$ todas as m tarefas fornecedoras da tarefa O para p_i (ver Figura 3-19).[47] Sejam também $q_i^1, \ldots, q_i^m$ as

[46] Se O é uma tarefa do tipo S, T ou R, p_i será o único produto de entrada de O.

[47] Não se impõe aqui que uma tarefa tenha, para cada produto de entrada, uma única tarefa fornecedora, que fornece toda a quantidade necessária daquele produto, à excepção do caso de uma tarefa de retalho (que terá

quantidades fornecidas de p_i por $O_1,\ldots,O_m$ a O e q_i a quantidade consumida por O. Então, tem de ser $q_i^1+\ldots+q_i^m=q_i$. Isto é, sendo os eventos de fim das tarefas O_k ($k=1,\ldots,m$) e o evento de início de O da forma:

$<\{<p_i,q_i^k>\},t_e^k,v_k>$ (para as tarefas O_k)

$<\{\ldots,<p_i,q_i>,\ldots\},t_s,v>$ (para a tarefa O)

as *restrições de continuidade externas* impõem que seja:

$$\sum_{k=1}^{m} q_i^k = q_i$$

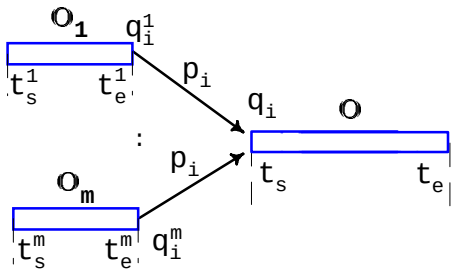


Figura 3-19- Restrições externas à entrada (com tarefas fornecedoras) de uma tarefa O.

Para o produto de saída p, de uma tarefa O, sejam $O_1,\ldots,O_r$ todas as r tarefas clientes da tarefa O para p (ver Figura 3-20).[48] Sejam também $q^1,\ldots,q^r$ as quantidades de p fornecidas por O a $O_1,\ldots,O_r$ e q a quantidade fornecida por O. Então, tem de ser $q^1+\ldots+q^r=q$. Isto é, sendo os eventos de início das tarefas O_j ($j=1,\ldots,r$) e o evento de fim de O da forma:

$<\{\ldots,<p,q^j>,\ldots\},t_s^j,v_j>$ (para as tarefas O_j)[49]

$<\{<p,q>\},t_e,v>$ (para a tarefa O)

as *restrições de continuidade externas* impõem que seja:

---

uma única tarefa fornecedora). Em modelos particulares, no entanto, esta restrição poderá ser imposta através da imposição de uma determinada configuração de rede.

[48] Não se impõe aqui que uma tarefa tenha uma única tarefa cliente que consome toda a quantidade entregue daquele produto, à excepção do caso de uma tarefa de matéria-prima (que terá uma única tarefa cliente). O caso de uma tarefa poder ter várias tarefas clientes para o mesmo produto de saída poderá acontecer, por exemplo, em processos em que o mesmo produto intermédio é utilizado como material, ou componente, de outros (mais de um) produto intermédios diferentes. Também em estruturas de rede divergentes (frequentes na parte de distribuição da rede de produção distribuição) isto acontece. Em modelos particulares, no entanto, aquela restrição poderá ser imposta através da imposição de uma determinada configuração de rede, ou através do modo como é construído um processo.

[49] Se O_j é uma tarefa do tipo S ou T, o par $<p,q^j>$ será o único elemento do conjunto de pares produto-quantidade do evento de início de O_j.

$\sum_{j=1}^{r} q^j = q$.

3.8.2 Restrições Temporais Externas

Estas restrições constrangem os valores de tempo dos eventos das tarefas, entre pares de tarefas *fornecedor-cliente*. Este tipo de restrições podem ser descritas, como se segue, para uma tarefa genérica $\mathbb{O}$.

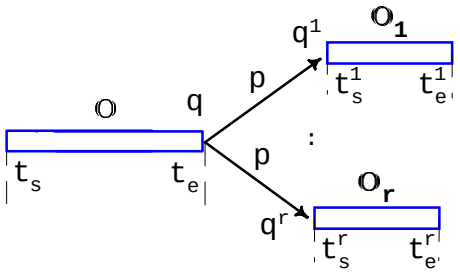


Figura 3-20- Restrições externas à saída (com tarefas clientes) de uma tarefa $\mathbb{O}$.

Sejam t_s e t_e os tempos de início e de fim de uma tarefa $\mathbb{O}$, respectivamente, e $t_s^1, \ldots, t_s^m$ e $t_e^1, \ldots, t_e^m$ os tempos de início e de fim, respectivamente, de cada uma das m tarefas fornecedoras $\mathbb{O}_1, \ldots, \mathbb{O}_m$, da tarefa $\mathbb{O}$ (ver Figura 3-19).

Então, se todas as tarefas são elementares, as *restrições temporais externas* impõem que seja:

$t_e^1 = t_e^2, \ldots, t_e^{m-1} = t_e^m$ e

$t_s = t_e^k$ (para qualquer k, $k=1, \ldots, m$)

Se todas as tarefas são não elementares, as *restrições temporais externas* impõem que seja:

$t_e^1 = t_e^2, \ldots, t_e^{m-1} = t_e^m$ e

$t_s > t_s^k$, $t_s \leq t_e^k$ e $t_e > t_e^k$ (para qualquer k, $k=1, \ldots, m$).

Sejam $t_s^1, \ldots, t_s^r$ e $t_e^1, \ldots, t_e^r$ os tempos de início e de fim, respectivamente, de cada uma das r tarefas clientes $\mathbb{O}_1, \ldots, \mathbb{O}_r$, da tarefa $\mathbb{O}$ (ver Figura 3-20).

Então, se todas as tarefas são elementares, as *restrições temporais externas* impõem que seja:

$t_s^1 = t_s^2, \ldots, t_s^{r-1} = t_s^r$ e

$t_e = t_s^j$ (para qualquer j, $j=1, \ldots, r$)

Se todas as tarefas são não elementares, as *restrições temporais externas* impõem que seja:

$t_s^1 = t_s^2, \ldots, t_s^{r-1} = t_s^r$ e

$t_s < t_s^j$, $t_e \geq t_s^j$ e $t_e < t_e^j$ (para qualquer j, $j=1, \ldots, r$).

3.9 Processo

Um *processo de rede* ou, simplesmente, *processo*, é um conjunto de tarefas ligadas por fluxos formando um grafo acíclico e conexo tal que, as tarefas de retalho do processo entregam um único produto final para o exterior de uma rede. Num mesmo processo, produtos são acumulados e processados por tarefas (de acumulação e de processamento) sendo transferidos de umas tarefas para outras no sentido dos fluxos, *i.e.*, no sentido de montante para jusante, para obter um único produto final.

Formalmente, um processo $\mathbb{RT}$ é descrito por um par:

$<\mathcal{O}, \mathcal{F}>$

sendo:

– $\mathcal{O}=\{\mathbb{O}_1, \ldots, \mathbb{O}_k, \ldots, \mathbb{O}_K\}$ o conjunto de todas as tarefas do processo contendo, pelo menos, uma tarefa de capacidade, uma tarefa de retalho e uma tarefa de matéria-prima;

– $\mathcal{F}=\{\ldots, \mathbb{F}_{x,y}, \ldots\}$ o conjunto de todos os fluxos entre as tarefas do processo contendo, por cada par de tarefas, no máximo um fluxo;

– $\mathcal{O}$ e $\mathcal{F}$ devem formar um grafo acíclico e conexo (e quando completamente instanciados devem satisfazer as restrições estruturais do modelo).

Um processo diz-se *escalonado* quando todas as tarefas do processo estão escalonadas e estão completamente instanciadas. Estas instanciações devem respeitar as restrições estruturais do modelo, designadamente as restrições internas e as restrições externas.

Numa rede física poderá existir, para um determinado horizonte temporal, um conjunto determinado de processos escalonados, $\mathcal{RT}=\{\mathbb{RT}_1, \ldots, \mathbb{RT}_j, \ldots, \mathbb{RT}_J\}$. Cada tarefa $\mathbb{O}_k^j$ (tarefa k do processo $\mathbb{RT}_j$) do conjunto de todas as tarefas dos processos (o conjunto $\bigcup_{j=1}^{J} \bigcup_{k=1}^{K} \{\mathbb{O}_k^j\}$) está escalonada num único nó da rede.

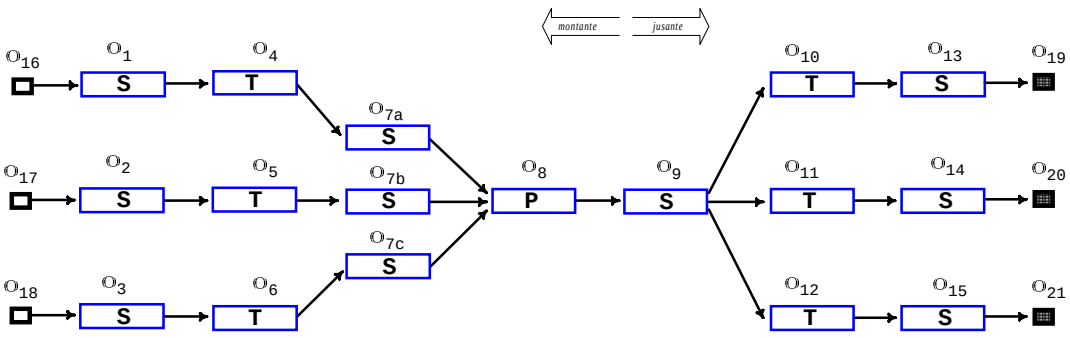


Figura 3-21- Um processo de rede.

Uma representação esquemática semelhante à que é usada para representar redes é usada para representar processos. Aqui, as tarefas são nós do grafo do processo e representam-se por barras; os fluxos de produtos entre as tarefas são representados por arcos direccionados entre as tarefas. A título de exemplo representa-se na Figura 3-21 um processo. Este processo é

escalonável na rede da Figura 3-13 (as tarefas estão indexadas de acordo com a numeração dos índices dos nós desta última figura; em particular, as tarefas $\mathbb{O}_{7a}$, $\mathbb{O}_{7b}$ e $\mathbb{O}_{7c}$ são escalonadas no nó v_7).

3.10 Conclusão

Neste capítulo descreveu-se a parte do modelo de escalonamento multi-agente apelidada de *nível físico*. As classes de objectos deste nível do modelo como *evento*, *nó*, *arco*, *rede*, *tarefa*, foram definidas em termos dos seus atributos e das operações necessárias para manipulação dos objectos de cada classe.

O *nível virtual*, exposto no Capítulo 4, complementa o nível físico do modelo e pressupõe as classes aqui definidas.

3.11 Lista Resumida de Símbolos Usados no Capítulo 3

Resumem-se na seguinte lista os símbolos usados no presente capítulo.

	Dimensões
$\mathbb{P}$	Conjunto de produtos
p, m	Identificador de produto (elemento de $\mathbb{P}$)
q	quantidade
t	Instante de tempo
$\mathbb{v}$	Conjunto de identificadores de nós (locais físicos)
v	Identificador de nó (local físico, elemento de $\mathbb{v}$)
	Eventos
$\mathbb{e}$	Conjunto de eventos
$\mathbb{e}$	Evento
$\mathbb{p}\mathbb{q}$	Conjunto de pares produto-quantidade
$\mathbb{p}\mathbb{q}$	Par produto-quantidade de um evento
	Tarefas, Fluxos, Processos
$\mathbb{O}$	Conjunto de tarefas
$\mathbb{O}$	Tarefa
TN	Tipo de tarefa (S, P, T, R ou M)
$\mathbb{F}$	Conjunto de fluxos
$\mathbb{F}$	Fluxo (elemento de $\mathbb{F}$)
$\mathbb{RT}$	Conjunto de processos de rede
$\mathbb{RT}$	Processo de rede
	Nós, Arcos, Rede Física
C	Designação genérica de capacidade (pode ser A ou W)
$C_{max}(t)$	Perfil de capacidade máxima
c(t)	Perfil de capacidade utilizada
$\mathbb{c}$(t)	Conjunto de perfis de capacidade utilizada
C(t)	Perfil de capacidade disponível
$A_{max}(t)$	Perfil de capacidade de acumulação máxima
a(t)	Perfil de capacidade de acumulação utilizada

$\mathcal{a}$(t)	Conjunto de perfis de capacidade de acumulação utilizada
A(t)	Perfil de capacidade de acumulação disponível
W_{max}(t)	Perfil de capacidade de processamento máxima
$\mathbb{w}$(t)	Perfil de capacidade de processamento utilizada
$\mathcal{w}$(t)	Conjunto de perfis de capacidade de processamento utilizada
W(t)	Perfil de capacidade de processamento disponível
ed(t)	Perfil de eventos de pedido
$\mathcal{ed}$(t)	Conjunto de perfis de eventos de pedido
TN	Tipo de nó (S, P, T, R ou M)
$\mathbb{p}$	Conjunto de produtos de saída de um nó
$\mathcal{m}$	Conjunto de produtos de entrada de um nó
$\mathcal{b}$	(Conjunto de quantidades associadas à) Lista de materiais de um nó
$\mathcal{r}$	(Conjunto de) Factores de conversão de capacidade
r	Factor de conversão de capacidade
$\mathcal{V}$	Conjunto de nós
$\mathbb{V}$	Nó
$\mathcal{E}$	Conjunto de arcos
$\mathbb{E}$	Arco
$\mathcal{pm}$	Conjunto de produtos associados a um arco
$\mathbb{RF}$	Rede física (rede do nível físico)
Nn	Número total de nós de uma rede física

4 Um Modelo para Escalonamento Multi-Agente — Nível Virtual

Neste capítulo descreve-se o *nível virtual* do modelo de escalonamento multi-agente. O *nível físico* foi descrito no capítulo anterior. No presente capítulo descrevem-se os agentes e a interacção entre agentes necessária à coordenação da actividade de escalonamento multi-agente.

4.1 Introdução

Na Tabela 4-1 identificam-se as classes de objectos principais definidas para o nível virtual do modelo, com as subclasses respectivas.

Tabela 4-1- Classes de objectos do nível virtual e sublcasses.

<table>
<tr><th>Classe</th><th colspan="4">Subclasses</th></tr>
<tr><td>dimensão</td><td colspan="4">local virtual</td></tr>
<tr><td>pedido</td><td colspan="4"></td></tr>
<tr><td>problema de escalonamento</td><td colspan="4"></td></tr>
<tr><td rowspan="5">agente</td><td rowspan="4">agente de rede</td><td rowspan="3">agente gestor</td><td colspan="2">agente acumulador</td></tr>
<tr><td>agente processador</td><td>agente produtor</td></tr>
<tr><td></td><td>agente transportador</td></tr>
<tr><td colspan="3">agente de retalho</td></tr>
<tr><td colspan="3">agente de matéria-prima</td></tr>
<tr><td colspan="4">agente supervisor</td></tr>
<tr><td>rede de EE</td><td colspan="4"></td></tr>
</table>

Estas classes de objectos são descritas nas secções que se seguem. No Apêndice A expõe-se um conjunto de diagramas de classes para o modelo desenvolvido que inclui as classes descritas no presente capítulo.

Para a descrição das classes de objectos do modelo no presente capítulo adoptaram-se as convenções anteriormente descritas no Capítulo 3 (na secção 3.1.1).

4.2 Dimensão Virtual

As dimensões são estendidas de modo a acomodar uma dimensão nova, denominada dimensão *local virtual*. A dimensão local virtual é definida por um conjunto:

$\mathcal{G}^r = \{g_0, g_1, \ldots, g_i, \ldots, g_{Nn}\}$

em que os g_i são locais virtuais, *i.e.*, identificadores de agentes (símbolos), identificando g_0 o *agente supervisor* e os restantes g_i (com $i \geq 1$ e $i \leq Nn$), os *agentes de rede* (ver as secções seguintes), num total de $Nn+1$ agentes.

4.3 Pedido

Um *pedido* é a informação elementar trocada entre agentes que interagem numa rede de Empresa Estendida (EE). Dependendo do contexto onde são trocados, os pedidos podem ser designados *pedidos locais* e *pedidos globais*. Pedidos locais são os pedidos trocados entre pares de agentes fornecedor-cliente; pedidos globais são pedidos do exterior à rede ou da rede ao exterior. Os pedidos são denotados pelo símbolo $\mathbb{d}$ com índices apropriados para distinguir diferentes pedidos, como se descreverá na presente secção.

Um *pedido* $\mathbb{d}$, destinado a um agente identificado por g_x, de uma determinada quantidade q de um produto p, a satisfazer num instante t (a data limite do pedido), é definido através de um evento simples[1] cujos valores associados de produto, quantidade, tempo e local são, respectivamente, p, q, t e g_x. Um pedido qualquer tem, portanto, a forma de um evento simples, em que o valor de local é interpretado como um identificador de agente (local virtual). O significado do pedido $<\{<p,q>\},t,g_x>$ é: "encomenda ao agente g_x de quantidade q de produto p para a data limite t".

4.3.1 Utilização dos Pedidos

Para escalonamento multi-agente, no ambiente rede de EE, a interacção entre agentes baseia-se na comunicação de pedidos. Os *agentes de retalho* e *agentes de matéria-prima* processam pedidos globais, que trocam com o *agente supervisor*; os *agentes gestores*[2] processam pedidos locais, que trocam entre si e com os agentes de retalho e agentes de matéria-prima.

Os agentes de retalho processam pedidos globais *do exterior à rede*. Um pedido deste tipo significa uma encomenda do exterior de determinada quantidade de um determinado produto, a satisfazer num dado instante pela rede. Os agentes de matéria-prima processam pedidos globais *da rede ao exterior*. Um pedido deste tipo significa uma encomenda da rede ao exterior de determinada quantidade de um determinado produto, a ser satisfeita num dado instante. Um agente gestor recebe pedidos locais de agentes clientes e envia pedidos locais a fornecedores. Para satisfazer um pedido local recebido, um agente gestor cria internamente uma tarefa escalonável no nó de rede física associado ao agente, para tornar disponível a quantidade de produto do pedido no instante de tempo do mesmo pedido. Os produtos de entrada necessários à tarefa são encomendados pelo agente aos agentes fornecedores através de pedidos locais, um pedido por cada produto de entrada. Os pedidos podem também ser usados para representar pedidos de re-escalonamento de um pedido anteriormente acordado entre um par fornecedor-cliente de agentes.

---

[1] Ver o Capítulo 3.

[2] Um *agente de retalho* é um agente que tem associado um nó da rede física que é um nó de retalho; um *agente de matéria-prima* é um agente que tem associado um nó da rede física que é um nó de matéria-prima; um *agente gestor* (ou *agente de capacidade*) é um agente que tem associado um nó da rede física que é um nó de capacidade. As classes *agente gestor*, *agente de retalho* e *agente de matéria-prima* são sublcasses da classe *agentes de rede*. As classes *agente de rede* e *agente supervisor* são subclasses da classe mais geral *agente*.

4.3.2 Operações com Pedidos

Como os pedidos são eventos simples, as operações aplicáveis a eventos simples PRODUTO, QUANTIDADE, TEMPO e LOCAL são,[3] também, aplicáveis a pedidos. Especificamente para pedidos, define-se a seguinte operação construtora (p, q, t e g denotam respectivamente um produto, uma quantidade, um valor de tempo e um identificador de agente):

PEDIDO(p,q,t,g) ::= EVENTO({PAR-PQ(p,q)},t,g).

4.3.3 Pedidos Complexos

Uma forma alternativa de modelar pedidos é a que considera uma classe geral *pedido* com subclasses *pedido simples* e *pedido complexo*. A classe *pedido simples* é idêntica à definida anteriormente e a classe *pedido complexo* é definida por um par:

<OP,$\mathcal{d}$>

em que OP∈{E,OU} e $\mathcal{d}$ é um conjunto de pedidos simples $\mathcal{d}=\{d_1,\ldots,d_k,\ldots,d_n\}$ em que os d_k apenas diferem nos valores de quantidade e tempo. Os operadores E e OU têm os significados de uma *conjunção de pedidos simples* e uma *disjunção exclusiva de pedidos simples*, respectivamente:

– <E,{<{<p,q_1>},t_1,g>,...,<{<p,q_k>},t_k,g>,...,<{<p,q_n>},t_n,g>}> significa "encomenda ao agente g do produto p nas a quantidades $q_1,\ldots,q_k,\ldots,q_n$ para as datas limite $t_1,\ldots,t_k,\ldots,t_n$, respectivamente";

– <OU,{<{<p,q_1>},t_1,g>,...,<{<p,q_k>},t_k,g>,...,<{<p,q_n>},t_n,g>}> significa "encomenda ao agente g do produto p na quantidade q_1 para a data limite t_1,... ou na quantidade q_k para a data limite t_k,... ou na quantidade q_n para a data limite t_n".

As operações necessárias para pedidos complexos são:

PEDIDO-SIMPLES?(d) ::= EVENTO-SIMPLES?(d).

PEDIDO-COMPLEXO?(<OP,$\mathcal{d}$>) ::=
(OP∈{E,OU}) ∧ ($\forall_{d_k}$ ($d_k\in\mathcal{d}$) : EVENTO-SIMPLES?(d_k)) ∧
($\forall_{d_i,d_j}$ ($d_i\in\mathcal{d}, d_j\in\mathcal{d}, d_i\neq d_j$) :
    (PRODUTO(d_i)=PRODUTO(d_j)) ∧ (LOCAL(d_i)=LOCAL(d_j))).

OP(<OP,$\mathcal{d}$>) ::= OP.

E-CONJ(<OP,$\mathcal{d}$>) ::= $\mathcal{d}$.

A operação construtora de pedido complexo é:

---

[3] O local associado a um pedido é um local virtual (*i.e.*, um identificador de um agente de rede) e, por isso, a operação LOCAL, quando aplicada a um pedido, produz o local virtual associado ao pedido.

PEDIDO-COMPLEXO(OP,d) ::= <OP,d>.

Pedidos complexos com o operador E permitem representar grupos de pedidos associados. Por exemplo, podem representar encomendas que advêm das necessidades de um produto de entrada para sub-tarefas de uma mesma tarefa, representar um pedido de re-escalonamento para conversão de um pedido simples anterior em pedidos de quantidades menores distribuídos ao longo do tempo (possivelmente devido ao re-escalonamento de uma tarefa por decomposição em sub-tarefas) ou, de um modo geral, pedidos que se repetem, de forma cíclica ou não, durante um certo intervalo de tempo.

Pedidos complexos com o operador OU permitem representar escolhas possíveis alternativas de uma encomenda do mesmo produto. Um caso particular de uso do operador OU em pedidos complexos é o que especifica escolhas possíveis alternativas para datas limite de uma mesma encomenda. Por exemplo:

<OU,{<{<p,q>},t_1,g>,...,<{<p,q>},t_k,g>,..., <{<p,q>},t_n,g>}>

significa "encomenda da quantidade q do produto p ao agente g, para a data limite t_1,... ou para a data limite t_k,... ou para a data limite t_n".

O uso de pedidos complexos é relegado para trabalho futuro.

4.3.4 Considerações Prévias e Pressupostos

Esta secção clarifica certos pormenores do modelo directa ou indirectamente relacionados com pedidos que, apesar de serem descritos em detalhe mais adiante neste trabalho, carecem de uma introdução prévia para compreensão das próximas secções. Para reduzir a complexidade do modelo alguns pressupostos simplificativos foram assumidos (enumerados no final da presente secção).

Uma rede de EE é composta por uma rede física — o nível físico do modelo — e uma rede de agentes — o nível virtual do modelo — implicitamente representada por um conjunto de *agentes de rede* e um *agente supervisor* único. Cada agente de rede, está associado a, ou gere, um único um nó da rede física e cada nó de rede física está associado a, ou é gerido por, um único agente de rede. Um agente de rede pode ser um *agente de retalho*, um *agente de matéria-prima* ou um *agente gestor* (*agente de capacidade*). Agentes de retalho e agentes de matéria-prima estão associados a nós de retalho e a nós de matéria-prima da rede física, respectivamente. Agentes gestores estão associados a nós de capacidade e gerem a capacidade desses nós.

As relações de cliente-fornecedor entre nós da rede física estendem-se aos agentes associados. Os agentes de uma rede de EE interagem comunicando informação relevante na forma de mensagens com conteúdo predefinido. A comunicação entre agentes ocorre entre pares de agentes. Entre um par de agentes comunicantes, uma mensagem ou introduz um pedido novo ou é a continuação de uma *conversação* sobre um pedido anteriormente introduzido. Neste último caso, a mensagem diz respeito a uma evolução do estado de interacção do par de agentes, no que respeita ao pedido. Uma vez introduzido, um pedido pode ser aceite ou rejeitado e uma vez aceite pode ser satisfeito, cancelado ou pode sofrer um re-escalonamento (requerido por um dos agentes, podendo ser ou não aceite pelo outro), existindo mensagens apropriadas para comunicar estes estados de interacção entre os agentes. Os pares possíveis de agentes comunicantes são:

- dois agentes de rede ligados por uma relação cliente-fornecedor;
- o agente supervisor e um agente de retalho;
- o agente supervisor e um agente de matéria-prima.

Agentes de retalho processam pedidos globais do exterior à rede. Estes pedidos são convertidos por agentes de retalho em pedidos locais e enviados a agentes fornecedores. Agentes de matéria-prima processam pedidos globais da rede ao exterior. Estes pedidos resultam da conversão de pedidos locais recebidos de agentes clientes pelos agentes de matéria-prima.

Problemas de escalonamento são colocados à rede de EE na forma de pedidos globais do exterior à rede, cada um associado a um horizonte temporal de escalonamento (horizonte temporal global). Este horizonte é um intervalo de tempo que inclui o instante de tempo (valor de tempo, ou data limite) do pedido e durante o qual todos os eventos (eventos que representam pedidos globais e locais e eventos de início e fim de tarefas) necessários à satisfação do pedido global do exterior à rede devem ocorrer. O pedido global pode ser re-escalonado, mas sempre dentro do horizonte temporal de escalonamento, *i.e.*, a data limite do pedido pode ser alterada mas não para após o tempo de fim, nem para antes do tempo de início do horizonte. Um pedido global do exterior à rede é introduzido pelo agente supervisor que o comunica ao agente de retalho a que é destinado. Este converte-o num pedido local que transmite ao agente gestor fornecedor apropriado. Os agentes gestores propagam para montante na rede (*i.e.*, para agentes fornecedores) pedidos para os fornecimentos necessários. Por fim, agentes de matéria-prima recebem pedidos locais de agentes gestores clientes, que convertem em pedidos globais da rede ao exterior e comunicam ao agente supervisor. Entre um par de agentes a troca de um novo pedido introduz uma nova *conversação* entre os agentes, uma sequência de trocas de mensagens acerca do pedido, com uma estrutura predefinida por um protocolo, que pode terminar com o pedido rejeitado, cancelado ou satisfeito. Esta actividade de interacção entre os agentes de uma rede de EE é a *actividade inter-agente*, ou *actividade comunicacional* e ocorre de forma distribuída (na dimensão tempo e na dimensão local) sempre que há uma nova encomenda do exterior.

Cada agente mantém um *estado interno* através de uma *actividade intra-agente* ditada por regras de comportamento. Este estado interno tem tanto a ver com o estado das conversações mantidas com outros agentes, o *estado comunicacional*, como com a resolução do problema de escalonamento que, localmente, cada pedido recebido pelo agente coloca e com a manutenção de um estado do nó associado ao agente, o *estado de capacidade*. Um pedido novo recebido por um agente notifica-o de que há um novo problema de escalonamento. Para cada problema de escalonamento que lhe é colocado, o agente mantém uma representação interna que corresponde a uma *perspectiva local* do problema de escalonamento colocado à rede. Para os agentes gestores esta representação interna é local ao agente e é apelidada de *problema local de escalonamento*. O agente deve criar e escalonar (no nó associado) as tarefas que são necessárias para satisfazer os pedidos correspondentes a cada problema local de escalonamento e enviar os pedidos dos fornecimentos necessários a essas tarefas. O problema de escalonamento colocado à rede como um todo é apelidado de *problema global de escalonamento*. O *agente supervisor* tem uma perspectiva particular dos problemas de escalonamento colocados à rede. Embora esta perspectiva possa ser apelidada de *perspectiva global* ela corresponde a uma representação interna do problema global de escalonamento que é incompleta, pois o agente supervisor apenas representa internamente os pedidos globais do exterior à rede e da rede ao exterior e o horizonte temporal de escalonamento, não conhecendo o que se passa entre os agentes de rede e dentro de cada agente de rede. A perspectiva dos agentes de retalho e de matéria-prima é, também, incompleta.

As actividades individuais dos agentes da rede de EE, inter-agente e intra-agente, contribuem para a actividade *global de escalonamento multi-agente* da rede de EE.

Os seguintes pressupostos são assumidos, no presente trabalho:

1. Para cada agente processador (cada agente com um nó processador associado), para além do nó processador associado ao agente, assume-se a existência de dois *nós acumuladores implícitos* (não representados explicitamente na rede física), colocados imediatamente a jusante e a montante do nó processador. Estes nós implícitos são nós fictícios com capacidade máxima de acumulação infinita (*i.e.*, com $A_{max}(t)=+\infty$ para qualquer t) e tais que, o nó acumulador a jusante consome os produtos de saída do nó processador e o nó acumulador a montante fornece todos os produtos de entrada do nó processador. Esta suposição permite que pares de agentes fornecedor-cliente possam escalonar tarefas de um mesmo processo de rede com folga temporal positiva entre a terminação da tarefa fornecedora e o início da tarefa cliente.[4]
2. Cada agente gestor cria, para satisfazer cada pedido recebido de um agente cliente, uma única tarefa e cada tarefa serve para satisfazer um único pedido de um agente cliente. Cada tarefa deve ser escalonada pelo agente afectando-lhe uma quantidade de capacidade constante. Isto garante, para as tarefas de processamento, que cada tarefa tenha uma duração fixa, a partir do momento em que é criada e é uma pré-condição para que o mecanismo para percepção de restrições temporais rígidas funcione correctamente. Para cada produto de entrada de cada tarefa criada o agente envia um único pedido a um agente fornecedor e cada pedido a um fornecedor serve para satisfazer todas as necessidades de um produto de entrada da tarefa.[5]
3. A rede física e a estrutura dos produtos finais da rede são tais que, os produtos de entrada correspondentes de cada produto de saída de qualquer nó têm conjuntos de produtos componentes (directos ou indirectos) disjuntos e nós fornecedores (directos ou indirectos) diferentes (ver exemplos de casos excluídos por este pressuposto na Figura 4-1-a1) e Figura 4-1-a2)).[6] Isto tem como consequência que, cada novo pedido à rede origina, para cada agente gestor de um nó com produtos de saída que são produtos intermédios do produto final pedido, um único pedido local novo recebido de um agente cliente. De acordo com

---

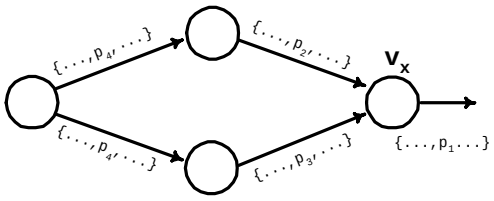
[4] Isto significa que se relaxam as restrições temporais externas descritas no Capítulo 3, de modo a acomodar as referidas folgas temporais positivas.

[5] O escalonamento envolvendo uma perspectiva de médio ou longo prazo (*i.e.*, sob a influência de previsão e planeamento) não é tratado no presente trabalho. Isto significa, por exemplo, que a quantidade de cada produto existente num dado instante em cada nó acumulador é sempre a necessária e suficiente para satisfazer os pedidos que foram realizados e ainda não satisfeitos. Está também a excluir-se a possibilidade de agregação (*batching*) e decomposição (*splitting*) de tarefas e de uma tarefa ter várias tarefas clientes. O tratamento de todos estes casos é relegado para trabalho futuro.

[6] *I.e.*, se $\mathbb{RF}$ é a rede física, p é um qualquer produto de saída de um qualquer nó $\mathbb{V}$, de $\mathbb{RF}$ e p_i e p_j são dois produtos de entrada desse nó tais que $p_i \neq p_j$, $p_i \in$ MATERIAIS-PARA-PRODUTO($\mathbb{V}$,p) e $p_j \in$ MATERIAIS-PARA-PRODUTO($\mathbb{V}$,p), se $\mathbb{V}_i$ e $\mathbb{V}_j$ são dois nós da rede tais que $\mathbb{V}_i \in$ NÓS-FORNECEDORES($\mathbb{RF}$,$\mathbb{V}$,p_i) e $\mathbb{V}_j \in$ NÓS-FORNECEDORES($\mathbb{RF}$,$\mathbb{V}$,p_j) então é: |COMPONENTES-DE-PRODUTO($\mathbb{RF}$,$\mathbb{V}_i$,p_i) $\cap$ COMPONENTES-DE-PRODUTO($\mathbb{RF}$,$\mathbb{V}_j$,p_j)|=0 (os componentes, directos ou indirectos de p, são diferentes); também, para quaisquer dois nós da rede $\mathbb{RF}$, $\mathbb{V}_i$ e $\mathbb{V}_j$, se é A-MONTANTE?(NÓS($\mathbb{RF}$),ARCOS($\mathbb{RF}$),$\mathbb{V}_i$,$\mathbb{V}$,p) e A-MONTANTE?(NÓS($\mathbb{RF}$), ARCOS($\mathbb{RF}$),$\mathbb{V}_j$,$\mathbb{V}$,p), então é: $\mathbb{V}_i, \neq \mathbb{V}_j$ (os fornecedores, directos ou indirectos, do nó $\mathbb{V}$ para o produto p, são diferentes).

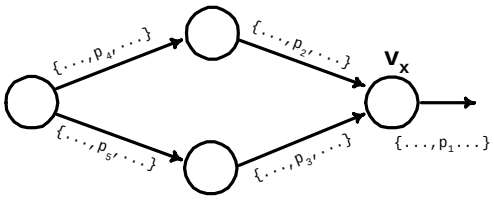
este pressuposto, por cada pedido global do exterior à rede um agente gestor (ou um agente de matéria-prima) particular receberá, ou nenhum pedido local, ou um único pedido local.

4. No ambiente de escalonamento multi-agente uma instância de modelo de rede de EE é única, apenas existindo uma única rede física com uma configuração fixa (relações de cliente-fornecedor preestabelecidas e capacidade máxima dos nós físicos fixa) e um conjunto de agentes que constituem, respectivamente, o nível físico e o nível virtual da instância do modelo. Assume-se que nada mais existe no ambiente (o exterior não é representado explicitamente). Isto traduz-se em: a) a acção do exterior manifesta-se na rede de EE através de pedidos globais do exterior à rede estabelecidos pelo *agente supervisor* e por ele transmitidos aos agentes de retalho; b) toda a capacidade dos nós geridos pelos agentes gestores é utilizável no escalonamento das tarefas necessárias à satisfação de pedidos dos seus agentes clientes na rede.[7]



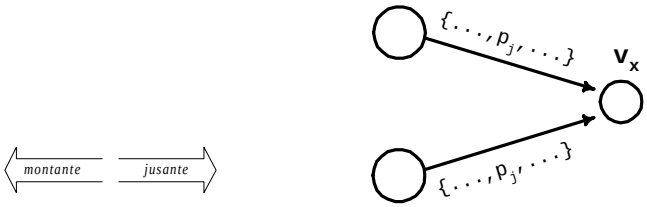
componentes do produto p_1: { p_2, p_3, p_4 }

a1) Componente p_4 repetido na estrutura do produto p_1.



componentes do produto p_1: { p_2, p_3, p_4, p_5 }

a2) p_4 e p_5 são componentes diferentes de p_1 fornecidos pelo mesmo nó.



b) Nó com mais de um fornecedor para o mesmo produto de entrada.

Figura 4-1- a1) e a2) são exemplos de casos de redes físicas excluídas pelo pressuposto 3 (componentes repetidos e fornecedores repetidos para um produto, respectivamente). b) é um exemplo de caso excluído pelo pressuposto 5 (fornecedores diferentes para um mesmo produto de entrada).

5. Cada nó de capacidade e cada nó de retalho da rede física têm, para cada produto de entrada, um único nó fornecedor (exemplifica-se um caso excluído por este pressuposto na Figura 4-1-b)).[8] Portanto, cada agente gestor e cada agente de retalho têm, para cada produto de entrada, um único agente fornecedor. Não há qualquer incerteza e necessidade

[7] Isto significa que o perfil de capacidade máxima de cada nó gerido por cada agente não sofre alterações durante a resolução de um problema de escalonamento pelo facto de os agentes da rede de EE realizarem tarefas para outras redes (simplesmente porque não realizam tarefas para outras redes).

[8] *I.e.*, se $\mathbb{RF}$ é a rede física, tem-se que `|NÓS-FORNECEDORES(`$\mathbb{RF}$`,`$\mathbb{V}$`,p)|=1`, para qualquer nó físico de capacidade ou de retalho $\mathbb{V}$, tal que $\mathbb{V} \in$ `NÓS(`$\mathbb{RF}$`)` e `TIPO(`$\mathbb{V}$`)`$\neq$`M` e com p$\in$ `MATERIAIS(`$\mathbb{V}$`)`.

de tomar decisão sobre a que fornecedor um agente deve enviar um pedido de um produto necessário, porque o agente só tem um único fornecedor para satisfazer o pedido.[9]

6. Assume-se que qualquer pedido global do exterior à rede poderá ser re-escalonado mas apenas por iniciativa da rede, apenas dentro dos limites temporais do horizonte temporal de escalonamento e que estes limites não são alterados.[10]

4.3.5 Escalonamento e Comunicação entre Agentes

A resolução de um problema de escalonamento multi-agente é levada a cabo de forma distribuída sendo necessário haver comunicação entre os diferentes agentes de rede. Como se disse na secção anterior, parte da actividade de cada agente é uma *actividade inter-agente*, ou *comunicacional*, que consiste na interacção de um agente com os outros agentes; a outra parte consiste numa *actividade intra-agente*, que tem a ver com a gestão interna da interacção e com o escalonamento das tarefas do agente.

Nesta secção descreve-se como os pedidos são usados na actividade comunicacional, necessária para o escalonamento multi-agente numa rede de EE e como a actividade comunicacional se relaciona com a actividade intra-agente. Descreve-se a seguir a notação e os parâmetros usados para escalonamento, nomeadamente no que respeita ao escalonamento numa perspectiva temporal, *i.e.*, que tem em conta as restrições temporais.

Na Tabela 4-2 e na Tabela 4-3 expõe-se a notação usada para os pedidos de rede de EE. Esta notação toma como referencial a rede e os pedidos globais do exterior à rede, permitindo distinguir pedidos locais trocados entre agentes, originados por diferentes pedidos globais do exterior à rede. Os índices `k`, `r`, `m` e `0` (zero) são usados de forma consistente para referir, respectivamente, agentes gestores, agentes de retalho, agentes de matéria-prima e o agente supervisor. Portanto, de um modo geral, g_k, g_r e g_m identificam, respectivamente, um agente gestor, um agente de retalho e um agente de matéria-prima e g_0 identifica o agente supervisor da rede de EE. Índices `i` e `j` podem ser usados ocasionalmente para designar qualquer agente de rede exceptuando o agente supervisor. `Nk`, `Nr` e `Nm` representam o número total de agentes gestores, de agentes de retalho e de agentes de matéria-prima de uma rede, respectivamente. O número total de agentes de rede é igual ao número total de nós da rede física, `Nn` (`Nn=Nk+Nm+Nr`); incluindo o supervisor o total de agentes é `Nn+1` (`Nn+1=`$|g^r|$. A notação empregue para pedidos é:

$$d^{\substack{\textit{agente-de-origem, agente-de-destino}\\ \textit{pedido-global, agente-de-retalho}}}$$

Em que *`agente-de-origem`*, *`agente-de-destino`*, *`agente-de-retalho`* e *`pedido-global`* são índices que indicam, respectivamente, o agente emissor do pedido, o agente receptor do pedido, o agente de retalho que processa o pedido global do exterior à rede que originou o pedido em questão e aquele pedido global. O último índice descrito permite diferenciar diferentes pedidos globais do exterior à rede recebidos pelo mesmo agente de retalho.

---

[9] Aspectos relacionados com escolhas de fornecedores alternativos para um mesmo fornecimento incluindo, possivelmente, negociação baseada em datas limite, ou preços, são relegados para trabalho futuro.

[10] Alterações em pedidos globais por iniciativa do exterior (incluindo a alteração do horizonte temporal de escalonamento) e negociação de atrasos na satisfação de um pedido global (re-escalonamento de pedidos globais para fora do horizonte temporal de escalonamento) com o exterior são relegados para trabalho futuro.

Tabela 4-2- Notação para pedidos globais.

	Notação	Descrição
pedidos globais do exterior à rede	$\mathbb{d}^{0,r}_{i_r,r}$	o pedido global do exterior à rede i_r do agente de retalho g_r (o índice i_r distingue pedidos diferentes de g_r).
pedidos globais da rede ao exterior	$\mathbb{d}^{m,0}_{i_r,r}$	pedido global da rede ao exterior do agente de matéria-prima g_m, originado pelo pedido global do exterior à rede $\mathbb{d}^{0,r}_{i_r,r}$;
	$\mathcal{d}'^{0}_{i_r,r} = \bigcup_{m=1}^{Nm^r_{i_r}} \{\mathbb{d}^{m,0}_{i_r,r}\}$	conjunto de todos os pedidos globais da rede ao exterior de agentes de matéria-prima, cada um originado pelo pedido global do exterior à rede $\mathbb{d}^{0,r}_{i_r,r}$ ($\lvert \mathcal{d}'^{0}_{i_r,r} \rvert = Nm_{i_r,r}$).

Tabela 4-3- Notação para pedidos locais.

	Notação	Descrição
contexto inter-agente, em geral	$\mathbb{d}^{i,j}_{i_r,r}$	pedido local enviado pelo cliente g_i e recebido pelo fornecedor g_j (ambos agentes de rede), originado por $\mathbb{d}^{0,r}_{i_r,r}$.
contexto intra-agente	$\mathbb{d}^{i,k}_{i_r,r}$	pedido local recebido pelo agente gestor g_k do cliente g_i, originado por $\mathbb{d}^{0,r}_{i_r,r}$;
	$\mathbb{d}^{k,j}_{i_r,r}$	pedido local enviado pelo agente gestor g_k ao fornecedor g_j, originado por $\mathbb{d}^{0,r}_{i_r,r}$.
do agente gestor g_k	$\mathcal{d}'^{k,}_{i_r,r} = \bigcup_{j=1}^{Nf^k_{i_r,r}} \{\mathbb{d}^{k,j}_{i_r,r}\}$	conjunto de todos os pedido locais enviados pelo agente gestor g_k a fornecedores (agente gestores ou agentes de matéria-prima) necessários à satisfação do pedido local $\mathbb{d}^{i,k}_{i_r,r}$ ($\lvert \mathcal{d}'^{k,}_{i_r,r} \rvert = Nf^{k}_{i_r,r}$).

Tabela 4-4- Notação para tarefas e processos de rede.

	Notação	Descrição
tarefas	$\mathbb{O}^{k}_{i_r,r}$	tarefa de capacidade do agente gestor g_k originada por, e para satisfazer um pedido local $\mathbb{d}^{i,k}_{i_r,r}$, de um cliente g_i;
	$\mathbb{O}^{r}_{i_r,r}$	tarefa de retalho do agente de retalho g_r originada pelo, e para satisfazer o pedido global $\mathbb{d}^{0,r}_{i_r,r}$;
	$\mathbb{O}^{m}_{i_r,r}$	tarefa de matéria-prima do agente de matéria-prima g_m, para satisfazer o pedido local $\mathbb{d}^{i,m}_{i_r,r}$.[11]
processos	$\mathbb{RT}_{i_r,r}$	processo de rede que satisfaz o pedido global $\mathbb{d}^{0,r}_{i_r,r}$.

[11] Tarefas de retalho e tarefas de matéria-prima são tarefas fictícias de um processo de rede que têm duração nula e são definidas através de um evento (ver o Capítulo 3) que, por sua vez, representa um pedido recebido pelo agente. Para um agente de retalho, a tarefa de retalho $\mathbb{O}^{r}_{i_r,r}$ e o pedido $\mathbb{d}^{0,r}_{i_r,r}$ são idênticos; para um agente de matéria-prima, a tarefa de matéria-prima $\mathbb{O}^{m}_{i_r,r}$ e o pedido $\mathbb{d}^{i,m}_{i_r,r}$ que o agente g_m recebe de um agente cliente g_i são idênticos (ver mais adiante).

Na Tabela 4-2 está descrita a notação para pedidos globais. $Nm_{i_r,r}$ é igual ao número de pedidos globais da rede ao exterior originados pelo pedido global do exterior à rede $\mathbb{d}^{0,r}_{i_r,r}$; $Nm_{i_r,r}$ é, também, igual ao número de agentes de matéria-prima envolvidos na satisfação do pedido global do exterior à rede $\mathbb{d}^{0,r}_{i_r,r}$ ($Nm_{i_r,r}$ serve, basicamente, para permitir enumerar todos aqueles pedidos globais da rede ao exterior, ou todos aqueles agentes de matéria-prima).

Na Tabela 4-3 está descrita a notação para pedidos locais. Se o agente gestor g_k está envolvido na satisfação do pedido global $\mathbb{d}^{0,r}_{i_r,r}$ então ele será receptor de um pedido local $\mathbb{d}^{i,k}_{i_r,r}$, de um seu cliente g_i. Os pedidos para os aprovisionamentos necessários dos produtos de entrada para a tarefa que g_k deve escalonar para a satisfação de $\mathbb{d}^{i,k}_{i_r,r}$ envolvem um número de agentes fornecedores de g_k igual a $Nf^{k}_{i_r,r}$; $Nf^{k}_{i_r,r}$ é, também, igual ao número de pedidos locais a enviar a fornecedores por cada pedido local $\mathbb{d}^{i,k}_{i_r,r}$ recebido de um cliente ($Nf^{k}_{i_r,r}$ serve, basicamente, para permitir enumerar todos aqueles agentes fornecedores, ou todos aqueles pedidos). Note-se que, para qualquer agente $Nf^{k}_{i_r,r} \geq 1$; em particular, para agentes acumuladores ou agentes transportadores $Nf^{k}_{i_r,r}=1$, pois nestes casos existirá apenas um único pedido $\mathbb{d}^{k,j}_{i_r,r}$ a um único fornecedor por cada pedido $\mathbb{d}^{i,k}_{i_r,r}$ recebido de um cliente.

Na Tabela 4-4 expõe-se, adicionalmente, a notação usada para tarefas e processos de rede.[12] Tarefas e processos são denotados pelos símbolos $\mathbb{O}$ e $\mathbb{RT}$, respectivamente. As notações empregues para tarefas e para processos são, respectivamente:

$$\mathbb{O}^{\,agente}_{pedido\text{-}global,\ agente\text{-}de\text{-}retalho} \qquad \text{e} \qquad \mathbb{RT}_{pedido\text{-}global,\ agente\text{-}de\text{-}retalho}$$

Em que os índices *agente*, *agente-de-retalho* e *pedido-global* indicam, respectivamente, o agente de rede que escalona a tarefa, o agente de retalho que processa o pedido global do exterior à rede que originou a existência da tarefa ou do processo e aquele pedido global daquele agente de retalho.

Note-se que, dados os pressupostos antes descritos, em particular 2, 3 e 4, cada novo pedido global origina, para cada agente gestor envolvido no processo que o pedido desencadeia, um único pedido local recebido. Se g_k é um agente gestor envolvido no processo, verifica-se:[13]

- $PRODUTO(\mathbb{d}^{i,k}_{i_r,r})=PRODUTO(\mathbb{O}^{k}_{i_r,r})$ e $QUANTIDADE(\mathbb{d}^{i,k}_{i_r,r})=QUANTIDADE(\mathbb{O}^{k}_{i_r,r})$;
- $Nf^{k}_{i_r,r}=|PQ\text{-}CONJ(ES(\mathbb{O}^{k}_{i_r,r}))|$ e há uma correspondência biunívoca entre cada par produto-quantidade, $\mathbb{pq}^{k,j}_{i_r,r}$ tal que $\mathbb{pq}^{k,j}_{i_r,r} \in PQ\text{-}CONJ(ES(\mathbb{O}^{k}_{i_r,r}))$ e cada pedido $\mathbb{d}^{k,j}_{i_r,r}$, sendo: $PRODUTO(\mathbb{pq}^{k,j}_{i_r,r})=PRODUTO(\mathbb{d}^{k,j}_{i_r,r})$ e $QUANTIDADE(\mathbb{pq}^{k,j}_{i_r,r})=QUANTIDADE(\mathbb{d}^{k,j}_{i_r,r})$.

Para exemplificar a notação introduzida nesta secção tome-se, como exemplo, a rede física da Figura 4-2-a). Esta rede tem treze nós de capacidade, identificados por v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12} e v_{13}, três nós de retalho, identificados por v_{14}, v_{15}, e v_{16} e quatro nós de matéria-prima, identificados por v_{17}, v_{18}, v_{19} e v_{20}. A rede tem os produtos de saída

[12] Ver o Capítulo 3.
[13] Para que sejam respeitadas as restrições de continuidade, ver o Capítulo 3.

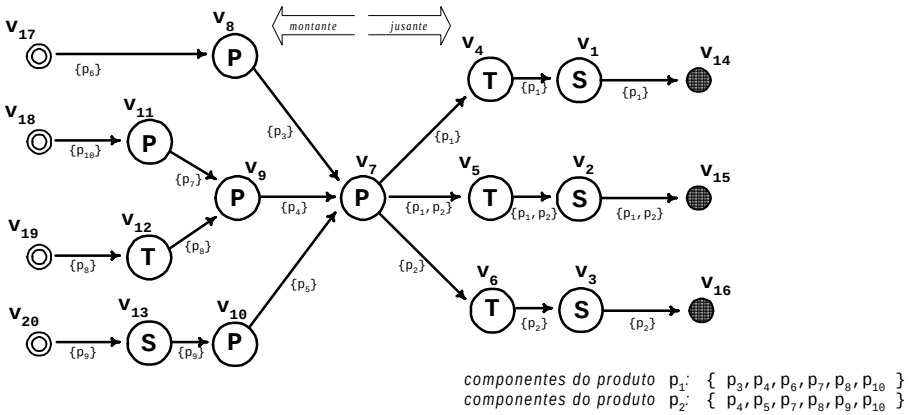
p_1, nos nós de retalho v_{14} e v_{15} e p_2, nos nós de retalho v_{15} e v_{16}. O produto p_1 é produzido no nó v_7 a partir dos componentes p_3 e p_4 e o produto p_2 é produzido no mesmo nó a partir de p_4 e p_5. Isto pode verificar-se por inspecção das etiquetas associadas aos arcos da rede (os conjuntos de produtos associados a cada arco), observando que p_3, p_4 e p_5 são os produtos que v_7 consome, que p_3 e p_4 estão incluídos no conjunto de componentes de p_1 (e que p_5 não está) e que p_4 e p_5 estão incluídos no conjunto de componentes de p_2 (e que p_3 não está), de acordo com o que é indicado em texto na figura. Cada agente de rede está associado a um nó portanto, o nível virtual contém treze agentes gestores ($Nk=13$), identificados por g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_9, g_{10}, g_{11}, g_{12} e g_{13}, três agentes de retalho ($Nr=3$), identificados por g_{14}, g_{15} e g_{16} e quatro agentes de matéria-prima ($Nm=4$), identificados por g_{17}, g_{18}, g_{19} e g_{20}, para além do agente supervisor, g_0. Na Figura 4-2-b) representam-se os agentes (cada nó v_i está associado ao agente g_i), juntamente com pedidos trocados entre os agentes no decurso da resolução do problema que a seguir se descreve. Os pedidos são representados por símbolos de pedido etiquetando arcos direccionados entre pares de agentes que, por sua vez, representam comunicação entre agentes.

Suponha-se que há três pedidos globais do exterior à rede $\mathbb{d}_{1,14}^{0,14}$, $\mathbb{d}_{1,15}^{0,15}$ e $\mathbb{d}_{2,15}^{0,15}$ de produtos p_1, p_2 e p_2, respectivamente (ver Figura 4-2-b)). De acordo com a notação, $\mathbb{d}_{1,14}^{0,14}$ é o pedido global 1 do agente de retalho g_{14}, $\mathbb{d}_{1,15}^{0,15}$ é o pedido global 1 do agente de retalho g_{15} e $\mathbb{d}_{2,15}^{0,15}$ é o pedido global 2 do agente de retalho g_{15}. A Figura 4-2-b) mostra os pedidos trocados entre os agentes que seriam originados por aqueles pedidos globais do exterior à rede. Na Figura 4-2-c) representam-se os três processos desencadeados para a satisfação destes três pedidos, tarefas fictícias incluídas (*i.e.*, as tarefas de retalho, dos agentes de retalho e as tarefas de matéria-prima, dos agentes de matéria-prima).

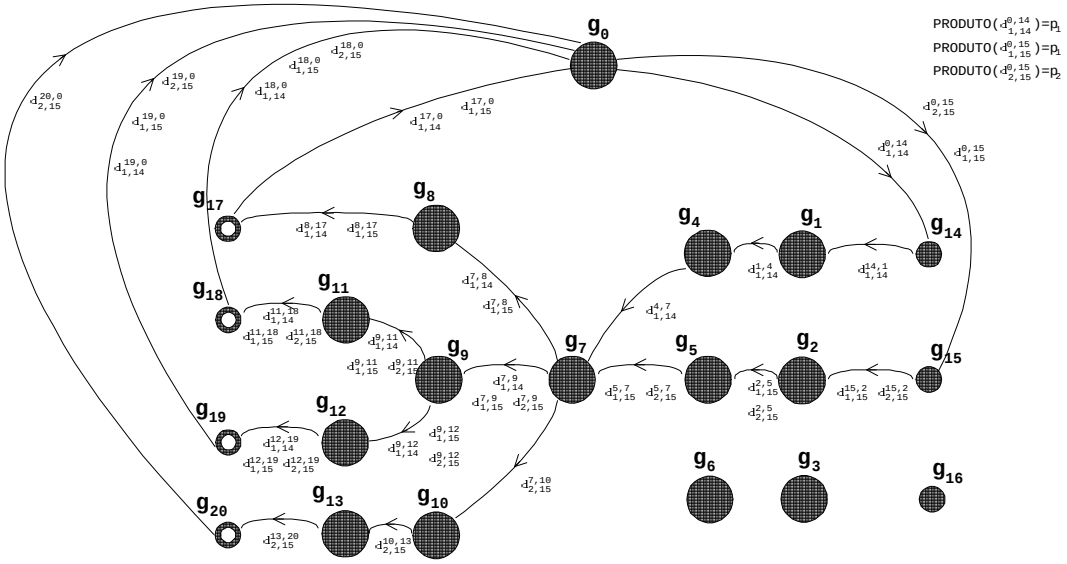
Para tratamento de um problema de escalonamento numa perspectiva temporal é necessário definir certos parâmetros. Estes dizem respeito a perspectivas temporais de problemas de escalonamento originados por um pedido global do exterior à rede genérico $\mathbb{d}_{i_r,r}^{0,r}$ e são descritos a seguir.

Para parâmetros de perspectiva local e internos relativamente a um agente usam-se símbolos com letras minúsculas; para parâmetros de uma perspectiva local e externos relativamente a um agente ou parâmetros da perspectiva global usam-se letras maiúsculas. A notação para estes parâmetros é exposta na Tabela 4-5 e na Tabela 4-6 (parâmetros de agentes gestores), na Tabela 4-7 (parâmetros de agentes de retalho), na Tabela 4-8 (parâmetros de agentes de matéria-prima) e na Tabela 4-9 (parâmetros do agente supervisor).

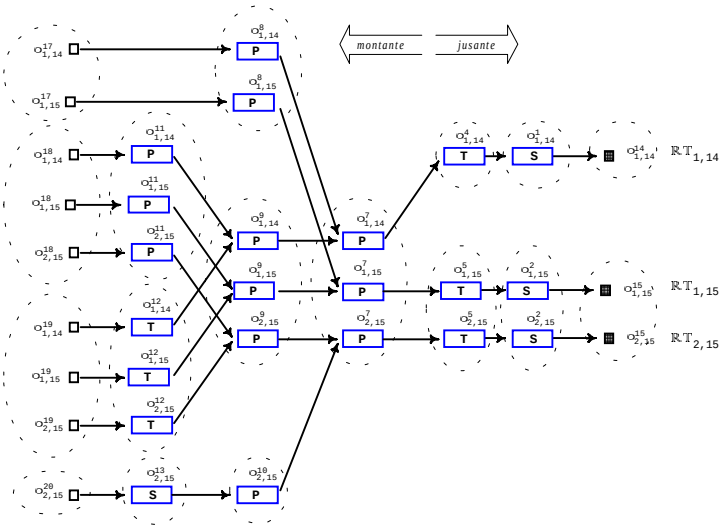
Para ilustração dos parâmetros usam-se figuras que representam graficamente escalonamentos de agentes individuais com uma única tarefa. Cada escalonamento é representado através de uma linha temporal junto à qual se mostram a tarefa do agente e os pedidos a jusante (do cliente) e a montante (aos fornecedores). Os intervalos de horizonte temporal e de horizonte temporal local com cada fornecedor (a descrever mais adiante) são também mostrados, a cor vermelha e a cor verde, respectivamente. O intervalo de uma tarefa é representado por um rectângulo, a cor azul, limitado por círculos indicando os eventos de fim e de início da tarefa. Pedidos são igualmente representados por círculos. Folgas temporais são representadas por setas (orientadas no sentido positivo da linha temporal se são positivas). Todos estes elementos são etiquetados por símbolos identificadores.



a) Nível físico.



b) Pedidos no nível virtual.



c) Processos $\mathbb{RT}_{1,14}$, $\mathbb{RT}_{1,15}$ e $\mathbb{RT}_{2,15}$ (e respectivas tarefas), para satisfação dos pedidos globais à rede.

Figura 4-2- Exemplo de interacção entre agentes, em b), numa rede de EE com a rede física, em a), com pedidos globais do exterior à rede $\mathbb{d}^{0,14}_{1,14}$, $\mathbb{d}^{0,15}_{1,15}$ e $\mathbb{d}^{0,15}_{2,15}$. Os processos desencadeados são representados em c).

A Figura 4-3 ilustra os parâmetros para escalonamento na perspectiva temporal para um agente gestor g_k, para os casos de agente processador (produtor ou transportador), na Figura 4-3-a) e de agente acumulador, na Figura 4-3-b). $d^{i,k}_{i_r,r}$ é o pedido local recebido por g_k, do cliente g_i e originado pelo pedido global à rede $d^{0,r}_{i_r,r}$; $O^{k}_{i_r,r}$ é a tarefa de g_k que satisfaz o pedido $d^{i,k}_{i_r,r}$. Os pedidos $d^{k,j}_{i_r,r}$ são pedidos aos fornecedores g_j de produtos de entrada para $O^{k}_{i_r,r}$ (um único pedido no caso de g_k ser um agente transportador ou acumulador). Representam-se também nas figuras os eventos de início e de fim, $es^{k}_{i_r,r}$ e $ee^{k}_{i_r,r}$, da tarefa $O^{k}_{i_r,r}$.

Tabela 4-5- Parâmetros temporais de escalonamento (intervalos e datas) para um agente gestor g_k e uma tarefa $O^{k}_{i_r,r}$.

	Notação	Descrição
Intervalos	$h^{k,j}_{i_r,r} = <rd^{k,j}_{i_r,r}, dd^{k}_{i_r,r}>$	horizonte temporal local com o fornecedor g_j;
	$h^{k}_{i_r,r} = <rd^{k}_{i_r,r}, dd^{k}_{i_r,r}>$	horizonte temporal local.
	$H^{k,j}_{i_r,r} = <RD^{k,j}_{i_r,r}, DD^{k}_{i_r,r}>$	horizonte temporal com o fornecedor g_j;
	$H^{k}_{i_r,r} = <RD^{k}_{i_r,r}, DD^{k}_{i_r,r}>$	horizonte temporal.
Datas	$dd^{k}_{i_r,r} = TEMPO(d^{i,k}_{i_r,r})$	data limite local;
	$rd^{k,j}_{i_r,r} = TEMPO(d^{k,j}_{i_r,r})$	data de lançamento local com o fornecedor g_j;
	$rd^{k}_{i_r,r} = \underset{j=1,\ldots,Nf^{k}_{i_r,r}}{MAX}(rd^{k,j}_{i_r,r})$	data de lançamento local.
	$DD^{k}_{i_r,r} = dd^{k}_{i_r,r} + FEJ^{k}_{i_r,r}$	data limite mais tarde;
	$RD^{k,j}_{i_r,r} = rd^{k,j}_{i_r,r} - FEM^{k,j}_{i_r,r}$	data de lançamento mais cedo com o fornecedor g_j;
	$RD^{k}_{i_r,r} = \underset{j=1,\ldots,Nf^{k}_{i_r,r}}{MAX}(RD^{k,j}_{i_r,r})$	data de lançamento mais cedo.
	$\mathcal{RD}^{k,}_{i_r,r} = \bigcup_{j=1}^{Nf^{k}_{i_r,r}} \{RD^{k,j}_{i_r,r}\}$	conjunto de datas de lançamento mais cedo.

O *horizonte temporal local* para a tarefa $O^{k}_{i_r,r}$, denotado por $h^{k}_{i_r,r}$, é o intervalo $<rd^{k}_{i_r,r}, dd^{k}_{i_r,r}>$, em que $dd^{k}_{i_r,r}$ é a *data limite local*, com $dd^{k}_{i_r,r} = TEMPO(d^{i,k}_{i_r,r})$ e $rd^{k}_{i_r,r}$ é a *data de lançamento local*, com $rd^{k}_{i_r,r} = \underset{j=1,\ldots,Nf^{k}_{i_r,r}}{MAX}(TEMPO(d^{k,j}_{i_r,r}))$. A duração do intervalo $h^{k}_{i_r,r}$ inclui a *folga interna* (local) *a montante*, $fim^{k}_{i_r,r}$ e a *folga interna* (local) *a jusante*, $fij^{k}_{i_r,r}$ de $O^{k}_{i_r,r}$.

As folgas $fim^{k}_{i_r,r}$ e $fij^{k}_{i_r,r}$ têm uma definição diferente para tarefas de processamento e para tarefas de acumulação, isto é, são calculadas de modo diferente pelos agentes processadores e

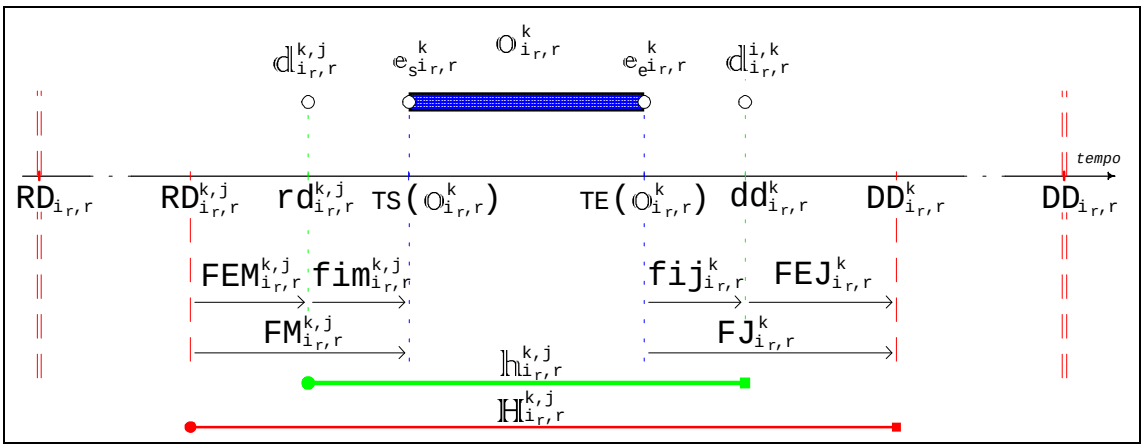
pelos agentes acumuladores. No caso de um agente processador $\mathtt{fim}^{k}_{i_r,r}$ e $\mathtt{fij}^{k}_{i_r,r}$ indicam respectivamente, o maior avanço que o evento de início de $\mathbb{O}^{k}_{i_r,r}$ ($\mathbb{e}\mathtt{s}^{k}_{i_r,r}$) pode sofrer sem comprometer a data $\mathtt{rd}^{k}_{i_r,r}$ (*i.e.*, sem que haja necessidade de g_k acordar alterações dos valores de tempo dos pedidos $\mathbb{d}^{k,j}_{i_r,r}$ com fornecedores) e o maior atraso que o evento de fim de $\mathbb{O}^{k}_{i_r,r}$ ($\mathbb{e}\mathtt{e}^{k}_{i_r,r}$) pode sofrer sem comprometer a data $\mathtt{dd}^{k}_{i_r,r}$ (*i.e.*, sem que haja necessidade de g_k acordar alteração do valor de tempo do pedido $\mathbb{d}^{i,k}_{i_r,r}$ com o cliente).

Tabela 4-6- Parâmetros temporais de escalonamento (folgas temporais) para um agente gestor g_k e relativamente a uma tarefa $\mathbb{O}^{k}_{i_r,r}$.

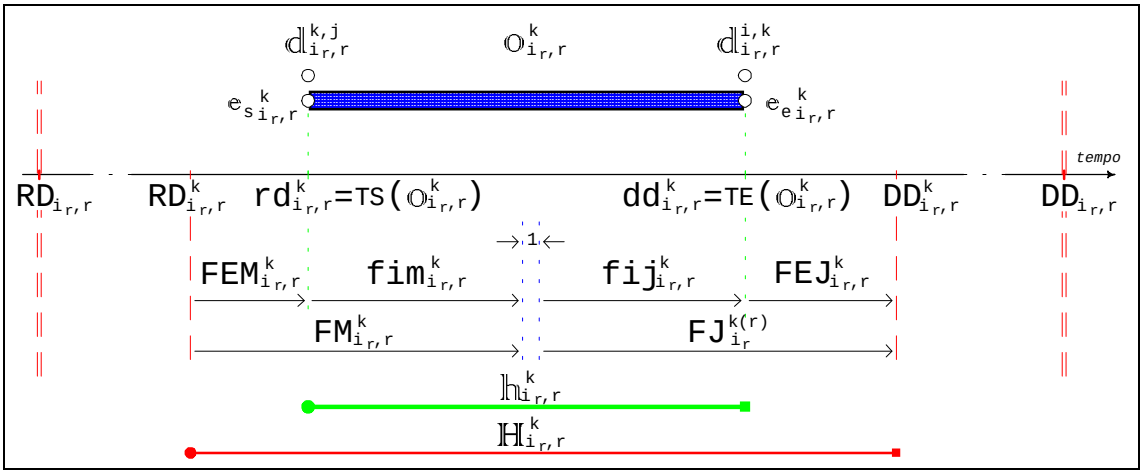
	Notação	**Descrição**
Folgas temporais	agente processador	
	$\mathtt{fij}^{k}_{i_r,r}=\mathtt{dd}^{k}_{i_r,r}-\mathtt{TE}(\mathbb{O}^{k}_{i_r,r})$	folga interna a jusante;
	$\mathtt{fim}^{k,j}_{i_r,r}=\mathtt{TS}(\mathbb{O}^{k}_{i_r,r})-\mathtt{rd}^{k,j}_{i_r,r}$	folga interna a montante com o fornecedor g_j;
	$\mathtt{fim}^{k}_{i_r,r}=\mathtt{TS}(\mathbb{O}^{k}_{i_r,r})-\mathtt{rd}^{k}_{i_r,r}$	folga interna a montante;
	agente acumulador	
	$\mathtt{fij}^{k}_{i_r,r}=\mathtt{CEILING}((\mathtt{DURAÇÃO}(\mathbb{O}^{k}_{i_r,r})-1)/2)$	folga interna a jusante;
	$\mathtt{fim}^{k}_{i_r,r}=\mathtt{DURAÇÃO}(\mathbb{O}^{k}_{i_r,r})-1-\mathtt{fij}^{k}_{i_r,r}$	folga interna a montante.
	$\mathtt{FEJ}^{k}_{i_r,r}=\mathtt{DD}^{k}_{i_r,r}-\mathtt{dd}^{k}_{i_r,r}$	folga externa a jusante;
	$\mathtt{FEM}^{k,j}_{i_r,r}=\mathtt{rd}^{k,j}_{i_r,r}-\mathtt{RD}^{k,j}_{i_r,r}$	folga externa a montante com o fornecedor g_j;
	$\mathtt{FEM}^{k}_{i_r,r}=\underset{j=1,\ldots,\mathtt{Nf}^{k}_{i_r,r}}{\mathtt{MIN}}(\mathtt{FEM}^{k,j}_{i_r,r})$	folga externa a montante.
	$\mathtt{FJ}^{k}_{i_r,r}=\mathtt{FEJ}^{k}_{i_r,r}+\mathtt{fij}^{k}_{i_r,r}$	folga a jusante;
	$\mathtt{FM}^{k,j}_{i_r,r}=\mathtt{FEM}^{k,j}_{i_r,r}+\mathtt{fim}^{k,j}_{i_r,r}$	folga a montante com o fornecedor g_j;
	$\mathtt{FM}^{k}_{i_r,r}=\underset{j=1,\ldots,\mathtt{Nf}^{k}_{i_r,r}}{\mathtt{MIN}}(\mathtt{FM}^{k,j}_{i_r,r})$	folga a montante.
	$\mathtt{FT}^{k}_{i_r,r}=\mathtt{FJ}^{k}_{i_r,r}+\mathtt{FM}^{k}_{i_r,r}$	folga total.
	$\mathtt{FJM}^{k,j}_{i_r,r}=\mathtt{FJ}^{k}_{i_r,r}+\mathtt{fim}^{k,j}_{i_r,r}$	folga jusante-montante para o fornecedor g_j;
	$\mathtt{FMJ}^{k,i}_{i_r,r}=\mathtt{FM}^{k}_{i_r,r}+\mathtt{fij}^{k}_{i_r,r}$	folga montante-jusante para o cliente g_i.

Para o caso de um agente acumulador, as definições usadas para agentes processadores não fariam sentido porque os eventos de início $\mathbb{e}\mathtt{s}^{k}_{i_r,r}$ e de fim $\mathbb{e}\mathtt{e}^{k}_{i_r,r}$, de $\mathbb{O}^{k}_{i_r,r}$ coincidem (nos valores de tempo) com o pedido enviado ao fornecedor $\mathbb{d}^{k,j}_{i_r,r}$ e o pedido recebido do cliente $\mathbb{d}^{i,k}_{i_r,r}$, respectivamente. Isto acontece porque o intervalo de tempo para escalonamento da tarefa no nó acumulador é sempre delimitado pelas datas dos pedidos $\mathbb{d}^{k,j}_{i_r,r}$ e $\mathbb{d}^{i,k}_{i_r,r}$, pois em

todos os instantes de tempo t_x tais que $t_x \geq$ TEMPO($d^{k,j}_{i_r,r}$) e $t_x <$ TEMPO($d^{i,k}_{i_r,r}$) a tarefa utiliza efectivamente capacidade de acumulação do nó gerido pelo agente. Para uma tarefa de acumulação a duração é imposta exogenamente, de modo que se pode considerar que toda a duração da tarefa contribui para a soma das folgas internas. Ora as tarefas de acumulação não são instantâneas e, partindo do princípio que uma tarefa de acumulação tem de existir, ela deverá durar, no mínimo, uma unidade de tempo. Deste modo, convenciona-se que, para qualquer tarefa de acumulação $O^k_{i_r,r}$, a soma das folgas internas é igual a DURAÇÃO($O^k_{i_r,r}$)-1, sendo ambas as folgas internas sempre não negativas.



a) Para uma tarefa de processamento $O^k_{i_r,r}$ de um agente gestor processador g_k.



b) Para uma tarefa de acumulação $O^k_{i_r,r}$ de um agente gestor acumulador g_k.

Figura 4-3- Parâmetros temporais de escalonamento para agentes gestores. São adicionalmente representados os eventos relevantes (por círculos) e as datas globais $RD_{i_r,r}$ e $DD_{i_r,r}$, referentes ao horizonte temporal global, $H_{i_r,r}$, do problema de escalonamento.

Para efeitos de definição, considera-se que o intervalo unitário de duração mínima, acima referido para uma tarefa de acumulação, é centrado relativamente ao intervalo da tarefa, pelo que as folgas internas fij e fim são *simétricas*, como expresso na Tabela 4-6 e representado na Figura 4-3-b); por consequência, também as folgas FJ e FM são *simétricas*. No entanto, como se verá na secção seguinte, para efeitos de identificação de conflitos temporais é

adequado considerar aquele intervalo deslocado à esquerda, com tempo de início igual a $TS(\mathbb{O}^{k}_{i_r,r})$, ou deslocado à direita, com tempo de fim igual a $TE(\mathbb{O}^{k}_{i_r,r})$ e fazer uso de folgas internas não simétricas; por consequência, também as folgas FJ e FM serão não simétricas. Neste último caso, na Tabela 4-6, apenas as igualdades que definem as folgas fij e fim não se aplicarão.

A *folga externa a montante* $FEM^{k}_{i_r,r}$ e a *folga externa a jusante* $FEJ^{k}_{i_r,r}$, permitem ao agente g_k obter a *data de lançamento mais cedo* $RD^{k}_{i_r,r}$ e a *data limite mais tarde* $DD^{k}_{i_r,r}$, respectivamente, de $\mathbb{O}^{k}_{i_r,r}$. Do ponto de vista dos pedidos, $RD^{k}_{i_r,r}$ e $DD^{k}_{i_r,r}$ indicam até quando o valor de $rd^{k}_{i_r,r}$ pode ser avançado e o valor de $dd^{k}_{i_r,r}$ pode ser atrasado, respectivamente, sem comprometer as datas globais $RD_{i_r,r}$ e $DD_{i_r,r}$ (ver adiante). Do ponto de vista da tarefa $\mathbb{O}^{k}_{i_r,r}$, $RD^{k}_{i_r,r}$ significa a data de início mais cedo, possível se todos os agentes a montante de g_k, no processo $\mathbb{RT}_{i_r,r}$ escalonassem as suas tarefas nas respectivas datas de início mais cedo; simetricamente, $DD^{k}_{i_r,r}$ significa a data de fim mais tarde, possível se todos os agentes a jusante de g_k, no processo $\mathbb{RT}_{i_r,r}$ escalonassem as suas tarefas nas respectivas datas de fim mais tarde.[14]

O *horizonte temporal* para $\mathbb{O}^{k}_{i_r,r}$ (para g_k), denotado por $\mathbb{H}^{k}_{i_r,r}$, é o intervalo $<RD^{k}_{i_r,r}, DD^{k}_{i_r,r}>$. A *folga a montante* de $\mathbb{O}^{k}_{i_r,r}$ (para g_k), $FM^{k}_{i_r,r}$ e a *folga a jusante* de $\mathbb{O}^{k}_{i_r,r}$ (para g_k), $FJ^{k}_{i_r,r}$ indicam em quanto tempo o evento de início da tarefa $\mathbb{O}^{k}_{i_r,r}$ pode sofrer avanço e em quanto tempo o evento de fim da mesma tarefa pode sofrer atraso, sem comprometer as datas globais $RD_{i_r,r}$ e $DD_{i_r,r}$ (ver adiante), respectivamente. A *folga total* de $\mathbb{O}^{k}_{i_r,r}$, $FT^{k}_{i_r,r}$, é a soma das folgas $FM^{k}_{i_r,r}$ e $FJ^{k}_{i_r,r}$.[15]

Os valores das folgas internas, $fim^{k}_{i_r,r}$ e $fij^{k}_{i_r,r}$, medem a flexibilidade temporal interna na perspectiva local do problema de escalonamento individual do agente g_k e deverão ser não negativos. Valores negativos de $fim^{k}_{i_r,r}$ ou $fij^{k}_{i_r,r}$ assinalam conflitos temporais de escalonamento *localmente aparentes*. Diz-se localmente aparentes porque o facto de o valor de uma folga interna ser negativa pode não corresponder obrigatoriamente a uma violação de restrições de precedência temporal entre duas tarefas, no problema global de escalonamento.

Por exemplo, o valor de $fij^{k}_{i_r,r}$ para uma tarefa de um agente processador pode ser negativo mas esta folga negativa poderá, possivelmente, ser compensada por uma folga positiva (de maior valor absoluto) existente no agente cliente (bastará que o par de agentes acorde num re-escalonamento da data limite de pedido para mais tarde de modo a que isso resulte uma

---

[14] Quer dizer, os valores temporais de $RD^{k}_{i_r,r}$ e de $DD^{k}_{i_r,r}$ significam limites rígidos para o tempo de início e o tempo de fim de $\mathbb{O}^{k}_{i_r,r}$. Estes limites não são negociáveis com os agentes cliente e fornecedor já que se assume que os limites temporais globais não são negociáveis com o exterior (*i.e.*, o horizonte temporal de escalonamento não muda), de acordo com o pressuposto 6, na secção 4.3.4.

[15] O valor da folga total $FT_{i_r,r}$, é igual para qualquer tarefa, $\mathbb{O}^{k}_{i_r,r}$ de qualquer agente gestor g_k num caminho de rede que seja, ou um *caminho crítico*, ou um *caminho quase crítico*, do mesmo processo de rede (ver mais adiante), além de que é o menor valor de folga total no processo $\mathbb{RT}_{i_r,r}$.

folga não negativa para ambos). Quando uma tarefa de agente é estabelecida, estas folgas serão, em princípio positivas pois, após recebido um novo pedido de um cliente, o agente cria e escalona a tarefa adequada e envia um conjunto de pedidos a fornecedores providenciando sempre de modo a que as folgas $\mathtt{fim}^{k}_{i_r,r}$ e $\mathtt{fij}^{k}_{i_r,r}$ sejam não negativas. No entanto, durante a resolução do problema de escalonamento é possível ocorrerem re-escalonamentos (sejam originados por causas externas ou internas ao agente) e podem, embora apenas temporariamente, ocorrer valores de folgas negativos.

Para agentes de retalho e agentes de matéria-prima, os parâmetros para escalonamento temporal são descritos na Tabela 4-7 e na Tabela 4-8, respectivamente, e são ilustrados na Figura 4-4. Nestes casos, por convenção, o valor de tempo do pedido $\mathbb{d}^{r,k}_{i_r,r}$ enviado pelo agente de retalho g_r ao agente fornecedor g_k, é igual ao valor de tempo da tarefa de retalho $\mathbb{O}^{r}_{i_r,r}$ e esta é idêntica ao pedido global do exterior à rede $\mathbb{d}^{0,r}_{i_r,r}$, quer dizer, verifica-se que:

$\mathbb{O}^{r}_{i_r,r}$=TAREFA($\mathbb{d}^{0,r}_{i_r,r}$) e TEMPO($\mathbb{d}^{r,k}_{i_r,r}$)=TEMPO($\mathbb{O}^{r}_{i_r,r}$);

Para além de que se verifica também que:[13]

PRODUTO($\mathbb{d}^{r,k}_{i_r,r}$)=PRODUTO($\mathbb{O}^{r}_{i_r,r}$) e QUANTIDADE($\mathbb{d}^{r,k}_{i_r,r}$)=QUANTIDADE($\mathbb{O}^{r}_{i_r,r}$).

Do mesmo modo, uma tarefa de matéria-prima $\mathbb{O}^{m}_{i_r,r}$, é sempre escalonada numa data igual ao valor de tempo do pedido $\mathbb{d}^{k,m}_{i_r,r}$, recebido pelo agente de matéria-prima do agente cliente, tarefa e pedido que são, aliás, idênticos; para além disso, a data do pedido global da rede ao exterior correspondente $\mathbb{d}^{m,0}_{i_r,r}$, é igual àquela data. Quer dizer, verifica-se que:

$\mathbb{O}^{m}_{i_r,r}$=TAREFA($\mathbb{d}^{k,m}_{i_r,r}$) e TEMPO($\mathbb{d}^{m,0}_{i_r,r}$)=TEMPO($\mathbb{O}^{m}_{i_r,r}$);

Para além de que se verifica também que:[13]

PRODUTO($\mathbb{d}^{m,0}_{i_r,r}$)=PRODUTO($\mathbb{O}^{m}_{i_r,r}$) e QUANTIDADE($\mathbb{d}^{m,0}_{i_r,r}$)=QUANTIDADE($\mathbb{O}^{m}_{i_r,r}$).

Também, por convenção, para o caso de agentes de retalho e agentes de matéria-prima, só existem folgas externas.

Para o caso do agente supervisor os parâmetros de escalonamento temporal são descritos na Tabela 4-9 e ilustrados na Figura 4-5. O *horizonte temporal global* (o horizonte temporal de escalonamento para o processo de rede) para satisfação do pedido global $\mathbb{d}^{0,r}_{i_r,r}$ é $\mathbb{H}_{i_r,r}$, sendo $\mathbb{H}_{i_r,r}$ o intervalo $<RD_{i_r,r}, DD_{i_r,r}>$, em que $DD_{i_r,r}$ é a *data limite global* e $RD_{i_r,r}$ é a *data de lançamento global*. As datas $RD_{i_r,r}$ e $DD_{i_r,r}$ são limites temporais globais rígidos dentro dos quais todos os eventos (*i.e.*, todos os pedidos e todos os eventos que definem todas as tarefas do processo de rede) necessários à satisfação do pedido global à rede $\mathbb{d}^{0,r}_{i_r,r}$, têm de ocorrer.

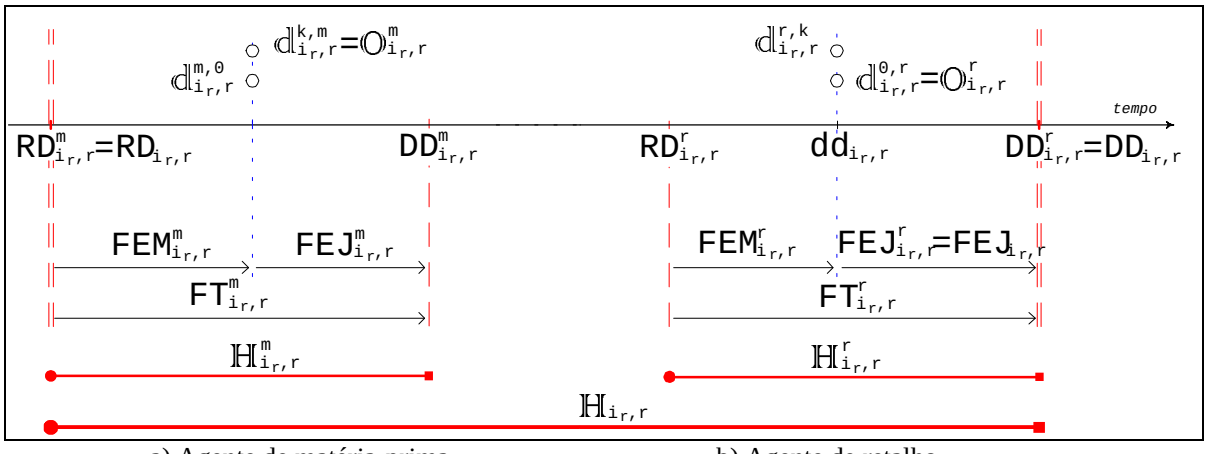
O valor da *folga externa a jusante*, $FEJ_{i_r,r}$ indica em quanto a data do pedido $\mathbb{d}^{0,r}_{i_r,r}$ ($dd_{i_r,r}$) pode ser atrasada sem exceder a data limite global $DD_{i_r,r}$ ou, alternativamente, em quanto esta última data poderia ser avançada.

Tabela 4-7- Parâmetros temporais de escalonamento (datas, folgas temporais e intervalos) para um agente de retalho g_r. Por convenção, as folgas internas são todas nulas.

	Notação	Descrição
Intervalos	$\mathbb{H}^{r}_{i_r,r} = <RD^{r}_{i_r,r}, DD^{r}_{i_r,r}>$	horizonte temporal.
Datas	$DD^{r}_{i_r,r} = DD_{i_r,r}$ $RD^{r}_{i_r,r} = DD^{r}_{i_r,r} - FT^{r}_{i_r,r}$	data limite mais tarde; data de lançamento mais cedo.
Folgas temporais	$FEJ^{r}_{i_r,r} = DD^{r}_{i_r,r} - TEMPO(\mathbb{d}^{0,r}_{i_r,r})$ $FEM^{r}_{i_r,r}$	folga externa a jusante; folga externa a montante.
	$FT^{r}_{i_r,r} = FEJ^{r}_{i_r,r} + FEM^{r}_{i_r,r}$	folga total.
	$FJM^{r,j}_{i_r,r} = FEJ^{r}_{i_r,r}$	folga jusante-montante.

Tabela 4-8- Parâmetros temporais de escalonamento (datas, folgas temporais e intervalos) para um agente de matéria-prima g_m. Por convenção, as folgas internas são todas nulas.

	Notação	Descrição
Intervalos	$\mathbb{H}^{m}_{i_r,r} = <RD^{m}_{i_r,r}, DD^{m}_{i_r,r}>$	horizonte temporal.
Datas	$DD^{m}_{i_r,r} = RD^{m}_{i_r,r} + FT^{m}_{i_r,r}$ $RD^{m}_{i_r,r} = RD_{i_r,r}$	data limite mais tarde; data de lançamento mais cedo.
Folgas temporais	$FEJ^{m}_{i_r,r}$ $FEM^{m}_{i_r,r} = TEMPO(\mathbb{d}^{m,0}_{i_r,r}) - RD^{m}_{i_r,r}$	folga externa a jusante; folga externa a montante.
	$FT^{m}_{i_r,r} = FEJ^{m}_{i_r,r} + FEM^{m}_{i_r,r}$	folga total.
	$FMJ^{m,i}_{i_r,r} = FEM^{m}_{i_r,r}$	folga montante-jusante.



a) Agente de matéria-prima. b) Agente de retalho.

Figura 4-4- Parâmetros temporais de escalonamento para um agente de matéria-prima e para um agente de retalho (os super-índices k dos pedidos para um caso e para o outro, dizem respeito, possivelmente, a agentes gestores diferentes). São adicionalmente representados os eventos relevantes (por círculos), as datas globais $RD_{i_r,r}$ e $DD_{i_r,r}$ e o horizonte temporal global, $\mathbb{H}_{i_r,r}$ do problema de escalonamento.

Tabela 4-9- Parâmetros temporais globais de escalonamento (datas, folgas temporais e intervalos), para o agente supervisor.

	Notação		Descrição
Intervalos	$\mathbb{H}_{i_r,r}$	$=<RD_{i_r,r}, DD_{i_r,r}>$	horizonte temporal global;
	$\mathbb{H}^0_{i_r,r}$	$=<RD^0_{i_r,r}, DD^0_{i_r,r}>$	horizonte de folga a jusante.
Datas	$dd_{i_r,r}$	$=TEMPO(\mathbb{d}^{0,r}_{i_r,r})$	data do pedido do exterior.
	$DD_{i_r,r}$		data limite global;
	$RD_{i_r,r}$		data de lançamento global.
	$DD^0_{i_r,r}$	$=DD_{i_r,r}$	data mais tarde para $\mathbb{d}^{0,r}_{i_r,r}$;
	$RD^0_{i_r,r}$	$=DD^0_{i_r,r}-FT_{i_r,r}$	data mais cedo para $\mathbb{d}^{0,r}_{i_r,r}$.
Folgas temporais	$FEJ_{i_r,r}$	$=DD_{i_r,r}-dd_{i_r,r}$	folga externa a jusante;
	$FEM_{i_r,r}$	$=\min_{m=1,\ldots,Nm^r_{i_r}}(TEMPO(\mathbb{d}^{m,0}_{i_r,r})-RD_{i_r,r})$	folga externa a montante.
	$FET_{i_r,r}$	$=FEJ_{i_r,r}+FEM_{i_r,r}$	folga externa total;
	$FIT_{i_r,r}$	$=FT_{i_r,r}-FET_{i_r,r}$	folga interna total.
	$FT_{i_r,r}$	$=FT^r_{i_r,r}$	folga total.

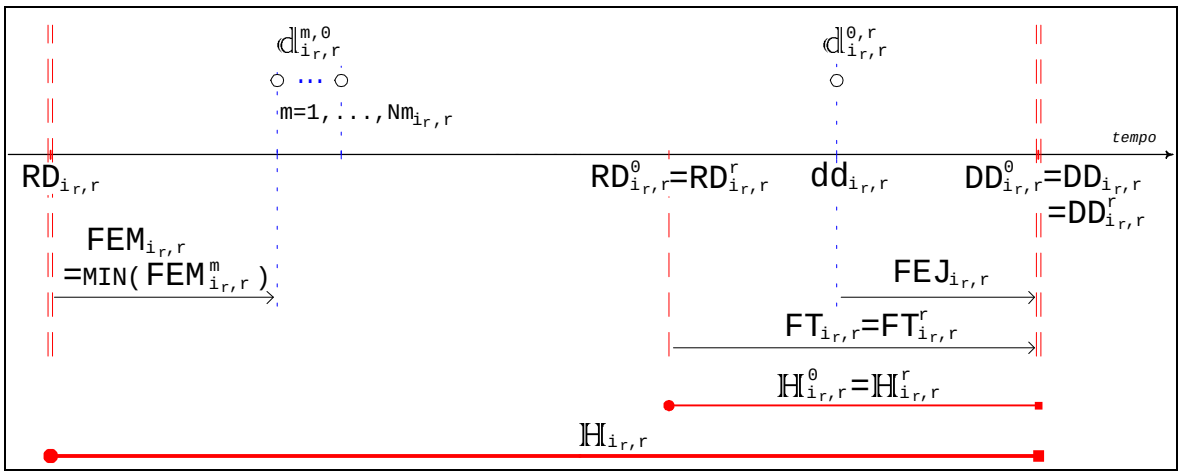


Figura 4-5- Parâmetros temporais globais de escalonamento para o agente supervisor. São adicionalmente representados os eventos relevantes (por círculos).

O valor da *folga externa a montante* $FEM_{i_r,r}$, indica em quanto a data de qualquer pedido $\mathbb{d}^{m,0}_{i_r,r}$ (com $m=1,\ldots,Nm_{i_r,r}$) pode ser avançada sem anteceder a data limite global $RD_{i_r,r}$ ou, alternativamente, em quanto esta última data poderia ser atrasada.

A *folga externa total* $FET_{i_r,r}$, é um indicador de flexibilidade temporal externa, já que é definida como sendo a soma de $FEJ_{i_r,r}$ e $FEM_{i_r,r}$. A *folga interna total* $FIT_{i_r,r}$ (definida como a diferença entre $FT_{i_r,r}$ e $FET_{i_r,r}$) é um indicador de flexibilidade temporal interna.

A *folga total* $FT_{i_r,r}$, mede a flexibilidade temporal global na perspectiva do problema de escalonamento de toda a rede. Se $FT_{i_r,r}<0$ o problema de escalonamento é globalmente

super-constrangido e não tem solução.[16] É possível ter valores de $\mathtt{FEJ}_{i_r,r}$ ou de $\mathtt{FEM}_{i_r,r}$ negativos e mesmo assim o valor de $\mathtt{FT}_{i_r,r}$ ser positivo, o que indicará um escalonamento global com atraso, ou com avanço, relativamente às datas $\mathtt{DD}_{i_r,r}$ e $\mathtt{RD}_{i_r,r}$, respectivamente.

4.3.6 Um Mecanismo de Coordenação para Escalonamento Temporal

Descrevem-se a seguir um mecanismo de coordenação e um conjunto de fases para resolução cooperativa de problemas de escalonamento multi-agente no contexto de rede de EE. Aquele mecanismo permite que os agentes gestores percepcionem, localmente, as restrições temporais globais rígidas de um problema de escalonamento de rede e é enquadrado nas fases de resolução.

As restrições temporais globais acima referidas, apelidadas conjuntamente de *restrições de datas limite*, são as *restrições de data de lançamento global* e as *restrições de data limite global*. Se o problema de escalonamento é introduzido por um pedido global do exterior à rede $\mathbb{d}^{0,r}_{i_r,r}$, as restrições de datas limite impõem que todos os pedidos locais necessários para a satisfação do pedido $\mathbb{d}^{0,r}_{i_r,r}$ tenham valores temporais durante o horizonte temporal global $\mathbb{H}_{i_r,r}$. Em termos da notação antes introduzida, as restrições de datas limite podem ser traduzidas, de forma resumida, pela verificação da seguinte condição, para cada agente gestor envolvido g_k:

$$\forall_j \ \ (j=1,\ldots,\mathtt{Nf}^k_{i_r,r}) \ : \ \mathtt{DURANTE?}(\mathbb{h}^{k,j}_{i_r,r}, \mathbb{H}_{i_r,r}) \qquad (1)$$

Para cada agente gestor g_k, envolvido num processo de rede, esta condição exprime que todos os intervalos de horizonte temporal local estão dentro do (ocorrem *durante* o) intervalo de horizonte temporal global do mesmo processo.

Adicionalmente, cada agente gestor g_k deve respeitar localmente, para cada tarefa $\mathbb{O}^k_{i_r,r}$, além das restrições de capacidade individuais, as *restrições de precedência temporal* de tarefas do processo de rede da tarefa. Estas restrições impõem que $\mathbb{O}^k_{i_r,r}$ deve preceder a tarefa do cliente $\mathbb{O}^i_{i_r,r}$ e suceder às tarefas do fornecedor $\mathbb{O}^j_{i_r,r}$, *i.e.*, deve verificar-se $\mathtt{TE}(\mathbb{O}^k_{i_r,r}) \leq \mathtt{TS}(\mathbb{O}^i_{i_r,r})$ e $\mathtt{TE}(\mathbb{O}^j_{i_r,r}) \leq \mathtt{TS}(\mathbb{O}^k_{i_r,r})$ (com $\mathtt{j=1,\ldots,Nf}^k_{i_r,r}$). No entanto, as tarefas de um agente são objectos do seu domínio privado e não são visíveis pelos outros agentes, de modo que, a informação associada às tarefas de cada agente não é conhecida pelos restantes. A informação que é sempre partilhada é a que está associada ao pedido local entre um par de agentes cliente-fornecedor. Relativamente ao agente gestor g_k, a informação que ele conhece é a que está associada ao pedido local $\mathbb{d}^{i,k}_{i_r,r}$, que o agente partilha com o cliente g_i e aos pedidos locais $\mathbb{d}^{k,j}_{i_r,r}$, que o agente partilha com os fornecedores g_j (com $\mathtt{j=1,\ldots,Nf}^k_{i_r,r}$), além da informação associada a $\mathbb{O}^k_{i_r,r}$. Deste modo, as restrições temporais de precedência entre tarefas do mesmo processo, não são directamente percepcionadas pelos agentes mas traduzem-se, localmente para cada agente g_k envolvido no

[16] Assume-se que a duração de qualquer tarefa de processamento é fixa e que os limites do horizonte temporal de escalonamento não mudam, de acordo com os pressupostos 2 e 6, na secção 4.3.4.

processo nas relações de precedência expressas por $\mathtt{TE}(\mathbb{O}^{k}_{i_r,r}) \leq \mathtt{TEMPO}(\mathbb{d}^{i,k}_{i_r,r})$ e $\mathtt{TEMPO}(\mathbb{d}^{k,j}_{i_r,r}) \leq \mathtt{TS}(\mathbb{O}^{k}_{i_r,r})$ (com $\mathtt{j=1,...,Nf}^{k}_{i_r,r}$). Em termos da notação antes introduzida, estas restrições podem ser traduzidas, de forma resumida, pela verificação da seguinte condição, para cada tarefa de cada agente gestor g_k:

$$\mathtt{DURANTE?}(\mathtt{INTERVALO}(\mathtt{TS}(\mathbb{O}^{k}_{i_r,r}),\mathtt{TE}(\mathbb{O}^{k}_{i_r,r})),\mathbb{h}^{k}_{i_r,r}) \qquad (2)$$

Para cada tarefa de cada agente gestor g_k, esta condição exprime que o intervalo da tarefa está dentro do (ocorre *durante* o) intervalo de horizonte temporal local do mesmo processo.[17]

Se no escalonamento de um processo todas as restrições de datas limite e todas as restrições de precedência temporal forem simultâneamente respeitadas, todas as tarefas $\mathbb{O}^{k}_{i_r,r}$ do processo estarão escalonadas dentro do horizonte temporal global $\mathbb{H}_{i_r,r}$. No entanto, na perspectiva local de cada agente gestor, apenas são percepcionadas, ainda que indirectamente, as restrições temporais de precedência.

O mecanismo de coordenação para resolução cooperativa de problemas de escalonamento multi-agente que vai ser exposto encara o escalonamento apenas na perspectiva temporal, isto é, apenas garante que as restrições temporais de escalonamento são respeitadas. Este mecanismo é baseado na partilha de determinada informação entre pares de agentes cliente-fornecedor, adicionalmente à informação associada aos pedidos. A informação adicional permite que cada agente conheça os limites temporais de escalonamento para as suas tarefas e dos valores de tempo dos pedidos de clientes e dos pedidos a fornecedores (os intervalos $\mathbb{H}^{k,j}_{i_r,r}$). Conforme se vai descrever, o agente pode então reconhecer, localmente, se o problema de escalonamento é ou não temporalmente super-constrangido, isto é, se tem ou não solução global possível, do ponto de vista temporal. Se o problema de escalonamento não for temporalmente super-constrangido, o conhecimento daqueles limites temporais permite que, juntamente com as restrições temporais de precedência, o agente possa respeitar também as restrições globais de datas limite. Isto abrange situações de recorrência a pedidos de re-escalonamento inter-agente (por exemplo, para reparar uma solução devido ao facto de existirem violações de restrições), que envolvem actividades adicionais de comunicação, a minimizar. Nestas situações os agentes podem evitar envolver-se em actividades de re-escalonamento com opções de escalonamento não viáveis em soluções globalmente possíveis, ou minimizar a actividade de re-escalonamento realizando os pedidos com datas de entrega dentro dos limites temporais acima referidos. O reconhecimento de um problema como temporalmente super-constrangido permite que o agente desista do problema (rejeitando o respectivo pedido local de cliente) com a certeza de a desistência não originar a inviabilização de uma solução global possível para o problema de escalonamento multi-agente, pois tal solução possível não existe. Note-se que, neste caso, a não desistência do problema iria envolver os agentes numa actividade de re-escalonamento inútil, em que procurariam

[17] Note-se que o intervalo de horizonte temporal local $\mathbb{h}^{k}_{i_r,r}$ ocorre sempre durante qualquer dos intervalos de horizonte temporal local com cada fornecedor $\mathbb{h}^{k,j}_{i_r,r}$ (ver a Tabela 4-5) e, portanto, afirmar que o intervalo de $\mathbb{O}^{k}_{i_r,r}$ ocorre durante $\mathbb{h}^{k}_{i_r,r}$ equivale a afirmar que o intervalo de $\mathbb{O}^{k}_{i_r,r}$ ocorre durante qualquer dos intervalos $\mathbb{h}^{k,j}_{i_r,r}$.

encontrar uma soluão possível quando ela não existe. Com o emprego do mecanismo de coordenação referido, a informação que fica disponível a um agente permite-lhe, além das folgas internas `fij` e `fim`, calcular também todas as restantes folgas `FT`, `FJ`, `FM`, `FEJ` e `FEM` (incluindo as folgas `fij`, `fim`, `FJ`, `FM` simétricas e as não simétricas, para o caso de um agente acumulador). Será mantendo valores não negativos das folgas que um agente vai garantir a não violação de restrições temporais.

Em resumo, o emprego do mecansimo de coodenação que se vem referindo permite que cada agente possa contribuir localmente, de forma cooperativa, para uma solução global possível do problema de escalonamento multi-agente. Os detalhes deste mecanismo são descritos a seguir.

Para um dado problema de escalonamento de rede, apelidem-se *soluções tempo-possíveis* as soluções que não violam restrições temporais, *soluções recurso-possíveis* as soluções que não violam restrições de capacidade e *soluções possíveis* as soluções que não violam restrições temporais nem violam restrições de capacidade. O conjunto de soluções possíveis é igual à intersecção do conjunto de soluções tempo-possíveis com o conjunto de soluções recurso-possíveis. Se ambos estes conjuntos forem não vazios e tiverem uma intersecção não vazia existirão soluções possíveis para o problema.[18] A Figura 4-6 ilustra este aspecto.

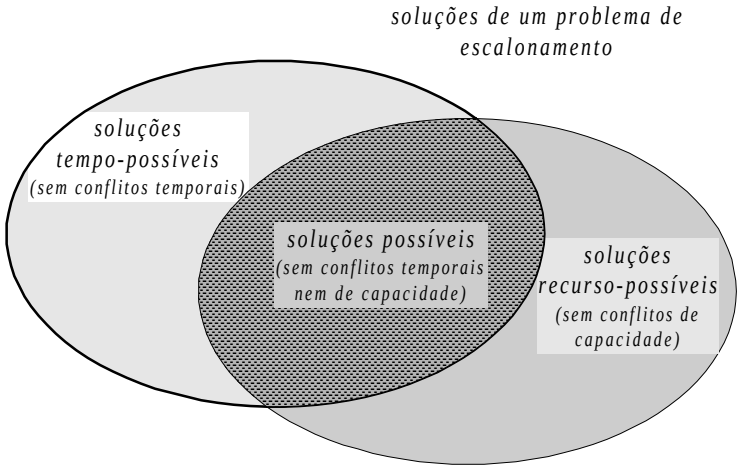


Figura 4-6- As soluções para um problema de escalonamento.

O uso do mecanismo de escalonamento temporal para resolução de problemas de escalonamento multi-agente baseia-se em restringir o conjunto de soluções ao conjunto de soluções tempo-possíveis, de modo a que a actividade de escalonamento e re-escalonamento dos agentes da rede se encaminhe apenas para soluções deste último conjunto. Se se detecta que o conjunto de soluções tempo-possíveis é vazio, a resolução do problema deve terminar por falha. Caso contrário, a resolução deve prosseguir procurando estabelecer uma solução tempo-possível que seja recurso-possível.

[18] Se for dado um critério de optimalidade, pode ainda definir-se o conjunto de soluções óptimas. Este é então um subconjunto do conjunto de soluções possíveis.

4.3.6.1 Taxonomia dos Conflitos Temporais

Para uma melhor compreensão dos tipos de conflitos temporais tratados apresenta-se uma classificação e caracterização destes conflitos. A classificação dos conflitos temporais é mostrada na Figura 4-7 e a caracterização deles é exposta a seguir. Assume-se que a duração de qualquer tarefa de processamento e os valores dos parâmetros do problema de escalonamento de rede (pedido global do exterior à rede e datas limite e de lançamento globais) são fixos. Um conflito é apelidado de *fatal* se, devido à sua existência o conjunto de soluções tempo-possíveis é vazio e, portanto, o problema não tem solução.

Conflito temporal absoluto - Atende-se ao tempo actual. Este tipo de conflitos advém da restrição de uma tarefa só poder ser escalonada num tempo futuro (*i.e.*, com tempo de início maior que o tempo actual). A detecção destes conflitos é apropriada em cenários de escalonamento dinâmico e reactivo, em que é necessário considerar o tempo actual *no momento de escalonamento de cada tarefa*. Pode ser:

Conflito absoluto local - Quando uma tarefa de agente gestor está escalonada com um valor de tempo de início não superior ao tempo actual, mas existe folga temporal a jusante suficiente para acomodar um re-escalonamento da tarefa para a frente no tempo (atraso) de modo àquela condição deixar de se verificar;

Conflito absoluto global - Condição semelhante à anterior mas não existindo folga temporal a jusante suficiente. É um conflito fatal.

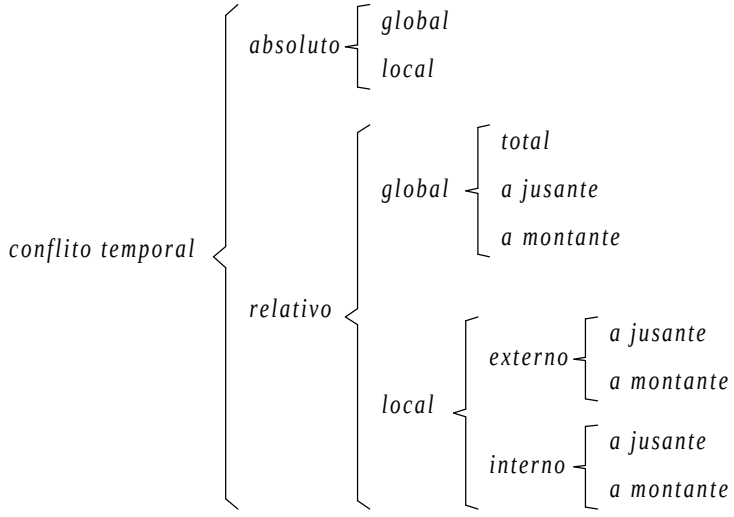


Figura 4-7- Classificação dos conflitos temporais.

Conflito temporal relativo - Não se atende ao tempo actual, mas apenas aos valores das folgas temporais que foram definidas anteriormente. Pode ser:

Conflito relativo global - Coexiste sempre com outros conflitos do mesmo tipo envolvendo mais de um agente gestor, e poderá ser ou não fatal. Quando não é fatal, para ser eliminado não basta que os agentes gestores recorram ao re-escalonamento de pedidos, sendo sempre necessário o re-escalonamento de tarefas. Este tipo de conflitos verifica-se quando há agentes gestores que detectam folgas `FT`, `FJ` ou `FM` negativas. Um conflito global pode ser:

Conflito relativo global total - Este conflito é fatal e verifica-se quando há agentes gestores com folgas `FT` negativas;

Conflito relativo global a jusante - Conflito não fatal. Verifica-se quando há agentes gestores com folgas `FJ` negativas. Para ser eliminado os agentes gestores têm de recorrer ao re-escalonamento de tarefas e de pedidos locais para trás no tempo (avanço);

Conflito relativo global a montante - Conflito não fatal. Verifica-se quando há agentes gestores com folgas `FM` negativas. Para ser eliminado os agentes gestores têm de recorrer ao re-escalonamento de tarefas e de pedidos locais para a frente no tempo (atraso).

Conflito relativo local - Este tipo de conflitos poderá afectar um grupo restrito de agentes (*e.g.*, um par de agentes cliente-fornecedor) ou não, mas nunca é fatal. Para sua eliminação basta que os agentes recorram apenas ao re-escalonamento de pedidos, ou apenas ao re-escalonamento de tarefas. Verifica-se quando há agentes gestores que detectam folgas `FEJ`, `FEM`, `fij` ou `fim` negativas. Um conflito local pode ser:

Conflito relativo local externo - Verifica-se quando há agentes gestores com folgas externas (`FEJ` ou `FEM`) negativas. Pode ser:

A jusante - Quando há agentes gestores com folgas `FEJ` negativas. Para eliminar estes conflitos basta que cada agente nessas condições acorde com o cliente o re-escalonamento para trás no tempo apropriado do pedido que origina o conflito;

A montante - Quando há agentes gestores com folgas `FEM` negativas. Para eliminar estes conflitos basta que cada agente nessas condições acorde com o fornecedor, ou fornecedores, os re-escalonamentos para a frente no tempo apropriados dos pedidos que originam conflitos.

Conflito relativo local interno - Verifica-se quando há agentes gestores com folgas internas (`fij` ou `fim`) negativas. Pode ser:

A jusante - Quando há agentes gestores com folgas `fij` negativas. Para eliminar estes conflitos basta que cada agente nessas condições re-escalone apropriadamente a tarefa que origina o conflito para trás no tempo;

A montante - Quando há agentes gestores com folgas `fim` negativas. Para eliminar estes conflitos basta que cada agente nessas condições re-escalone apropriadamente a tarefa que origina o conflito para a frente no tempo.

Este trabalho não aborda conflitos temporais absolutos. Cenários possíveis de ambientes de escalonamento dinâmico e reactivo, com tratamento de eventuais conflitos daquele tipo são relegados para trabalho futuro. Deste modo, apenas se atenderá a conflitos temporais relativos e os conflitos temporais que forem referidos a partir de agora são conflitos temporais relativos.

4.3.6.2 Regras para Identificação de Conflitos Temporais

A identificação do tipo de um conflito temporal por um agente gestor é apropriada para que lhe possa dar o tratamento adequado, de modo a eliminá-lo. Esta identificação pode ser simplificada se se fizer a assumpção a seguir descrita.

A não violação de restrições temporais de folgas internas (`fij` e `fim`) apenas depende de cada agente individual. No caso de um agente acumulador estas folgas são sempre não negativas. No caso de um agente processador, nenhum pedido de cliente deve ter um valor de

tempo menor que o do evento de fim da tarefa que satisfará o pedido (*i.e.*, a folga $\mathtt{fij}$ deve ser não negativa), nem nenhum pedido a um fornecedor deve ter um valor de tempo maior do que o do evento de início da tarefa para a qual o pedido é necessário (*i.e.*, a folga $\mathtt{fim}$ deve ser não negativa). Assume-se então que qualquer agente, para cada problema, mantém sempre folgas internas $\mathtt{fij}$ e $\mathtt{fim}$ não negativas. Com esta assumpção verifica-se, na definição do mecanismo de escalonamento temporal, o seguinte:

– Põe-se de parte a necessidade de identificação de conflitos temporais locais internos, restando apenas identificar conflitos globais (totais, a jusante e a montante) e conflitos locais externos (a jusante e a montante);

– Conflitos globais a montante e a jusante aparecem sempre acompanhados por conflitos locais externos: sempre que ocorrem conflitos globais a jusante ocorrem simultaneamente conflitos locais externos a jusante e sempre que ocorrem conflitos globais a montante ocorrem simultaneamente conflitos locais externos a montante.

O agente pode então identificar o tipo de um conflito aplicando apenas uma sequência de cinco regras condicionais. Para um agente gestor genérico g_k envolvido na satisfação do pedido global do exterior à rede $d^{0,r}_{i_r,r}$ estas regras são descritas a seguir. Para o caso particular dos agentes acumuladores, a definição empregue nas condições das regras para $\mathtt{FJ}^k_{i_r,r}$ é a de uma folga $\mathtt{FJ}$ não simétrica com o intervalo de duração mínima da tarefa deslocado à esquerda e a definição para $\mathtt{FM}^k_{i_r,r}$ é a de uma folga $\mathtt{FM}$ não simétrica com o intervalo de duração mínima da tarefa deslocado à direita (consultem-se as excepções para o cálculo destas folgas mais abaixo).

Regra 0 - Se $\mathtt{FT}^k_{i_r,r}<0$, o agente g_k conclui que há um conflito temporal global total. O problema é temporalmente super-constrangido e, portanto, não tem solução.

Regra 1 - Se $\mathtt{FJ}^k_{i_r,r}<0$ e $\mathtt{FEJ}^k_{i_r,r}<0$, o agente g_k conclui que há um conflito temporal global a jusante e um conflito temporal local externo a jusante.

Regra 2 - Se $\mathtt{FM}^k_{i_r,r}<0$ e $\mathtt{FEM}^k_{i_r,r}<0$, o agente g_k conclui que há um conflito temporal global a montante e um conflito temporal local externo a montante.

Regra 3 - Se $\mathtt{FEJ}^k_{i_r,r}<0$ e $\mathtt{FJ}^k_{i_r,r}\geq 0$, o agente g_k conclui que há um conflito temporal local externo a jusante.

Regra 4 - Se $\mathtt{FEM}^k_{i_r,r}<0$ e $\mathtt{FM}^k_{i_r,r}\geq 0$, o agente g_k conclui que há um conflito temporal local externo a montante.

Das regras expostas a regra 0 deve ser aplicada em primeiro lugar. Apenas no caso em que não se conclua que o problema é temporalmente super-constrangido se devem depois aplicar as regras 1, 2, 3 e 4.

No caso particular de agentes acumuladores, as condições das regras 1, 2, 3 e 4 fazem uso de folgas $\mathtt{FJ}^k_{i_r,r}$ e $\mathtt{FM}^k_{i_r,r}$ não simétricas, cujo cálculo é baseado em folgas internas $\mathtt{fij}^k_{i_r,r}$ e $\mathtt{fim}^k_{i_r,r}$ não simétricas com definição diferente da exposta na Tabela 4-6. O cálculo das folgas não simétricas é descrito a seguir (para as folgas $\mathtt{FEJ}^k_{i_r,r}$ e $\mathtt{FEM}^k_{i_r,r}$ as definições expostas na Tabela 4-6 mantêm-se).

Para as regras 1 e 3, que detectam conflitos a jusante, o intervalo mínimo de duração (unitária) da tarefa de acumulação considerado deve estar deslocado no extremo à esquerda dentro do intervalo da tarefa. Tem-se então que, para aplicar as regras 1 e 3 no caso de agentes acumuladores, as definições de folgas internas a utilizar são:

$\mathtt{fij}^{k}_{i_r,r}=\mathtt{DURAÇÃO}(\mathbb{O}^{k}_{i_r,r})-1$

$\mathtt{fim}^{k}_{i_r,r}=0$

Para o cálculo da folga $\mathtt{FJ}^{k}_{i_r,r}$ usa-se depois a definição exposta na Tabela 4-6.

Para as regras 2 e 4, que detectam conflitos a montante, o intervalo mínimo de duração (unitária) da tarefa de acumulação considerado deve estar deslocado no extremo à direita dentro do intervalo da tarefa. Tem-se então que, para aplicar as regras 2 e 4 no caso de agentes acumuladores, as definições de folgas internas a utilizar são:

$\mathtt{fij}^{k}_{i_r,r}=0$

$\mathtt{fim}^{k}_{i_r,r}=\mathtt{DURAÇÃO}(\mathbb{O}^{k}_{i_r,r})-1$

Para o cálculo da folga $\mathtt{FM}^{k}_{i_r,r}$ usa-se depois a definição exposta na Tabela 4-6.

As regras 1, 2, 3 e 4 permitem tratamentos diferenciados por parte do agente na eliminação dos conflitos temporais. Estes tratamentos serão referidos por tratamentos de Caso 1, Caso 2, Caso 3 e Caso 4 (a serem respectivamente associados às regras 1, 2, 3, e 4) de re-escalonamento para eliminação de conflitos temporais. Para maior simplicidade, a partir de agora, para as regras que detectam mais de um conflito, apenas será referido o conflito de maior extensão temporal. Deste modo, para a regra 1 apenas se referirá o conflito temporal global a jusante e para a regra 2 apenas se referirá o conflito temporal global a montante.

As condições das cinco regras expostas serão reformuladas, mais adiante, em termos de predicados apropriados à detecção de conflitos temporais.[19]

Por uma questão de prioridade de tratamento aos conflitos globais (mais graves porque a sua eliminação envolve sempre re-escalonamentos de tarefas e re-escalonamentos numa extensão temporal maior e abrangendo possivelmente mais agentes), para um mesmo problema de escalonamento o agente deve aplicar as regras 1 e 2 antes de aplicar as regras 3 e 4. Adicionalmente, assim que uma regra é aplicada, se identifica um conflito temporal, o respectivo tratamento de eliminação do conflito deve ser aplicado de seguida.

4.3.6.3 Fases da Resolução de Problemas de Escalonamento de Rede

Para uso adequado do mecanismo na resolução cooperativa de um problema de escalonamento de rede preconiza-se um método composto pela seguinte sequência de fases:

[19] Ver a secção 4.4.5.2.

Fase 1 - Estabelecimento de Pedidos Locais

O objectivo desta fase é determinar se o problema é ou não temporalmente super-constrangido e, no caso de o não ser, estabelecer uma solução inicial.

Os agentes gestores dão início à actividade de escalonamento para o problema com a propagação, para montante, de mensagens de pedidos locais, contendo os pedidos de fornecimento adequados, por parte de cada agente gestor envolvido no problema, de modo a que as folgas internas, `fij` e `fim`, possam ser não negativas. A seguir os agentes propagam, para jusante, mensagens de aceitação ou rejeição dos mesmos pedidos. Cada agente gestor aceita ou rejeita o pedido do cliente só após ter recebido mensagens de aceitação ou rejeição de todos os seus pedidos de fornecimento. Se um ou mais dos pedidos de fornecimento são rejeitados, o agente rejeita o pedido do cliente e cancela todos os pedidos de fornecimento que foram aceites. Quando um agente gestor obteve aceitação de todos os seus pedidos de fornecimento, determina a seguir, usando a regra 0 (com informação obtida pelo mecanismo de escalonamento temporal, a detalhar mais adiante) se o problema é temporalmente super-constrangido. Em caso de o ser, o agente rejeita o pedido local que recebeu do cliente e cancela todos os pedidos de fornecimento. Se o problema não é temporalmente super-constrangido, o agente aceita o pedido do cliente e escalona a sua tarefa de modo a ter folgas internas, `fij` e `fim`, não negativas. No primeiro caso há pedidos locais rejeitados ou cancelados na rede e a resolução termina nesta fase por falha. No segundo caso nenhum pedido local foi rejeitado, ou aceite e depois cancelado e todos os agentes escalonaram as tarefas apropriadas. Neste último caso, portanto, uma solução inicial foi estabelecida e a resolução prossegue na fase seguinte.

Fase 2 - Eliminação de Conflitos Temporais

O objectivo desta fase é reparar a solução estabelecida na fase anterior através da eliminação dos conflitos temporais que nela existam.

Cada agente gestor envolvido no problema conhece já, da fase anterior, os limites temporais para os pedidos de cliente e a fornecedores e os limites temporais para os eventos de início e fim da sua tarefa (em ambos os casos mediante informação obtida pelo mecanismo de escalonamento temporal). Pode portanto determinar se há, ou não, violação de restrições temporais na solução estabelecida na fase anterior, aplicando as regras 1, 2, 3 e 4, acima descritas. No caso de não haver, a resolução passa à fase seguinte com a solução anteriormente estabelecida. No caso de haver violações de restrições temporais, cada agente que as detecta procede a acções de re-escalonamento de pedidos e, se necessário, também da tarefa, de modo a eliminar estas violações sem gerar outras do mesmo tipo (para cada caso de conflito temporal identificado por cada uma das regras 1, 2, 3 e 4, um tratamento específico é aplicado, a detalhar mais adiante). O re-escalonamento de um pedido consiste em enviar (a um cliente ou fornecedor) uma mensagem de pedido de re-escalonamento adequado, com vista a deslocar o valor de tempo do pedido para dentro dos limites temporais aceitáveis. O re-escalonamento de uma tarefa consiste em deslocar a tarefa no tempo de modo a que os valores de tempo dos eventos de início e fim da mesma fiquem dentro dos limites temporais aceitáveis para a tarefa. Desde que permita eliminar uma violação de restrição temporal sem criar outras, um pedido de re-escalonamento desta fase *tem de ser sempre aceite* pelo agente que o recebe, uma vez que o que está em causa é estabelecer uma solução isenta de conflitos temporais. Para que isto aconteça, a mensagem que contém o pedido de re-escalonamento deverá conter também a justificação apropriada. A justificação deve incluir o tipo de folga e o valor negativo de folga detectado (folgas `FJ`, `FM`, `FEJ` e `FEM` respectivamente nos casos de

conflito temporal global a jusante, global a montante, local externo a jusante e local externo a montante). No final desta fase uma solução isenta de conflitos temporais é estabelecida.

O mecanismo de coordenação para escalonamento temporal permite guiar as acções dos agentes não apenas nas fases 1 e 2 de resolução de um problema de escalonamento, mas durante todo o intervalo de tempo em que o problema existe. Adicionalmente, existe uma fase 3 de resolução do problema que tem a ver com a eliminação de conflitos de capacidade. Esta fase envolve, no entanto, não apenas um problema de escalonamento mas vários, no caso geral. A resolução de um particular problema de escalonamento de rede na fase 3, sobrepõe-se temporalmente, e interage com, a resolução de outros possíveis problemas de escalonamento de rede que estejam na mesma fase já que, para cada agente, tarefas de vários problemas de escalonamento competem pela capacidade do nó gerido pelo agente.

Fase 3 - Eliminação de Conflitos de Capacidade

O objectivo desta fase é reparar a solução estabelecida na fase anterior através da eliminação de todos os conflitos de capacidade existentes, sem que disso resultem conflitos temporais. Um procedimento de eliminação de conflitos de capacidade apropriado para esta fase deverá recorrer ao re-escalonamento de tarefas e ao re-escalonamento de pedidos e, em última instância, recorrer ao cancelamento de pedidos. A seguir descreve-se um procedimento possível de uma forma muito geral.

Cada agente gestor que detecte conflitos de capacidade deve re-escalonar as suas tarefas, de modo a que o número desses conflitos seja minimizado e de modo a que não sejam originados conflitos temporais que não possam ser eliminados a seguir, através de re-escalonamentos de pedidos. Em caso de se originarem conflitos temporais o agente deve interagir com clientes ou fornecedores, trocando pedidos de re-escalonamento que sejam adequados para eliminação dos conflitos (os pedidos de re-escalonamento podem ou não ser aceites pelos agentes gestores destinatários e resultarão, no caso negativo, em tentativas falhadas de re-escalonamento do agente que os envia, persistindo para ele, os conflitos de capacidade). Para cada conflito de capacidade envolvendo um conjunto de tarefas de um agente gestor, se o agente não consegue eliminar o conflito pelo procedimento descrito até aqui, deverá desfazer o escalonamento de uma ou mais dessas tarefas, cancelar os respectivos pedidos do cliente e a fornecedores e recomeçar o procedimento de eliminação de conflitos de capacidade. Isto deverá ser repetido até que as soluções estabelecidas não envolvam conflitos de capacidade.

A fase 3 durará, para cada problema de escalonamento, até ao momento em que todos os pedidos locais do problema são satisfeitos (*i.e.*, até que o tempo actual seja igual ao maior valor de tempo de pedido local).

O procedimento de eliminação de conflitos de capacidade sugerido para a fase 3 é muito geral. Não se prescreve, por exemplo, que tarefa ou tarefas num conflito de capacidade o agente escolhe para re-escalonar, como as vai re-escalonar, que pedidos de re-escalonamento o agente realizará se forem necessários (o que é particularmente importante no caso de a capacidade requerida pelas tarefas se aproximar da, ou exceder a, capacidade disponível do nó do agente). Poderão existir múltiplas escolhas, aparentemente boas (*i.e.*, escolhas que não aumentem o número de conflitos) para o agente individualmente mas algumas podem mais tarde vir a revelar-se más escolhas para rede (por não evitarem o aparecimento de conflitos noutros agentes), havendo necessidade de retroceder de uma solução estabelecida. É necessário um mecanismo de coordenação entre os agentes para escalonamento baseado na capacidade, que

permita conduzir cada agente para escolhas boas *para o agente e para a rede*, de uma forma rápida.[20] Este aspecto foi relegado para trabalho futuro.

Passa-se de seguida aos detalhes do mecanismo de coordenação para escalonamento temporal.

4.3.6.4 Escalonamento Temporal na Fase 1

Para descrever a informação que os agentes de uma rede devem trocar entre si, para realizarem o mecanismo de coordenação para escalonamento temporal, vai fazer-se uso de exemplos de casos simulados. Os casos referem-se a um problema de escalonamento introduzido na rede da Figura 4-2, por um pedido global do exterior à rede $\mathbb{d}_{1,14}^{0,14}$, um pedido de produto p_1, que é encaminhado pelo agente supervisor da rede ao agente de retalho g_{14}. Para além do agente de retalho g_{14}, estão envolvidos na resolução do problema os seguintes agentes gestores:

- o agente acumulador g_1, fornecedor de g_{14} para o produto p_1;
- o agente transportador g_4, fornecedor de g_1 para o produto p_1;
- o agente produtor g_7, fornecedor de g_4 para o produto p_1;
- o agente produtor g_8, fornecedor de g_7 para o produto p_3 (p_3 é um componente de p_1);
- o agente produtor g_9, fornecedor de g_7 para o produto p_4 (p_4 é um componente de p_1);
- o agente produtor g_{11}, fornecedor de g_9 para o produto p_7 (p_7 é um componente de p_4);
- o agente transportador g_{12}, fornecedor de g_9 para o produto p_8 (p_8 é um componente de p_4).

Adicionalmente, estão envolvidos também os seguintes agentes de matéria-prima:

- o agente de matéria-prima g_{17}, fornecedor de g_8 para o produto p_6 (p_6 é um componente de p_3);
- o agente de matéria-prima g_{18}, fornecedor de g_{11} para o produto p_{10} (p_{10} é um componente de p_7);
- o agente de matéria-prima g_{19}, fornecedor de g_{12} para o produto p_8.

Durante a resolução do problema as tarefas que têm de ser escalonadas são as do processo $\mathbb{RT}_{1,14}$, representado na Figura 4-2-c). Destas, as tarefas de capacidade são: $\mathbb{O}_{1,14}^{1}$ (do agente g_1), $\mathbb{O}_{1,14}^{4}$ (do agente g_4), $\mathbb{O}_{1,14}^{7}$ (do agente g_7), $\mathbb{O}_{1,14}^{8}$ (do agente g_8), $\mathbb{O}_{1,14}^{9}$ (do agente g_9), $\mathbb{O}_{1,14}^{11}$ (do agente g_{11}) e $\mathbb{O}_{1,14}^{12}$ (do agente g_{12}). Nas simulações ignoram-se restrições de capacidade e as durações das tarefas de capacidade são fixas (1, 2, 3, 2, 3, 3 e 1 unidades de tempo para as tarefas dos agentes g_1, g_4, g_7, g_8, g_9, g_{11} e g_{12}, respectivamente). Também as folgas internas para os agentes processadores (produtores e transportadores) são fixas.[21]

Em cada figura representam-se: a) os dados de entrada de um problema, incluindo as durações das tarefas (d) e as folgas internas dos agentes gestores (fim e fij), e b) o escalonamento na forma de um gráfico de Gantt do processo de rede. As zonas riscadas na horizontal entre as áreas de dados de cada agente em a) pretendem sugerir as relações de cliente-fornecedor. No escalonamento em b) é usada uma linha temporal para cada tarefa de agente gestor. Para cada tarefa representam-se o intervalo da tarefa, a cor azul, os intervalos de horizonte temporal

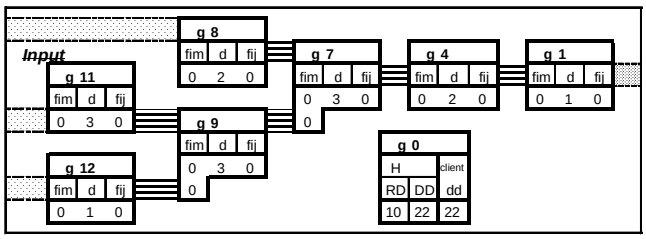
---

[20] Por exemplo, em [Sycara 1991a] e [Sadeh 1994] descrevem-se abordagens ao escalonamento que se baseiam em escalonar prioritariamente tarefas em recursos por cuja capacidade há maior competição.

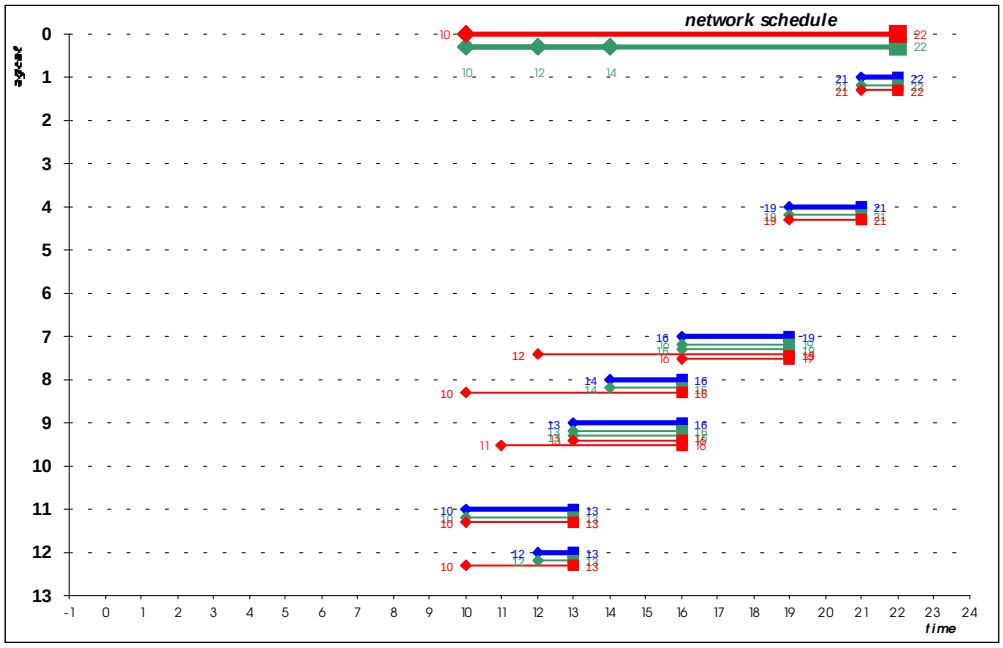
[21] Relativamente à tarefa do agente acumulador, uma vez que a duração dela é unitária, as folgas internas são nulas.

local (os $\mathbb{h}$'s) com cada fornecedor, a cor verde e os intervalos de horizonte temporal (os $\mathbb{H}$'s) com cada fornecedor, a cor vermelha. Também são representados para o agente supervisor (e, portanto, para toda a rede) o intervalo de horizonte temporal global, a cor vermelha, e os intervalos entre o tempo do pedido ao agente de retalho e cada um dos tempos dos pedidos dos agentes de matéria-prima (três intervalos, no caso), a cor verde, numa linha temporal distinta.

Suponha-se que, na fase 1 de resolução um problema de escalonamento os agentes gestores introduzem folgas temporais internas mínimas possíveis e o problema é tal que não haverá folga externa na rede (valores de $\mathtt{FEJ}_{i_r,r}$ e $\mathtt{FEM}_{i_r,r}$ nulos). Então, a folga total $\mathtt{FT}_{i_r,r}$ na rede será nula, um caso limite de problema não temporalmente super-constrangido. Este tipo de casos é ilustrado pela simulação da Figura 4-8 para um pedido global do exterior à rede $\mathbb{d}^{0,14}_{1,14}$, com $\mathtt{dd}_{1,14}$=22 e $\mathbb{H}_{1,14}$=<10,22> (*i.e.*, TEMPO($\mathbb{d}^{0,14}_{1,14}$)=22, $\mathtt{RD}_{1,14}$=10 e $\mathtt{DD}_{1,14}$=22). O caminho de rede contendo as tarefas cujas folgas totais são nulas é, então, o *caminho crítico* do processo de rede $\mathbb{RT}_{i_r,r}$.



a) Dados de entrada do problema.



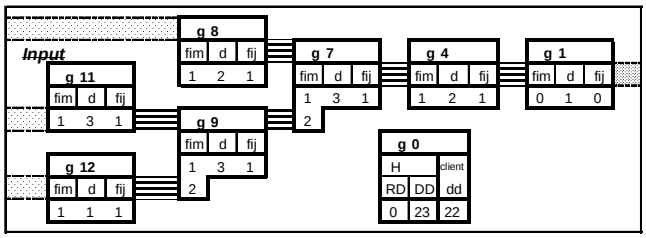
b) Escalonamento.

Figura 4-8- Simulação do processo $\mathbb{RT}_{1,14}$ originado pelo pedido $\mathbb{d}^{0,14}_{1,14}$ (ver Figura 4-2), com $\mathtt{dd}_{1,14}$=22 e $\mathbb{H}_{1,14}$=<10,22>. Neste caso a folga total na rede é 0. O caminho crítico do processo contém as tarefas dos agentes gestores g_1, g_4, g_7, g_9 e g_{11} (as tarefas de capacidade $\mathbb{O}^{1}_{1,14}$, $\mathbb{O}^{4}_{1,14}$, $\mathbb{O}^{7}_{1,14}$, $\mathbb{O}^{9}_{1,14}$ e $\mathbb{O}^{11}_{1,14}$), que percepcionam este valor de folga total.

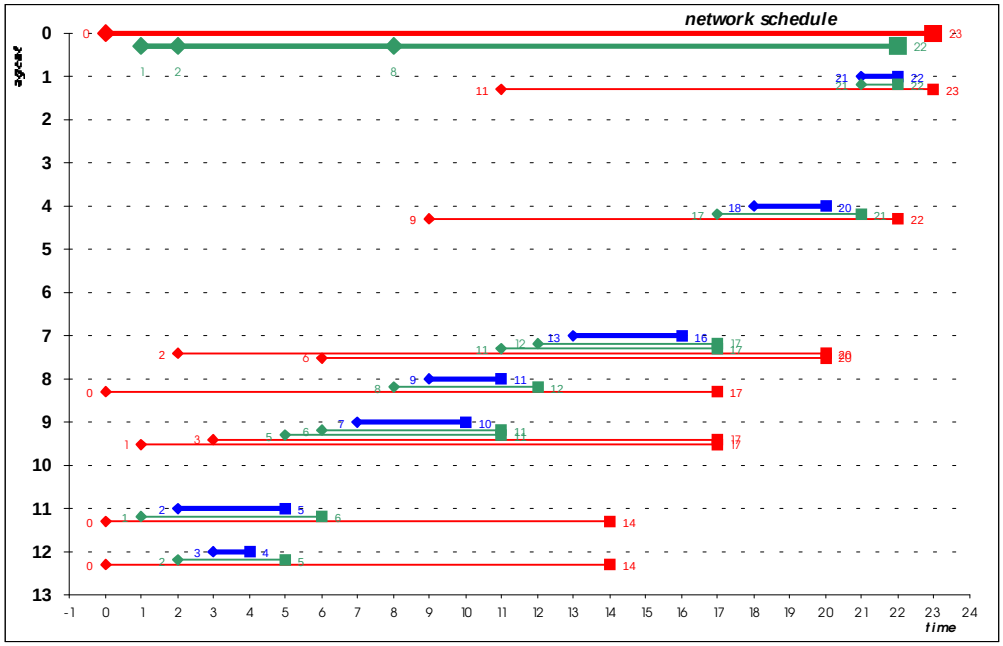
Se cada agente gestor conhecesse os limites do intervalo de horizonte temporal da sua tarefa (o mais restrito dos intervalos de horizonte temporal com cada um dos seus fornecedores) poderia calcular a folga total, subtraindo a duração da tarefa à duração daquele intervalo. No exemplo da Figura 4-8 os agentes g_1, g_4, g_7, g_9 e g_{11} percepcionariam folgas totais nulas, e iguais à folga total da rede, pois as tarefas destes agentes estão no caminho crítico do processo de rede.

Existindo folga externa positiva, existirá sempre um *caminho quase crítico*, contendo tarefas de agentes cujas folgas totais são iguais à folga total da rede; as restantes tarefas terão folgas totais maiores que a folga total da rede. Este tipo de casos não temporalmente super-constrangidos, é ilustrado pela simulação da Figura 4-9 para um pedido $\mathbb{d}^{0,14}_{1,14}$, com $dd_{1,14}$=22 e $\mathbb{H}_{1,14}$=<0,23> (*i.e.*, TEMPO($\mathbb{d}^{0,14}_{1,14}$)=22, $RD_{1,14}$=0 e $DD_{1,14}$=23).

Se cada agente gestor conhecesse os limites do intervalo de horizonte temporal da sua tarefa poderia calcular a folga total. No exemplo da Figura 4-9 os agentes g_1, g_4, g_7, g_9 e g_{11} percepcionariam folgas totais iguais à folga total da rede (igual a 11 unidades de tempo), pois as tarefas destes agentes estão no caminho quase crítico do processo de rede.



a) Dados de entrada do problema.



b) Escalonamento.

Figura 4-9- Simulação do processo $\mathbb{RT}_{1,14}$ originado pelo pedido $\mathbb{d}^{0,14}_{1,14}$ (ver Figura 4-2), com $dd_{1,14}$=22 e $\mathbb{H}_{1,14}$=<0,23>. Neste caso a folga total na rede é 11. O caminho quase crítico do processo contém as tarefas dos agentes gestores g_1, g_4, g_7, g_9 e g_{11} (as tarefas de capacidade $\mathbb{O}^{1}_{1,14}$, $\mathbb{O}^{4}_{1,14}$, $\mathbb{O}^{7}_{1,14}$, $\mathbb{O}^{9}_{1,14}$ e $\mathbb{O}^{11}_{1,14}$), que percepcionam este valor de folga total.

Assumindo que os agentes podem conhecer os limites do intervalo de horizonte temporal da sua tarefa, se o problema de escalonamento for temporalmente super-constrangido existirá pelo menos um agente gestor que percepciona uma folga total negativa. Nesse caso, cada agente nessas condições deve rejeitar o pedido do cliente e cancelar os pedidos aos fornecedores. Estas acções têm como resultado a rejeição do pedido global do exterior à rede (bem como o cancelamento de todos os pedidos globais da rede ao exterior) e evitam tentativas de estabelecimento de uma solução fatalmente condenadas ao fracasso, pois o problema não tem solução.

Para que um agente g_k possa determinar os limites do intervalo de horizonte temporal $RD^{k}_{i_r,r}$ e $DD^{k}_{i_r,r}$, propõe-se que sejam propagados valores de folgas externas de agente para agente.

Estes valores de folga serão incluídos como conteúdo das mensagens de pedido e de aceitação de pedido.[22] Cada agente gestor envolvido na satisfação do mesmo pedido global do exterior à rede envia, em cada mensagem de pedido a cada fornecedor g_j, o valor de folga jusante-montante apropriado $FJM^{k,j}_{i_r,r}$. O agente fornecedor interpretará este valor como sendo o valor da sua folga externa a jusante $FEJ^{j}_{i_r,r}$. De um modo simétrico, cada agente gestor g_k envia, na mensagem de aceitação de pedido ao cliente g_i, o valor de folga montante-jusante $FMJ^{k,i}_{i_r,r}$. O agente cliente interpretará este valor como sendo o valor da sua folga externa a montante $FEM^{i,k}_{i_r,r}$ com o fornecedor g_k.

Cada agente g_k pode então calcular $RD^{k}_{i_r,r}$ e $DD^{k}_{i_r,r}$ através de (ver Tabela 4-5):

$$DD^{k}_{i_r,r} = dd^{k}_{i_r,r} + FEJ^{k}_{i_r,r}$$

$$RD^{k,j}_{i_r,r} = rd^{k,j}_{i_r,r} - FEM^{k,j}_{i_r,r}$$

$$RD^{k}_{i_r,r} = \underset{j=1,\ldots,Nf^{k}_{i_r,r}}{MAX} \left(RD^{k,j}_{i_r,r} \right)$$

Para agentes acumuladores apenas um cálculo é necessário para $RD^{k}_{i_r,r}$, já que uma tarefa de acumulação tem apenas um fornecedor (idem para agentes transportadores e tarefas de transporte). Os valores de $RD^{k}_{i_r,r}$ e $DD^{k}_{i_r,r}$ permitem calcular a folga total do seguinte modo:

$FT^{k}_{i_r,r} = DD^{k}_{i_r,r} - RD^{k}_{i_r,r} - DURAÇÃO(\mathcal{O}^{k}_{i_r,r})$ para agentes processadores,

$FT^{k}_{i_r,r} = DD^{k}_{i_r,r} - RD^{k}_{i_r,r} - 1$ para agentes acumuladores

Alternativamente, a folga total poderá ser calculada através da igualdade (ver Tabela 4-6):

$$FT^{k}_{i_r,r} = FJ^{k}_{i_r,r} + FM^{k}_{i_r,r}$$

já que $FJ^{k}_{i_r,r}$ depende da folga $FEJ^{k}_{i_r,r}$ (obtida do cliente) e da folga $fij^{k}_{i_r,r}$ (conhecida, uma vez que, para agentes processadores depende do tempo do pedido do cliente e do tempo de fim da tarefa e para agentes acumuladores depende da duração da tarefa) e $FM^{k}_{i_r,r}$ depende das folgas $FEM^{k,j}_{i_r,r}$ (obtidas dos fornecedores) e das folgas $fim^{k,j}_{i_r,r}$ (conhecidas, uma vez que, para agentes processadores dependem dos tempos dos pedidos aos fornecedores e do tempo

[22] Ver a secção 4.4.7, onde os protocolos de interacção entre os agentes são descritos.

de início da tarefa e para agentes acumuladores a única folga interna depende da duração da tarefa).

Com os valores de $RD^{k,j}_{i_r,r}$ e $DD^{k}_{i_r,r}$ o agente pode calcular a folga total $FT^{k}_{i_r,r}$ e saber se o problema é temporalmente super-constrangido. Quando o problema não é temporalmente super-constrangido, estes valores, para além de indicarem os limites temporais a serem tidos em conta para possíveis re-escalonamentos da tarefa do agente ($RD^{k}_{i_r,r} = \underset{j=1,\ldots,Nf^{k}_{i_r,r}}{MAX}(RD^{k,j}_{i_r,r})$ e $DD^{k}_{i_r,r}$), são também usados como limites temporais para os valores de tempo aceitáveis dos pedidos de re-escalonamento. Quer dizer, se for necessário re-escalonar pedidos, cada valor de $RD^{k,j}_{i_r,r}$ deve ser tomado pelo agente como o limite inferior para o valor de tempo do pedido ao fornecedor g_j e o valor de $DD^{k}_{i_r,r}$ deve ser tomado como o limite superior para o valor de tempo do pedido de cliente. Isto aplica-se tanto a pedidos de re-escalonamento feitos pelo agente ao cliente ou aos fornecedores como a pedidos de re-escalonamento recebidos pelo agente do cliente ou dos fornecedores. O conhecimento dos valores $RD^{k,j}_{i_r,r}$ e $DD^{k}_{i_r,r}$ para o problema de escalonamento, obtido através da partilha de folgas externas entre agentes é, portanto, útil não apenas na fase 1 mas também nas fases 2 e 3 de resolução do problema.

Este mecanismo é simples e não envolve partilha de informação interna explícita,[23] pois os valores de folga (jusante-montante e montante-jusante) que os agentes trocam entre si correspondem a somas de valores de folgas de outros agentes (a jusante e a montante) do agente.

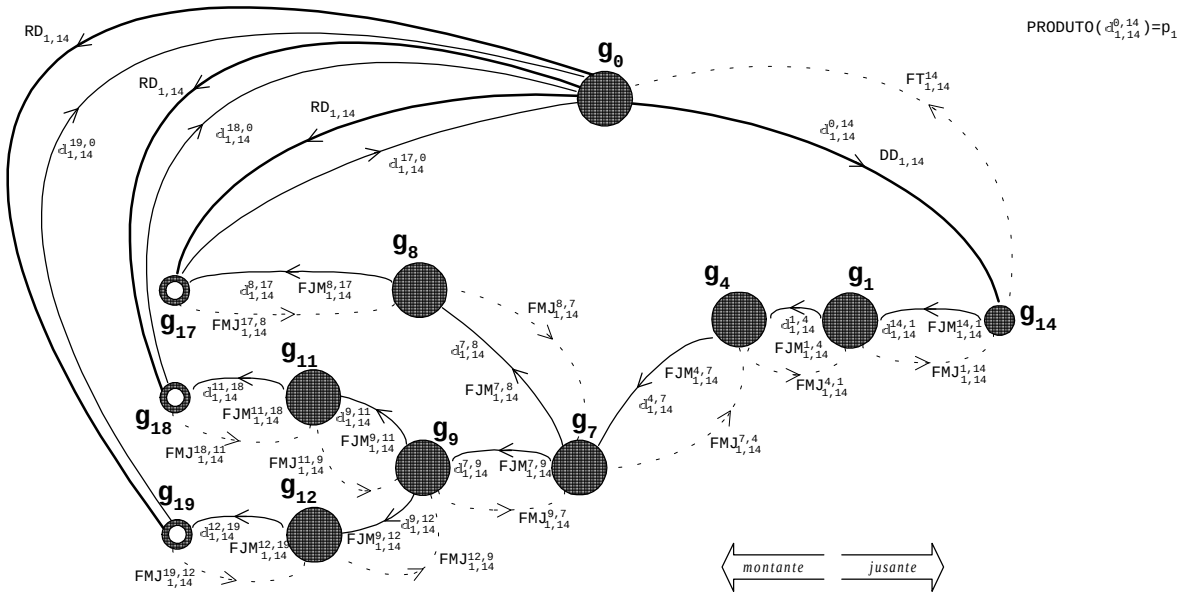


Figura 4-10- Propagação de pedidos e folgas temporais para o problema simulado na Figura 4-8 e na Figura 4-9.

Na Figura 4-10 exemplifica-se o mecanismo de propagação de pedidos e de folgas temporais na fase 1 de resolução de um problema de escalonamento não temporalmente

[23] Se os agentes representam empresas ou organizações independentes a privacidade é um aspecto importante.

super-constrangido. Mostra-se, para o caso do problema simulado na Figura 4-8 e na Figura 4-9, que pedidos e folgas temporais são comunicados entre os agentes.

Neste caso, a fase 1 consiste nos seguintes passos:

1. O agente supervisor transmite o pedido global do exterior à rede, $\mathbb{d}_{1,14}^{0,14}$ e a data limite global, $DD_{1,14}$, ao agente de retalho g_{14};
2. O agente de retalho g_{14} envia o pedido local $\mathbb{d}_{1,14}^{14,1}$ e a folga jusante-montante $FJM_{1,14}^{14,1}$ para o fornecedor g_1, iniciando assim a propagação de pedidos locais e folgas jusante-montante para montante na rede;
3. Cada agente gestor g_k envolvido na satisfação do pedido global do exterior à rede (estão envolvidos os agentes gestores g_1, g_4, g_7, g_8, g_9, g_{11} e g_{12}) envia para os fornecedores mensagens contendo pedidos locais $\mathbb{d}_{i_r,r}^{k,j}$ e folgas $FJM_{i_r,r}^{k,j}$ (setas a cheio, na Figura 4-10). Propagam-se deste modo os pedidos e as folgas jusante-montante para montante na rede;
4. Os agentes de matéria-prima envolvidos, g_{17}, g_{18} e g_{19}, após receberem os pedidos locais e folgas dos agentes clientes g_8, g_{11} e g_{12}, transmitem pedidos globais da rede ao exterior $\mathbb{d}_{1,14}^{17,0}$, $\mathbb{d}_{1,14}^{18,0}$ e $\mathbb{d}_{1,14}^{19,0}$ ao agente supervisor;
5. O agente supervisor transmite aos agentes de matéria-prima g_{17}, g_{18} e g_{19}, a data de lançamento global, $RD_{1,14}$;
6. Os agentes de matéria-prima g_{17}, g_{18} e g_{19}, transmitem aos agentes gestores g_8, g_{11} e g_{12} as folgas $FMJ_{1,14}^{17,8}$, $FMJ_{1,14}^{18,11}$ e $FMJ_{1,14}^{19,12}$;
7. Cada agente gestor g_k envolvido reconhece que o problema não é temporalmente super-constrangido e envia para o cliente mensagens de aceitação contendo a folga $FMJ_{i_r,r}^{k,i}$ (setas a tracejado, na Figura 4-10). Propagam-se deste modo as folgas montante-jusante para jusante na rede;
8. O agente de retalho g_{14}, após receber a folga $FMJ_{1,14}^{1,14}$ do agente g_1, comunica ao agente supervisor a aceitação do pedido global e envia-lhe a folga total $FT_{1,14}^{14}$.

Se o problema for temporalmente super-constrangido isso é será detectado durante o passo 7. Nesse caso, existiriam agentes gestores que rejeitam pedidos de cliente e cancelam pedidos a fornecedores e ter-se-ia, alternativamente:

7. Cada agente gestor g_k envolvido que reconhece o problema como temporalmente super-constrangido envia uma mensagem de rejeição de pedido do cliente e mensagens de cancelamento de pedidos aos fornecedores para cada pedido de fornecimento que foi aceite (os restantes agentes envolvidos procedem inicialmente da mesma forma que anteriormente descrito para o passo 7). Também, se um cliente cancela um pedido antes aceite pelo agente, o agente deve cancelar os pedidos aceites por fornecedores; adicionalmente, se um ou mais dos fornecedores cancelam pedidos que aceitaram do agente, o agente deve cancelar os restantes pedidos a fornecedores e rejeitar o pedido de cliente (caso ainda não lhe tenha dado resposta) ou cancelá-lo (caso já o tenha antes aceite);[24]

[24] Neste passo 7 poderão ocorrer mensagens de cancelamento mútuo do mesmo pedido entre um par de agentes.

8. O agente de retalho g_{14} (que recebeu uma rejeição do fornecedor g_1) comunica ao agente supervisor a rejeição do pedido global do exterior e os agentes de matéria-prima g_{17}, g_{18} e g_{19} (que receberam cancelamentos dos clientes g_8, g_{11} e g_{12}, respectivamente) comunicam ao agente supervisor o cancelamento dos respectivos pedidos globais ao exterior.

Serão definidos, mais adiante, protocolos de interacção entre os agentes, no contexto dos quais se enquadram a propagação de pedidos locais e folgas através da rede.[25]

4.3.6.5 Escalonamento Temporal na Fase 2

Nesta fase há que eliminar conflitos temporais que existam na solução estabelecida na fase anterior. Dadas as regras de identificação de tipos de conflitos temporais 1, 2, 3 e 4,[26] apresentam-se, a seguir, exemplos ilustrativos de casos tipo de conflitos abrangidos pelas regras. Adicionalmente, propõem-se os procedimentos gerais de tratamento para eliminação apropriados a cada caso de conflito para um agente gestor genérico g_k, para os casos em que g_k é um agente processador ou é um agente acumulador.

Para cada caso de conflito temporal detectado através de uma das regras 1, 2, 3 ou 4 podem, em geral, existir várias opções para eliminação do conflito. No entanto, cada um dos procedimentos de tratamento aqui propostos descreve um *tratamento minimal* apropriado ao conflito detectado. Por tratamento minimal entende-se um conjunto de acções de re-escalonamento (re-escalonamentos de pedidos, re-escalonamento da tarefa ou ambos) envolvendo os *deslocamentos no tempo mínimos necessários* para eliminar o conflito temporal detectado sem originar outros conflitos temporais.

Recorda-se que, para o caso dos agentes acumuladores, as folgas `fij`, `fim`, `FJ` e `FM` empregues são de folgas não simétricas (diferentes das expostas na Tabela 4-6 e diferentes conforme se trata da aplicação das regras 1 e 3, ou da aplicação das regras 2 e 4), de acordo com o que foi definido.[26]

Os exemplos baseiam-se em casos de hipotéticos para o agente processador g_7 e o agente acumulador g_1 da rede da Figura 4-2, para um problema de escalonamento introduzido por um pedido global do exterior à rede $\mathbb{d}_{i,14}^{0,14}$, dirigido ao agente de retalho g_{14} (à semelhança do que acontece nos exemplos de simulações até aqui mostrados, para g_7 o cliente é g_4 e os fornecedores são g_8 e g_9; para g_1 o cliente é g_{14} e o fornecedor é g_4).

A representação gráfica para cada exemplo de cada caso é semelhante à usada na secção 4.3.5. É representado um escalonamento preexistente de um agente individual (um escalonamento contendo um conflito temporal) e o escalonamento resultante daquele após aplicação do tratamento adequado. Cada escalonamento é representado através de uma linha temporal junto à qual se mostram a tarefa do agente e os pedidos a jusante (do cliente) e a montante (aos fornecedores). Os intervalos de horizonte temporal, a cor vermelha, e de horizonte temporal local com cada fornecedor, a cor verde, são também mostrados. O intervalo de uma tarefa é representado por um rectângulo limitado por círculos indicando os eventos de fim e de início da tarefa. Este intervalo é representado a cor azul, com excepção da parte que está violando uma restrição temporal (se for o caso), que é representada a cor vermelha. Pedidos são representados por círculos a negro, com a excepção de pedidos que violam uma restrição temporal num conflito que esteja em questão, caso em que são representados a vermelho;

[25] Ver a secção 4.4.7.
[26] Ver a secção 4.3.6.2.

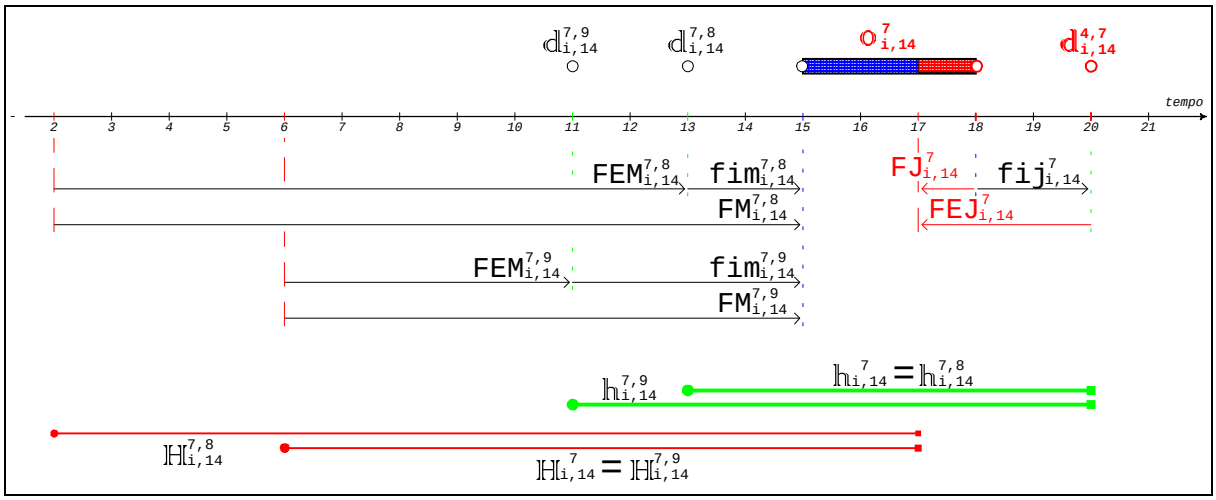
adicionalmente, pedidos que após parte do procedimento de tratamento adequado ficam (temporariamente) violando uma restrição temporal são assinalados a verde. Folgas positivas são representadas por setas a negro e orientadas no sentido positivo da linha temporal e folgas negativas a cor vermelha e no sentido inverso. Todos estes elementos são etiquetados por símbolos identificadores com a cor respectiva. Veja-se, como exemplo, a Figura 4-11.

No texto, os símbolos t_s e t_e são usados como abreviaturas para os tempos de início e de fim de uma tarefa em questão (*i.e.*, para uma tarefa $\mathbb{O}^{k}_{i_r,r}$ é $t_s = TS(\mathbb{O}^{k}_{i_r,r})$ e $t_e = TE(\mathbb{O}^{k}_{i_r,r})$), respectivamente. Na descrição dos procedimentos de tratamento um símbolo com uma plica significa um parâmetro cujo valor foi alterado (*e.g.*, t_s' é o novo valor, alterado, de t_s).

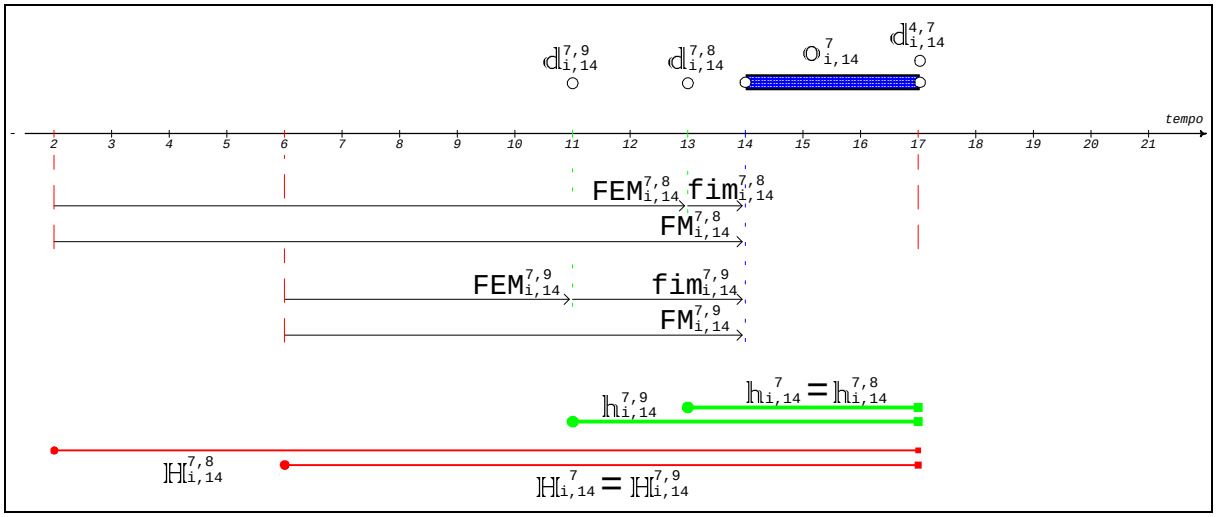
Caso 1 - Conflito Temporal Global a Jusante

Este caso é detectado pela regra 1 se as folgas a jusante e externa a jusante são negativas ($FJ^{k}_{i_r,r}<0$ e $FEJ^{k}_{i_r,r}<0$, para um agente gestor g_k).

Para agentes processadores exemplificam-se duas situações possíveis abrangidas pelo caso 1 na Figura 4-11 (em que não são necessários re-escalonamentos de pedidos a fornecedores) e na Figura 4-12 (em que há necessidade de re-escalonamentos de pedidos a fornecedores).



a) Escalonamento com conflito temporal global a jusante.

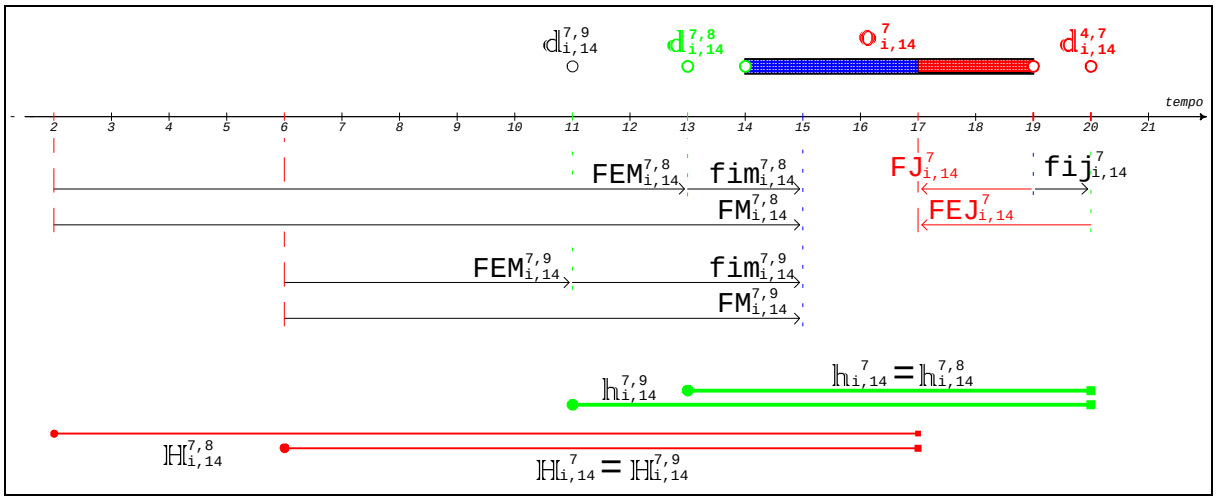


b) Escalonamento após tratamento minimal.

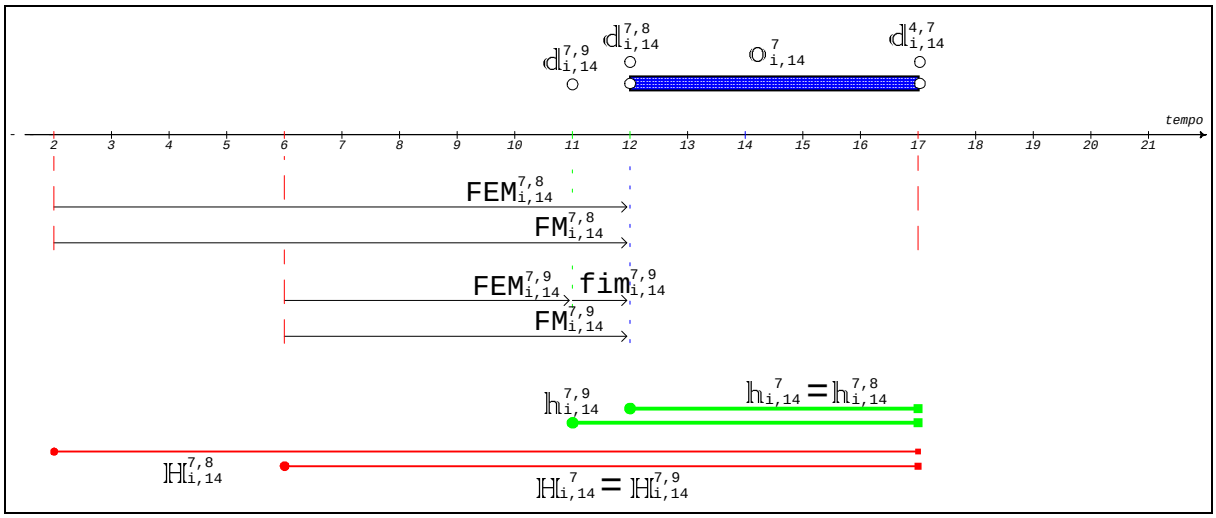
Figura 4-11- Caso 1 de conflito temporal (conflito temporal global a jusante) para agentes processadores. Neste exemplo não há necessidade de re-escalonamento de pedidos a fornecedores.

Para a situação na Figura 4-11-a) tem-se: $DD^{7}_{i,14}=17$, $RD^{7,8}_{i,14}=2$, $RD^{7,9}_{i,14}=6$, $dd^{7}_{i,14}=20$, $rd^{7,8}_{i,14}=13$, $rd^{7,9}_{i,14}=11$, $t_e=18$, $t_s=15$ ($DURAÇÃO(\mathbb{O}^{7}_{i,14})=3$), resultando nos seguintes valores de folga: $FJ^{7}_{i,14}=-1$ e $FEJ^{7}_{i,14}=-3$. Para a situação na Figura 4-12-a) tem-se: $DD^{7}_{i,14}=17$, $RD^{7,8}_{i,14}=2$, $RD^{7,9}_{i,14}=6$, $dd^{7}_{i,14}=20$, $rd^{7,8}_{i,14}=13$, $rd^{7,9}_{i,14}=11$, $t_e=19$, $t_s=14$

(DURAÇÃO($\mathbb{O}^{7}_{i,14}$)=5), resultando nos seguintes valores de folga: $FJ^{7}_{i,14}$=-2 e $FEJ^{7}_{i,14}$=-3.



a) Escalonamento com conflito temporal global a jusante.



b) Escalonamento após tratamento minimal.

Figura 4-12- Caso 1 de conflito temporal (conflito temporal global a jusante) para agentes processadores. Neste exemplo há necessidade de re-escalonamento de pedidos a fornecedores.

O tratamento minimal consiste num re-escalonamento para trás da tarefa, um re-escalonamento para trás do pedido do cliente e, se necessário, re-escalonamentos para trás de pedidos a fornecedores. A seguinte sequência de passos descreve o tratamento minimal, para um agente processador genérico g_k:

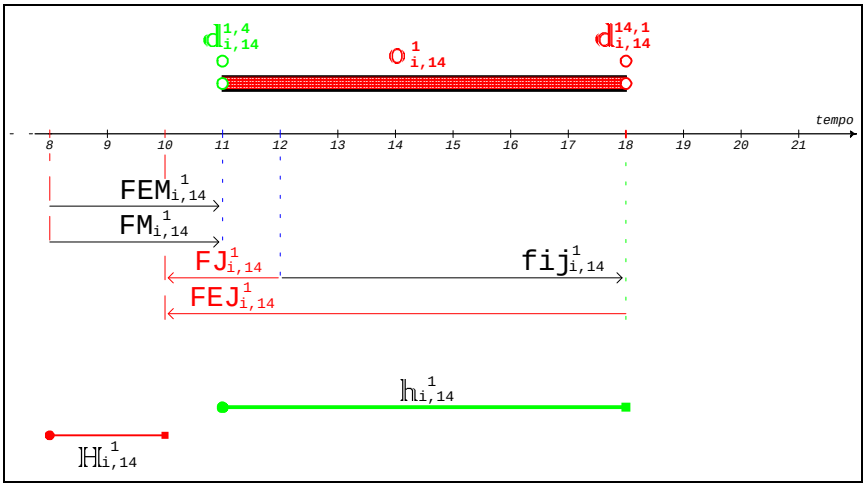
a) Realizar re-escalonamento da tarefa, com $t_e'=DD^{k}_{i_r,r}$ e $t_s'=t_e'-DURAÇÃO(\mathbb{O}^{k}_{i_r,r})$;

b) Enviar pedido de re-escalonamento ao cliente, com $dd^{k}_{i_r,r}'=DD^{k}_{i_r,r}$ (justificado por $FJ^{k}_{i_r,r}$);

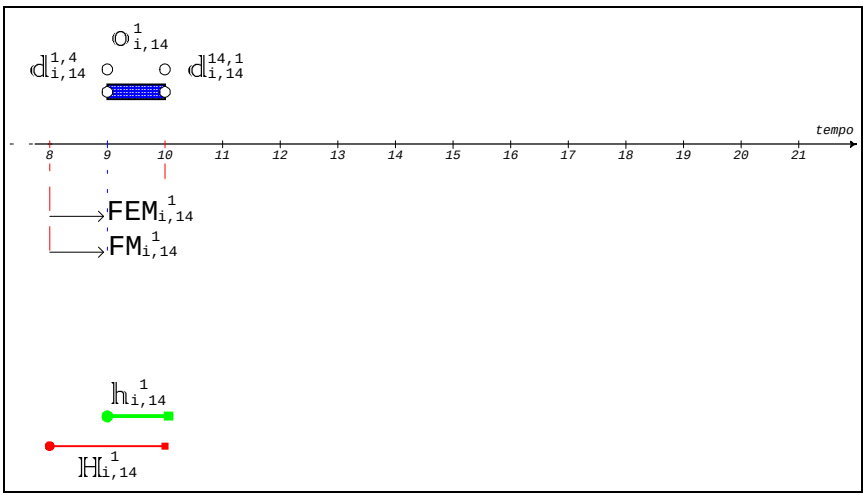
c) Para todo o $rd^{k,j}_{i_r,r}$ ($j=1,\ldots,Nf^{k}_{i_r,r}$) tal que $rd^{k,j}_{i_r,r} > t_s'$, enviar pedido de re-escalonamento ao fornecedor g_j, com $rd^{k,j}_{i_r,r} '= t_s'$ (justificado por $FJ^{k}_{i_r,r}$).

Após o tratamento minimal, o escalonamento na Figura 4-11-a) é convertido no escalonamento na Figura 4-11-b) com $dd^{7}_{i,14}=17$, $t_e=17$ e $t_s=14$ (os valores dos restantes parâmetros permanecem iguais); o escalonamento na Figura 4-12-a) é convertido no escalonamento na Figura 4-12-b) com $dd^{7}_{i,14}=17$, $rd^{7,8}_{i,14}=12$, $t_e=17$ e $t_s=12$ (os valores dos restantes parâmetros permanecem iguais).

Para agentes acumuladores o caso 1 é exemplificado na Figura 4-13.



a) Escalonamento com conflito temporal global a jusante.



b) Escalonamento após tratamento minimal.

Figura 4-13- Caso 1 de conflito temporal (conflito temporal global a jusante) para agentes acumuladores.

Para a situação na Figura 4-13-a) tem-se: $DD^{1}_{i,14}=10$, $RD^{1}_{i,14}=8$, $dd^{1}_{i,14}=t_e=18$, $rd^{1}_{i,14}=t_s=11$ (DURAÇÃO($\mathbb{O}^{1}_{i,14}$)=7), resultando nos seguintes valores de folga: $FJ^{1}_{i,14}=-2$ e $FEJ^{1}_{i,14}=-8$.

O tratamento minimal consiste num re-escalonamento para trás da tarefa, um re-escalonamento para trás do pedido do cliente e um re-escalonamento para trás do pedido do fornecedor. A seguinte sequência de passos descreve o tratamento minimal, para um agente acumulador genérico g_k:

a) Realizar re-escalonamento da tarefa, com $t_e' = DD^k_{i_r,r}$ e $t_s' = t_e' - 1$;

b) Enviar pedido de re-escalonamento ao cliente, com $dd^k_{i_r,r}' = DD^k_{i_r,r}$ (justificado por $FJ^k_{i_r,r}$);

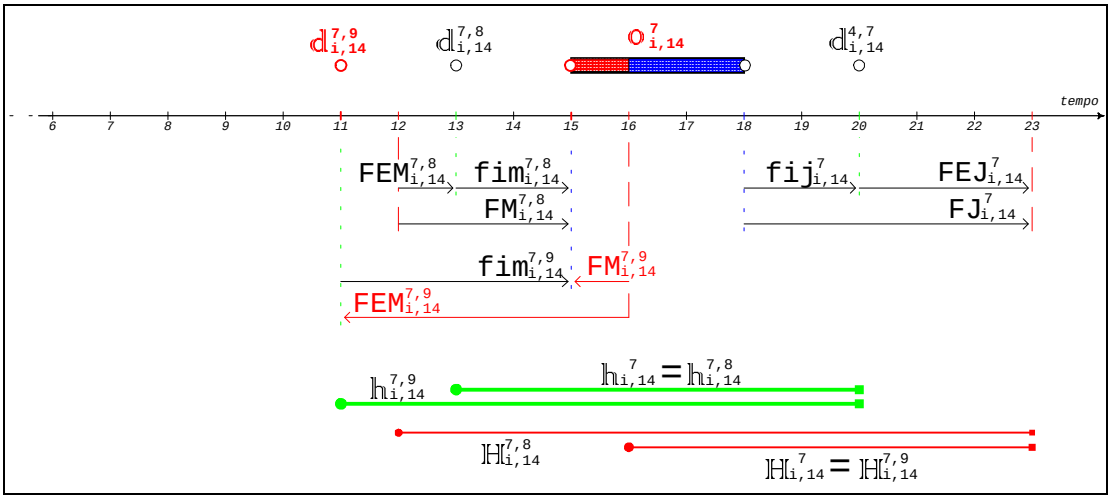
c) Enviar pedido de re-escalonamento ao fornecedor, com $rd^k_{i_r,r}' = t_s'$ (justificado por $FJ^k_{i_r,r}$).

Após o tratamento minimal, o escalonamento na Figura 4-13-a) é convertido no escalonamento na Figura 4-13-b) com $dd^1_{i,14} = t_e = 10$ e $rd^1_{i,14} = t_s = 9$, alterando-se a duração da tarefa para 1.

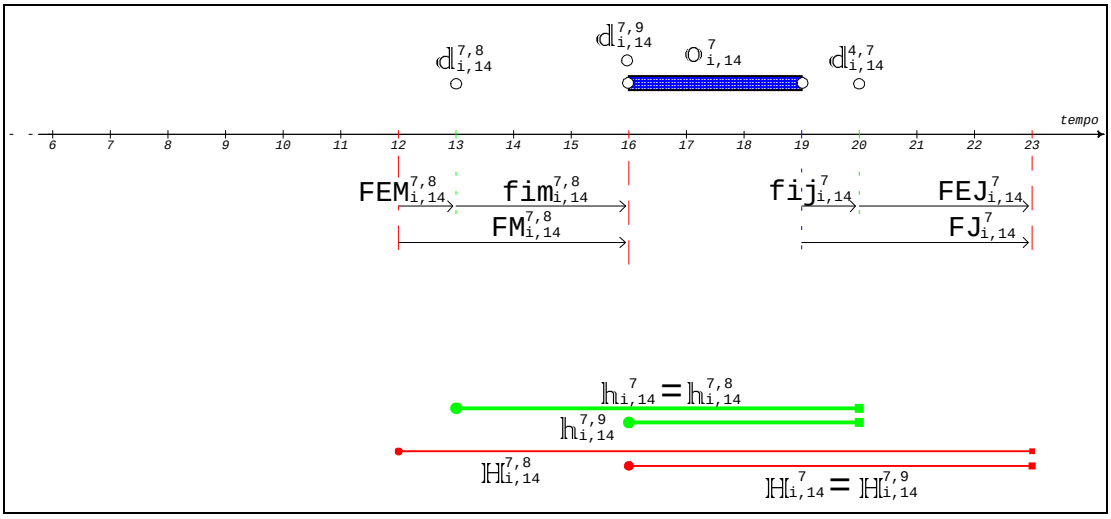
Caso 2 - Conflito Temporal Global a Montante

Este caso é detectado pela regra 2 se as folgas a montante e externa a montante são negativas ($FM^{k}_{i_r,r}<0$ e $FEM^{k}_{i_r,r}<0$, para um agente gestor g_k).

Para agentes processadores exemplificam-se duas situações possíveis abrangidas pelo caso 2 na Figura 4-14 (em que não é necessário re-escalonamento do pedido do cliente) e na Figura 4-15 (em que há necessidade de re-escalonamento do pedido do cliente).



a) Escalonamento com conflito temporal global a montante.

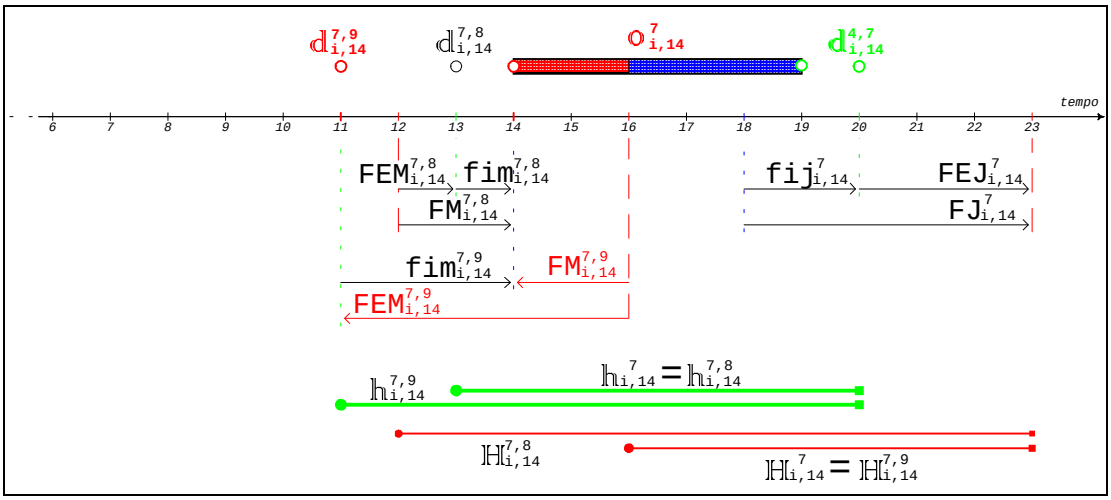


b) Escalonamento após tratamento minimal.

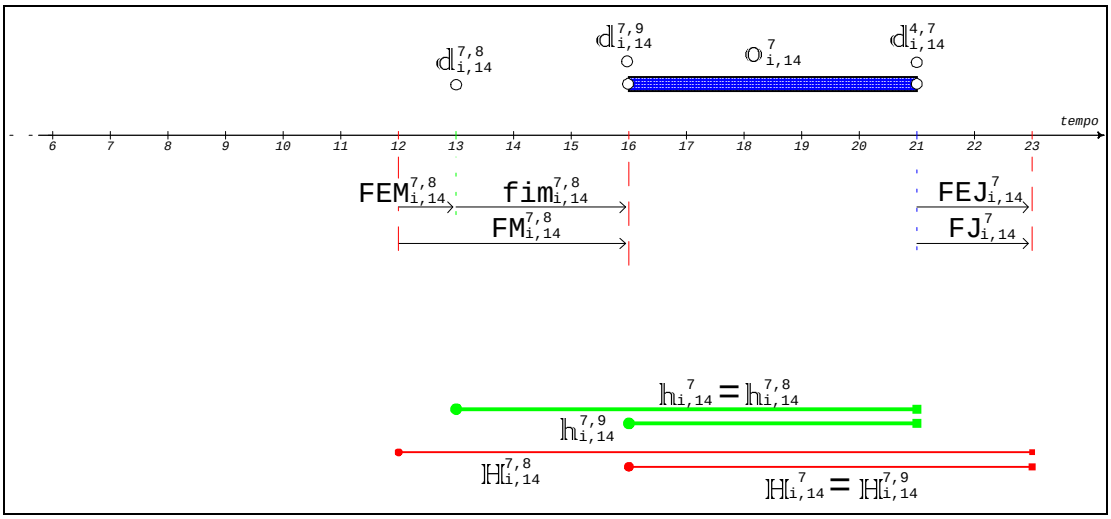
Figura 4-14- Caso 2 de conflito temporal (conflito temporal global a montante) para agentes processadores. Neste exemplo não há necessidade de re-escalonamento do pedido do cliente.

Para a situação na Figura 4-14-a) tem-se: $DD^{7}_{i,14}=23$, $RD^{7,8}_{i,14}=12$, $RD^{7,9}_{i,14}=16$, $dd^{7}_{i,14}=20$, $rd^{7,8}_{i,14}=13$, $rd^{7,9}_{i,14}=11$, $t_e=18$, $t_s=15$ ($DURAÇÃO(\mathcal{O}^{7}_{i,14})=3$), resultando nos seguintes valores de folga: $FM^{7}_{i,14}=-1$ e $FEM^{7}_{i,14}=-5$. Para a situação na Figura 4-15-a) tem-se: $DD^{7}_{i,14}=23$, $RD^{7,8}_{i,14}=12$, $RD^{7,9}_{i,14}=16$, $dd^{7}_{i,14}=20$, $rd^{7,8}_{i,14}=13$, $rd^{7,9}_{i,14}=11$, $t_e=19$, $t_s=14$

(DURAÇÃO($\mathbb{O}^{7}_{i,14}$)=5), resultando nos seguintes valores de folga: $FM^{7}_{i,14}$=-2 e $FEM^{7}_{i,14}$=-5.



a) Escalonamento com conflito temporal global a montante.



b) Escalonamento após tratamento minimal.

Figura 4-15- Caso 2 de conflito temporal (conflito temporal global a montante) para agentes processadores. Neste exemplo há necessidade de re-escalonamento do pedido do cliente.

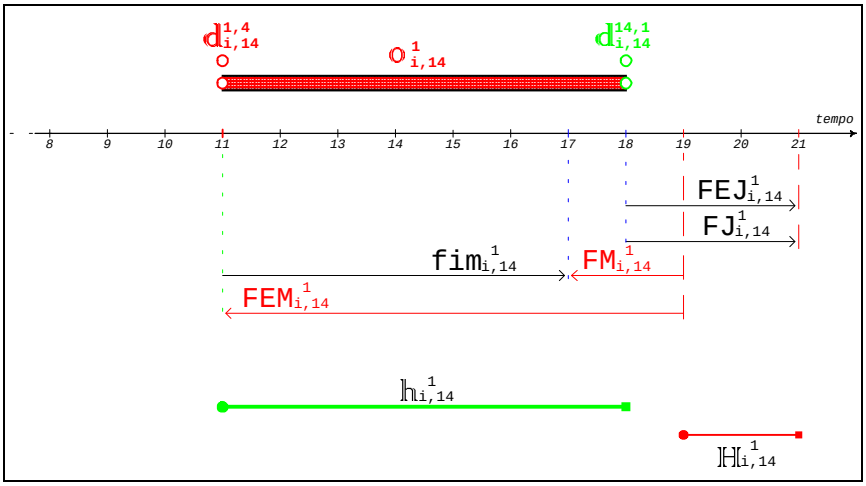
O tratamento minimal consiste num re-escalonamento para a frente da tarefa, um re-escalonamento para a frente dos pedidos a fornecedores que originam folgas externas a montante negativas e, se necessário, um re-escalonamento para a frente do pedido do cliente. A seguinte sequência de passos descreve o tratamento minimal, para um agente processador genérico g_k:

a) Realizar re-escalonamento da tarefa, com $t_s'=RD^{k}_{i_r,r}$ (com $RD^{k}_{i_r,r} = \underset{j=1,\ldots,Nf^{k}_{i_r,r}}{MAX}(RD^{k,j}_{i_r,r})$ e $t_e'=t_s'+$DURAÇÃO($\mathbb{O}^{k}_{i_r,r}$);

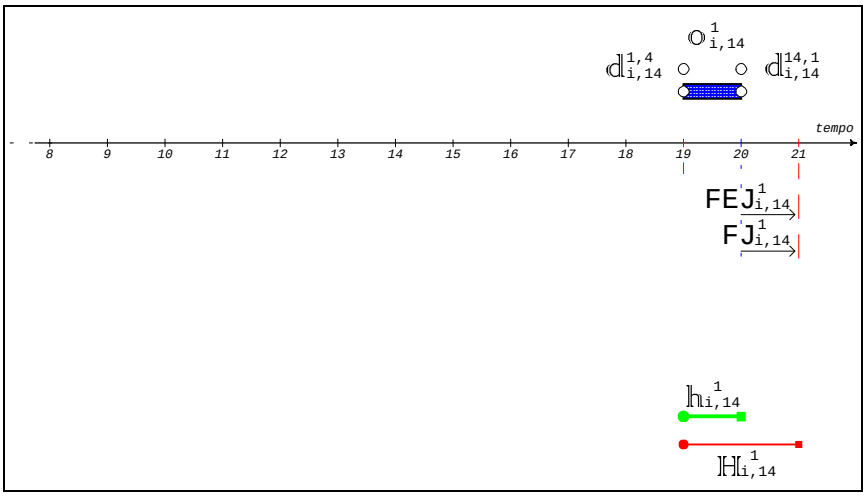
b) Para todo o $rd^{k,j}_{i_r,r}$ ($j=1,\ldots,Nf^{k}_{i_r,r}$) tal que $rd^{k,j}_{i_r,r} < RD^{k,j}_{i_r,r}$, enviar pedido de re-escalonamento ao fornecedor g_j, com $rd^{k,j}_{i_r,r}{}' = RD^{k,j}_{i_r,r}$ (justificado por $FM^{k,j}_{i_r,r}$);

c) Se $dd^{k}_{i_r,r} < t_e'$, enviar pedido de re-escalonamento ao cliente, com $dd^{k}_{i_r,r}{}' = t_e'$ (justificado por $FM^{k}_{i_r,r}$, com $FM^{k}_{i_r,r} = \underset{j=1,\ldots,Nf^{k}_{i_r,r}}{MIN}(FM^{k,j}_{i_r,r})$).

Após o tratamento minimal, o escalonamento na Figura 4-14-a) é convertido no escalonamento na Figura 4-14-b) com $rd^{7,9}_{i,14}=16$, $t_e=19$ e $t_s=16$ (os valores dos restantes parâmetros permanecem iguais); o escalonamento na Figura 4-15-a) é convertido no escalonamento na Figura 4-15-b) com $dd^{7}_{i,14}=21$, $rd^{7,9}_{i,14}=16$, $t_e=21$ e $t_s=16$ (os valores dos restantes parâmetros permanecem iguais).



a) Escalonamento com conflito temporal global a montante.



b) Escalonamento após tratamento minimal.

Figura 4-16- Caso 2 de conflito temporal (conflito temporal global a montante) para agentes acumuladores.

Para agentes acumuladores o caso 2 é exemplificado na Figura 4-16.

Para a situação na Figura 4-16-a) tem-se: $DD^{1}_{i,14}=21$, $RD^{1}_{i,14}=19$, $dd^{1}_{i,14}=t_e=18$, $rd^{1}_{i,14}=t_s=11$ ($DURAÇÃO(\mathbb{O}^{1}_{i,14})=7$), resultando nos seguintes valores de folga: $FM^{1}_{i,14}=-2$ e $FEM^{1}_{i,14}=-8$.

O tratamento minimal consiste num re-escalonamento para a frente da tarefa, um re-escalonamento para a frente do pedido do fornecedor, e um re-escalonamento para a frente do pedido do cliente. A seguinte sequência de passos descreve o tratamento minimal, para um agente acumulador genérico g_k:

a) Realizar re-escalonamento da tarefa, com $t_s'=RD^{k}_{i_r,r}$ e $t_e'=t_s'+1$;

b) Enviar pedido de re-escalonamento ao fornecedor, com $rd^{k}_{i_r,r}'=RD^{k}_{i_r,r}$ (justificado por $FM^{k}_{i_r,r}$);

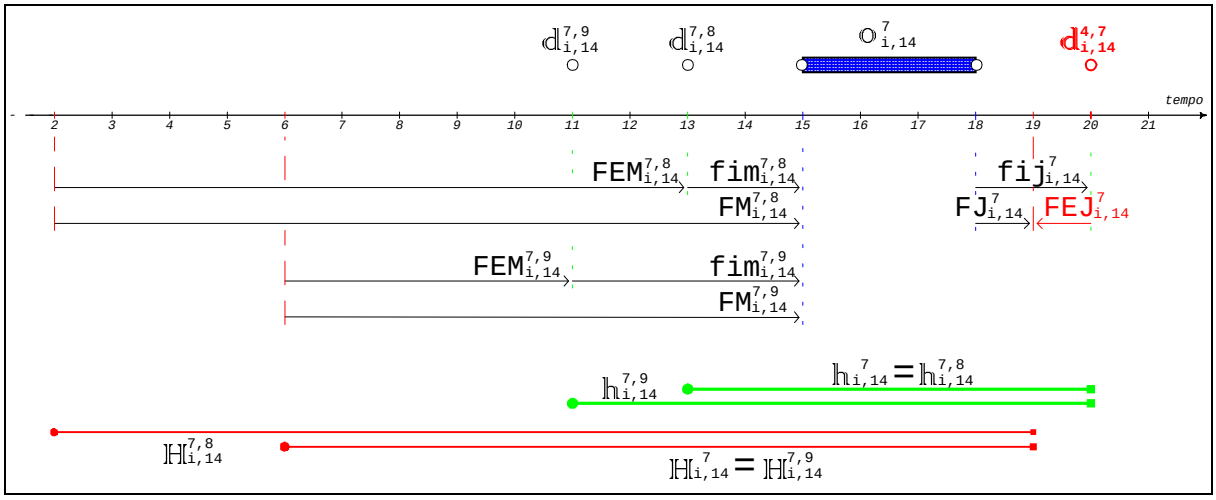
c) Enviar pedido de re-escalonamento ao cliente, com $dd^{k}_{i_r,r}'=t_e'$ (justificado por $FM^{k}_{i_r,r}$).

Após o tratamento minimal, o escalonamento na Figura 4-16-a) é convertido no escalonamento na Figura 4-16-b) com $dd^{1}_{i,14}=t_e=20$ e $rd^{1}_{i,14}=t_s=19$, alterando-se a duração da tarefa para 1.

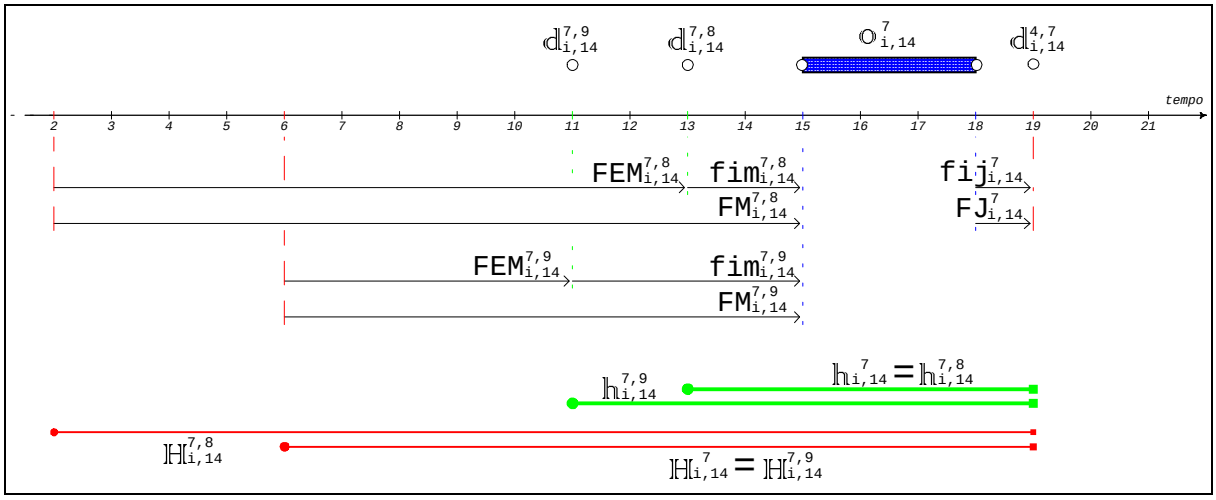
Caso 3 - Conflito Temporal Local Externo a Jusante

Este caso é detectado pela regra 3 se a folga externa a jusante é negativa e a folga a jusante é não negativa ($FEJ^{k}_{i_r,r}<0$ e $FJ^{k}_{i_r,r}\geq 0$, para um agente gestor g_k).

Para agentes processadores o caso 3 é exemplificado na Figura 4-17.



a) Escalonamento com conflito temporal local externo a jusante.



b) Escalonamento após tratamento minimal.

Figura 4-17- Caso 3 de conflito temporal (conflito temporal local externo a jusante) para agentes processadores.

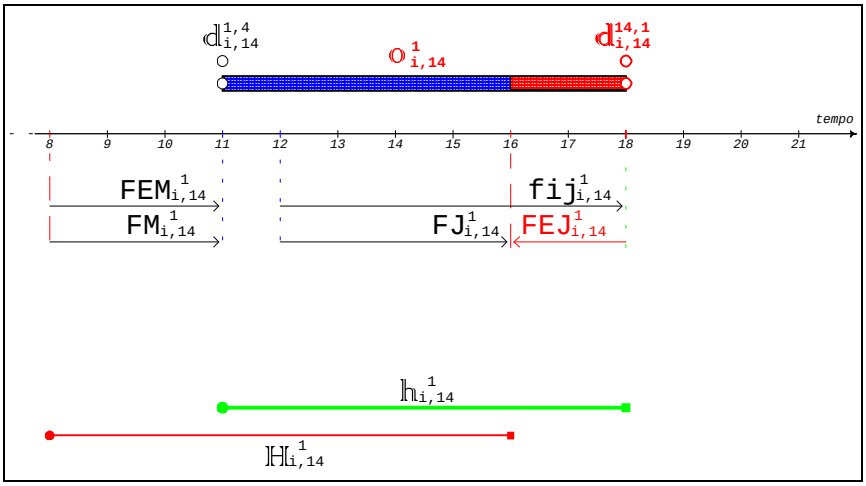
Para a situação na Figura 4-17-a) tem-se: $DD^{7}_{i,14}=19$, $RD^{7,8}_{i,14}=2$, $RD^{7,9}_{i,14}=6$, $dd^{7}_{i,14}=20$, $rd^{7,8}_{i,14}=13$, $rd^{7,9}_{i,14}=11$, $t_e=18$, $t_s=15$ ($DURAÇÃO(\mathbb{O}^{7}_{i,14})=3$), resultando nos seguintes valores de folga: $FJ^{7}_{i,14}=1$ e $FEJ^{7}_{i,14}=-1$.

O tratamento minimal consiste num re-escalonamento para trás do pedido do cliente. A seguinte sequência de passos (com um único passo) descreve o tratamento minimal, para um agente processador genérico g_k:

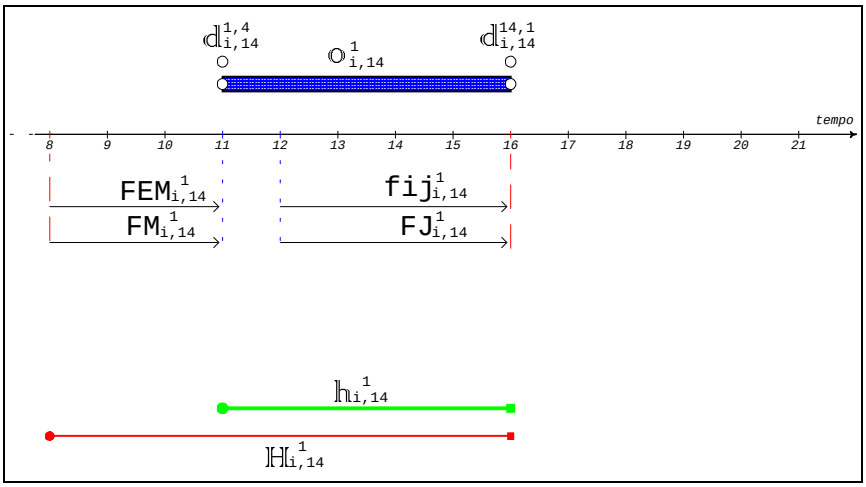
a) Enviar pedido de re-escalonamento ao cliente, com $dd^{k}_{i_r,r}$'=$DD^{k}_{i_r,r}$ (justificado por $FEJ^{k}_{i_r,r}$).

Após o tratamento minimal, o escalonamento na Figura 4-17-a) é convertido no escalonamento na Figura 4-17-b) com $dd^{7}_{i,14}$=19 (os valores dos restantes parâmetros permanecem iguais).

Para agentes acumuladores o caso 3 é exemplificado na Figura 4-18.



a) Escalonamento com conflito temporal local externo a jusante.



b) Escalonamento após tratamento minimal.

Figura 4-18- Caso 3 de conflito temporal (conflito temporal local externo a jusante) para agentes acumuladores.

Para a situação na Figura 4-18-a) tem-se: $DD^{1}_{i,14}$=16, $RD^{1}_{i,14}$=8, $dd^{1}_{i,14}$=t_e=18, $rd^{1}_{i,14}$=t_s=11 (DURAÇÃO($\mathbb{O}^{1}_{i,14}$)=7), resultando nos seguintes valores de folga: $FJ^{1}_{i,14}$=4 e $FEJ^{1}_{i,14}$=-2.

O tratamento minimal consiste numa redução da duração da tarefa e num re-escalonamento para trás do pedido do cliente. A seguinte sequência de passos descreve o tratamento minimal, para um agente acumulador genérico g_k:

a) Realizar re-escalonamento da tarefa, com $t_e' = DD^k_{i_r,r}$;

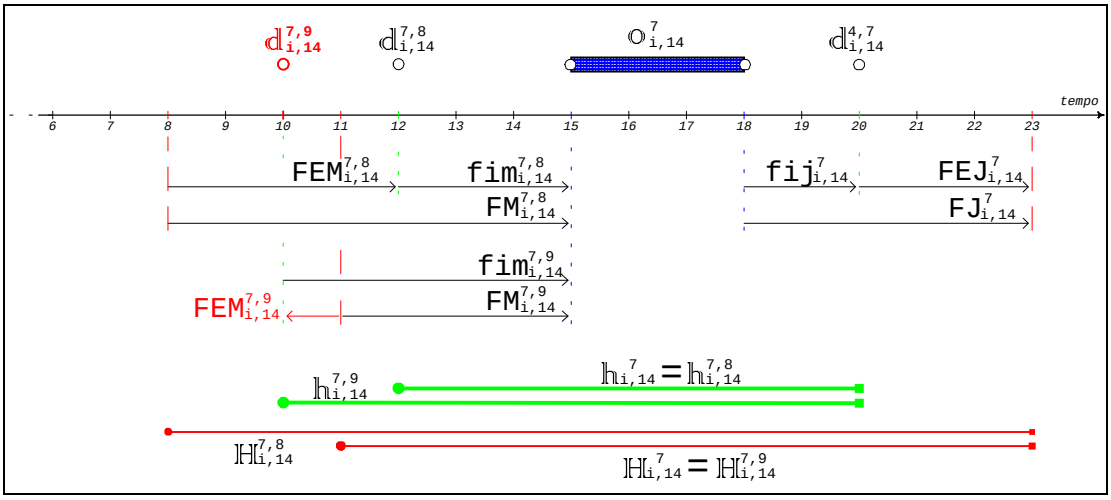
b) Enviar pedido de re-escalonamento ao cliente, com $dd^k_{i_r,r}' = DD^k_{i_r,r}$ (justificado por $FEJ^k_{i_r,r}$).

Após o tratamento minimal, o escalonamento na Figura 4-18-a) é convertido no escalonamento na Figura 4-18-b) com $dd^1_{i,14} = t_e = 16$, alterando-se a duração da tarefa para 5 (os valores dos restantes parâmetros permanecem iguais).

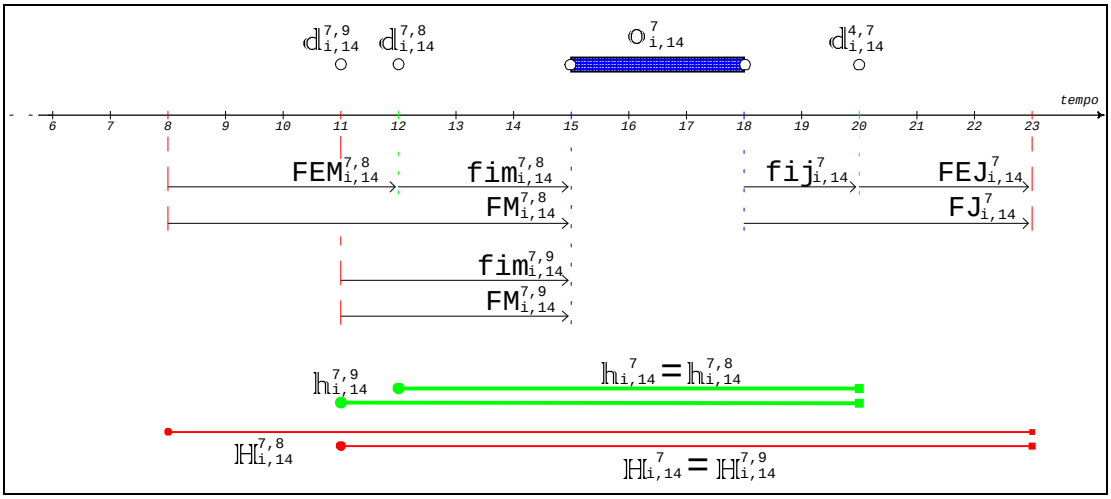
Caso 4 - Conflito Temporal Local Externo a Montante

Este caso é detectado pela regra 4 se a folga externa a montante é negativa e a folga a montante é não negativa ($FEM^{k}_{i_r,r}<0$ e $FM^{k}_{i_r,r}\geq 0$, para um agente gestor g_k).

Para agentes processadores o caso 4 é exemplificado na Figura 4-19.



a) Escalonamento com conflito temporal local externo a montante.



b) Escalonamento após tratamento minimal.

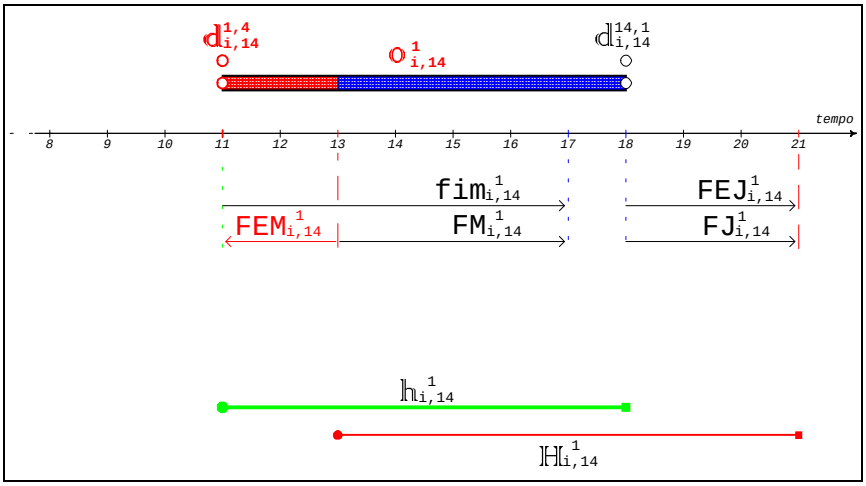
Figura 4-19- Caso 4 de conflito temporal (conflito temporal local externo a montante) para agentes processadores.

Para a situação na Figura 4-19-a) tem-se: $DD^{7}_{i,14}=23$, $RD^{7,8}_{i,14}=8$, $RD^{7,9}_{i,14}=11$, $dd^{7}_{i,14}=20$, $rd^{7,8}_{i,14}=12$, $rd^{7,9}_{i,14}=10$, $t_e=18$, $t_s=15$ ($DURAÇÃO(\mathcal{O}^{7}_{i,14})=3$), resultando nos seguintes valores de folga: $FM^{7}_{i,14}=4$ e $FEM^{7}_{i,14}=-1$.

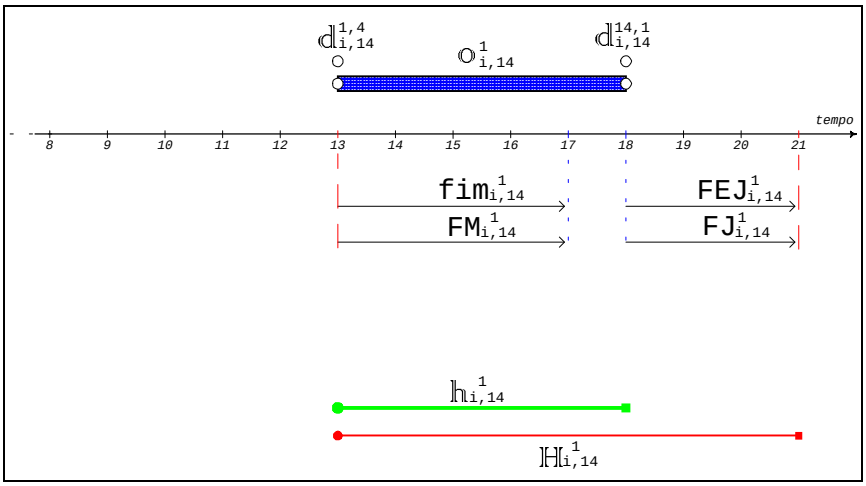
O tratamento minimal consiste num re-escalonamento para a frente dos pedidos a fornecedores que originam folgas externas a montante negativas. A seguinte sequência de passos (com um único passo) descreve o tratamento minimal, para um agente processador genérico g_k:

a) Para todo o $rd^{k,j}_{i_r,r}$ ($j=1,\ldots,Nf^{k}_{i_r,r}$) tal que $rd^{k,j}_{i_r,r}<RD^{k,j}_{i_r,r}$, enviar pedido de re-escalonamento ao fornecedor g_j, com $rd^{k,j}_{i_r,r}{}'=RD^{k,j}_{i_r,r}$ (justificado por $FEM^{k,j}_{i_r,r}$).

Após o tratamento minimal, o escalonamento na Figura 4-19-a) é convertido no escalonamento na Figura 4-19-b) com $rd^{7,9}_{i,14}=11$ (os valores dos restantes parâmetros permanecem iguais).



a) Escalonamento com conflito temporal local externo a montante.



b) Escalonamento após tratamento minimal.

Figura 4-20- Caso 4 de conflito temporal (conflito temporal local externo a montante) para agentes acumuladores.

Para agentes acumuladores o caso 4 é exemplificado na Figura 4-20.

Para a situação na Figura 4-20-a) tem-se: $DD^{1}_{i,14}=21$, $RD^{1}_{i,14}=13$, $dd^{1}_{i,14}=t_e=18$, $rd^{1}_{i,14}=t_s=11$ (DURAÇÃO($\mathbb{O}^{1}_{i,14}$)=7), resultando nos seguintes valores de folga: $FM^{1}_{i,14}=4$ e $FEM^{1}_{i,14}=-2$.

O tratamento minimal consiste numa redução da duração da tarefa e num re-escalonamento para a frente do pedido do fornecedor. A seguinte sequência de passos descreve o tratamento minimal, para um agente acumulador genérico g_k:

a) Realizar re-escalonamento da tarefa, com $t_s' = RD^k_{i_r,r}$;

b) Enviar pedido de re-escalonamento ao fornecedor, com $rd^k_{i_r,r}' = RD^k_{i_r,r}$ (justificado por $FEM^k_{i_r,r}$).

Após o tratamento minimal, o escalonamento na Figura 4-20-a) é convertido no escalonamento na Figura 4-20-b) com $rd^1_{i,14} = t_s = 13$, alterando-se a duração da tarefa para 5 (os valores dos restantes parâmetros permanecem iguais).

4.4 Agente

O presente trabalho, em especial no que respeita ao presente capítulo, é enquadrável na área de Inteligência Artificial Distribuída (IAD), já que se encara a resolução de problemas de escalonamento distribuída por vários agentes autónomos. As definições para *agente*, bem como a caracterização dos agentes, diferem na literatura de IAD. Adopta-se aqui a definição de que um *agente* é um processo computacional autónomo, possivelmente inteligente e possivelmente pro-activo, que está situado num ambiente com o qual interage; a interacção pode incluir acções que modificam o estado do ambiente e comunicação com outros agentes que possam nele existir.[27] A caracterização dos agentes pode fazer-se usando um conjunto mínimo de quatro propriedades, que são a *reactividade*, a *autonomia*, a *pro-actividade* e a *persistência* (ver o Apêndice D). A seguir, caracterizam-se os agentes de uma rede de EE em termos destas e outras propriedades.

1. *Reactividade* - Numa rede de EE, em particular os agentes gestores, reagem a pedidos dos agentes clientes (aceitando-os ou rejeitando-os) e adaptam dinamicamente as soluções locais de escalonamento das suas tarefas de acordo com restrições locais e globais;
2. *Autonomia* - Os agentes têm controlo próprio sobre as suas decisões e acções. Numa rede de EE, cada agente gestor tem uma perspectiva local dos problemas de escalonamento da rede, escalona tarefas no recurso por ele gerido de acordo com essa perspectiva local e podendo fazer uso de preferências individuais de escalonamento,[28] faz pedidos a fornecedores de acordo com necessidades de fornecimento de cada tarefa, pode fazer pedidos de re-escalonamento a clientes e fornecedores se necessário, e decidir aceitar ou rejeitar pedidos de re-escalonamento de clientes e fornecedores, ou decidir cancelar um pedido de cliente anteriormente aceite, se necessário.
3. *Pro-actividade* - Um agente é pro-activo se actua guiado por objectivos, actuando não apenas em resposta ao ambiente. Numa rede de EE, todos os agentes procuram cooperar de modo a satisfazer o objectivo global de satisfazer, dentro de certos limites temporais globais dados, cada pedido global do exterior à rede. Para cada agente gestor, o objectivo global traduz-se no objectivo individual de satisfazer cada pedido dos seus clientes dentro de limites temporais locais determinados. No que respeita à actividade de curto prazo de

[27] Esta definição assume à partida que se trata de agentes artificiais.

[28] De acordo com a sequência de três fases para resolução cooperativa de problemas de escalonamento multi-agente proposta na secção anterior, esta possibilidade existe apenas na fase 3.

escalonamento, os agentes são, basicamente, reactivos.[29] No entanto, cada agente pode ter, e fazer uso de, preferências individuais no que respeita à forma como afecta a capacidade disponível do nó físico no escalonamento de tarefas e pode tentar optimizar a utilização da capacidade de acordo com aquelas preferências individuais. Além disso, no que respeita a pedidos previamente aceites, pode enviar pedidos de re-escalonamento;

4. *Persistência* - Um agente é persistente se é um processo em execução de forma temporalmente contínua. Numa rede de EE os agentes persistem ao longo do tempo realizando escalonamento estático, *i.e.*, para um conjunto de tarefas e um horizonte temporal fixos e predeterminados, ou dinâmico, *i.e.*, tarefas a escalonar vão surgindo ao longo do tempo.

Também a *comunicabilidade*, a *aprendizagem*, a *mobilidade*, a *flexibilidade* e o facto de se assumirem ou não como *personagens* são propriedades frequentemente usadas para caracterização dos agentes. A seguir caracterizam-se os agentes de uma rede EE em termos destas propriedades adicionais.

5. *Comunicabilidade* - Um agente tem esta propriedade se comunica com outros agentes (agentes humanos incluídos). Numa rede de EE os agentes comunicam através de mensagens as quais veiculam, basicamente, pedidos;

6. *Aprendizagem* - Diz-se que um agente aprende se é capaz de se adaptar, alterando o seu comportamento com base na experiência prévia. No presente trabalho, os agentes de uma rede de EE apenas realizam a actividade de curto prazo de escalonamento e têm um comportamento "inato" ditado por um conjunto de regras, fixo para cada agente (mas que pode ser diferente de agente para agente), que se traduz no modo como cada agente afecta a sua capacidade disponível às tarefas a realizar. Portanto, os agentes não aprendem. No entanto, em trabalho futuro, a aprendizagem deverá vir a ser considerada, traduzindo-se em aspectos de médio-longo prazo (previsão, planeamento) influenciando os aspectos de curto prazo do escalonamento;

7. *Mobilidade* - Um agente é móvel se é capaz de se transportar de uma máquina para outra. Os agentes de uma rede de EE têm associados locais de rede física bem determinados e não são móveis;

8. *Flexibilidade* - Um agente é flexível se as suas acções não são pré-programadas. As acções individuais de cada agente de uma rede de EE são pré-programadas. No entanto, não pode dizer-se que o resultado global das acções, na perspectiva da rede como um todo,[30] seja pré-programado, dependendo esse resultado também das interacções entre os agentes e do grau de coordenação inter-agente na actividade de escalonamento;

9. *Personagem* - Diz-se que um agente é (ou "incarna") um personagem se tem uma personalidade realista (*i.e.*, credível) e exibe estado emocional. Os agentes de uma rede de EE não têm estas características, quer dizer, não têm estado emocional e o seu comportamento não é afectado por emoções.[31]

---

[29] A pro-actividade será mais proeminente num contexto de médio-longo prazo, mas no presente trabalho apenas a actividade de curto prazo, *i.e.*, o escalonamento, interessa.

[30] Por exemplo, se um pedido do exterior é ou não satisfeito a tempo, pela rede, ou é ou não rejeitado, ou cancelado, mesmo que haja capacidade disponível na rede para o satisfazer, por exemplo.

[31] Um aspecto interessante e que pode ser explorado em trabalho futuro, é o da influência de um componente emocional no comportamento de cada agente, eventualmente diferente de agente para agente. Por exemplo, os agentes podem ser sensíveis ao número de pedidos que lhes são dirigidos pelos clientes ficando mais

Outros aspectos são a *honestidade* e a *cooperatividade* dos agentes. Neste trabalho assume-se que os agentes são honestos em termos da informação que trocam entre si e são cooperantes na actividade de escalonamento multi-agente.

Para caracterizar um agente autónomo empregam-se, adicionalmente, os seguintes parâmetros: *ambiente* em que o agente está imerso, *capacidades sensoriais* (*i.e.*, de percepção do ambiente) do agente, *acções possíveis* do agente, *motivações* e uma *arquitectura para escolha de acções*. Em termos destes parâmetros os agentes de uma rede de EE podem ser descritos do seguinte modo:

- *Ambiente* - O ambiente em que cada agente existe é o de uma rede de EE. O ambiente de uma rede de EE é composto por uma rede física (o nível físico), um conjunto de agentes (o nível virtual) e, implicitamente, o exterior, de onde surgem pedidos. Na rede física existem nós físicos (macro-recursos) de uma rede de produção e distribuição e tarefas que usam a capacidade dos nós físicos. Os agentes realizam escalonamento de tarefas nos nós da rede física;
- *Capacidades sensoriais* - Todos os agentes podem comunicar através de mensagens. Adicionalmente, cada agente de rede de EE responsável por gerir um nó da rede física tem acesso ao estado desse nó;[32]
- *Acções possíveis* - Os agentes de rede podem enviar e receber mensagens a outros agentes. Entre pares de agentes cliente-fornecedor estas mensagens contêm pedidos de produtos ou avisos de aceitação ou rejeição de pedidos e pedidos de re-escalonamento, cancelamento e satisfação de pedidos previamente aceites. Agentes especiais situados nos extremos da rede, denominados *agentes de retalho* e *agentes de matéria-prima*, juntamente com um agente especial, único, o *agente supervisor*, funcionam como fronteira, ou interface, com o exterior, aceitando pedidos do exterior à rede, nos extremos a jusante, ou remetendo pedidos da rede ao exterior, nos extremos a montante. Para além desta *actividade inter-agente*, ou *comunicacional*, existe uma *actividade intra-agente*. Em particular, cada agente gestor tem uma actividade intra-agente que se traduz na gestão interna da interacção com os outros agentes e na criação das tarefas necessárias para tornar disponíveis os produtos pedidos pelos seus clientes (na quantidade e no instante de tempo pedidos) e no escalonamento dessas tarefas através da afectação de capacidade do nó gerido pelo agente;
- *Motivações* - Implicitamente existe a motivação de satisfazer o maior número de pedidos do exterior à rede. Em cada agente gestor isto traduz-se em tentar satisfazer o maior número de pedidos dos seus clientes, respeitando as restrições de escalonamento;
- *Arquitectura para escolha de acções* - Esta arquitectura baseia-se num conjunto de componentes apropriados à gestão da actividade intra-agente, através dos quais o agente representa internamente o estado da interacção com o mundo exterior (conversações com outros agentes) e o estado do nó associado ao agente (o estado de capacidade, no caso dos

---

"nervosos". Poderá haver agentes cujo comportamento se degrade, ou se torne mais eficiente, à medida que aumenta o número de pedidos por unidade de tempo. Será, então, interessante, estudar como varia em eficiência o comportamento global da rede com estes agentes "sensíveis" que reagem de formas diferentes.

[32] Considera-se que o nó associado a um agente é parte integrante do agente e que o ambiente, ou mundo exterior, com o qual o agente interage se limita a um subconjunto dos restantes agentes de rede de EE, com os quais pode comunicar. Alternativamente, poderia considerar-se que o agente e o nó são objectos distintos e que o nó faz parte do ambiente do agente. Neste caso, as capacidades sensoriais do agente abrangeriam não só capacidades de comunicação com outros agentes, mas também capacidades de percepção do estado do nó associado.

agentes gestores, *i.e.*, tarefas escalonadas, capacidade disponível). Inclui-se também, na arquitectura, um conjunto de regras que conduzem a actividade do agente, apelidado de *comportamento*.

Nesta secção propõe-se uma arquitectura apropriada à resolução de problemas de escalonamento multi-agente, para cada classe de agente. De acordo com as tarefas especializadas desempenhadas, os agentes de rede de EE são classificados em:

- *Agente de rede* - É um agente ao qual está associado um nó da rede física (um nó é associado a um único agente e um agente é associado a um único nó). A classe agente de rede especializa-se, dependendo do tipo de nó associado, nas seguintes subclasses:
  - *Agente gestor*, ou *agente de capacidade* - Agente com um nó de capacidade associado;
  - *Agente de retalho* - Agente com um nó de retalho associado;
  - *Agente de matéria-prima* - Agente que com um nó de matéria-prima associado;
- *Agente supervisor*, ou *supervisor* - O *agente supervisor* é um agente único numa rede de EE que introduz novos problemas de escalonamento na rede e actua, juntamente com agentes de retalho e agentes de matéria-prima, como fronteira da rede com o exterior.[33] Ao contrário de outros agentes de rede, o agente supervisor não tem um nó físico associado.

Estas classes de agentes são descritas a seguir.

4.4.1 Agente Gestor

Um *agente gestor*, ou *agente de capacidade*, gere a capacidade de um nó de capacidade da rede física, *i.e.*, um nó processador ou um nó acumulador. Um objecto da classe agente gestor $\mathbb{G}_k$, é descrito por um tuplo de onze elementos:

$<g_k, Cl, Fr, Pcl, Pfr, Msg\text{-}in, Msg\text{-}out, Convs, PE\text{-}Conj, \mathbb{V}_k, Comportamento>$

Os primeiros sete elementos (g_k, Cl, Fr, Pcl, Pfr, $Msg\text{-}in$ e $Msg\text{-}out$) constituem uma *interface* com o mundo exterior do agente (*i.e.*, os restantes agentes da rede[34]), os três seguintes ($Convs$, $PE\text{-}Conj$ e $\mathbb{V}_k$) descrevem o *estado interno* do agente e o último elemento ($Comportamento$) descreve o *comportamento* do agente. Estes elementos estão presentes na arquitectura representada na Figura 4-21 e são, a seguir, descritos.

A interface:

- g_k é o identificador do agente ($g_k \in \mathbb{g}$ e $g_k \neq g_0$);

[33] O agente supervisor também poderia influir na actividade de escalonamento através de introdução de informação de previsão da procura à rede em períodos futuros e de troca de informação relativa a planeamento de médio-longo prazo com os agentes de rede. A actividade de médio-longo prazo referida está, no entanto, reservada para trabalho futuro. O foco de atenção no presente trabalho é a actividade de escalonamento.

[34] No presente trabalho, para os agentes gestores, a visibilidade do mundo exterior limita-se aos agentes clientes e fornecedores, aos quais está ligado por relações de cliente-fornecedor e com os quais pode comunicar. Estes agentes estão identificados no conjunto produzido pela operação `OUTROS` (ver a secção 4.4.1.1).

- $\mathcal{Cl}$ ($\mathcal{Cl}=\{\ldots,g_i,\ldots\}$) é o conjunto de identificadores dos agentes clientes do agente, *i.e.*, dos agentes que gerem nós da rede física clientes do nó gerido pelo agente ($\mathcal{Cl}\subset\mathcal{g}$, $g_k\notin\mathcal{Cl}$ e $g_0\notin\mathcal{Cl}$);
- $\mathcal{Fr}$ ($\mathcal{Fr}=\{\ldots,g_j,\ldots\}$) é o conjunto de identificadores dos agentes fornecedores do agente, *i.e.*, dos agentes que gerem nós da rede física fornecedores do nó gerido pelo agente ($\mathcal{Fr}\subset\mathcal{g}$, $g_k\notin\mathcal{Fr}$ e $g_0\notin\mathcal{Fr}$);
- $\mathcal{Pcl}$ ($\mathcal{Pcl}=\{\ldots,\mathbb{pg}_i,\ldots\}$) é o conjunto de todos os pares produto-local-virtual, $\mathbb{pg}_i=<p_{k_i},g_i>$, que associam cada produto p_{k_i} de saída do nó gerido pelo agente a cada identificador de agente cliente do produto g_i ($p_{k_i}\in\texttt{PRODUTOS}(\mathbb{V}_k)$, $g_i\in\mathcal{Cl}$);
- $\mathcal{Pfr}$ ($\mathcal{Pfr}=\{\ldots,\mathbb{pg}_j,\ldots\}$) é o conjunto de todos os pares produto-local-virtual, $\mathbb{pg}_j=<p_{k_j},g_j>$, que associam cada produto p_{k_j} de entrada do nó gerido pelo agente a cada identificador de agente fornecedor do produto g_j ($p_{k_j}\in\texttt{MATERIAIS}(\mathbb{V}_k)$, $g_j\in\mathcal{Fr}$);
- $\mathcal{Msg\text{-}in}$ ($\mathcal{Msg\text{-}in}=\{\ldots,\mathbb{msg\text{-}in}_x,\ldots\}$) é a *caixa de correio de entrada* do agente. É uma fila de mensagens recebidas e ainda não processadas, de agentes clientes e fornecedores, ordenada pela ordem temporal de recepção das mensagens;
- $\mathcal{Msg\text{-}out}$ ($\mathcal{Msg\text{-}out}=\{\ldots,\mathbb{msg\text{-}out}_y,\ldots\}$) é a *caixa de correio de saída* do agente. É uma fila de mensagens para agentes clientes e fornecedores, ordenada pela ordem temporal de envio das mensagens.

O estado interno do agente:

- $\mathcal{Convs}$ ($\mathcal{Convs}=\{\ldots,\mathbb{conv}^{k,j}_{i_r,r},\ldots,\mathbb{conv}^{i,k}_{i_r,r},\ldots\}$) é o conjunto actual de *conversações* com outros agentes, clientes e fornecedores. Cada conversação $\mathbb{conv}^{i,k}_{i_r,r}$ representa o estado interno do agente no que respeita a um pedido $\mathbb{dl}^{i,k}_{i_r,r}$ de um agente cliente g_i e cada conversação $\mathbb{conv}^{k,j}_{i_r,r}$ representa o estado interno do agente no que respeita a um pedido $\mathbb{dl}^{k,j}_{i_r,r}$, a um agente fornecedor g_j. Cada conversação no conjunto $\mathcal{Convs}$ tem associado um identificador de problema de escalonamento que a associa ao problema de escalonamento acerca do qual a conversação é mantida;[35]
- $\mathcal{PE\text{-}Conj}$ ($\mathcal{PE\text{-}Conj}=\{\ldots,\mathbb{PE}^{k}_{i_r,r},\ldots\}$) é o conjunto actual de problemas de escalonamento de agente gestor;
- $\mathbb{V}_k$ é o nó de capacidade associado ao agente g_k ($\mathbb{V}_k\in\mathcal{V}$ e $\texttt{TIPO}(\mathbb{V}_k)\in\{\texttt{S},\texttt{P},\texttt{T}\}$).

O comportamento:

- $\mathcal{Comportamento}$ é um componente que determina o comportamento do agente gestor g_k, em função das interacções com o exterior e do estado interno. Propõe-se que este componente seja realizado através de um conjunto de regras, do tipo condição-acção, a ser aplicado por um interpretador de regras, mediante o estado dos restantes componentes do

[35] Este identificador é obtido da mensagem (recebida de um cliente) que iniciou a conversação sobre um novo problema de escalonamento. Ver as secções 4.4.5 e 4.4.6.

agente.[36] Deste modo, tem-se que: $Comportamento = \{\ldots, regra_u, \ldots\}$, em que $regra_u$ é uma regra.

A interacção do agente como o mundo exterior dá-se pela recepção e envio de mensagens de e para outros agentes através de *Msg-in* e de *Msg-out*. O conjunto *Msg-in* é modificado por junção de mensagens, enviadas por agentes clientes e fornecedores, destinadas a g_k e que g_k vai processar. O processamento de uma mensagem consiste em excluir a mensagem de *Msg-in* e alterar o estado interno (*Convs*, *PE-Conj* e $\mathbb{V}_k$) de acordo o estado e o conteúdo da mensagem. Dependendo do estado interno, e de acordo com protocolos a definir, o agente deverá, possivelmente, enviar mensagens a outros agentes. Isto ocorre por junção das mensagens ao conjunto *Msg-out*. Estas mensagens são posteriormente recebidas pelos agentes a que se destinam e são excluídas de *Msg-out*.

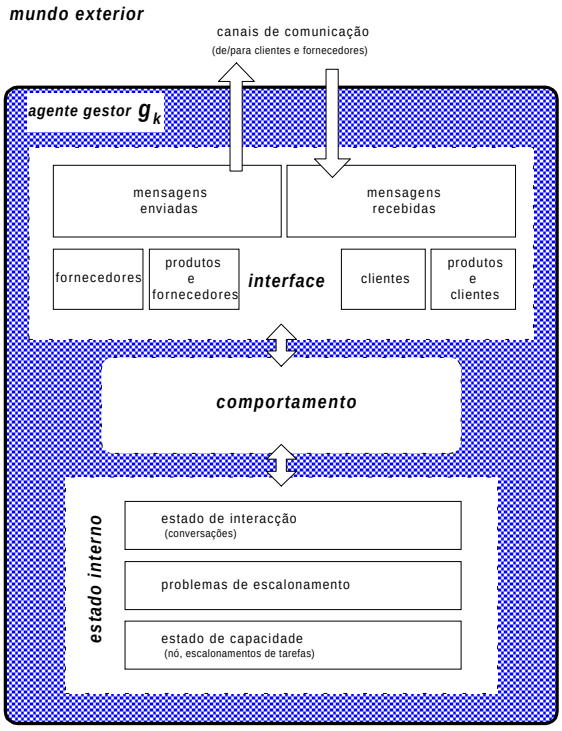


Figura 4-21- Arquitectura de um agente gestor.

O elemento *Convs* representa o *estado de interacção* com os outros agentes, *i.e.*, a parte do estado interno de g_k que diz respeito à *actividade comunicacional*, ou *actividade inter-agente*. O elemento $\mathbb{V}_k$ representa o *estado de capacidade*, *i.e.*, a parte do estado interno de g_k que diz respeito à *actividade intra-agente*, ou *actividade de escalonamento* (o estado de capacidade do nó gerido, incluindo as tarefas escalonadas no nó). O elemento *PE-Conj* estabelece uma ligação entre estas duas componentes do estado interno do agente, pois representa uma *lista de problemas de escalonamento actuais*, acerca dos quais o agente dialoga com os outros

[36] No entanto, *Comportamento* poderá ser realizado de outro modo qualquer. Por exemplo, através de procedimentos (condicionalmente executados dependendo do estado dos restantes componentes do agente) da linguagem de programação usada na realização computacional do modelo.

agentes (o estado actual destes diálogos é mantido em *Convs*) e para os quais mantém as soluções actuais (em $\mathbb{V}_k$).

Cada conversação em *Convs* regista a história da interacção do agente g_k com outro agente, relativa a um pedido recebido, ou enviado, pelo agente g_k. Cada problema de escalonamento $\mathbb{PE}^{k}_{i_r,r}$ em *PE-Conj* é o problema local de escalonamento do agente g_k originado pelo problema global de escalonamento $\mathbb{PE}_{i_r,r}$. A cada $\mathbb{PE}^{k}_{i_r,r}$ em *PE-Conj* corresponde uma conversação $conv^{i,k}_{i_r,r}$ em *Convs*, com um cliente g_i, relativa ao pedido $d^{i,k}_{i_r,r}$, tal que $d^{i,k}_{i_r,r}$=PEDIDO-DE-JUSANTE($\mathbb{PE}^{k}_{i_r,r}$). A cada $\mathbb{PE}^{k}_{i_r,r}$ em *PE-Conj* corresponde uma ou mais conversações $conv^{k,j}_{i_r,r}$ em *Convs*, com fornecedores g_j (com $j=1,\ldots,Nf^{k}_{i_r,r}$), cada uma relativa a um dos pedidos $d^{k,j}_{i_r,r}$, sendo $d^{k,j}_{i_r,r} \in$ PEDIDOS-A-MONTANTE($\mathbb{PE}^{k}_{i_r,r}$). Adicionalmente, no nó $\mathbb{V}_k$ é escalonada a tarefa $\mathbb{O}^{k}_{i_r,r}$ ($\mathbb{O}^{k}_{i_r,r}$=TAREFA($\mathbb{PE}^{k}_{i_r,r}$)).[37]

O comportamento dos agentes gestores será mais detalhado adiante,[38] em particular no que respeita à interacção com os outros agentes de uma rede de EE.

4.4.1.1 Operações para Agentes Gestores

Descrevem-se a seguir as operações primitivas para agentes gestores. As operações selectoras são:

LOCAL(<g_k, *Cl*, *Fr*, *Pcl*, *Pfr*, *Msg-in*, *Msg-out*,
Convs, *PE-Conj*, $\mathbb{V}_k$, *Comportamento*>) ::= g_k.

CLIENTES(<g_k, *Cl*, *Fr*, *Pcl*, *Pfr*, *Msg-in*, *Msg-out*,
Convs, *PE-Conj*, $\mathbb{V}_k$, *Comportamento*>) ::= *Cl*.

FORNECEDORES(<g_k, *Cl*, *Fr*, *Pcl*, *Pfr*, *Msg-in*, *Msg-out*,
Convs, *PE-Conj*, $\mathbb{V}_k$, *Comportamento*>) ::= *Fr*.

PRODUTOS-CLIENTES(<g_k, *Cl*, *Fr*, *Pcl*, *Pfr*, *Msg-in*, *Msg-out*,
Convs, *PE-Conj*, $\mathbb{V}_k$, *Comportamento*>) ::= *Pcl*.

MATERIAIS-FORNECEDORES(<g_k, *Cl*, *Fr*, *Pcl*, *Pfr*, *Msg-in*, *Msg-out*,
Convs, *PE-Conj*, $\mathbb{V}_k$, *Comportamento*>) ::= *Pfr*.

CORREIO-IN(<g_k, *Cl*, *Fr*, *Pcl*, *Pfr*, *Msg-in*, *Msg-out*,
Convs, *PE-Conj*, $\mathbb{V}_k$, *Comportamento*>) ::= *Msg-in*.

CORREIO-OUT(<g_k, *Cl*, *Fr*, *Pcl*, *Pfr*, *Msg-in*, *Msg-out*,

---

[37] As operações PEDIDO-DE-JUSANTE, PEDIDOS-A-MONTANTE e TAREFA produzem, respectivamente, o pedido de cliente, o conjunto de pedidos a fornecedores e a tarefa associados a um problema de escalonamento do agente. Ver a secção 4.4.5.2.

[38] Ver as secções 4.4.7 e 4.4.8.

Convs, *PE-Conj*, $\mathbb{V}_k$, *Comportamento*>) ::= *Msg-out*.

CONVERSAÇÕES(<g_k, *Cl*, *Fr*, *Pcl*, *Pfr*, *Msg-in*, *Msg-out*,
Convs, *PE-Conj*, $\mathbb{V}_k$, *Comportamento*>) ::= *Convs*.

PROBLEMAS-DE-ESCALONAMENTO(<g_k, *Cl*, *Fr*, *Pcl*, *Pfr*, *Msg-in*, *Msg-out*,
Convs, *PE-Conj*, $\mathbb{V}_k$, *Comportamento*>) ::= *PE-Conj*.

NÓ(<g_k, *Cl*, *Fr*, *Pcl*, *Pfr*, *Msg-in*, *Msg-out*,
Convs, *PE-Conj*, $\mathbb{V}_k$, *Comportamento*>) ::= $\mathbb{V}_k$.

COMPORTAMENTO(<g_k, *Cl*, *Fr*, *Pcl*, *Pfr*, *Msg-in*, *Msg-out*,
Convs, *PE-Conj*, $\mathbb{V}_k$, *Comportamento*>) ::= *Comportamento*.

A operação construtora é:

AGENTE-GESTOR(g_k, *Cl*, *Fr*, *Pcl*, *Pfr*, *Msg-in*, *Msg-out*,
Convs, *PE-Conj*, $\mathbb{V}_k$, *Comportamento*) ::=
<g_k, *Cl*, *Fr*, *Pcl*, *Pfr*, *Msg-in*, *Msg-out*, *Convs*, *PE-Conj*, $\mathbb{V}_k$, *Comportamento*>.

As seguintes operações são, por conveniência, adicionalmente definidas, para qualquer agente gestor $\mathbb{G}_k$.

PRODUTOS($\mathbb{G}_k$) ::= PRODUTOS(NÓ($\mathbb{G}_k$)).

MATERIAIS($\mathbb{G}_k$) ::= MATERIAIS(NÓ($\mathbb{G}_k$)).

OUTROS($\mathbb{G}_k$) ::= CLIENTES($\mathbb{G}_k$) $\cup$ FORNECEDORES($\mathbb{G}_k$).

Para pares produto-local-virtual (elementos dos componentes *Pcl* e *Pfr*) assume-se a seguinte operação construtora:

PAR-PL(p,g) ::= <p,g>.

Para um agente gestor, $\mathbb{G}_k$, o fornecedor de um dado produto de entrada p (com p$\in$MATERIAIS($\mathbb{G}_k$)) é identificado pela seguinte operação:

FORNECEDOR($\mathbb{G}_k$,p) ::=
g_i : ($\exists p_x$: (p_x=p) $\wedge$ ($p_x\in$MATERIAIS($\mathbb{G}_k$)) $\wedge$
(PAR-PL(p_x,g_i)$\in$MATERIAIS-FORNECEDORES($\mathbb{G}_k$))).

As operações seguintes, TIPOS-MSG-GESTOR e MODELOS-CONV-GESTOR definem, respectivamente, o conjunto de tipos de mensagens e o conjunto de modelos de conversação

conhecidos pelos agentes gestores. Assume-se que os modelos de conversação *Pedido-de-Cliente* e *Pedido-a-Fornecedor* são conhecidos pelo agente e estão acessíveis por `PEDIDO-DE-CLIENTE()` e `PEDIDO-A-FORNECEDOR()`.[39]

```
TIPOS-MSG-GESTOR() ::=
{ pedido,aceitação,rejeição,satisfação,cancelamento,
  pedido-re,aceitação-re,rejeição-re }.

MODELOS-CONV-GESTOR() ::=
{ PEDIDO-DE-CLIENTE(),PEDIDO-A-FORNECEDOR() }.
```

O predicado `EXISTE-PE?` testa se existe, no conjunto de problemas de escalonamento de um agente gestor $\mathbb{G}_k$, um problema de escalonamento dado o identificador de problema de escalonamento `Id-PE`.[40] A operação `PE` permite obter um problema de escalonamento do agente $\mathbb{G}_k$, dado o identificador de problema `Id-PE` e é aplicável se, e só se, se verifica `EXISTE-PE?(`$\mathbb{G}_k$`,Id-PE)`. Estas operações são definidas, respectivamente, como se segue.

```
EXISTE-PE?(Gk,Id-PE) ::=
∃ℙℰ : (ℙℰ∈PROBLEMAS-DE-ESCALONAMENTO(Gk)) ∧ (ID-PE(ℙℰ)=Id-PE).

PE(Gk,Id-PE) ::=
ℙℰ  : (ℙℰ∈PROBLEMAS-DE-ESCALONAMENTO(Gk)) ∧
      (ID-PE(ℙℰ)=Id-PE).
```

A operação `CONVS-AGENTE` permite seleccionar o conjunto de conversações de um agente $\mathbb{G}_k$ com um agente interlocutor, identificado por g_i.[41]

```
CONVS-AGENTE(Gk,gi) ::=
{ conv :  (conv∈CONVERSAÇÕES(Gk)) ∧
          ((INICIANTE(conv)=gi) ∨ (DESTINATÁRIO(conv)=gi)) }.
```

O predicado `EXISTE-CONV?` testa se, no conjunto de conversações de um agente $\mathbb{G}_k$, já existe uma conversação com o agente identificado por g_i com o identificador de problema de escalonamento associado dado por `Id-PE`. Dada uma mensagem recebida de outro agente, esta operação permite ao agente que a recebe, por exemplo, reconhecer a conversação no

---

[39] Um modelo de conversação define um padrão predefinido de interacção para uma conversação entre um par de agentes e está associado a um protocolo de interacção. Mensagens e modelos de conversação são introduzidos na secção 4.4.6 e, no contexto dos protocolos de interacção entre agentes, descritos em detalhe na secção 4.4.7, para cada classe de agente.

[40] A operação de problema de escalonamento `ID-PE` produz o identificador de um problema de escalonamento dado. Este identificador identifica um problema de escalonamento e permite também seleccionar as conversações do agente que são relevantes para o problema. Ver as secções 4.4.5 e 4.4.6.

[41] As operações de conversação `INICIANTE` e `DESTINATÁRIO` produzem, respectivamente, o identificador do agente que iniciou a conversação, ou o identificador do outro agente envolvido na conversação, dada a conversação. Ver a secção 4.4.6.

contexto da qual a mensagem deve ser entendida, ou reconhecer que se trata de uma mensagem que estabelece uma nova conversação. Neste último caso, que acontece quando é recebido um novo pedido de um agente cliente, o agente deverá, então, criar internamente uma nova conversação, com um identificador de problema de escalonamento igual ao presente na mensagem recebida,[42] que juntará ao seu conjunto de conversações. Esta operação é definida como se segue.

EXISTE-CONV?($\mathbb{G}_k$,g_i,Id-PE) ::=
$\exists_{conv}$: ($conv \in$ CONVS-AGENTE($\mathbb{G}_k$,g_i)) $\wedge$ (ID-PE($conv$)=Id-PE).

A operação CONV-AGENTE permite aceder à conversação com o agente g_i com o identificador de problema de escalonamento associado Id-PE. Esta operação é aplicável se, e só se, se verifica EXISTE-CONV?($\mathbb{G}_k$,g_i,Id-PE) e é definida do seguinte modo:

CONV-AGENTE($\mathbb{G}_k$,g_i,Id-PE) ::=
$conv$: ($conv \in$ CONVS-AGENTE($\mathbb{G}_k$,g_i)) $\wedge$ (ID-PE($conv$)=Id-PE).

As operações CONVS-CLIENTES e CONVS-FORNECEDORES produzem todas as conversações com clientes e todas as conversações com fornecedores, respectivamente, e são definidas como se segue.

CONVS-CLIENTES($\mathbb{G}_k$) ::=
{ $conv$: ($conv \in$ CONVERSAÇÕES($\mathbb{G}_k$)) $\wedge$
(DESTINATÁRIO($conv$)=LOCAL($\mathbb{G}_k$)) }.

CONVS-FORNECEDORES($\mathbb{G}_k$) ::=
{ $conv$: ($conv \in$ CONVERSAÇÕES($\mathbb{G}_k$)) $\wedge$
(INICIANTE($conv$)=LOCAL($\mathbb{G}_k$)) }.

As operações CONV-CLIENTE e CONVS-FORNECEDORES, a seguir definidas, permitem aceder, para um dado agente gestor $\mathbb{G}_k$ e para um dado identificador de problema de escalonamento Id-PE, à conversação do agente com o cliente e ao conjunto de conversações com os fornecedores, relevantes para aquele problema de escalonamento.

CONV-CLIENTE($\mathbb{G}_k$,Id-PE) ::=
$conv$: ($conv \in$ CONVS-CLIENTES($\mathbb{G}_k$)) $\wedge$ (ID-PE($conv$)=Id-PE).

CONVS-FORNECEDORES($\mathbb{G}_k$,Id-PE) ::=
{ $conv$: ($conv \in$ CONVS-FORNECEDORES($\mathbb{G}_k$)) $\wedge$
(ID-PE($conv$)=Id-PE) }.

---

[42] Uma mensagem contém sempre o identificador do agente que a envia e o identificador de problema de escalonamento associado à conversação no contexto da qual a mensagem é trocada. Ver a secção 4.4.6.3.

A operação CONVS-ESTADO selecciona o conjunto de conversações de um agente $\mathbb{G}_k$ que estão no estado dado Estado.[43]

```
CONVS-ESTADO(Gk,Estado) ::=
{ conv :  (conv∈CONVERSAÇÕES(Gk)) ∧ (ESTADO(conv)=Estado) }.
```

A modificação do estado de um agente gestor ocorre por modificação dos componentes *Msg-in*, *Msg-out*, *Convs*, *PE-Conj* e $\mathbb{V}_k$, de acordo com as regras contidas em *Comportamento* e de acordo com a troca de mensagens com o mundo exterior ao agente. A modificação de cada um destes componentes isoladamente é definida pelas operações que se seguem.

Para a colocação de uma mensagem a enviar na caixa de correio de saída (componente *Msg-out*) de um agente gestor, que deve ocorrer quando o agente pretende enviar a mensagem a outro agente, define-se a operação:

```
PÕE-CORREIO-OUT(Gk,msg) ::=
AGENTE-GESTOR(  LOCAL(Gk),
                CLIENTES(Gk),
                FORNECEDORES(Gk),
                PRODUTOS-CLIENTES(Gk),
                MATERIAIS-FORNECEDORES(Gk),
                CORREIO-IN(Gk),
                JUNTA-NO-FIM(CORREIO-OUT(Gk),msg),
                CONVERSAÇÕES(Gk),
                PROBLEMAS-DE-ESCALONAMENTO(Gk),
                NÓ(Gk),
                COMPORTAMENTO(Gk)).
```

As seguintes operações descrevem o acesso do agente às mensagens da caixa de correio de entrada (componente *Msg-in*) de um agente gestor. O predicado HÁ-CORREIO-IN? testa se a caixa de correio de entrada de um agente gestor não está vazia e é definido deste modo:

```
HÁ-CORREIO-IN?(Gk) ::= |CORREIO-IN(Gk)| > 0.
```

A operação OBTÉM-MSG-IN produz a próxima mensagem na caixa de correio de entrada (*i.e.*, a primeira mensagem que foi colocada na caixa, das que lá estão presentes). Esta operação é aplicável se, e só se, se verifica HÁ-CORREIO-IN?($\mathbb{G}_k$) e é definida deste modo:

```
OBTÉM-MSG-IN(Gk) ::= PRIMEIRO(CORREIO-IN(Gk)).
```

---

[43] O *estado* de uma conversação (dado pela operação ESTADO) de um agente evolui, de estado para estado, através de *transições*, de acordo com um *modelo de conversação* e de acordo com o estado de interacção do agente com o agente interlocutor com que é mantida a conversação. Ver a secção 4.4.6.

A operação MSGS-IN-PE produz o conjunto ordenado de mensagens presentes na caixa de correio de entrada que têm a ver com um dado problema de escalonamento, identificado por Id-PE:[44]

MSGS-IN-PE($\mathbb{G}_k$,Id-PE) ::=
{ msg : ($msg \in$ CORREIO-IN($\mathbb{G}_k$)) $\wedge$ (ID-PE(msg)=Id-PE) }.

A retirada de uma dada mensagem da caixa de correio de entrada, o que deve ocorrer após o acesso do agente à mensagem, é descrita pela operação:

```
RETIRA-CORREIO-IN(Gk,msg) ::=
AGENTE-GESTOR(  LOCAL(Gk),
                CLIENTES(Gk),
                FORNECEDORES(Gk),
                PRODUTOS-CLIENTES(Gk),
                MATERIAIS-FORNECEDORES(Gk),
                CORREIO-IN(Gk)\{msg},
                CORREIO-OUT(Gk),
                CONVERSAÇÕES(Gk),
                PROBLEMAS-DE-ESCALONAMENTO(Gk),
                NÓ(Gk),
                COMPORTAMENTO(Gk)).
```

Enquanto que as operações acima definidas para recepção e envio de mensagens são para uso interno do agente gestor, as que a seguir se definem destinam-se a ser usadas pelo sistema de processamento de correio. O sistema de processamento de correio copia cada mensagem na caixa de correio de saída de cada agente para a caixa de correio de entrada do agente destinatário, de modo a garantir, para cada par de agentes comunicantes, a recepção das mensagens pelo destinatário pela mesma ordem que foram enviadas pelo remetente. O predicado HÁ-CORREIO-OUT? testa se a caixa de correio de saída de um agente não está vazia. A operação OBTÉM-MSG-OUT produz a próxima mensagem na caixa de correio de saída de um agente (*i.e.*, a primeira mensagem que foi colocada na caixa, das que lá estão presentes). A operação RETIRA-CORREIO-OUT retira a próxima mensagem da caixa de correio de saída de um agente; a operação PÕE-CORREIO-IN coloca uma mensagem endereçada a um agente na caixa de correio de entrada dele. Estas operações são definidas a seguir (as operações OBTÉM-MSG-OUT e RETIRA-CORREIO-OUT só são aplicáveis a um agente $\mathbb{G}_k$ se se verifica previamente HÁ-CORREIO-OUT?($\mathbb{G}_k$)).

HÁ-CORREIO-OUT?($\mathbb{G}_k$) ::= |CORREIO-OUT($\mathbb{G}_k$)| > 0.

OBTÉM-MSG-OUT($\mathbb{G}_k$) ::= PRIMEIRO(CORREIO-OUT($\mathbb{G}_k$)).

RETIRA-CORREIO-OUT($\mathbb{G}_k$) ::=

[44] A operação de mensagem ID-PE produz, para uma dada mensagem, o identificador do problema de escalonamento a que a mensagem diz respeito. Ver a secção 4.4.6.4.

```
AGENTE-GESTOR(  LOCAL(𝔾k),
                CLIENTES(𝔾k),
                FORNECEDORES(𝔾k),
                PRODUTOS-CLIENTES(𝔾k),
                MATERIAIS-FORNECEDORES(𝔾k),
                CORREIO-IN(𝔾k),
                RETIRA-PRIMEIRO(CORREIO-OUT(𝔾k)),
                CONVERSAÇÕES(𝔾k),
                PROBLEMAS-DE-ESCALONAMENTO(𝔾k),
                NÓ(𝔾k),
                COMPORTAMENTO(𝔾k)).

PÕE-CORREIO-IN(𝔾k,𝕞𝕤𝕘) ::=
AGENTE-GESTOR(  LOCAL(𝔾k),
                CLIENTES(𝔾k),
                FORNECEDORES(𝔾k),
                PRODUTOS-CLIENTES(𝔾k),
                MATERIAIS-FORNECEDORES(𝔾k),
                JUNTA-NO-FIM(CORREIO-IN(𝔾k),𝕞𝕤𝕘),
                CORREIO-OUT(𝔾k),
                CONVERSAÇÕES(𝔾k),
                PROBLEMAS-DE-ESCALONAMENTO(𝔾k),
                NÓ(𝔾k),
                COMPORTAMENTO(𝔾k)).
```

A operação INTRODUZ-CONV-NOVA? permite saber se, para um agente gestor, uma mensagem trocada com um outro agente introduz ou não uma conversação nova, de acordo com os modelos de conversação conhecidos.[45]

```
INTRODUZ-CONV-NOVA?(𝔾k,𝕞𝕤𝕘) ::=
(¬ EXISTE-CONV?(𝔾k,ORIGEM(𝕞𝕤𝕘),ID-PE(𝕞𝕤𝕘))) ∧
(∃Modelo:
  (Modelo∈MODELOS-CONV-GESTOR()) ∧
  TRANSIÇÃO-INICIAL?( Modelo,
                      ETIQUETA( SE(ORIGEM(𝕞𝕤𝕘)=LOCAL(𝔾k),S,R),
                                TIPO-DE-MENSAGEM(𝕞𝕤𝕘)))).
```

[45] As operações de mensagem ORIGEM e TIPO-DE-MENSAGEM produzem o identificador do agente que envia a mensagem e o tipo de uma mensagem, respectivamente. Uma *etiqueta* é associada a cada transição de um modelo de conversação; uma etiqueta é um par que associa um sentido (denotado pelos símbolos S ou R, para mensagem enviada e recebida, respectivamente) a um tipo de mensagem. A operação TRANSIÇÃO-INICIAL? é um predicado que testa se, para um dado modelo de conversação, existe uma transição partindo do estado inicial com uma dada etiqueta associada. Ver a secção 4.4.6.

A operação MODELO-CONV-NOVA produz o modelo de conversação para a conversação que, para o agente $\mathbb{G}_k$, é introduzida pela mensagem, msg, dada.[46] Esta operação é aplicável se, e só se, se verifica INTRODUZ-CONV-NOVA?($\mathbb{G}_k$,msg).

```
MODELO-CONV-NOVA(Gk,msg) ::=
Modelo  :
  (Modelo∈MODELOS-CONV-GESTOR()) ∧
  TRANSIÇÃO-INICIAL?( Modelo,
                      ETIQUETA( SE(ORIGEM(msg)=LOCAL(Gk),S,R),
                                TIPO-DE-MENSAGEM(msg))).
```

A manipulação do conjunto de conversações (introduzir uma conversação nova, retirar uma conversação existente ou operar uma transição numa conversação) de um agente gestor é traduzida pelas operações a seguir definidas. A operação INTRODUZ-CONV, para um dado agente $\mathbb{G}_k$, introduz uma nova conversação no estado inicial, dado um identificador de problema de escalonamento Id-PE, o modelo de conversação $Modelo$ e os identificadores dos agentes iniciante e destinatário De e Para.[47] No caso de a nova conversação estar sendo estabelecida por outro agente, o identificador Id-PE é obtido da mensagem recebida (com Id-PE=ID-PE(msg)). No caso de a nova conversação estar sendo estabelecida pelo próprio agente (*i.e.*, o agente vai enviar a um fornecedor uma mensagem contendo um novo pedido), Id-PE é o identificador de problema de escalonamento recebido na mensagem que iniciou o pedido de cliente relevante ao problema. A operação INTRODUZ-CONV é definida como se segue:[48]

```
INTRODUZ-CONV(Gk,Id-PE,Modelo,De,Para) ::=
AGENTE-GESTOR(  LOCAL(Gk),
                CLIENTES(Gk),
                FORNECEDORES(Gk),
                PRODUTOS-CLIENTES(Gk),
                MATERIAIS-FORNECEDORES(Gk),
                CORREIO-IN(Gk),
                CORREIO-OUT(Gk),
                CONVERSAÇÕES(Gk)∪
                    CONVERSAÇÃO-INICIAL(  Id-PE
                                          Modelo,
                                          De,
                                          Para),
                PROBLEMAS-DE-ESCALONAMENTO(Gk),
                NÓ(Gk),
                COMPORTAMENTO(Gk)).
```

[46] O conjunto de modelos de conversação conhecidos pelo agente deve ser tal que, se se verifica INTRODUZ-CONV-NOVA?($\mathbb{G}_k$,msg), apenas existe um único modelo que contém uma transição inicial com sentido e etiqueta iguais às de msg.

[47] Estes devem ser tais que, ou (De=LOCAL($\mathbb{G}_k$))∧(Para∈OUTROS($\mathbb{G}_k$)) ou (De∈OUTROS($\mathbb{G}_k$))∧(Para=LOCAL($\mathbb{G}_k$)).

[48] A operação CONVERSAÇÃO-INICIAL produz uma conversação no seu estado inicial, ver a secção 4.4.6.6.

A operação RETIRA-CONV retira a conversação $conv$ do conjunto de conversações de um agente G_k. Esta operação deve ser usada quando a conversação está no seu estado final, *i.e.*, quando se verifica que CONVERSAÇÃO-TERMINADA?($conv$) [49] e a conversação não é mais necessária.

```
RETIRA-CONV(Gk,conv) ::=
AGENTE-GESTOR(  LOCAL(Gk),
                CLIENTES(Gk),
                FORNECEDORES(Gk),
                PRODUTOS-CLIENTES(Gk),
                MATERIAIS-FORNECEDORES(Gk),
                CORREIO-IN(Gk),
                CORREIO-OUT(Gk),
                CONVERSAÇÕES(Gk)\{conv},
                PROBLEMAS-DE-ESCALONAMENTO(Gk),
                NÓ(Gk),
                COMPORTAMENTO(Gk)).
```

Para a modificação do estado de uma conversação de um agente gestor G_k, dada a conversação $conv$ e a mensagem recebida, ou a enviar, msg, define-se a operação TRANSITA-CONV.[50]

```
TRANSITA-CONV(Gk,conv,msg) ::=
AGENTE-GESTOR(  LOCAL(Gk),
                CLIENTES(Gk),
                FORNECEDORES(Gk),
                PRODUTOS-CLIENTES(Gk),
                MATERIAIS-FORNECEDORES(Gk),
                SE( ORIGEM(msg)=LOCAL(Gk),
                    CORREIO-IN(Gk),
                    CORREIO-IN(Gk)\{msg}),
                SE( ORIGEM(msg)=LOCAL(Gk),
                    JUNTA-NO-FIM(CORREIO-OUT(Gk),msg),
                    CORREIO-OUT(Gk)),
                SUBSTITUI(  CONVERSAÇÕES(Gk),
                            conv,
                            TRANSITA( conv,
                                      SE( ORIGEM(msg)=
                                            LOCAL(Gk),
                                          S,
                                          R),
                                      msg)),
                PROBLEMAS-DE-ESCALONAMENTO(Gk),
```

[49] A operação CONVERSAÇÃO-TERMINADA? é um predicado que testa se uma conversação está no seu estado final. Ver a secção 4.4.6.6.

[50] A operação TRANSITA aqui usada, opera uma transição numa conversação. Ver a secção 4.4.6.6.

```
                NÓ(𝔾k),
                COMPORTAMENTO(𝔾k)).
```

As operações para manipulação do conjunto de problemas de escalonamento (juntar ao, ou retirar do componente *PE-Conj* um problema de escalonamento dado) de um agente gestor é traduzida, de um modo semelhante, pelas operações a seguir definidas (onde $\mathbb{PE}$ denota um problema de escalonamento do agente gestor $\mathbb{G}_k$):

```
JUNTA-PROBLEMA(𝔾k,ℙ𝔼) ::=
AGENTE-GESTOR(  LOCAL(𝔾k),
                CLIENTES(𝔾k),
                FORNECEDORES(𝔾k),
                PRODUTOS-CLIENTES(𝔾k),
                MATERIAIS-FORNECEDORES(𝔾k),
                CORREIO-IN(𝔾k),
                CORREIO-OUT(𝔾k),
                CONVERSAÇÕES(𝔾k),
                PROBLEMAS-DE-ESCALONAMENTO(𝔾k)∪{ℙ𝔼},
                NÓ(𝔾k),
                COMPORTAMENTO(𝔾k)).

RETIRA-PROBLEMA(𝔾k,ℙ𝔼) ::=
AGENTE-GESTOR(  LOCAL(𝔾k),
                CLIENTES(𝔾k),
                FORNECEDORES(𝔾k),
                PRODUTOS-CLIENTES(𝔾k),
                MATERIAIS-FORNECEDORES(𝔾k),
                CORREIO-IN(𝔾k),
                CORREIO-OUT(𝔾k),
                CONVERSAÇÕES(𝔾k),
                PROBLEMAS-DE-ESCALONAMENTO(𝔾k)\{ℙ𝔼},
                NÓ(𝔾k),
                COMPORTAMENTO(𝔾k)).
```

As modificações do estado de um agente gestor $\mathbb{G}_k$ que correspondem à efectivação ou ao cancelamento do escalonamento de uma tarefa, são descritas pelas duas operações a seguir definidas.[51]

```
ESCALONA-TAREFA(𝔾k,𝕆) ::=
AGENTE-GESTOR(  LOCAL(𝔾k),
                CLIENTES(𝔾k),
                FORNECEDORES(𝔾k),
                PRODUTOS-CLIENTES(𝔾k),
                MATERIAIS-FORNECEDORES(𝔾k),
```

[51] Estas operações fazem uso das operações `ESCALONA` e `DES-ESCALONA`, definidas no Capítulo 3.

```
                CORREIO-IN(𝔾k),
                CORREIO-OUT(𝔾k),
                CONVERSAÇÕES(𝔾k),
                PROBLEMAS-DE-ESCALONAMENTO(𝔾k),
                ESCALONA(NÓ(𝔾k),𝕆),
                COMPORTAMENTO(𝔾k)).

DES-ESCALONA-TAREFA(𝔾k,𝕆) ::=
AGENTE-GESTOR(  LOCAL(𝔾k),
                CLIENTES(𝔾k),
                FORNECEDORES(𝔾k),
                PRODUTOS-CLIENTES(𝔾k),
                MATERIAIS-FORNECEDORES(𝔾k),
                CORREIO-IN(𝔾k),
                CORREIO-OUT(𝔾k),
                CONVERSAÇÕES(𝔾k),
                PROBLEMAS-DE-ESCALONAMENTO(𝔾k),
                DES-ESCALONA(NÓ(𝔾k),𝕆),
                COMPORTAMENTO(𝔾k)).
```

As operações C-CONFLITO-CONJ e T-CONFLITO-CONJ, a seguir definidas, produzem, para um dado agente gestor $\mathbb{G}_k$, o conjunto de tarefas escalonadas envolvidas em conflitos de capacidade, ou em conflitos temporais, respectivamente.[52]

```
C-CONFLITO-CONJ(𝔾k) ::=
{ 𝕆j :  ESCALONADA?(NÓ(𝔾k),𝕆j) ∧
        (∃ℙℝ,ti : (ℙℝ∈PROBLEMAS-DE-ESCALONAMENTO(𝔾k)) ∧
                  (𝕆j=TAREFA(ℙℝ)) ∧
                  (𝕆j∈C-CONFLITO-CONJ(NÓ(𝔾k),ti))) }.

T-CONFLITO-CONJ(𝔾k) ::=
{ 𝕆j :  ESCALONADA?(NÓ(𝔾k),𝕆j) ∧
        (∃ℙℝ  : (ℙℝ∈PROBLEMAS-DE-ESCALONAMENTO(𝔾k)) ∧
                  (𝕆j=TAREFA(ℙℝ)) ∧ T-CONFLITO?(ℙℝ) }.
```

Por comodidade define-se também uma operação construtora de tarefas TAREFA para agentes acumuladores e para agentes processadores, respectivamente.[53]

```
TAREFA(𝔾k,p,q,ts,te) ::= TAREFA(NÓ(𝔾k),p,q,ts,te).
```

---

[52] A operação C-CONFLITO-CONJ para nós (aqui usada na definição da operação C-CONFLITO-CONJ para agentes), testa se existe um conflito de capacidade num nó e num instante dados e foi definida no Capítulo 3. A operação T-CONFLITO? testa se um dado problema de escalonamento tem um conflito temporal e é definida na secção 4.4.5.2.

[53] Estas operações fazem uso da operação TAREFA, definida no Capítulo 3.

TAREFA($\mathbb{G}_k$,p,q,t_s,cval) ::= TAREFA(NÓ($\mathbb{G}_k$),p,q,t_s,cval).

Por fim, a operação TEMPO-ACTUAL é um relógio da rede de EE (t_{agora} denota o valor de tempo actual da rede de EE, igual para toda a rede):

TEMPO-ACTUAL() ::= t_{agora}.

4.4.2 Agente de Retalho

Um *agente de retalho* tem associado um nó de retalho da rede física. Difere de um agente gestor no facto de comunicar com o agente supervisor, não ter clientes, não dispor de capacidade e, portanto, não escalonar tarefas de capacidade. As tarefas que o agente de retalho escalona (tarefas de retalho) são tarefas fictícias, idênticas a pedidos do exterior à rede, que lhe são enviados pelo agente supervisor. Um objecto da classe agente de retalho $\mathbb{G}_r$, é descrito por um tuplo de dez elementos:

<g_r, g_0, *Fr*, *Pfr*, *Msg-in*, *Msg-out*, *Convs*, *PE-Conj*, $\mathbb{V}_r$, *Comportamento*>

Os primeiros seis elementos (g_r, g_0, *Fr*, *Pfr*, *Msg-in*, *Msg-out*) constituem a *interface* com o mundo exterior (*i.e.*, os agentes gestores fornecedores e o agente supervisor), os três seguintes (*Convs*, *PE-Conj* e $\mathbb{V}_k$) descrevem o *estado* interno do agente e o último elemento (*Comportamento*) descreve o *comportamento* do agente. Estes elementos estão presentes na arquitectura representada na Figura 4-22-b) e são, a seguir, descritos.

Para a interface:

- g_r é o identificador do agente ($g_r \in g$ e $g_r \neq g_0$);
- g_0 é o identificador do agente supervisor da rede ($g_0 \in g$);
- *Fr*, *Pfr*, *Msg-in* e *Msg-out* têm um significado semelhante ao que têm para um agente gestor, com a diferença de *Msg-in* e *Msg-out* apenas servirem para mensagens de, e para, os agentes gestores fornecedores e o agente supervisor.

Para o estado interno do agente:

- *Convs* (*Convs*={...,$conv^{0,r}_{i_r,r}$,...,$conv^{r,k}_{i_r,r}$,...}) é o conjunto actual de *conversações* com outros agentes, *i.e.*, o agente supervisor e os agentes gestores fornecedores;
- *PE-Conj* (*PE-Conj*={...,$\mathbb{PE}^{r}_{i_r,r}$,...}) é o conjunto actual de problemas de escalonamento de agente de retalho;

– $\mathbb{V}_r$ é o nó de retalho associado ao agente g_r ($\mathbb{V}_r \in \mathcal{V}$ e TIPO($\mathbb{V}_r$)=R).

Para o comportamento:

– *Comportamento* é o componente que determina o comportamento do agente de retalho g_r em função das interacções com o exterior e do estado interno (um conjunto de regras, do tipo condição-acção, *i.e.*, *Comportamento*={...,$regra_u$,...}, em que $regra_u$ é uma regra).

O comportamento dos agentes de retalho será mais detalhado adiante, em particular no que respeita à interacção com os outros agentes de uma rede de EE.[54] No que respeita a esta interacção, as diferenças essenciais relativamente aos agentes gestores (que advêm do facto de os agentes de retalho funcionarem como parte da fronteira entre a rede e o exterior, servindo de intermediário entre o agente supervisor e os agentes gestores) podem resumir-se assim:

– Um agente de retalho não cancela pedidos enviados aos fornecedores que tenham sido aceites. Um pedido a um fornecedor, uma vez aceite pelo fornecedor só pode ser cancelado por iniciativa deste;

– Um agente de retalho não envia pedidos de re-escalonamento para fornecedores, mas pode receber pedidos de re-escalonamento de fornecedores;

– Um agente de retalho não rejeita (aceita sempre) pedidos de re-escalonamento recebidos dos fornecedores. Cada pedido de re-escalonamento de um fornecedor recebido pelo agente de retalho origina um re-escalonamento do pedido do exterior à rede correspondente, que é comunicado ao agente supervisor;

– Um agente de retalho não rejeita nem cancela pedidos do exterior à rede (remetidos pelo agente supervisor), a não ser que o pedido correspondente ao fornecedor seja rejeitado ou cancelado, respectivamente, por iniciativa do fornecedor.

4.4.2.1 Operações para Agentes de Retalho

As operações primitivas selectoras para agentes de retalho são LOCAL, FORNECEDORES, MATERIAIS-FORNECEDORES, CORREIO-IN, CORREIO-OUT, CONVERSAÇÕES, PROBLEMAS-DE-ESCALONAMENTO, NÓ e COMPORTAMENTO, que operam de forma similar às operações para agentes gestores. Adicionalmente, define-se a seguinte operação:

SUPERVISOR(<g_r, g_0, *Fr*, *Pfr*, *Msg-in*, *Msg-out*,
Convs, *PE-Conj*, $\mathbb{V}_r$, *Comportamento*>) ::= g_0.

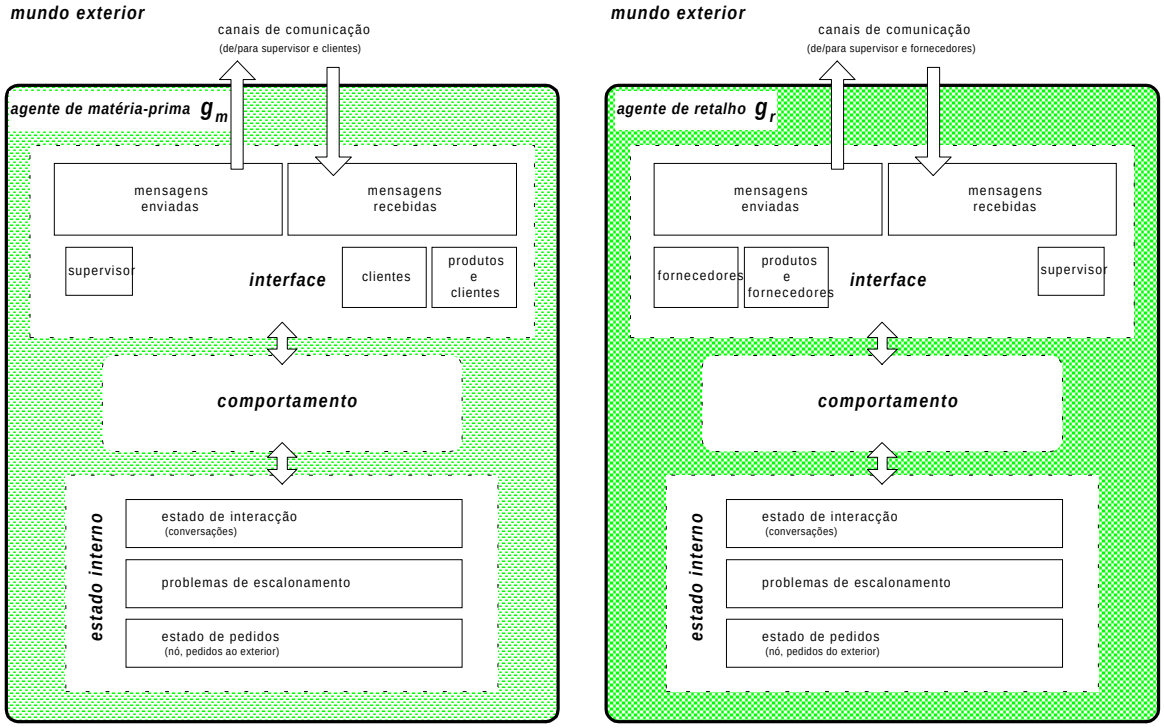
A operação primitiva construtora é:

AGENTE-DE-RETALHO(g_r, g_0, *Fr*, *Pfr*, *Msg-in*, *Msg-out*,
Convs, *PE-Conj*, $\mathbb{V}_r$, *Comportamento*) ::=
<g_r, g_0, *Fr*, *Pfr*, *Msg-in*, *Msg-out*, *Convs*, *PE-Conj*, $\mathbb{V}_r$, *Comportamento*>.

[54] Ver a secção 4.4.7. O detalhe é menor do que para os agentes gestores devido a razões de espaço e dada a relevância dos agentes gestores.

A operação MATERIAIS opera de forma similar à operação de agentes gestores. A operação OUTROS é definida como se segue:

$$\texttt{OUTROS}(\mathbb{G}_r) ::= \{\texttt{SUPERVISOR}(\mathbb{G}_r)\} \cup \texttt{FORNECEDORES}(\mathbb{G}_r).$$



a) Agente de matéria-prima. b) Agente de retalho.

Figura 4-22- Arquitecturas de agentes de matéria-prima e de retalho.

As operações TIPOS-MSG-RETALHO e MODELOS-CONV-RETALHO definem, respectivamente, o conjunto de tipos de mensagens e o conjunto de modelos de conversação conhecidos pelos agentes de retalho. Assume-se que os modelos de conversação *Pedido-para-Montante* e *Pedido-do-Exterior* são conhecidos pelo agente e estão acessíveis por PEDIDO-PARA-MONTANTE() e PEDIDO-DO-EXTERIOR().

```
TIPOS-MSG-RETALHO() ::=
{ pedido,aceitação,rejeição,satisfação,cancelamento,pedido-re,
  aceitação-re,re-escalonamento }.
```

```
MODELOS-CONV-RETALHO() ::=
{ PEDIDO-PARA-MONTANTE(),PEDIDO-DO-EXTERIOR() }.
```

Assume-se definida a operação construtora de pares produto-local-virtual PAR-PL.[55] As operações FORNECEDOR, EXISTE-PE?, PE, CONVS-AGENTE, EXISTE-CONV?, CONV-AGENTE, CONVS-FORNECEDORES e CONVS-ESTADO são definidas de forma similar às operações definidas para agentes gestores. Para processamento dos componentes *Msg-in*, *Msg-out*, *Convs*, *PE-Conj* e $\mathbb{V}_r$ de um agente de retalho usam-se as operações PÕE-CORREIO-OUT, HÁ-CORREIO-IN?, OBTÉM-MSG-IN, MSGS-IN-PE, RETIRA-CORREIO-IN, HÁ-CORREIO-OUT?, OBTÉM-MSG-OUT, RETIRA-CORREIO-OUT, PÕE-CORREIO-IN, INTRODUZ-CONV, RETIRA-CONV, TRANSITA-CONV, JUNTA-PROBLEMA, RETIRA-PROBLEMA, ESCALONA-TAREFA, DES-ESCALONA-TAREFA e TEMPO-ACTUAL, que operam de forma similar às operações definidas para agentes gestores. As operações INTRODUZ-CONV-NOVA? e MODELO-CONV-NOVA são definidas como se segue.

```
INTRODUZ-CONV-NOVA?(𝔾r,𝕞𝕤𝕘) ::=
(¬ EXISTE-CONV?(𝔾r,ORIGEM(𝕞𝕤𝕘),ID-PE(𝕞𝕤𝕘))) ∧
(∃Modelo:
  (Modelo∈MODELOS-CONV-RETALHO()) ∧
  TRANSIÇÃO-INICIAL?( Modelo,
                      ETIQUETA( SE(ORIGEM(𝕞𝕤𝕘)=LOCAL(𝔾r),S,R),
                                TIPO-DE-MENSAGEM(𝕞𝕤𝕘)))).

MODELO-CONV-NOVA(𝔾r,𝕞𝕤𝕘) ::=
Modelo  :
  (Modelo∈MODELOS-CONV-RETALHO()) ∧
  TRANSIÇÃO-INICIAL?( Modelo,
                      ETIQUETA( SE(ORIGEM(𝕞𝕤𝕘)=LOCAL(𝔾r),S,R),
                                TIPO-DE-MENSAGEM(𝕞𝕤𝕘))).
```

4.4.3 Agente de Matéria-Prima

Um *agente de matéria-prima* tem associado um nó de matéria-prima da rede física. É funcionalmente simétrico do agente de retalho e difere de um agente gestor no facto de comunicar com o agente supervisor, não ter fornecedores, não dispor de capacidade e, portanto, não escalonar tarefas de capacidade. As tarefas que o agente de matéria-prima escalona (tarefas de matéria-prima) são tarefas fictícias, idênticas aos pedidos que recebe dos clientes, a partir dos quais gera pedidos da rede ao exterior, que remete para o agente supervisor. Um objecto da classe agente de matéria-prima $\mathbb{G}_m$, é descrito por um tuplo de dez elementos:

$<g_m, g_0, Cl, Pcl, Msg\text{-}in, Msg\text{-}out, Convs, PE\text{-}Conj, \mathbb{V}_m, Comportamento>$

De um modo semelhante ao que acontece com o agente de retalho, para o agente de matéria-prima, os primeiros seis elementos (g_m, g_0, *Cl*, *Pcl*, *Msg-in*, *Msg-out*) constituem uma *interface* com o mundo exterior (*i.e.*, os agentes gestores clientes e o agente supervisor),

[55] Ver a secção 4.4.1.1.

os três seguintes (*Convs*, *PE-Conj* e $\mathbb{V}_k$) descrevem o *estado* interno e último o elemento (*Comportamento*) descreve o *comportamento* do agente. Estes elementos estão presentes na arquitectura representada na Figura 4-22-a) e são, a seguir, descritos.

Para a interface:

- g_m é o identificador do agente ($g_m \in \mathcal{g}$ e $g_m \neq g_0$);
- g_0 é o identificador do agente supervisor da rede ($g_0 \in \mathcal{g}$);
- *Cl*, *Pcl*, *Msg-in* e *Msg-out* têm um significado semelhante ao que têm para um agente gestor, com a diferença de *Msg-in* e *Msg-out* apenas servirem para mensagens de, e para, os agentes gestores clientes e o agente supervisor.

Para o estado interno do agente:

- *Convs* (*Convs*=$\{\ldots, conv^{k,m}_{i_r,r}, \ldots, conv^{m,0}_{i_r,r}, \ldots\}$) é o conjunto actual de *conversações* com outros agentes, *i.e.*, os agentes gestores clientes e o agente supervisor;
- *PE-Conj* (*PE-Conj*=$\{\ldots, \mathbb{PE}^{m}_{i_r,r}, \ldots\}$) é o conjunto actual de problemas de escalonamento de agente de matéria-prima;
- $\mathbb{V}_m$ é o nó de matéria-prima associado ao agente g_m ($\mathbb{V}_m \in \mathcal{V}$ e $\texttt{TIPO}(\mathbb{V}_m)=\texttt{M}$).

Para o comportamento:

- *Comportamento* determina o comportamento do agente de matéria-prima g_m em função das interacções com o exterior e do estado interno. De forma semelhante que para o agente gestor, e para o agente de retalho, assume-se que se trata de um conjunto de regras do tipo condição-acção.

O comportamento dos agentes de matéria-prima será mais detalhado adiante,[54] em particular no que respeita à interacção com os outros agentes de uma rede de EE. No que respeita a esta interacção, as diferenças essenciais relativamente aos agentes gestores (que advêm do facto de os agentes de matéria-prima funcionarem como parte da fronteira entre a rede e o exterior, servindo de intermediário entre o agente supervisor e os agentes gestores) podem resumir-se assim:

- Um agente de matéria-prima não rejeita (aceita sempre) pedidos recebidos dos clientes, que comunica ao agente supervisor;
- Um agente de matéria-prima não cancela pedidos recebidos dos clientes. Um pedido de cliente só pode ser cancelado por iniciativa do cliente;
- Um agente de matéria-prima não envia pedidos de re-escalonamento para clientes, mas pode receber pedidos de re-escalonamento de clientes;
- Um agente de matéria-prima não rejeita (aceita sempre) pedidos de re-escalonamento recebidos dos clientes. Cada pedido de re-escalonamento de um cliente recebido pelo agente de matéria-prima origina um re-escalonamento do pedido da rede ao exterior correspondente que é comunicado ao agente supervisor;
- Um agente de matéria-prima não cancela pedidos da rede ao exterior (remetidos ao agente supervisor), a não ser que o correspondente pedido do cliente seja cancelado por iniciativa do cliente.

4.4.3.1 Operações para Agentes de Matéria-Prima

As operações primitivas selectoras para agentes de matéria-prima são LOCAL, CLIENTES, PRODUTOS-CLIENTES, CORREIO-IN, CORREIO-OUT, CONVERSAÇÕES, PROBLEMAS-DE-ESCALONAMENTO, NÓ e COMPORTAMENTO, que operam de forma similar às operações para agentes gestores. Adicionalmente, existe a operação SUPERVISOR, que opera do mesmo modo que a operação para agentes de retalho.

A operação primitiva construtora é:

AGENTE-DE-MATÉRIA-PRIMA(g_m, g_0, *Cl*, *Pcl*, *Msg-in*, *Msg-out*, *Convs*, *PE-Conj*, $\mathbb{V}_m$, *Comportamento*) ::=
<g_m, g_0, *Cl*, *Pcl*, *Msg-in*, *Msg-out*, *Convs*, *PE-Conj*, $\mathbb{V}_m$, *Comportamento*>.

A operação PRODUTOS é também definida para agentes de matéria-prima e opera de forma similar à operação de agentes gestores. A operação OUTROS é definida como se segue:

OUTROS($\mathbb{G}_m$) ::= {SUPERVISOR($\mathbb{G}_m$)} $\cup$ CLIENTES($\mathbb{G}_m$).

As operações TIPOS-MSG-MATÉRIA-PRIMA e MODELOS-CONV-MATÉRIA-PRIMA definem, respectivamente, o conjunto de tipos de mensagens e o conjunto de modelos de conversação conhecidos pelos agentes de matéria-prima. Assume-se que os modelos de conversação *Pedido-de-Jusante* e *Pedido-para-o-Exterior* são conhecidos pelo agente e estão acessíveis por PEDIDO-DE-JUSANTE() e PEDIDO-PARA-O-EXTERIOR().

```
TIPOS-MSG-MATÉRIA-PRIMA() ::=
{ pedido,aceitação,satisfação,cancelamento,pedido-re,
  aceitação-re,re-escalonamento }.
```

```
MODELOS-CONV-MATÉRIA-PRIMA() ::=
{ PEDIDO-DE-JUSANTE(),PEDIDO-PARA-O-EXTERIOR() }.
```

As operações EXISTE-PE?, PE, CONVS-AGENTE, EXISTE-CONV?, CONV-AGENTE, CONVS-CLIENTES, CONV-CLIENTE, CONVS-ESTADO e são definidas de forma similar às operações definidas para agentes gestores. Para processamento dos componentes *Msg-in*, *Msg-out*, *Convs*, *PE-Conj* e $\mathbb{V}_m$ de um agente de matéria-prima usam-se as operações PÕE-CORREIO-OUT, HÁ-CORREIO-IN?, OBTÉM-MSG-IN, MSGS-IN-PE, RETIRA-CORREIO-IN, HÁ-CORREIO-OUT?, OBTÉM-MSG-OUT, RETIRA-CORREIO-OUT, PÕE-CORREIO-IN, INTRODUZ-CONV, RETIRA-CONV, TRANSITA-CONV, JUNTA-PROBLEMA, RETIRA-PROBLEMA, ESCALONA-TAREFA, DES-ESCALONA-TAREFA e TEMPO-ACTUAL, que operam de forma similar às operações definidas para agentes gestores. As operações INTRODUZ-CONV-NOVA? e MODELO-CONV-NOVA são definidas como se segue.

```
INTRODUZ-CONV-NOVA?(𝔾m, msg) ::=
(¬ EXISTE-CONV?(𝔾m,ORIGEM(msg),ID-PE(msg))) ∧
```

```
(∃Modelo:
  (Modelo∈MODELOS-CONV-MATÉRIA-PRIMA()) ∧
  TRANSIÇÃO-INICIAL?( Modelo,
                      ETIQUETA( SE(ORIGEM(msg)=LOCAL(Gm),S,R),
                                TIPO-DE-MENSAGEM(msg)))).

MODELO-CONV-NOVA(Gm,msg) ::=
Modelo :
  (Modelo∈MODELOS-CONV-MATÉRIA-PRIMA()) ∧
  TRANSIÇÃO-INICIAL?( Modelo,
                      ETIQUETA( SE(ORIGEM(msg)=LOCAL(Gm),S,R),
                                TIPO-DE-MENSAGEM(msg))).
```

4.4.4 Agente Supervisor

Um objecto da classe agente supervisor G_0, é descrito por um tuplo de dez elementos:

$<g_0, Rg, Mg, Prg, Pmg, Msg\text{-}in, Msg\text{-}out, Convs, PE\text{-}Conj, Comportamento>$

Os primeiros sete elementos (g_0, Rg, Mg, Prg, Pmg, $Msg\text{-}in$ e $Msg\text{-}out$) constituem uma *interface* do agente supervisor com o seu mundo exterior (*i.e.*, os agentes de retalho, os agentes de matéria-prima e o exterior da rede de EE), os dois elementos seguintes ($Convs$ e $PE\text{-}Conj$) descrevem o *estado* interno do agente e o último elemento ($Comportamento$) descreve o *comportamento* do agente. Estes elementos estão presentes na arquitectura representada na Figura 4-23 e são, a seguir, descritos.

Para a interface:

- g_0 é o identificador do agente ($g_0 \in gr$);
- Rg é o conjunto de identificadores dos agentes de retalho da rede ($|Rg|=Nr$, $Rg \subset gr$ e $g_0 \notin Rg$);
- Mg é o conjunto de identificadores dos agentes de matéria-prima da rede ($|Mg|=Nm$, $Mg \subset gr$ e $g_0 \notin Mg$);
- Prg ($Prg=\{...,<p_{k_r}, g_r>,...\}$) é um conjunto contendo todos os pares produto-local-virtual $<p_{k_r}, g_r>$ que associam cada produto p_{k_r}, de saída da rede a cada identificador de agente de retalho cliente desse produto na rede g_r ($p_{k_r} \in$ MATERIAIS(NÓ(G_r)), $g_r \in Rg$);
- Pmg ($Pmg=\{...,<p_{k_m}, g_m>,...\}$) é um conjunto contendo todos os pares produto-local-virtual $<p_{k_m}, g_m>$ que associam cada produto p_{k_m}, de entrada da rede a cada identificador de agente de matéria-prima fornecedor do produto na rede g_m ($p_{k_m} \in$ PRODUTOS(NÓ(G_m)), $g_m \in Mg$).

Msg-in e *Msg-out* têm um significado semelhante ao que têm para os restantes agentes, com a diferença de *Msg-in* e *Msg-out* apenas servirem para mensagens de, e para, os agentes de retalho e de matéria-prima da rede e de, e para, o exterior da rede.

Para o estado interno do agente:

– *Convs* (*Convs*=$\{\ldots, \mathrm{conv}^{0,r}_{i_r,r}, \ldots, \mathrm{conv}^{m,0}_{i_r,r}, \ldots\}$) é o conjunto actual de *conversações* com os agentes de retalho e os agentes de matéria-prima;

– *PE-Conj* (*PE-Conj*=$\{\ldots, \mathrm{PE}_{i_r,r}, \ldots\}$) é o conjunto actual de problemas de escalonamento de agente supervisor.

Para o comportamento:

– *Comportamento* é o componente que determina o comportamento do agente de matéria-prima g_m em função das interacções com o exterior e do estado interno (um conjunto de regras do tipo condição-acção).

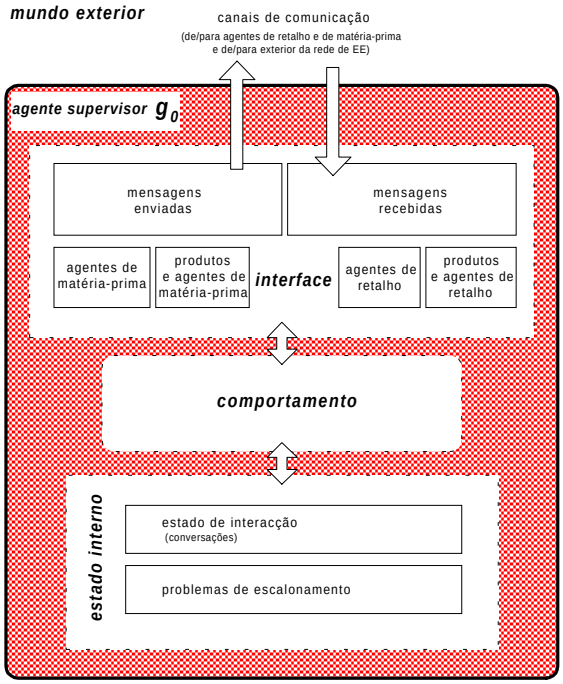


Figura 4-23- Arquitectura do agente supervisor.

Basicamente, a actividade do agente supervisor resume-se à introdução de novos problemas de escalonamento mediante pedidos globais à rede recebidos do exterior. Estes pedidos são recebidos em mensagens especiais na caixa de correio de entrada (o componente *Msg-in*) e são comunicados pelo agente supervisor aos agentes de retalho a que são destinados. De forma simétrica, pedidos da rede ao exterior recebidos pelo agente supervisor dos agentes de matéria-prima são remetidos pelo agente supervisor ao exterior em mensagens especiais colocadas na caixa de correio de saída (o componente *Msg-out*). Estas mensagens de comunicação com o exterior são distintas das mensagens que o agente supervisor troca com os

agentes da rede.[56] Assume-se que o exterior coloca na caixa de correio de entrada do agente supervisor pedidos do exterior à rede e retira da caixa de correio de saída do mesmo agente pedidos da rede ao exterior que lá tenham sido colocados.

4.4.4.1 Operações para o Agente Supervisor

As operações primitivas selectoras para o agente supervisor são LOCAL, CORREIO-IN, CORREIO-OUT, CONVERSAÇÕES, PROBLEMAS-DE-ESCALONAMENTO, COMPORTAMENTO, que operam de forma similar às operações para agentes gestores. Adicionalmente, definem-se as operações:

AGENTES-DE-RETALHO(<g_0, *Rg*, *Mg*, *Prg*, *Pmg*, *Msg-in*, *Msg-out*, *Convs*, *PE-Conj*, *Comportamento*>) ::= *Rg*.

AGENTES-DE-MATÉRIA-PRIMA(<g_0, *Rg*, *Mg*, *Prg*, *Pmg*, *Msg-in*, *Msg-out*, *Convs*, *PE-Conj*, *Comportamento*>) ::= *Mg*.

PRODUTOS-AGENTES(<g_0, *Rg*, *Mg*, *Prg*, *Pmg*, *Msg-in*, *Msg-out*, *Convs*, *PE-Conj*, *Comportamento*>) ::= *Prg*.

MATERIAIS-AGENTES(<g_0, *Rg*, *Mg*, *Prg*, *Pmg*, *Msg-in*, *Msg-out*, *Convs*, *PE-Conj*, *Comportamento*>) ::= *Pmg*.

A operação primitiva construtora é:

AGENTE-SUPERVISOR(g_0, *Rg*, *Mg*, *Prg*, *Pmg*, *Msg-in*, *Msg-out*, *Convs*, *PE-Conj*, *Comportamento*) ::=
<g_0, *Rg*, *Mg*, *Prg*, *Pmg*, *Msg-in*, *Msg-out*, *Convs*, *PE-Conj*, *Comportamento*>.

Assume-se definida a operação construtora de pares produto-local-virtual PAR-PL.[57] As duas operações que se definem a seguir produzem os conjuntos de produtos que o agente supervisor reconhece como sendo o conjunto de produtos de saída da rede (operação PRODUTOS) e o conjunto de produtos de entrada da rede (operação MATERIAIS).[58]

PRODUTOS($\mathbb{G}_0$) ::=
{ p_{k_r} : ($\exists g_r$ ($g_r \in$ AGENTES-DE-RETALHO($\mathbb{G}_0$)) :
($PAR\text{-}PL(p_{k_r}, g_r) \in$ PRODUTOS-AGENTES($\mathbb{G}_0$))) }.

MATERIAIS($\mathbb{G}_0$) ::=

---

[56] Ver as secções 4.4.6.3 e 4.4.7.
[57] Ver a secção 4.4.1.1.
[58] Note-se que, funcionando o agente supervisor como interface da rede de EE com o mundo exterior à rede, os produtos que a rede pode tornar disponíveis ao exterior são os que estão efectivamente incluídos no conjunto produzido por PRODUTOS (e apenas esses) e os produtos que a rede pode consumir do exterior são os que estão efectivamente incluídos no conjunto produzido por MATERIAIS (e apenas esses).

$\{ p_{k_m} : (\exists g_m \quad (g_m \in \text{AGENTES-DE-MATÉRIA-PRIMA}(\mathbb{G}_0)) :$
$(\text{PAR-PL}(p_{k_m}, g_m) \in \text{MATERIAIS-AGENTES}(\mathbb{G}_0))) \}.$

A operação OUTROS é definida como se segue:

OUTROS($\mathbb{G}_0$) ::=
AGENTES-DE-RETALHO($\mathbb{G}_0$) $\cup$ AGENTES-DE-MATÉRIA-PRIMA($\mathbb{G}_0$).

As operações TIPOS-MSG-SUPERVISOR e MODELOS-CONV-SUPERVISOR definem, respectivamente, o conjunto de tipos de mensagens e o conjunto de modelos de conversação conhecidos pelo agente supervisor. Assume-se que os modelos de conversação *Pedido-Global-à-Saída* e *Pedido-Global-à-Entrada* são conhecidos pelo agente e estão acessíveis por PEDIDO-GLOBAL-À-SAÍDA() e PEDIDO-GLOBAL-À-ENTRADA().

TIPOS-MSG-SUPERVISOR() ::=
{ pedido,aceitação,rejeição,satisfação,cancelamento,
re-escalonamento,pedido-in,pedido-out }.

MODELOS-CONV-SUPERVISOR() ::=
{ PEDIDO-GLOBAL-À-SAÍDA(),PEDIDO-GLOBAL-À-ENTRADA() }.

As mensagens do tipo pedido-in e do tipo pedido-out, do conjunto TIPOS-MSG-SUPERVISOR(), são mensagens especiais que o agente supervisor pode trocar com o exterior para receber pedidos globais do exterior à rede (tipo pedido-in), ou enviar pedidos globais da rede ao exterior (tipo pedido-out). Estas mensagens não são empregues no contexto dos modelos de conversação do conjunto MODELOS-CONV-SUPERVISOR().

As operações EXISTE-PE?, PE, CONVS-AGENTE, EXISTE-CONV?, CONV-AGENTE e CONVS-ESTADO são definidas de forma similar às operações definidas para agentes gestores. A operação GERA-ID-PE serve para gerar um novo identificador para um novo problema de escalonamento[59] e é definida como se segue.

GERA-ID-PE($\mathbb{G}_0$) ::=
Id-PE : ($\forall_{conv}$ (conv$\in$CONVERSAÇÕES($\mathbb{G}_0$)) : (ID-PE(conv)$\neq$Id-PE)).

[59] O identificador do novo problema de escalonamento, denotado por Id-PE na definição de GERA-ID-PE, poderá, por exemplo, ser um contador dos problemas de escalonamento que existiram até ao momento. Existe uma correspondência biunívoca entre identificador novo e cada pedido global do exterior à rede, cada processo desencadeado pelo pedido, cada par $<g_r, i_r>$, em que g_r é o identificador do agente de retalho que recebeu o pedido global à rede e i_r um índice que distingue problemas de escalonamento diferentes, colocados ao mesmo agente de retalho g_r. Numa realização computacional, o identificador de problema de escalonamento permite rastrear a propagação de pedidos na rede e o escalonamento e re-escalonamento de tarefas do processo desencadeado pelo mesmo pedido global, já que cada problema de escalonamento local de cada agente de rede e cada uma das conversações relevantes ao problema para cada agente têm associados este identificador.

Para processamento dos componentes *Msg-in*, *Msg-out*, *Convs* e *PE-Conj* do agente supervisor definem-se as operações PÕE-CORREIO-OUT, HÁ-CORREIO-IN?, OBTÉM-MSG-IN, MSGS-IN-PE, RETIRA-CORREIO-IN, HÁ-CORREIO-OUT?, OBTÉM-MSG-OUT, RETIRA-CORREIO-OUT, PÕE-CORREIO-IN, INTRODUZ-CONV, RETIRA-CONV, TRANSITA-CONV, JUNTA-PROBLEMA, RETIRA-PROBLEMA e TEMPO-ACTUAL, que operam de forma similar às operações definidas para agentes gestores. As operações INTRODUZ-CONV-NOVA? e MODELO-CONV-NOVA são definidas como se segue.

```
INTRODUZ-CONV-NOVA?(G₀,msg) ::=
(¬ EXISTE-CONV?(G₀,ORIGEM(msg),ID-PE(msg))) ∧
(∃Modelo:
  (Modelo∈MODELOS-CONV-SUPERVISOR()) ∧
  TRANSIÇÃO-INICIAL?( Modelo,
                      ETIQUETA( SE(ORIGEM(msg)=LOCAL(G₀),S,R),
                                TIPO-DE-MENSAGEM(msg)))).

MODELO-CONV-NOVA(G₀,msg) ::=
Modelo :
  (Modelo∈MODELOS-CONV-SUPERVISOR()) ∧
  TRANSIÇÃO-INICIAL?( Modelo,
                      ETIQUETA( SE(ORIGEM(msg)=LOCAL(G₀),S,R),
                                TIPO-DE-MENSAGEM(msg))).
```

4.4.5 Problema de Escalonamento

Nesta secção descreve-se como é modelado um *problema de escalonamento multi-agente* na perspectiva de cada agente. A informação relevante associada a cada problema de escalonamento multi-agente é representada internamente, num agente que esteja envolvido na resolução do problema, como um elemento do conjunto dos problemas de escalonamento *PE-Conj*. A secção presente diz respeito, portanto, aos elementos deste componente da arquitectura dos agentes (ver a secção anterior), simbolicamente referidos pelo símbolo PE.

Como uma rede de EE é um sistema distribuído composto por vários agentes, existem perspectivas diferentes de um mesmo problema de escalonamento para agentes diferentes da uma mesma rede. Isto quer dizer que o mesmo problema de escalonamento é visto, e representado, por cada um dos agentes, de modo diferente e que nenhum agente detém toda a informação relativa ao problema; esta informação está distribuída por vários agentes. As representações internas que se propõem são, no entanto, estruturalmente similares.

De um modo geral, a representação de um problema de escalonamento inclui sempre um *pedido de jusante* (um pedido de um cliente, no caso de um agente gestor), um *pedido* ou um *conjunto de pedidos a montante* (pedidos a fornecedores, no caso de um agente gestor), *datas* limite que definem um horizonte temporal para satisfação dos pedidos e poderá incluir, ou não, uma *tarefa*.

Também, um agente pode estar ou não envolvido no (*i.e.*, envolvido na resolução do) problema, conforme recebe ou não um pedido associado ao problema. Se não está envolvido, não possui qualquer representação do problema; se está envolvido, mantém uma representação interna do problema que corresponde à sua perspectiva do problema de escalonamento multi-agente. Para os agentes gestores envolvidos, esta perspectiva do problema é apelidada de *perspectiva local* do problema ou *problema de escalonamento local*; para o agente supervisor é apelidada de *perspectiva global* ou *problema de escalonamento global* (como mais adiante se verá, isto não significa que o agente supervisor detém toda a informação de cada problema de escalonamento multi-agente). No contexto da sequência de fases de resolução de problemas de escalonamento proposta anteriormente,[60] a representação interna de um problema de escalonamento é gerada na fase 1 e torna-se obsoleta no final da fase 3 ou antes, se houver desistência do problema por parte de agentes gestores envolvidos.

A seguir clarifica-se melhor a noção de envolvimento de um agente num problema de escalonamento, para as diferentes classes de agentes:

Dado um problema de escalonamento global definido por um pedido global do exterior à rede $\mathbb{d}^{0,r}_{i_r,r}$ e um intervalo de horizonte temporal $\mathbb{H}_{i_r,r}$:

- O agente supervisor g_0 está envolvido no problema. g_0 recebe, do exterior, o pedido global do exterior à rede $\mathbb{d}^{0,r}_{i_r,r}$ e recebe, dos agentes de matéria prima g_m (ver mais abaixo) pedidos globais da rede ao exterior $\mathbb{d}^{m,0}_{i_r,r}$ (com $m=1,...,Nm_{i_r,r}$), que remete para o exterior;
- O agente de retalho g_r está envolvido no problema. g_r recebe de g_0 o pedido global $\mathbb{d}^{0,r}_{i_r,r}$;
- Se g_i designa o agente gestor fornecedor de g_r para o produto $\texttt{PRODUTO}(\mathbb{d}^{0,r}_{i_r,r})$, g_i está envolvido no problema. g_i recebe de g_r o pedido local $\mathbb{d}^{r,i}_{i_r,r}$ (do mesmo produto que $\mathbb{d}^{0,r}_{i_r,r}$) Os agentes gestores g_k fornecedores, directos ou indirectos, de produtos componentes, directos ou indirectos, de $\texttt{PRODUTO}(\mathbb{d}^{r,i}_{i_r,r})$ estão igualmente envolvidos no problema. Cada g_k recebe de um cliente g_i um pedido local $\mathbb{d}^{i,k}_{i_r,r}$;
- Os agentes de matéria-prima g_m (com $m=1,...,Nm_{i_r,r}$) estão envolvidos no problema. Cada g_m recebe de um agente gestor cliente g_j um pedido local $\mathbb{d}^{j,m}_{i_r,r}$;
- Mais nenhum agente está envolvido no problema.

4.4.5.1 Problema de Escalonamento de Agente Gestor

Os agentes gestores têm uma perspectiva local de um problema de escalonamento. Para um agente gestor genérico g_k, um problema de escalonamento $\mathbb{PE}^{k}_{i_r,r}$ é definido por um tuplo de seis elementos:

$<\texttt{Id-PE},\texttt{DD}^{k}_{i_r,r},\mathbb{RD}^{k,}_{i_r,r},\mathbb{d}^{i,k}_{i_r,r},\mathbb{d}^{k,}_{i_r,r},\mathbb{O}^{k}_{i_r,r}>$

em que:

---

[60] Ver a secção 4.3.6.

– Id-PE é um identificador que associa o problema de escalonamento local do agente às conversações relevantes, associadas ao problema, com o cliente e com os fornecedores e ao problema de escalonamento global correspondente;

– $DD^{k}_{i_r,r}$ é a data limite mais tarde;

– $\mathcal{RD}^{k,}_{i_r,r}$ ($\mathcal{RD}^{k,}_{i_r,r}=\{RD^{k,1}_{i_r,r},...,RD^{k,j}_{i_r,r},...,RD^{k,Nf^{k}_{i_r,r}}_{i_r,r}\}$) é o conjunto ordenado de datas de lançamento mais cedo com os fornecedores g_j (com $j=1,...,Nf^{k}_{i_r,r}$);

– $\mathbb{d}^{i,k}_{i_r,r}$ é um pedido local de um agente cliente g_i, originado pelo pedido global $\mathbb{d}^{0,r}_{i_r,r}$;

– $\mathcal{d}^{k,}_{i_r,r}$ ($\mathcal{d}^{k,}_{i_r,r}=\{\mathbb{d}^{k,1}_{i_r,r},...,\mathbb{d}^{k,j}_{i_r,r},...,\mathbb{d}^{k,Nf^{k}_{i_r,r}}_{i_r,r}\}$) é o conjunto ordenado de todos os pedidos locais para agentes fornecedores g_j (com $j=1,...,Nf^{k}_{i_r,r}$), necessários à satisfação do pedido local $\mathbb{d}^{i,k}_{i_r,r}$;

– $\mathbb{O}^{k}_{i_r,r}$ é a tarefa do agente g_k para satisfação do pedido local $\mathbb{d}^{i,k}_{i_r,r}$.

Id-PE é obtido a partir da mensagem recebida do cliente que estabeleceu a conversação do cliente. O pedido $\mathbb{d}^{i,k}_{i_r,r}$ é enviado a g_k pelo cliente g_i e a data $DD^{k}_{i_r,r}$ e as datas em $\mathcal{RD}^{k,}_{i_r,r}$ são calculadas pelo agente g_k com base nas folgas $FEJ^{k}_{i_r,r}$ e $FEM^{k,j}_{i_r,r}$, fornecidas pelo cliente e pelos fornecedores, respectivamente. $\mathcal{d}^{k,}_{i_r,r}$ e $\mathbb{O}^{k}_{i_r,r}$ são parte de uma solução do problema de escalonamento e dependem de decisões (locais) do agente g_k. Os intervalos $\mathbb{h}^{k,j}_{i_r,r}$ e $\mathbb{H}^{k,j}_{i_r,r}$ (o que inclui $\mathbb{h}^{k}_{i_r,r}$ e $\mathbb{H}^{k}_{i_r,r}$), bem como todas as folgas temporais, estão implicitamente representados no problema de escalonamento $\mathbb{PE}^{k}_{i_r,r}$. Desde a geração de um problema $\mathbb{PE}^{k}_{i_r,r}$ e sua inclusão em *PE-Conj* até à sua exclusão de *PE-Conj*, apenas os componentes $\mathbb{d}^{i,k}_{i_r,r}$, $\mathcal{d}^{k,}_{i_r,r}$, $\mathbb{O}^{k}_{i_r,r}$ podem ser alterados, no caso de existirem re-escalonamentos dos pedidos (por acordo com o cliente ou com os fornecedores) ou da tarefa.[61]

4.4.5.2 Operações com Problemas de Escalonamento de Agente Gestor

Nesta secção definem-se operações aplicáveis a problemas de escalonamento de agentes gestores. Adicionalmente, no final da secção, as condições das regras para detecção de conflitos temporais, anteriormente expostas,[62] são reformuladas em termos de algumas daquelas operações.

A seguir definem-se a as operações selectoras de problema de escalonamento de agente gestor.

$ID\text{-}PE(<Id\text{-}PE, DD^{k}_{i_r,r}, \mathcal{RD}^{k,}_{i_r,r}, \mathbb{d}^{i,k}_{i_r,r}, \mathcal{d}^{k,}_{i_r,r}, \mathbb{O}^{k}_{i_r,r}>) ::= Id\text{-}PE.$

$DD(<Id\text{-}PE, DD^{k}_{i_r,r}, \mathcal{RD}^{k,}_{i_r,r}, \mathbb{d}^{i,k}_{i_r,r}, \mathcal{d}^{k,}_{i_r,r}, \mathbb{O}^{k}_{i_r,r}>) ::= DD^{k}_{i_r,r}.$

---

[61] A data $DD^{k}_{i_r,r}$ e as datas em $\mathcal{RD}^{k,}_{i_r,r}$ mantêm-se inalteradas, pois assume-se que não há alteração das datas $DD_{i_r,r}$ e $RD_{i_r,r}$ do problema global de escalonamento. Ver o pressuposto 6 na secção 4.3.4.

[62] Ver a secção 4.3.6.2.

$RD\text{-}CONJ(<Id\text{-}PE, DD^{k}_{i_r,r}, \mathcal{RD}^{k,}_{i_r,r}, \mathcal{d}^{i,k}_{i_r,r}, \mathcal{d}^{k,}_{i_r,r}, \mathbb{O}^{k}_{i_r,r}>) ::= \mathcal{RD}^{k,}_{i_r,r}.$

$RD(<Id\text{-}PE, DD^{k}_{i_r,r}, \{..., RD^{k,j}_{i_r,r}, ...\}, \mathcal{d}^{i,k}_{i_r,r}, \mathcal{d}^{k,}_{i_r,r}, \mathbb{O}^{k}_{i_r,r}>, g_j) ::= RD^{k,j}_{i_r,r}.$

$RD(\mathbb{PE}^{k}_{i_r,r}) ::= MAX(RD^{k,j}_{i_r,r}) : RD^{k,j}_{i_r,r} \in RD\text{-}CONJ(\mathbb{PE}^{k}_{i_r,r}).$

$PEDIDO\text{-}DE\text{-}JUSANTE(<Id\text{-}PE, DD^{k}_{i_r,r}, \mathcal{RD}^{k,}_{i_r,r}, \mathcal{d}^{i,k}_{i_r,r}, \mathcal{d}^{k,}_{i_r,r}, \mathbb{O}^{k}_{i_r,r}>) ::= \mathcal{d}^{i,k}_{i_r,r}.$

$PEDIDOS\text{-}A\text{-}MONTANTE(<Id\text{-}PE, DD^{k}_{i_r,r}, \mathcal{RD}^{k,}_{i_r,r}, \mathcal{d}^{i,k}_{i_r,r}, \mathcal{d}^{k,}_{i_r,r}, \mathbb{O}^{k}_{i_r,r}>) ::= \mathcal{d}^{k,}_{i_r,r}.$

$PEDIDO\text{-}A\text{-}MONTANTE($
$<Id\text{-}PE, DD^{k}_{i_r,r}, \mathcal{RD}^{k,}_{i_r,r}, \mathcal{d}^{i,k}_{i_r,r}, \{..., \mathcal{d}^{k,j}_{i_r,r}, ...\}, \mathbb{O}^{k}_{i_r,r}>, g_j) ::= \mathcal{d}^{k,j}_{i_r,r}.$

$TAREFA(<Id\text{-}PE, DD^{k}_{i_r,r}, \mathcal{RD}^{k,}_{i_r,r}, \mathcal{d}^{i,k}_{i_r,r}, \mathcal{d}^{k,}_{i_r,r}, \mathbb{O}^{k}_{i_r,r}>) ::= \mathbb{O}^{k}_{i_r,r}.$

A operação construtora é:

$PE(Id\text{-}PE, DD^{k}_{i_r,r}, \mathcal{RD}^{k,}_{i_r,r}, \mathcal{d}^{i,k}_{i_r,r}, \mathcal{d}^{k,}_{i_r,r}, \mathbb{O}^{k}_{i_r,r}) ::=$
$<Id\text{-}PE, DD^{k}_{i_r,r}, \mathcal{RD}^{k,}_{i_r,r}, \mathcal{d}^{i,k}_{i_r,r}, \mathcal{d}^{k,}_{i_r,r}, \mathbb{O}^{k}_{i_r,r}>.$

A seguir, definem-se operações adicionais para qualquer problema de escalonamento de agente gestor $\mathbb{PE}^{k}_{i_r,r}$. Estas operações pressupõem as igualdades expostas na Tabela 4-5 e na Tabela 4-6.

As três operações seguintes permitem obter, respectivamente, a data limite local, a data de lançamento local com um fornecedor e a data de lançamento local:

$DD\text{-}LOCAL(\mathbb{PE}^{k}_{i_r,r}) ::= TEMPO(PEDIDO\text{-}DE\text{-}JUSANTE(\mathbb{PE}^{k}_{i_r,r})).$

$RD\text{-}LOCAL(\mathbb{PE}^{k}_{i_r,r}, g_j) ::= TEMPO(PEDIDO\text{-}A\text{-}MONTANTE(\mathbb{PE}^{k}_{i_r,r}, g_j)).$

$RD\text{-}LOCAL(\mathbb{PE}^{k}_{i_r,r}) ::=$
$MAX(TEMPO(\mathcal{d}^{k,j}_{i_r,r})) : \mathcal{d}^{k,j}_{i_r,r} \in PEDIDOS\text{-}A\text{-}MONTANTE(\mathbb{PE}^{k}_{i_r,r}).$

Para o caso de problema de escalonamento de agente processador, as folgas temporais interna a jusante, interna a montante com um fornecedor e a montante são obtidas, respectivamente, através das operações:

$FIJ(\mathbb{PE}^{k}_{i_r,r}) ::= DD\text{-}LOCAL(\mathbb{PE}^{k}_{i_r,r}) - TE(TAREFA(\mathbb{PE}^{k}_{i_r,r})).$

$FIM(\mathbb{PE}^{k}_{i_r,r}, g_j) ::= TS(TAREFA(\mathbb{PE}^{k}_{i_r,r})) - RD\text{-}LOCAL(\mathbb{PE}^{k}_{i_r,r}, g_j).$

$FIM(\mathbb{PE}^{k}_{i_r,r}) ::= TS(TAREFA(\mathbb{PE}^{k}_{i_r,r})) - RD\text{-}LOCAL(\mathbb{PE}^{k}_{i_r,r}).$

Para o caso de problema de escalonamento de agente acumulador, as folgas temporais internas a jusante e a montante simétricas são dadas pelas operações:

$FIJ(\mathbb{PE}^{k}_{i_r,r}) ::= CEILING((DURAÇÃO(TAREFA(\mathbb{PE}^{k}_{i_r,r})) - 1) / 2).$

$FIM(\mathbb{PE}^{k}_{i_r,r}) ::= DURAÇÃO(TAREFA(\mathbb{PE}^{k}_{i_r,r})) - 1 - FIJ(\mathbb{PE}^{k}_{i_r,r}).$

No caso de problema de escalonamento de agente processador, as folgas temporais a jusante e a montante são dadas por:

$FJ(\mathbb{PE}^{k}_{i_r,r}) ::= DD(\mathbb{PE}^{k}_{i_r,r}) - TE(TAREFA(\mathbb{PE}^{k}_{i_r,r})).$

$FM(\mathbb{PE}^{k}_{i_r,r}) ::= TS(TAREFA(\mathbb{PE}^{k}_{i_r,r})) - RD(\mathbb{PE}^{k}_{i_r,r}).$

No caso de problema de escalonamento de agente acumulador, as folgas temporais a jusante e a montante são dadas por:

$FJ(\mathbb{PE}^{k}_{i_r,r}) ::= DD(\mathbb{PE}^{k}_{i_r,r}) - TE(TAREFA(\mathbb{PE}^{k}_{i_r,r})) + FIJ(\mathbb{PE}^{k}_{i_r,r}).$

$FM(\mathbb{PE}^{k}_{i_r,r}) ::= TS(TAREFA(\mathbb{PE}^{k}_{i_r,r})) - RD(\mathbb{PE}^{k}_{i_r,r}) + FIM(\mathbb{PE}^{k}_{i_r,r}).$

Para qualquer problema de escalonamento de agente gestor a folga total é dada por:

$FT(\mathbb{PE}^{k}_{i_r,r}) ::= FJ(\mathbb{PE}^{k}_{i_r,r}) + FM(\mathbb{PE}^{k}_{i_r,r}).$

Para qualquer problema de escalonamento de agente gestor as folgas temporais externa a jusante, externa a montante com um fornecedor e a montante são dadas por:

$FEJ(\mathbb{PE}^{k}_{i_r,r}) ::= DD(\mathbb{PE}^{k}_{i_r,r}) - TEMPO(PEDIDO\text{-}DE\text{-}JUSANTE(\mathbb{PE}^{k}_{i_r,r})).$

$FEM(\mathbb{PE}^{k}_{i_r,r}, g_j) ::=$
$TEMPO(PEDIDO\text{-}A\text{-}MONTANTE(\mathbb{PE}^{k}_{i_r,r}, g_j)) - RD(\mathbb{PE}^{k}_{i_r,r}, g_j).$

$FEM(\mathbb{PE}^{k}_{i_r,r}) ::= \underset{j=1,\ldots,Nf^{k}_{i_r,r}}{MIN} (FEM(\mathbb{PE}^{k}_{i_r,r}, g_j)).$

Para qualquer problema de escalonamento de agente gestor, as folgas temporais jusante-montante (a comunicar a fornecedores) e montante-jusante (a comunicar ao cliente) são dadas por:

$\mathtt{FJM}(\mathbb{PE}^{k}_{i_r,r}, g_j) ::= \mathtt{FJ}(\mathbb{PE}^{k}_{i_r,r}) + \mathtt{FIM}(\mathbb{PE}^{k}_{i_r,r}, g_j).$

$\mathtt{FMJ}(\mathbb{PE}^{k}_{i_r,r}) ::= \mathtt{FM}(\mathbb{PE}^{k}_{i_r,r}) + \mathtt{FIJ}(\mathbb{PE}^{k}_{i_r,r}).$

Os valores da folga externa a jusante $\mathtt{FEJ}^{k}_{i_r,r}$, comunicado pelo cliente e das folgas externas a montante $\mathtt{FEM}^{k,j}_{i_r,r}$, comunicados por cada fornecedor g_j, juntamente com o pedido do cliente, $\mathbb{d}^{i,k}_{i_r,r}$ e os pedidos aos fornecedores em $\mathbb{d}^{k,}_{i_r,r}$, permitem calcular a data $\mathtt{DD}^{k}_{i_r,r}$ e as datas em $\mathbb{RD}^{k,}_{i_r,r}$. As seguintes operações definem este cálculo:

$\mathtt{DD\text{-}CALC}(\mathbb{d}^{i,k}_{i_r,r}, \mathtt{FEJ}^{k}_{i_r,r}) ::= \mathtt{TEMPO}(\mathbb{d}^{i,k}_{i_r,r}) + \mathtt{FEJ}^{k}_{i_r,r}.$

$\mathtt{RD\text{-}CALC}(\mathbb{d}^{k,j}_{i_r,r}, \mathtt{FEM}^{k,j}_{i_r,r}) ::= \mathtt{TEMPO}(\mathbb{d}^{k,j}_{i_r,r}) - \mathtt{FEM}^{k,j}_{i_r,r}.$

As operações seguintes dizem respeito a conflitos temporais relativos envolvendo problemas de escalonamento. São predicados que testam se existem conflitos temporais globais ou conflitos temporais locais externos de um tipo particular, para um dado problema de escalonamento.

Para qualquer caso de problema de escalonamento de agente gestor a seguinte operação testa a existência de conflito global total:

$\mathtt{T\text{-}CONFLITO\text{-}GLOBAL\text{-}TOTAL?}(\mathbb{PE}^{k}_{i_r,r}) ::= \mathtt{FT}(\mathbb{PE}^{k}_{i_r,r}) < 0.$

Para o caso de problema de escalonamento de agente processador as seguintes operações testam, respectivamente, a existência de um conflito global a jusante e de um conflito global a montante:

$\mathtt{T\text{-}CONFLITO\text{-}GLOBAL\text{-}JUSANTE?}(\mathbb{PE}^{k}_{i_r,r}) ::= \mathtt{FJ}(\mathbb{PE}^{k}_{i_r,r}) < 0.$

$\mathtt{T\text{-}CONFLITO\text{-}GLOBAL\text{-}MONTANTE?}(\mathbb{PE}^{k}_{i_r,r}) ::= \mathtt{FM}(\mathbb{PE}^{k}_{i_r,r}) < 0.$

Para o caso de problema de escalonamento de agente acumulador as seguintes operações testam, respectivamente, a existência de um conflito global a jusante e de um conflito global a montante (estas operações usam, implicitamente, folgas internas não simétricas):

$\mathtt{T\text{-}CONFLITO\text{-}GLOBAL\text{-}JUSANTE?}(\mathbb{PE}^{k}_{i_r,r}) ::=$
$\mathtt{DD}(\mathbb{PE}^{k}_{i_r,r}) - \mathtt{TS}(\mathtt{TAREFA}(\mathbb{PE}^{k}_{i_r,r})) - 1 < 0.$

T-CONFLITO-GLOBAL-MONTANTE?($\mathbb{PE}^{k}_{i_r,r}$) ::=
TE(TAREFA($\mathbb{PE}^{k}_{i_r,r}$)) - 1 - RD($\mathbb{PE}^{k}_{i_r,r}$) < 0.

Para qualquer caso de problema de escalonamento de agente gestor as duas operações seguintes testam, respectivamente, a existência de um conflito local externo a jusante e de um conflito local externo a montante:

T-CONFLITO-EXTERNO-JUSANTE?($\mathbb{PE}^{k}_{i_r,r}$) ::= FEJ($\mathbb{PE}^{k}_{i_r,r}$) < 0.

T-CONFLITO-EXTERNO-MONTANTE?($\mathbb{PE}^{k}_{i_r,r}$) ::= FEM($\mathbb{PE}^{k}_{i_r,r}$) < 0.

Para qualquer caso de problema de escalonamento de agente gestor as duas operações seguintes testam, respectivamente, se existem conflitos temporais globais e se existem ou conflitos temporais locais externos:

T-CONFLITO-GLOBAL?($\mathbb{PE}^{k}_{i_r,r}$) ::=
T-CONFLITO-GLOBAL-TOTAL?($\mathbb{PE}^{k}_{i_r,r}$) $\vee$
T-CONFLITO-GLOBAL-JUSANTE?($\mathbb{PE}^{k}_{i_r,r}$) $\vee$
T-CONFLITO-GLOBAL-MONTANTE?($\mathbb{PE}^{k}_{i_r,r}$).

T-CONFLITO-EXTERNO?($\mathbb{PE}^{k}_{i_r,r}$) ::=
T-CONFLITO-EXTERNO-JUSANTE?($\mathbb{PE}^{k}_{i_r,r}$) $\vee$
T-CONFLITO-EXTERNO-MONTANTE?($\mathbb{PE}^{k}_{i_r,r}$).

A operação T-CONFLITO? testa se existe algum conflito temporal relativo, dos que acima foram referidos.

T-CONFLITO?($\mathbb{PE}^{k}_{i_r,r}$) ::=
T-CONFLITO-GLOBAL?($\mathbb{PE}^{k}_{i_r,r}$) $\vee$ T-CONFLITO-EXTERNO?($\mathbb{PE}^{k}_{i_r,r}$).

Por último, reformulam-se as condições das regras para detecção de conflitos temporais,[62] em termos de operações acima definidas. Para um problema de escalonamento de agente gestor $\mathbb{PE}^{k}_{i_r,r}$, as condições referidas são:

Condição para a regra 0: T-CONFLITO-GLOBAL-TOTAL?($\mathbb{PE}^{k}_{i_r,r}$);

Condição para a regra 1: T-CONFLITO-GLOBAL-JUSANTE?($\mathbb{PE}^{k}_{i_r,r}$) $\wedge$
T-CONFLITO-EXTERNO-JUSANTE?($\mathbb{PE}^{k}_{i_r,r}$);

Condição para a regra 2: $\texttt{T-CONFLITO-GLOBAL-MONTANTE?}(\mathbb{PE}^{k}_{i_r,r}) \wedge$
$\texttt{T-CONFLITO-EXTERNO-MONTANTE?}(\mathbb{PE}^{k}_{i_r,r})$;

Condição para a regra 3: $\texttt{T-CONFLITO-EXTERNO-JUSANTE?}(\mathbb{PE}^{k}_{i_r,r}) \wedge$
$(\neg\ \texttt{T-CONFLITO-GLOBAL-JUSANTE?}(\mathbb{PE}^{k}_{i_r,r}))$;

Condição para a regra 4: $\texttt{T-CONFLITO-EXTERNO-MONTANTE?}(\mathbb{PE}^{k}_{i_r,r}) \wedge$
$(\neg\ \texttt{T-CONFLITO-GLOBAL-MONTANTE?}(\mathbb{PE}^{k}_{i_r,r}))$.

4.4.5.3 Problema de Escalonamento de Agente de Retalho

Os agentes de retalho têm uma *perspectiva mista* de um problema de escalonamento, *i.e.*, uma perspectiva que é, em parte global e, em parte, local. Para um agente de retalho genérico g_r, um problema de escalonamento $\mathbb{PE}^{r}_{i_r,r}$ é definido pelo tuplo de cinco elementos:

$$\texttt{<Id-PE,}\texttt{DD}^{r}_{i_r,r}\texttt{,}\texttt{RD}^{r}_{i_r,r}\texttt{,}\mathbb{d}^{0,r}_{i_r,r}\texttt{,}\mathbb{d}^{r,k}_{i_r,r}\texttt{>}$$

em que:

- `Id-PE` é um identificador que associa o problema de escalonamento do agente às conversações relevantes, associadas ao problema, com o supervisor e com o fornecedor e ao problema de escalonamento global;
- $\texttt{DD}^{r}_{i_r,r}$ é a data limite mais tarde;
- $\texttt{RD}^{r}_{i_r,r}$ é a data de lançamento mais cedo;
- $\mathbb{d}^{0,r}_{i_r,r}$ é o pedido global do exterior à rede;
- $\mathbb{d}^{r,k}_{i_r,r}$ é o pedido local para o agente gestor g_k, fornecedor do produto do pedido $\mathbb{d}^{0,r}_{i_r,r}$.

`Id-PE`, $\mathbb{d}^{0,r}_{i_r,r}$ e $\texttt{DD}^{r}_{i_r,r}$ são dados pelo agente supervisor e $\texttt{RD}^{r}_{i_r,r}$ é calculada através de $\mathbb{d}^{r,k}_{i_r,r}$ e de $\texttt{FEM}^{r}_{i_r,r}$. Esta última folga é obtida do fornecedor g_k. A folga $\texttt{FEJ}^{r}_{i_r,r}$, a comunicar ao fornecedor g_k é calculável usando $\texttt{DD}^{r}_{i_r,r}$ e $\mathbb{d}^{0,r}_{i_r,r}$. A folga total $\texttt{FT}^{r}_{i_r,r}$, a comunicar ao agente supervisor, é calculável através das folgas $\texttt{FEJ}^{r}_{i_r,r}$ e $\texttt{FEM}^{r}_{i_r,r}$. O pedido para o fornecedor $\mathbb{d}^{r,k}_{i_r,r}$, é construído a partir de $\mathbb{d}^{0,r}_{i_r,r}$ e a tarefa de retalho $\mathbb{O}^{r}_{i_r,r}$ é representada pelo pedido $\mathbb{d}^{0,r}_{i_r,r}$ ($\mathbb{O}^{r}_{i_r,r}$=`TAREFA`($\mathbb{d}^{0,r}_{i_r,r}$)).

4.4.5.4 Operações com Problemas de Escalonamento de Agente de Retalho

Definem-se a seguir as operações selectoras de problema de escalonamento de agente de retalho.

$$\texttt{ID-PE(<Id-PE,}\texttt{DD}^{r}_{i_r,r}\texttt{,}\texttt{RD}^{r}_{i_r,r}\texttt{,}\mathbb{d}^{0,r}_{i_r,r}\texttt{,}\mathbb{d}^{r,k}_{i_r,r}\texttt{>) ::= Id-PE.}$$

$$\texttt{DD(<Id-PE,}\texttt{DD}^{r}_{i_r,r}\texttt{,}\texttt{RD}^{r}_{i_r,r}\texttt{,}\mathbb{d}^{0,r}_{i_r,r}\texttt{,}\mathbb{d}^{r,k}_{i_r,r}\texttt{>) ::=}\ \texttt{DD}^{r}_{i_r,r}\texttt{.}$$

$\mathtt{RD(<Id\text{-}PE,DD}^{r}_{i_r,r}\mathtt{,RD}^{r}_{i_r,r}\mathtt{,}\mathbb{d}^{0,r}_{i_r,r}\mathtt{,}\mathbb{d}^{r,k}_{i_r,r}\mathtt{>)\ ::=\ RD}^{r}_{i_r,r}\mathtt{.}$

$\mathtt{PEDIDO\text{-}DE\text{-}JUSANTE(<Id\text{-}PE,DD}^{r}_{i_r,r}\mathtt{,RD}^{r}_{i_r,r}\mathtt{,}\mathbb{d}^{0,r}_{i_r,r}\mathtt{,}\mathbb{d}^{r,k}_{i_r,r}\mathtt{>)\ ::=}\ \mathbb{d}^{0,r}_{i_r,r}\mathtt{.}$

$\mathtt{PEDIDO\text{-}A\text{-}MONTANTE(<Id\text{-}PE,DD}^{r}_{i_r,r}\mathtt{,RD}^{r}_{i_r,r}\mathtt{,}\mathbb{d}^{0,r}_{i_r,r}\mathtt{,}\mathbb{d}^{r,k}_{i_r,r}\mathtt{>)\ ::=}\ \mathbb{d}^{r,k}_{i_r,r}\mathtt{.}$

A operação construtora é:

$\mathtt{PE(Id\text{-}PE,DD}^{r}_{i_r,r}\mathtt{,RD}^{r}_{i_r,r}\mathtt{,}\mathbb{d}^{0,r}_{i_r,r}\mathtt{,}\mathbb{d}^{r,k}_{i_r,r}\mathtt{)\ ::=\ <Id\text{-}PE,DD}^{r}_{i_r,r}\mathtt{,RD}^{r}_{i_r,r}\mathtt{,}\mathbb{d}^{0,r}_{i_r,r}\mathtt{,}\mathbb{d}^{r,k}_{i_r,r}\mathtt{>.}$

Adicionalmente, a operação TAREFA é definida, para qualquer problema de escalonamento de agente de retalho $\mathbb{PE}^{r}_{i_r,r}$:

$\mathtt{TAREFA(}\mathbb{PE}^{r}_{i_r,r}\mathtt{)\ ::=\ TAREFA(PEDIDO\text{-}DE\text{-}JUSANTE(}\mathbb{PE}^{r}_{i_r,r}\mathtt{)).}$

A seguir, definem-se operações adicionais para qualquer problema de escalonamento de agente de retalho $\mathbb{PE}^{r}_{i_r,r}$. Estas operações pressupõem as igualdades expostas na Tabela 4-7.

Para as folgas temporais externa a jusante e total, definem-se as operações:

$\mathtt{FEJ(}\mathbb{PE}^{r}_{i_r,r}\mathtt{)\ ::=\ DD(}\mathbb{PE}^{r}_{i_r,r}\mathtt{)\ -\ TEMPO(PEDIDO\text{-}DE\text{-}JUSANTE(}\mathbb{PE}^{r}_{i_r,r}\mathtt{)).}$

$\mathtt{FT(}\mathbb{PE}^{r}_{i_r,r}\mathtt{)\ ::=\ DD(}\mathbb{PE}^{r}_{i_r,r}\mathtt{)\ -\ RD(}\mathbb{PE}^{r}_{i_r,r}\mathtt{).}$

O valor da folga externa a montante $\mathtt{FEM}^{r}_{i_r,r}$, comunicado pelo fornecedor g_j, juntamente com o pedido ao fornecedor $\mathbb{d}^{r,k}_{i_r,r}$, permite calcular a data $\mathtt{RD}^{r}_{i_r,r}$. A seguinte operação descreve este cálculo:

$\mathtt{RD\text{-}CALC(}\mathbb{d}^{r,k}_{i_r,r}\mathtt{,FEM}^{r}_{i_r,r}\mathtt{)\ ::=\ TEMPO(}\mathbb{d}^{r,k}_{i_r,r}\mathtt{)\ -\ FEM}^{r}_{i_r,r}\mathtt{.}$

A seguinte operação permite calcular a folga $\mathtt{FEM}^{r}_{i_r,r}$, dado um problema de escalonamento $\mathbb{PE}^{k}_{i_r,r}$:

$\mathtt{FEM(}\mathbb{PE}^{r}_{i_r,r}\mathtt{)\ ::=\ TEMPO(PEDIDO\text{-}DE\text{-}JUSANTE(}\mathbb{PE}^{r}_{i_r,r}\mathtt{))\ -\ RD(}\mathbb{PE}^{r}_{i_r,r}\mathtt{).}$

Os seguintes predicados testam se existem conflitos temporais num problema de escalonamento $\mathbb{PE}^{r}_{i_r,r}$.

$\mathtt{T\text{-}CONFLITO\text{-}TOTAL?(}\mathbb{PE}^{r}_{i_r,r}\mathtt{)\ ::=\ FT(}\mathbb{PE}^{r}_{i_r,r}\mathtt{)\ <\ 0.}$

T-CONFLITO-JUSANTE?($\mathbb{PE}^{r}_{i_r,r}$) ::= FEJ($\mathbb{PE}^{r}_{i_r,r}$) < 0.

T-CONFLITO-MONTANTE?($\mathbb{PE}^{r}_{i_r,r}$) ::= FEM($\mathbb{PE}^{r}_{i_r,r}$) < 0.

Por último, o predicado T-CONFLITO? testa se existe algum conflito temporal (dos que acima foram referidos):

T-CONFLITO?($\mathbb{PE}^{r}_{i_r,r}$) ::=
T-CONFLITO-TOTAL?($\mathbb{PE}^{r}_{i_r,r}$) $\vee$ T-CONFLITO-JUSANTE?($\mathbb{PE}^{r}_{i_r,r}$) $\vee$
T-CONFLITO-MONTANTE?($\mathbb{PE}^{r}_{i_r,r}$).

4.4.5.5 Problema de Escalonamento de Agente de Matéria-Prima

Os agentes de matéria-prima, tal como os agentes de retalho, têm uma *perspectiva mista* de um problema de escalonamento. Para um agente de matéria-prima genérico g_m, um problema de escalonamento $\mathbb{PE}^{m}_{i_r,r}$ é representado pelo tuplo de cinco elementos:

<Id-PE,$DD^{m}_{i_r,r}$,$RD^{m}_{i_r,r}$,$\mathbb{d}^{k,m}_{i_r,r}$,$\mathbb{d}^{m,0}_{i_r,r}$>

em que:

– Id-PE é um identificador que associa o problema de escalonamento do agente às conversações relevantes com o cliente e com o supervisor e ao problema de escalonamento global;

– $DD^{m}_{i_r,r}$ é a data limite mais tarde;

– $RD^{m}_{i_r,r}$ é a data de lançamento mais cedo;

– $\mathbb{d}^{k,m}_{i_r,r}$ é o pedido local de um agente cliente g_k;

– $\mathbb{d}^{m,0}_{i_r,r}$ é o pedido global da rede ao exterior para satisfazer o pedido local $\mathbb{d}^{k,m}_{i_r,r}$.

Id-PE é obtido a partir da mensagem recebida (do cliente) que estabeleceu a conversação do cliente. $\mathbb{d}^{k,m}_{i_r,r}$ é recebido de g_k, $RD^{m}_{i_r,r}$ é obtida do supervisor e $DD^{m}_{i_r,r}$ é calculada através de $\mathbb{d}^{k,m}_{i_r,r}$ e de $FEJ^{m}_{i_r,r}$. Esta última folga é obtida do cliente g_k. A folga $FEM^{m}_{i_r,r}$, a comunicar ao cliente g_k, é calculável usando $RD^{m}_{i_r,r}$ e $\mathbb{d}^{k,m}_{i_r,r}$. A folga total $FT^{m}_{i_r,r}$ é calculável através das folgas $FEJ^{m}_{i_r,r}$ e $FEM^{m}_{i_r,r}$. O pedido para o agente supervisor $\mathbb{d}^{m,0}_{i_r,r}$ é construído a partir de $\mathbb{d}^{k,m}_{i_r,r}$. A tarefa de matéria-prima $\mathbb{O}^{m}_{i_r,r}$ é representada pelo pedido $\mathbb{d}^{k,m}_{i_r,r}$ ($\mathbb{O}^{m}_{i_r,r}$=TAREFA($\mathbb{d}^{k,m}_{i_r,r}$)).

4.4.5.6 Operações com Problemas de Escalonamento de Agente de Matéria-Prima

Definem-se a seguir operações com problemas de escalonamento de agente de matéria-prima:

$\texttt{ID-PE(<Id-PE,DD}^{m}_{i_r,r}\texttt{,RD}^{m}_{i_r,r}\texttt{,}\mathbb{d}^{k,m}_{i_r,r}\texttt{,}\mathbb{d}^{m,0}_{i_r,r}\texttt{>) ::= Id-PE.}$

$\texttt{DD(<Id-PE,DD}^{m}_{i_r,r}\texttt{,RD}^{m}_{i_r,r}\texttt{,}\mathbb{d}^{k,m}_{i_r,r}\texttt{,}\mathbb{d}^{m,0}_{i_r,r}\texttt{>) ::= DD}^{m}_{i_r,r}\texttt{.}$

$\texttt{RD(<Id-PE,DD}^{m}_{i_r,r}\texttt{,RD}^{m}_{i_r,r}\texttt{,}\mathbb{d}^{k,m}_{i_r,r}\texttt{,}\mathbb{d}^{m,0}_{i_r,r}\texttt{>) ::= RD}^{m}_{i_r,r}\texttt{.}$

$\texttt{PEDIDO-DE-JUSANTE(<Id-PE,DD}^{m}_{i_r,r}\texttt{,RD}^{m}_{i_r,r}\texttt{,}\mathbb{d}^{k,m}_{i_r,r}\texttt{,}\mathbb{d}^{m,0}_{i_r,r}\texttt{>) ::= }\mathbb{d}^{k,m}_{i_r,r}\texttt{.}$

$\texttt{PEDIDO-A-MONTANTE(<Id-PE,DD}^{m}_{i_r,r}\texttt{,RD}^{m}_{i_r,r}\texttt{,}\mathbb{d}^{k,m}_{i_r,r}\texttt{,}\mathbb{d}^{m,0}_{i_r,r}\texttt{>) ::= }\mathbb{d}^{m,0}_{i_r,r}\texttt{.}$

A operação construtora é:

$\texttt{PE(Id-PE,DD}^{m}_{i_r,r}\texttt{,RD}^{m}_{i_r,r}\texttt{,}\mathbb{d}^{k,m}_{i_r,r}\texttt{,}\mathbb{d}^{m,0}_{i_r,r}\texttt{) ::= <Id-PE,DD}^{m}_{i_r,r}\texttt{,RD}^{m}_{i_r,r}\texttt{,}\mathbb{d}^{k,m}_{i_r,r}\texttt{,}\mathbb{d}^{m,0}_{i_r,r}\texttt{>.}$

Adicionalmente, a operação TAREFA é também definida para qualquer problema de escalonamento de agente de matéria-prima $\mathbb{PE}^{m}_{i_r,r}$:

$\texttt{TAREFA(}\mathbb{PE}^{m}_{i_r,r}\texttt{) ::= TAREFA(PEDIDO-DE-JUSANTE(}\mathbb{PE}^{m}_{i_r,r}\texttt{)).}$

A seguir, definem-se operações adicionais para qualquer problema de escalonamento de agente de matéria-prima $\mathbb{PE}^{m}_{i_r,r}$. Estas operações pressupõem as igualdades expostas na Tabela 4-8.

Para as folgas temporais externa a montante e total, definem-se as operações:

$\texttt{FEM(}\mathbb{PE}^{m}_{i_r,r}\texttt{) ::= TEMPO(PEDIDO-DE-JUSANTE(}\mathbb{PE}^{m}_{i_r,r}\texttt{)) - RD(}\mathbb{PE}^{m}_{i_r,r}\texttt{).}$

$\texttt{FT(}\mathbb{PE}^{m}_{i_r,r}\texttt{) ::= DD(}\mathbb{PE}^{m}_{i_r,r}\texttt{) - RD(}\mathbb{PE}^{m}_{i_r,r}\texttt{).}$

O valor da folga externa a jusante $\texttt{FEJ}^{m}_{i_r,r}$, comunicado pelo cliente g_k, juntamente com o pedido do cliente $\mathbb{d}^{k,m}_{i_r,r}$, permite calcular a data $\texttt{DD}^{m}_{i_r,r}$. A seguinte operação descreve este cálculo:

$\texttt{DD-CALC(}\mathbb{d}^{k,m}_{i_r,r}\texttt{,FEJ}^{m}_{i_r,r}\texttt{) ::= TEMPO(}\mathbb{d}^{k,m}_{i_r,r}\texttt{) + FEJ}^{m}_{i_r,r}\texttt{.}$

A seguinte operação permite calcular a folga $\texttt{FEM}^{m}_{i_r,r}$ dado um problema de escalonamento $\mathbb{PE}^{m}_{i_r,r}$:

$\texttt{FEJ(}\mathbb{PE}^{m}_{i_r,r}\texttt{) ::= DD(}\mathbb{PE}^{m}_{i_r,r}\texttt{) - TEMPO(PEDIDO-DE-JUSANTE(}\mathbb{PE}^{m}_{i_r,r}\texttt{)).}$

Os predicados T-CONFLITO-TOTAL?, T-CONFLITO-JUSANTE?, T-CONFLITO-MONTANTE? e T-CONFLITO? testam se existem conflitos temporais num dado problema de escalonamento de agente de matéria-prima e definem-se da mesma forma que as operações com os mesmos identificadores para problemas de escalonamento de agente de retalho.

4.4.5.7 Problema de Escalonamento de Agente Supervisor

O agente supervisor tem uma *perspectiva global* de um problema de escalonamento. Para o agente supervisor g_0, um problema de escalonamento $\mathbb{PE}_{i_r,r}$ é representado por um tuplo de seis elementos:

$<\text{Id-PE}, DD_{i_r,r}, RD_{i_r,r}, \mathbb{d}^{0,r}_{i_r,r}, \mathcal{d}'^{0}_{i_r,r}, RD^{0}_{i_r,r}>$

em que:

- Id-PE é um identificador que identifica univocamente o problema de escalonamento global $\mathbb{PE}_{i_r,r}$;
- $DD_{i_r,r}$ é a data limite global (e também a data mais tarde para $\mathbb{d}^{0,r}_{i_r,r}$);
- $RD_{i_r,r}$ é a data de lançamento global;
- $\mathbb{d}^{0,r}_{i_r,r}$ é o pedido global do exterior à rede;
- $\mathcal{d}'^{0}_{i_r,r}$ ($\mathcal{d}'^{0}_{i_r,r}=\{\mathbb{d}^{1,0}_{i_r,r}, ..., \mathbb{d}^{m,0}_{i_r,r}, ..., \mathbb{d}^{Nm_{i_r,r},0}_{i_r,r}\}$) é o conjunto de todos os pedidos globais da rede ao exterior originados pelos agentes de matéria-prima g_m (com $m=1,...,Nm_{i_r,r}$), necessários para que a rede satisfaça o pedido global $\mathbb{d}^{0,r}_{i_r,r}$;
- $RD^{0}_{i_r,r}$ é a data mais cedo para $\mathbb{d}^{0,r}_{i_r,r}$.

Id-PE é um identificador novo, gerado pelo agente supervisor de cada vez que cria um problema de escalonamento novo.[63] $\mathbb{d}^{0,r}_{i_r,r}$, $DD_{i_r,r}$ e $RD_{i_r,r}$ são dados globais, enquanto que $\mathcal{d}'^{0}_{i_r,r}$ e $RD^{0}_{i_r,r}$ são parte de uma solução do problema de escalonamento. O agente supervisor g_0, comunica $\mathbb{d}^{0,r}_{i_r,r}$ e $DD_{i_r,r}$ ao agente de retalho g_r e comunica $RD_{i_r,r}$ a cada um dos agentes de matéria-prima envolvidos g_m (com $m=1,...,Nm_{i_r,r}$). Os pedidos em $\mathcal{d}'^{0}_{i_r,r}$ são recebidos por g_0 destes agentes de matéria-prima. A data $RD^{0}_{i_r,r}$ é calculada com base nos valores de $DD_{i_r,r}$ e da folga total $FT_{i_r,r}$ fornecida pelo agente de retalho g_r. Para calcular os valores das folgas $FEJ_{i_r,r}$, $FEM_{i_r,r}$, $FET_{i_r,r}$ e $FIT_{i_r,r}$ o agente supervisor faz uso das datas $DD_{i_r,r}$, $RD_{i_r,r}$, da folga $FT_{i_r,r}$, do pedido $\mathbb{d}^{0,r}_{i_r,r}$ e dos pedidos em $\mathcal{d}'^{0}_{i_r,r}$.

4.4.5.8 Operações com Problemas de Escalonamento de Agente Supervisor

Definem-se a seguir operações de problema de escalonamento de agente supervisor:

---

[63] Através da operação GERA-ID-PE, ver a secção 4.4.4.1.

$\texttt{ID-PE(<Id-PE,DD}_{i_r,r}\texttt{,RD}_{i_r,r}\texttt{,}\mathbb{d}^{0,r}_{i_r,r}\texttt{,}\mathcal{d}'^{0}_{i_r,r}\texttt{,RD}^{0}_{i_r,r}\texttt{>) ::= Id-PE.}$

$\texttt{DD(<Id-PE,DD}_{i_r,r}\texttt{,RD}_{i_r,r}\texttt{,}\mathbb{d}^{0,r}_{i_r,r}\texttt{,}\mathcal{d}'^{0}_{i_r,r}\texttt{,RD}^{0}_{i_r,r}\texttt{>) ::= DD}_{i_r,r}\texttt{.}$

$\texttt{RD(<Id-PE,DD}_{i_r,r}\texttt{,RD}_{i_r,r}\texttt{,}\mathbb{d}^{0,r}_{i_r,r}\texttt{,}\mathcal{d}'^{0}_{i_r,r}\texttt{,RD}^{0}_{i_r,r}\texttt{>) ::= RD}_{i_r,r}\texttt{.}$

$\texttt{RD-PEDIDO(<Id-PE,DD}_{i_r,r}\texttt{,RD}_{i_r,r}\texttt{,}\mathbb{d}^{0,r}_{i_r,r}\texttt{,}\mathcal{d}'^{0}_{i_r,r}\texttt{,RD}^{0}_{i_r,r}\texttt{>) ::= RD}^{0}_{i_r,r}\texttt{.}$

$\texttt{PEDIDO-DE-JUSANTE(<Id-PE,DD}_{i_r,r}\texttt{,RD}_{i_r,r}\texttt{,}\mathbb{d}^{0,r}_{i_r,r}\texttt{,}\mathcal{d}'^{0}_{i_r,r}\texttt{,RD}^{0}_{i_r,r}\texttt{>) ::= }\mathbb{d}^{0,r}_{i_r,r}\texttt{.}$

$\texttt{PEDIDOS-A-MONTANTE(<Id-PE,DD}_{i_r,r}\texttt{,RD}_{i_r,r}\texttt{,}\mathbb{d}^{0,r}_{i_r,r}\texttt{,}\mathcal{d}'^{0}_{i_r,r}\texttt{,RD}^{0}_{i_r,r}\texttt{>) ::= }\mathcal{d}'^{0}_{i_r,r}\texttt{.}$

A operação construtora é:

$\texttt{PE(Id-PE,DD}_{i_r,r}\texttt{,RD}_{i_r,r}\texttt{,}\mathbb{d}^{0,r}_{i_r,r}\texttt{,}\mathcal{d}'^{0}_{i_r,r}\texttt{,RD}^{0}_{i_r,r}\texttt{) ::=}$
$\texttt{<Id-PE,DD}_{i_r,r}\texttt{,RD}_{i_r,r}\texttt{,}\mathbb{d}^{0,r}_{i_r,r}\texttt{,}\mathcal{d}'^{0}_{i_r,r}\texttt{,RD}^{0}_{i_r,r}\texttt{>.}$

Adicionalmente, definem-se as seguintes operações para qualquer problema de escalonamento de agente supervisor $\mathbb{PE}_{i_r,r}$ (estas operações pressupõem as igualdades expostas na Tabela 4-9).

$\texttt{FEJ(}\mathbb{PE}_{i_r,r}\texttt{) ::= DD(}\mathbb{PE}_{i_r,r}\texttt{) - TEMPO(}\mathbb{d}^{0,r}_{i_r,r}\texttt{).}$

$\texttt{FEM(}\mathbb{PE}_{i_r,r}\texttt{) ::=}$
$\texttt{MIN(TEMPO(}\mathbb{d}^{m,0}_{i_r,r}\texttt{) - RD(}\mathbb{PE}_{i_r,r}\texttt{)) : }\mathbb{d}^{m,0}_{i_r,r}\texttt{∈PEDIDOS-A-MONTANTE(}\mathbb{PE}_{i_r,r}\texttt{).}$

$\mathtt{FT}(\mathbb{PE}_{i_r,r})$ ::= $\mathtt{DD}(\mathbb{PE}_{i_r,r})$ - RD-PEDIDO($\mathbb{PE}_{i_r,r}$).

$\mathtt{FET}(\mathbb{PE}_{i_r,r})$ ::= $\mathtt{FEJ}(\mathbb{PE}_{i_r,r})$ + $\mathtt{FEM}(\mathbb{PE}_{i_r,r})$.

$\mathtt{FIT}(\mathbb{PE}_{i_r,r})$ ::= $\mathtt{FT}(\mathbb{PE}_{i_r,r})$ - $\mathtt{FET}(\mathbb{PE}_{i_r,r})$.

Para calcular $\mathtt{RD}^0_{i_r,r}$ o agente supervisor usa a data limite global e a folga total comunicada pelo agente de retalho. A operação que descreve este cálculo é:

RD-PEDIDO-CALC($\mathtt{DD}_{i_r,r}$,$\mathtt{FT}^r_{i_r,r}$) ::= $\mathtt{DD}_{i_r,r}$ - $\mathtt{FT}^r_{i_r,r}$.

Os predicados T-CONFLITO-TOTAL?, T-CONFLITO-JUSANTE?, T-CONFLITO-MONTANTE? e T-CONFLITO? testam se existem conflitos temporais num dado problema de escalonamento de agente supervisor e definem-se da mesma forma que as operações com os mesmos identificadores para problemas de escalonamento de agente de retalho ou de matéria-prima.

4.4.6 Conversação

Nesta secção descrevem-se os componentes da arquitectura dos agentes apropriados para a representação interna do estado de interacção. Este estado é registado nos elementos $\mathtt{conv}$ do componente Convs dos agentes, designados por *conversações*. Descreve-se o que são conversações e mensagens, como as conversações são modificadas em função de mensagens recebidas ou enviadas e ainda algumas classes adicionais de objectos relacionadas com conversações.

4.4.6.1 Modelos de Conversação

Para descrever uma conversação é usado um *modelo de conversação* para cada classe de conversação possível. Um modelo de conversação é uma máquina de estados finita e determinística que descreve os estados e as transições entre estados possíveis para instâncias de conversações da mesma classe.[64] Um modelo de conversação é descrito por um tuplo de quatro valores:

<Início,Fim,$\mathit{Estados}$,$\mathit{Transições}$>

Em que:

– Início é um número natural que indica o estado inicial da classe de conversação (Início$\in \mathit{Estados}$);

[64] Ver, por exemplo, [Mandrioli 1987]. Na secção 4.4.7 descrevem-se os modelos de conversação empregues na realização computacional do modelo, para cada classe de agente, incluindo os respectivos diagramas de estado. Dada a simplicidade dos modelos de conversação propostos (o número de estados é limitado e não ocorrem acções em paralelo), são expressos com base em máquinas de estado finitas. Para modelar comportamentos mais complexos, no entanto, autómatos de registos (*augmented transition networks*, ou ATNs) ou redes de Petri, seriam preferíveis.

– `Fim` é um número natural que indica o estado final da classe de conversação (`Fim`$\in$ *Estados* e `Fim`$\neq$`Início`);

– *Estados* é o conjunto de estados possíveis do modelo de conversação, um subconjunto finito dos números naturais, *i.e.*, *Estados*=`{...,Estado`$_i$`,...}`, em que `Estado`$_i$ é um número natural;

– *Transições* descreve o grafo de estados do modelo de conversação e é um conjunto finito de transições possíveis, indicando os caminhos possíveis desde o estado inicial (`Início`) até ao estado final (`Fim`), dependendo da classe de conversação. *Transições*=`{...,`transição$_k$`,...}`, em que cada elemento transição$_k$ é uma transição (ver mais adiante).

São usados modelos de conversação diferentes para cada classe de agente. A comunicação entre agentes para coordenação no escalonamento de tarefas baseia-se na teoria dos *actos do discurso* (ver [Searle 1969]). No contexto de um modelo de conversação particular, cada transição do grafo de estados é associada a um *acto*[65] através de uma *etiqueta* que indica o tipo de mensagem e se a mensagem é recebida ou enviada pelo agente. Cada etiqueta é um par:

`<Sentido,Tipo-de-mensagem>`

Em que:

– `Sentido`$\in$`{S,R}`, com `S` indicando envio e `R` indicando recepção de mensagem;

– `Tipo-de-mensagem` é a classe de mensagem (existindo um número fixo de classes de mensagens, ver adiante).

Cada elemento transição$_k$, do componente *Transições* de um modelo de conversação, é descrito por um triplo:

`<Partida,Chegada,`etiqueta`>`

Em que:

– `Partida` é o estado de partida da transição (`Partida`$\in$ *Estados*);

– `Chegada` é o estado de chegada da transição (`Chegada`$\in$ *Estados*);

– etiqueta é uma etiqueta que indica o acto associado à recepção ou envio de uma mensagem, que permite efectuar a transição do estado `Partida` para o estado `Chegada`.

Os modelos de conversação devem ser construídos de modo a que, no mesmo modelo, dado um qualquer estado de partida do modelo e uma etiqueta de transição partindo desse estado, o estado de chegada é único (a máquina de estados do modelo de conversação é determinística). Para isso, transições para estados diferentes a partir de um mesmo estado de partida devem ter etiquetas diferentes (etiquetas com sentidos e/ou tipos de mensagens associados diferentes). Adicionalmente, cada modelo do conjunto dos modelos de conversação conhecido por uma classe de agentes, deve ser distinto dos restantes no que respeita às transições iniciais (os conjuntos de transições partindo do estado inicial de cada modelo devem ser disjuntos).

---

[65] Em geral, um acto de discurso, mas existe, em particular, um tipo de mensagem que não significa um acto de discurso, mas sim um acto físico (a mensagem do tipo `satisfação`, ver a secção 4.4.7).

4.4.6.2 Operações com Etiquetas, Transições e Modelos de Conversação

Descrevem-se aqui as operações necessárias para operar em etiquetas, transições e modelos de conversação. Para etiquetas tem-se:

```
SENTIDO(<Sentido,Tipo-de-mensagem>) ::= Sentido.

TIPO-DE-MENSAGEM(<Sentido,Tipo-de-mensagem>) ::=
Tipo-de-mensagem.

ETIQUETA(Sentido,Tipo-de-mensagem) ::=
<Sentido,Tipo-de-mensagem>.
```

Para transições tem-se:

```
PARTIDA(<Partida,Chegada,etiqueta>) ::= Partida.

CHEGADA(<Partida,Chegada,etiqueta>) ::= Chegada.

ETIQUETA(<Partida,Chegada,etiqueta>) ::= etiqueta.

TRANSIÇÃO(Partida,Chegada,etiqueta) ::=
<Partida,Chegada,etiqueta>.
```

Para modelos de conversação têm-se as seguintes operações primitivas:

```
ESTADO-INICIAL(<Início,Fim,Estados,Transições>) ::= Início.

ESTADO-FINAL(<Início,Fim,Estados,Transições>) ::= Fim.

ESTADOS(<Início,Fim,Estados,Transições>) ::= Estados.

TRANSIÇÕES(<Início,Fim,Estados,Transições>) ::= Transições.

MODELO-DE-CONVERSAÇÃO(Início,Fim,Estados,Transições) ::=
<Início,Fim,Estados,Transições>.
```

A operação TRANSIÇÕES-POSSÍVEIS devolve o conjunto de transições possíveis de um modelo dado Modelo, a partir de um estado dado Estado, tal que Estado∈ESTADOS(Modelo):

```
TRANSIÇÕES-POSSÍVEIS(Modelo,Estado) ::=
{ transição_k :    (transição_k∈TRANSIÇÕES(Modelo)) ∧
                   (PARTIDA(transição_k)=Estado) }.
```

O predicado TRANSITÁVEL? testa se, para um dado modelo de conversação, existe uma transição partindo de um estado com uma determinada etiqueta:

```
TRANSITÁVEL?(Modelo,Estado,etiqueta) ::=
(Estado∈ESTADOS(Modelo)) ∧ (Estado≠ESTADO-FINAL(Modelo)) ∧
(∃transição_k : (transição_k∈TRANSIÇÕES-POSSÍVEIS(Modelo,Estado)) ∧
                (ETIQUETA(transição_k)=etiqueta)).
```

O predicado TRANSIÇÃO-INICIAL? testa se, para um dado modelo de conversação, existe uma transição partindo do estado inicial com uma dada etiqueta associada. Esta operação pode ser usada para identificar, num conjunto de modelos de conversação conhecidos, qual o modelo de conversação da conversação iniciada por uma mensagem (cujo sentido e tipo são os da etiqueta dada como parâmetro).

```
TRANSIÇÃO-INICIAL?(Modelo,etiqueta) ::=
TRANSITÁVEL?(Modelo,ESTADO-INICIAL(Modelo),etiqueta).
```

Para um modelo, um estado e uma etiqueta dados, a operação PRÓXIMO-ESTADO devolve o estado de chegada da transição do modelo que tem como estado de partida e etiqueta o estado e a etiqueta dados.

```
PRÓXIMO-ESTADO(Modelo,Estado,etiqueta) ::=
EstadoX :
 (EstadoX∈ESTADOS(Modelo)) ∧
 (∃transição_k : (transição_k∈TRANSIÇÕES-POSSÍVEIS(Modelo,Estado)) ∧
                 (ETIQUETA(transição_k)=etiqueta) ∧
                 (CHEGADA(transição_k)=EstadoX)).
```

A operação PRÓXIMA-TRANSIÇÃO aceita os mesmos parâmetros que a anterior mas devolve, em vez do estado de chegada, a transição que leva ao estado de chegada.

```
PRÓXIMA-TRANSIÇÃO(Modelo,Estado,etiqueta) ::=
transição_k : (transição_k∈TRANSIÇÕES-POSSÍVEIS(Modelo,Estado)) ∧
              (ETIQUETA(transição_k)=etiqueta)).
```

Tanto PRÓXIMO-ESTADO como PRÓXIMA-TRANSIÇÃO apenas são aplicáveis se, para o modelo dado, houver uma transição partindo do estado dado com a etiqueta dada (*i.e.*, se se verificar TRANSITÁVEL?(Modelo,Estado,etiqueta)).

4.4.6.3 Mensagens

As mensagens que os agentes podem enviar e/ou receber têm associado um *tipo*, ou classe, de mensagem, que identifica a "performativa", *i.e.*, o acto de discurso associado à mensagem (ver o Apêndice D). No total, para todas as classes de agentes, existem onze tipos diferentes de mensagens: pedido, aceitação, rejeição, satisfação, cancelamento,

`pedido-re`, `aceitação-re`, `rejeição-re`, `re-escalonamento`, `pedido-in` e `pedido-out`. Estes tipos de mensagens enquadram-se em diferentes protocolos de interacção entre os agentes e cada agente conhece apenas um subconjunto deste conjunto de tipos de mensagens, que pode utilizar em conversações com outros agentes.[66] As mensagens dos tipos `pedido-in` e `pedido-out` são distintas das restantes e só são conhecidas do agentes supervisor, sendo apenas usadas para comunicar com o exterior da rede de EE.

Exceptuando mensagens dos tipos `pedido-in` e `pedido-out`, uma mensagem é um tuplo de cinco elementos:

`<Tipo-de-mensagem,De,Para,Id-PE,`*Conteúdo*`>`

Em que:

- `Tipo-de-mensagem` identifica o acto associado à mensagem (os tipos de mensagens possíveis são `pedido`, `aceitação`, `rejeição`, `satisfação`, `cancelamento`, `pedido-re`, `aceitação-re`, `rejeição-re` ou `re-escalonamento`);
- `De` identifica o agente que enviou a mensagem (`De`$\in g$);
- `Para` identifica o agente a que é enviada a mensagem (`Para`$\in g$);
- `Id-PE` é o identificador de problema de escalonamento ao qual a mensagem se refere e que identifica também a conversação entre os agentes `De` e `Para`, no contexto da qual a mensagem é trocada (esta pode ser uma conversação preexistente, ou uma conversação nova que a mensagem estabelece);
- *Conteúdo* é o conteúdo da mensagem, um conjunto ordenado cujos elementos dependem do tipo de mensagem.

Apenas para mensagens dos tipos `pedido-in` e `pedido-out`, uma mensagem é um par:

`<Tipo-de-mensagem,`Conteúdo`>`

Em que:

- `Tipo-de-mensagem` identifica o acto associado à mensagem (tipos de mensagens possíveis: `pedido-in` ou `pedido-out`);
- Conteúdo ou contém informação associada a um pedido do exterior à rede, no caso de uma mensagem de tipo `pedido-in`, ou a um pedido da rede ao exterior, no caso de uma mensagem de tipo `pedido-out`.

---

[66] As operações que produzem os conjuntos de tipos de mensagens que cada agente pode receber ou enviar são definidas nas secções 4.4.1, 4.4.2, 4.4.3 e 4.4.4 e os protocolos de interacção que empregam aqueles tipos de mensagens são descritos em detalhe na secção 4.4.7. A utilização de um protocolo de comunicação entre agentes como o KQML, encarado como padrão de linguagem de comunicação inter-agente na área de IAD, foi posta de parte já que se pretendia um conjunto limitado de performativas específicas do domínio do escalonamento, para comunicação entre um grupo de agentes cooperantes fechado e relativamente homogéneo. O KQML é bastante genérico: contém cerca de 40 tipos de mensagens diferentes e foi pensado para abranger comunicação entre agentes de sistemas multi-agente heterogéneos (ver [Finin 1993] e ver o Apêndice D). O formato das mensagens é, no entanto, inspirado no do KQML, simplificado (os componentes de mensagem `De`, `Para` e *Conteúdo* correspondem aos campos de mensagem de KQML `:sender`, `:receiver` e `:content`, respectivamente; o componente `Id-PE` corresponde a ambos os campos `:reply-with` e `:in-reply-to`, no KQML; não existem componentes correspondentes aos campos `:language`, `:ontology` e `:force` do KQML).

O conteúdo dos diferentes tipos de mensagens no contexto de diferentes protocolos de interacção entre os agentes de diferentes classes, bem como os tipos de mensagens `pedido-in` e `pedido-out`, que não fazem parte de nenhum protocolo, é descrito na secção 4.4.7.

4.4.6.4 Operações com Mensagens

Definem-se as seguintes operações com mensagens de qualquer tipo, à excepção dos tipos `pedido-in` e `pedido-out`:

`TIPO-DE-MENSAGEM(<Tipo-de-mensagem,De,Para,Id-PE,`*Conteúdo*`>) ::= Tipo-de-mensagem.`

`ORIGEM(<Tipo-de-mensagem,De,Para,Id-PE,`*Conteúdo*`>) ::= De.`

`DESTINO(<Tipo-de-mensagem,De,Para,Id-PE,`*Conteúdo*`>) ::= Para.`

`ID-PE(<Tipo-de-mensagem,De,Para,Id-PE,`*Conteúdo*`>) ::= Id-PE.`

`CONTEÚDO(<Tipo-de-mensagem,De,Para,Id-PE,`*Conteúdo*`>) ::=` *Conteúdo*`.`

`MENSAGEM(Tipo-de-mensagem,De,Para,Id-PE,`*Conteúdo*`) ::= <Tipo-de-mensagem,De,Para,`*Conteúdo*`>.`

Nas mensagens dos tipos `pedido` e `pedido-re`, o componente *Conteúdo* contém um pedido. Apenas para estes tipos de mensagens, assume-se como definida uma operação selectora identificada por `PEDIDO`, que permite aceder ao pedido nelas contido. Também, nas mensagens dos tipos `pedido`, `pedido-re` e `aceitação` o *Conteúdo* contém uma folga temporal. Para estes tipos de mensagens, assume-se como definida uma operação selectora identificada por `FOLGA`, que permite aceder a esta folga.

Para mensagens dos tipos `pedido-in` e `pedido-out` definem-se as operações seguintes:

`TIPO-DE-MENSAGEM(<Tipo-de-mensagem,`Conteúdo`>) ::= Tipo-de-mensagem.`

`CONTEÚDO(<Tipo-de-mensagem,`Conteúdo`>) ::=` Conteúdo`.`

`MENSAGEM(Tipo-de-mensagem,`Conteúdo`) ::= <Tipo-de-mensagem,`Conteúdo`>.`

As mensagens dos tipos `pedido-in` e `pedido-out` contêm um pedido. Para estes tipos de mensagens, assume-se como definida uma operação selectora identificada por `PEDIDO`, que permite aceder ao pedido nelas contido. No caso particular das mensagens de tipo `pedido-in`, o conteúdo da mensagem inclui também um intervalo de horizonte temporal

global $\mathbb{H}$. Para este tipo de mensagens, assume-se como definida uma operação selectora identificada por HORIZONTE que permite aceder ao intervalo nelas contido.

4.4.6.5 Conversações

Uma *conversação* é um tuplo de oito elementos:

<Id-PE,Modelo,De,Para,Estado,*Transições*,*Msg-in*,*Msg-out*>

Em que:

- Id-PE é o identificador do problema de escalonamento a que a conversação diz respeito;
- Modelo é um modelo de conversação inter-agente (descrito anteriormente). Existe um conjunto de modelos de conversação para cada classe de agente e Modelo é um desses modelos;
- De é o identificador do agente que iniciou a conversação (De$\in g$);
- Para é o identificador do agente interlocutor do agente que iniciou a conversação (Para$\in g$);
- Estado é um número natural que indica o estado actual da conversação e que é um estado possível de Modelo (*i.e.*, Estado$\in$ESTADOS(Modelo));
- *Transições* é o conjunto ordenado das transições efectuadas desde o estado inicial da conversação até ao estado actual, pela ordem em que foram efectuadas;
- *Msg-in* é o conjunto actual de mensagens recebidas pelo agente no contexto da conversação. Este conjunto é ordenado pela ordem temporal de recepção das mensagens;[67]
- *Msg-out* é o conjunto actual de mensagens enviadas pelo agente no contexto da conversação. Este conjunto é ordenado pela ordem temporal de envio das mensagens.[67]

4.4.6.6 Operações com Conversações

Definem-se a seguintes operações selectoras para conversações:

```
ID-PE(
    <Id-PE,Modelo,De,Para,Estado,Transições,Msg-in,Msg-out>) ::=
Id-PE.

MODELO-DE-CONVERSAÇÃO(
    <Id-PE,Modelo,De,Para,Estado,Transições,Msg-in,Msg-out) ::=
Modelo.

INICIANTE(
    <Id-PE,Modelo,De,Para,Estado,Transições,Msg-in,Msg-out>) ::=
De.
```

[67] Tal como as caixas de correio dos agentes.

```
DESTINATÁRIO(
    <Id-PE,Modelo,De,Para,Estado,Transições,Msg-in,Msg-out>) ::=
Para.

ESTADO(
    <Id-PE,Modelo,De,Para,Estado,Transições,Msg-in,Msg-out>) ::=
Estado.

TRANSIÇÕES(
    <Id-PE,Modelo,De,Para,Estado,Transições,Msg-in,Msg-out>) ::=
Transições.

MENSAGENS-RECEBIDAS(
    <Id-PE,Modelo,De,Para,Estado,Transições,Msg-in,Msg-out>) ::=
Msg-in.

MENSAGENS-ENVIADAS(
    <Id-PE,Modelo,De,Para,Estado,Transições,Msg-in,Msg-out>) ::=
Msg-out.
```

A operação primitiva construtora é:

```
CONVERSAÇÃO(
      Id-PE,Modelo,De,Para,Estado,Transições,Msg-in,Msg-out) ::=
<Id-PE,Modelo,De,Para,Estado,Transições,Msg-in,Msg-out>.
```

A operação `CONVERSAÇÃO-INICIAL` cria uma conversação no seu estado inicial:

```
CONVERSAÇÃO-INICIAL(Id-PE,Modelo,De,Para) ::=
CONVERSAÇÃO(  Id-PE,
              Modelo,
              De,
              Para,
              ESTADO-INICIAL(Modelo),
              {},
              {},
              {}).
```

A operação `TRANSIÇÕES-POSSÍVEIS` devolve o conjunto de transições possíveis a partir do estado actual de uma dada conversação `conv`:

```
TRANSIÇÕES-POSSÍVEIS(conv) ::=
TRANSIÇÕES-POSSÍVEIS(MODELO-DE-CONVERSAÇÃO(conv),ESTADO(conv)).
```

O predicado TRANSITÁVEL? testa se, para uma conversação conv existe uma transição que parte do estado actual de *conv* associada à recepção ou envio de uma mensagem msg (relativamente ao parâmetro Sentido deve ser R, para recepção ou S, para envio):

```
TRANSITÁVEL?(conv,Sentido,msg) ::=
TRANSITÁVEL?( MODELO-DE-CONVERSAÇÃO(conv),
              ESTADO(conv),
              ETIQUETA(Sentido,TIPO-DE-MENSAGEM(msg))) ∧
(((ORIGEM(msg)=INICIANTE(conv)) ∧
  (DESTINO(msg)=DESTINATÁRIO(conv))) ∨
 ((ORIGEM(msg)=DESTINATÁRIO(conv)) ∧
  (DESTINO(msg)=INICIANTE(conv)))).
```

A operação TRANSITA exprime a modificação da conversação conv por realização de uma transição possível para o próximo estado, dados uma mensagem msg (recebida ou enviada no contexto da conversação conv) e o sentido da mensagem Sentido (R, para recepção ou S, para envio). Esta operação é aplicável só quando se verifica TRANSITÁVEL?(conv,Sentido,msg) e é definida do seguinte modo:

```
TRANSITA(conv,Sentido,msg) ::=
CONVERSAÇÃO(
    ID-PE(conv),
    MODELO-DE-CONVERSAÇÃO(conv),
    INICIANTE(conv),
    DESTINATÁRIO(conv),
    PRÓXIMO-ESTADO( MODELO-DE-CONVERSAÇÃO(conv),
                    ESTADO(conv),
                    ETIQUETA( Sentido,
                              TIPO-DE-MENSAGEM(msg))),
    JUNTA-NO-FIM(
      TRANSIÇÕES(conv),
      PRÓXIMA-TRANSIÇÃO(  MODELO-DE-CONVERSAÇÃO(conv),
                          ESTADO(conv),
                          ETIQUETA( Sentido,
                                    TIPO-DE-MENSAGEM(msg)))),
    SE( Sentido=R,
        JUNTA-NO-FIM(MENSAGENS-RECEBIDAS(conv),msg),
        MENSAGENS-RECEBIDAS(conv)),
    SE( Sentido=S,
        JUNTA-NO-FIM(MENSAGENS-ENVIADAS(conv), msg),
        MENSAGENS-ENVIADAS(conv))).
```

As duas operações seguintes produzem, para uma dada conversação, a última transição operada e o estado antes da última transição, respectivamente (estas operações são aplicáveis apenas se a conversação está num estado que não é o estado inicial).

```
ÚLTIMA-TRANSIÇÃO(conv) ::= ÚLTIMO(TRANSIÇÕES(conv)).

ÚLTIMO-ESTADO(conv) ::= PARTIDA(ÚLTIMO(TRANSIÇÕES(conv))).
```

O predicado `CONVERSAÇÃO-TERMINADA?` testa se uma dada conversação está terminada, *i.e.*, se está no estado final:

```
CONVERSAÇÃO-TERMINADA?(conv) ::=
(ESTADO(conv)=ESTADO-FINAL(MODELO-DE-CONVERSAÇÃO(conv))).
```

4.4.7 Protocolo de Interacção de Alto Nível

Nesta secção descreve-se o protocolo de interacção entre os agentes de redes de EE. Como se vai descrever, este protocolo traduz-se, efectivamente, num conjunto de vários protocolos apropriados à interacção entre agentes de várias classes.

A actividade inter-agente ocorre através da troca de mensagens entre pares de agentes, no contexto de conversações. O termo conversação designa, no contexto da literatura de IAD, um conjunto de declarações, ou expressões, numa linguagem preestabelecida, de alguma forma comunicadas (proferidas, escritas, codificadas em mensagens) entre um agente *autor* e um agente *interlocutor* ou agentes *interlocutores* que, em geral, se assume ter uma estrutura (uma sequência ou sequências aceitáveis das declarações comunicadas) predefinida.[68] No contexto do presente trabalho define-se uma *conversação* como sendo uma troca de mensagens de tipos predefinidos entre um par de agentes de uma rede de EE, com uma sequência predefinida descrita por um *modelo de conversação*. As conversações têm estados e transições de estado de acordo com o modelo de conversação associado. Como foi descrito anteriormente, cada um dos dois agentes envolvidos numa conversação mantém uma representação actualizada da conversação; estas representações são igualmente apelidadas de *conversações* e são mantidas como elementos do componente *Convs* de cada agente.[69]

Para cada um dos agentes envolvidos numa conversação, cada transição de um estado para outro da conversação ocorre quando do envio ou da recepção de uma mensagem. A transição pode ser originada por causas externas a um agente (recepção de uma mensagem de um cliente ou de um fornecedor) ou por alterações no *estado interno* do agente (possibilidade ou impossibilidade de aceitar ou satisfazer um pedido de um cliente ou de aceitar um pedido de re-escalonamento de um cliente ou fornecedor, necessidade de fazer um pedido de fornecimento, necessidade de pedir o re-escalonamento de um pedido). O estado actual de uma conversação é o estado de interacção do agente com o agente interlocutor, o que inclui o conjunto das mensagens recebidas ou enviadas juntamente com as transições associadas efectuadas desde um estado inicial.

Numa rede de EE, uma conversação é sempre um diálogo entre um par de agentes acerca de um pedido de um dos agentes dirigido ao outro agente. O agente que inicia a conversação é o autor (ou a origem) da primeira mensagem trocada no contexto da conversação e é apelidado de *iniciante* da conversação; o outro agente é o interlocutor (ou o destino) da primeira

---

[68] Ver, por exemplo, [Martial 1992], [Parunak 1996b], [Weiss 1999] ou [Ferber 1999].
[69] Ver as secções 4.4.1, 4.4.2, 4.4.3, 4.4.4 e 4.4.6.

mensagem trocada e é apelidado de *destinatário* da conversação. No caso dos agentes gestores, o iniciante é o agente, do par de agentes, localizado mais a jusante na rede e assume o papel de cliente; o destinatário é o agente localizado mais a montante e assume o papel de um fornecedor. Em todos os modelos de conversação que vão ser definidos, a primeira mensagem do iniciante para o destinatário é uma mensagem de tipo `pedido`, que estabelece um novo pedido e introduz uma nova conversação sobre um novo problema de escalonamento. Estes acontecimentos desencadeiam, no agente destinatário, o início da fase 1 de resolução de problemas de escalonamento para o novo problema de escalonamento.[70]

Uma vez estabelecida, ambos os agentes mantêm uma representação interna actualizada da conversação.[71] Os pares possíveis de agentes comunicantes são:

- dois agentes de rede ligados pela relação cliente-fornecedor (incluídos os casos de um agente de retalho com um agente gestor e de um agente gestor com um agente de matéria-prima);
- o agente supervisor e um agente de retalho;
- o agente supervisor e um agente de matéria-prima.

Um protocolo de interacção descreve um padrão de comunicação através de uma sequência permitida de mensagens trocadas entre agentes, juntamente com as restrições ao conteúdo dessas mensagens.[72] No contexto deste trabalho, define-se *protocolo de interacção* como sendo um conjunto de modelos de conversação apropriados para a interacção, através comunicação por mensagens, de um par de agentes comunicantes numa rede de EE. Existem, cinco protocolos, cada um associado a um conjunto de dois modelos de conversação.[73] São protocolos de interacção de alto nível, relativamente aos quais se fazem certas assumpções, a seguir descritas:

- Cada modelo de conversação descreve apenas o *comportamento externo* de um agente de uma determinada classe em interacção com um segundo agente, *i.e.*, descreve apenas a componente externa, ou visível, resultante da actividade intra-agente do agente;[74]
- Os modelos de conversação que o agente conhece pertencem a um conjunto bem definido e que não muda;
- Os tipos de mensagens que é possível cada agente individual receber ou enviar pertencem a um conjunto bem definido e conhecido pelo agente e qualquer mensagem recebida por um agente, ou se insere no contexto de uma conversação existente, ou introduz uma conversação nova entre o agente e o agente remetente;

---

[70] Ver a secção 4.3.6.3.
[71] Ver a secção 4.4.6.5.
[72] Ver, por exemplo, [Weiss 1999] ou [Odell 2000].
[73] Os protocolos definidos são inspirados na linguagem de coordenação COOL (ver [Barbuceanu 1995a]) desenvolvida para inter-agência num ambiente logístico multi-agente.
[74] O comportamento interno proposto para os agentes de uma rede de EE é descrito, em separado, na secção 4.4.8. Qualquer que seja o comportamento interno, ele deve ser coerente com os protocolos de interacção de alto nível, incluindo no que respeita ao *significado*. Se é permitido atribuir aos agentes qualidades de comportamento e estados mentais tal como aos humanos, poderia dizer-se que se impõe que os agentes sejam, além de cooperantes, realistas e honestos nas suas crenças e intenções. Por exemplo, se um agente gestor aceita um pedido de um cliente, tem de ser porque ele *acredita* que pode satisfazê-lo (tendo verificado, inclusive, que isso lhe é possível) *e* tem a *intenção* de satisfazê-lo (o agente alterou o seu estado interno de modo a poder satisfazer o pedido).

- Entre um par de agentes comunicantes, as mensagens enviadas por uma determinada ordem por um agente, são sempre *todas* recebidas e *pela mesma ordem* pelo agente receptor;
- O conteúdo de cada mensagem recebida é correctamente interpretado pelo agente receptor, de acordo com o que se define mais adiante para cada modelo de conversação;
- Detalhes físicos respeitantes à comunicação não são especificados (por exemplo, detalhes sobre a tecnologia e o meio físico usado, falhas técnicas na comunicação).

Na definição dos modelos de conversação assumiu-se, como pré-requisito, a necessidade de os agentes poderem realizar a comunicação básica apropriada para um ambiente logístico de produção e distribuição. Para os agentes gestores, em particular, assume-se que esta comunicação básica inclui as seguintes formas de comunicação através de mensagens:

1. A comunicação de um pedido de um cliente a um fornecedor (significando uma encomenda a satisfazer em data dada). O tipo de mensagem apropriado é `pedido`;
2. A comunicação da aceitação, ou da rejeição, de um pedido por parte do fornecedor (significando a aceitação ou a rejeição da encomenda). Os tipos de mensagem apropriados são `aceitação` e `rejeição`;
3. A comunicação, do cliente ao fornecedor ou do fornecedor ao cliente, de um pedido de re-escalonamento de um pedido anteriormente aceite pelo fornecedor (significando um pedido para atrasar ou avançar a data de entrega da encomenda). O tipo de mensagem apropriado é `pedido-re`;
4. A comunicação, do cliente ao fornecedor ou do fornecedor ao cliente, da aceitação, ou da rejeição, de um pedido de re-escalonamento anteriormente realizado (significando a aceitação ou rejeição de um pedido para atrasar ou avançar a data de entrega da encomenda). Os tipos de mensagem apropriados são `aceitação-re` e `rejeição-re`;
5. A comunicação, do cliente ao fornecedor ou do fornecedor ao cliente, do cancelamento de um pedido anteriormente aceite (significando a desistência, ou cancelamento, da encomenda por parte do cliente, ou do fornecedor). O tipo de mensagem apropriada é `cancelamento`;
6. A comunicação ao cliente da satisfação de um pedido anteriormente aceite pelo fornecedor, por parte do fornecedor (que, no contexto do presente trabalho significa o acto da entrega da encomenda na data de entrega actual). O tipo de mensagem apropriada é `satisfação`.

Relativamente às fases de resolução de problemas de escalonamento multi-agente propostas anteriormente,[70] estas formas de comunicação enquadram-se do seguinte modo:

- 1 ocorre na fase 1;
- 2 e 5 podem ocorrer na fase 1;
- 3 e 4 podem ocorrer nas fases 2 e 3;
- 5 e 6 podem ocorrer na fase 3.

Os modelos de conversação são descritos em detalhe mais adiante, bem como o comportamento dos agentes gestores, em particular no que respeita à actividade interna.[75]

[75] Ver as secções 4.4.7.1, 4.4.7.2, 4.4.7.3 e 4.4.7.4 (para os modelos de conversação) e a secção 4.4.8 (para o comportamento dos agentes gestores).

Na Figura 4-24, cada protocolo e cada modelo do conjunto de modelos de conversação de cada protocolo é associado a um par de classes de agentes comunicantes. Os protocolos são resumidamente descritos a seguir.

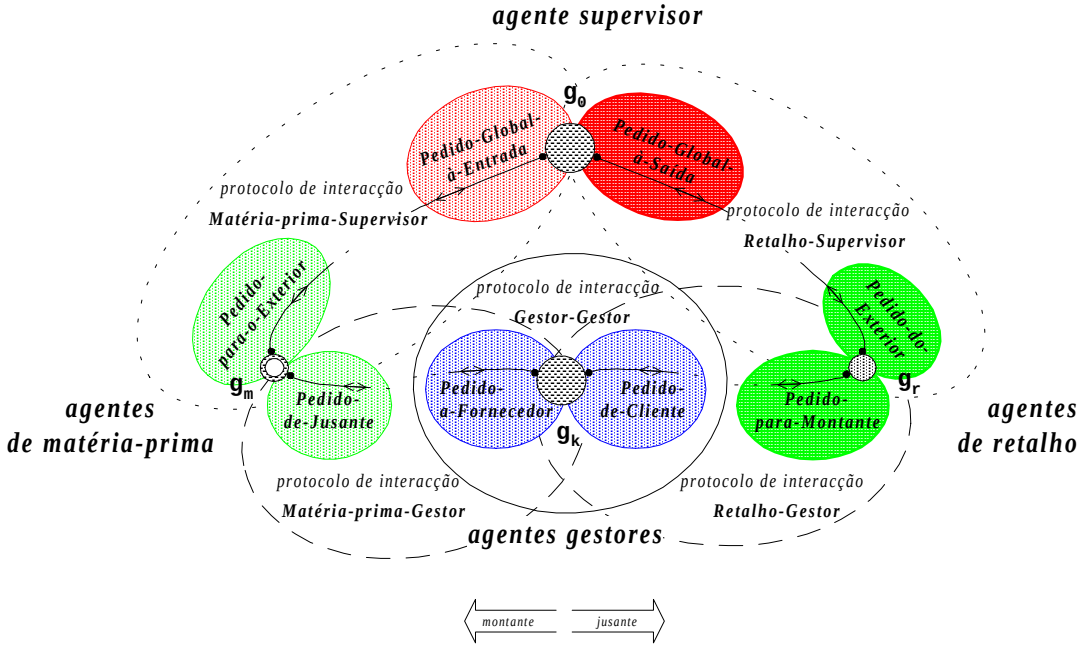


Figura 4-24- Modelos de conversação para cada classe de agente (elipses cheias a cor), para interacção com outros agentes, agrupados por protocolo de interacção (elipses a traço negro).

1. Protocolo *Gestor-Gestor* - Protocolo de interacção entre dois agentes gestores. Inclui os modelos de conversação *Pedido-de-Cliente* e *Pedido-a-Fornecedor* (simétricos). Estes modelos são definidos, respectivamente, pelas operações `PEDIDO-DE-CLIENTE` e `PEDIDO-A-FORNECEDOR`;
2. Protocolo *Retalho-Gestor* - Protocolo de interacção entre um agente de retalho e um agente gestor. Inclui os modelos de conversação *Pedido-de-Cliente*, para o agente gestor e *Pedido-para-Montante*, para o agente de retalho (não simétricos), definidos pelas operações `PEDIDO-DE-CLIENTE` e `PEDIDO-PARA-MONTANTE`, respectivamente;
3. Protocolo *Matéria-prima-Gestor* - Protocolo de interacção entre um agente de matéria-prima e um agente gestor. Inclui os modelos de conversação *Pedido-a-Fornecedor*, para o agente gestor e *Pedido-de-Jusante*, para o agente de matéria-prima (não simétricos), definidos pelas operações `PEDIDO-A-FORNECEDOR` e `PEDIDO-DE-JUSANTE`, respectivamente;
4. Protocolo *Retalho-Supervisor* - Protocolo de interacção entre um agente de retalho e o agente supervisor. Inclui os modelos de conversação *Pedido-do-Exterior*, para o agente de retalho e *Pedido-Global-à-Saída*, para o agente supervisor (simétricos), definidos pelas operações `PEDIDO-DO-EXTERIOR` e `PEDIDO-GLOBAL-À-SAÍDA`, respectivamente;

5. Protocolo *Matéria-prima-Supervisor* - Protocolo de interacção entre um agente de matéria-prima e o agente supervisor. Inclui os modelos de conversação *Pedido-para-o-Exterior*, para o agente de matéria-prima e *Pedido-Global-à-Entrada*, para o agente supervisor (simétricos), definidos pelas operações `PEDIDO-PARA-O-EXTERIOR` e `PEDIDO-GLOBAL-À-ENTRADA`, respectivamente.

Na Figura 4-25 cada um dos protocolos é descrito através de um diagrama de sequência. Os diagramas de colaboração na Figura 4-26 mostram os tipos de mensagens trocadas entre agentes de cada classe em vários casos de interacção possível no contexto dos protocolos, nomeadamente: a) estabelecimento de pedidos locais, em que se incluem dois casos: quando não há e quando há rejeição de um pedido local, b) cancelamento e c) satisfação.

Com os diagramas de sequência expostos na Figura 4-27 pretende-se descrever a interacção entre agentes gestores, agentes clientes e agentes fornecedores envolvidos na satisfação do mesmo pedido global do exterior à rede para aqueles três casos de interacção e, adicionalmente mais um: d) re-escalonamento. Este último, é subdividido em quatro casos de re-escalonamento: re-escalonamento pedido pelo agente ao cliente, re-escalonamento pedido pelo agente a um fornecedor, re-escalonamento pedido pelo cliente e re-escalonamento pedido por um fornecedor.[76] Também o caso b), de cancelamento, é subdividido em três casos: cancelamento por iniciativa do agente, cancelamento pelo cliente e cancelamento por um ou mais fornecedores.

No contexto das fases de resolução de problemas de escalonamento multi-agente, os casos acima referidos incluídos em a) enquadram-se na fase 1, o caso b) enquadra-se na fase 3,[77] o caso c) enquadra-se na fase 3 e os casos incluídos em d) enquadram-se nas fases 2 e 3.

Na Tabela 4-10 mostram-se os modelos de conversação associados a cada protocolo e os tipos de mensagens usados pelos modelos de conversação no contexto do protocolo. Incluem-se também as mensagens especiais `pedido-in` e `pedido-out`, para comunicação do agente supervisor com o exterior da rede de EE, apesar de não estarem associadas a protocolo algum.

A cada classe de agente está associado um conjunto de dois modelos de conversação (ver a Figura 4-24). Estes conjuntos agrupam os modelos de conversação por cada classe de agente e são definidos pelas operações, `MODELOS-CONV-GESTOR`, `MODELOS-CONV-RETALHO`, `MODELOS-CONV-MATÉRIA-PRIMA` e `MODELOS-CONV-SUPERVISOR`.[78] Por classe de agente, os modelos de conversação associados são:

Para agentes gestores (modelos do conjunto `MODELOS-CONV-GESTOR()`):

– *Pedido-de-Cliente* - Para interacção de um agente gestor com um agente cliente. Este último pode ser um outro agente gestor ou um agente de retalho;

– *Pedido-a-Fornecedor* - Para interacção de um agente gestor com um agente fornecedor. Este último pode ser um outro agente gestor ou um agente de matéria-prima.

---

[76] *Diagramas de sequência*, *diagramas de colaboração*, *diagramas de estado* e *diagramas de actividade* fazem parte da linguagem de análise e desenho orientado por objectos UML (*Unified Modeling Language*, ver [Rumbaugh 1999]). Estes tipos de diagramas é usado para descrever a interacção e o comportamento dinâmico dos agentes, como proposto em [Odell 2000].

[77] Os cancelamentos que podem ocorrer na fase 1 são incluídos em a), na Figura 4-27.

[78] Estas operações foram definidas no contexto das secções 4.4.1, 4.4.2, 4.4.3 e 4.4.4.

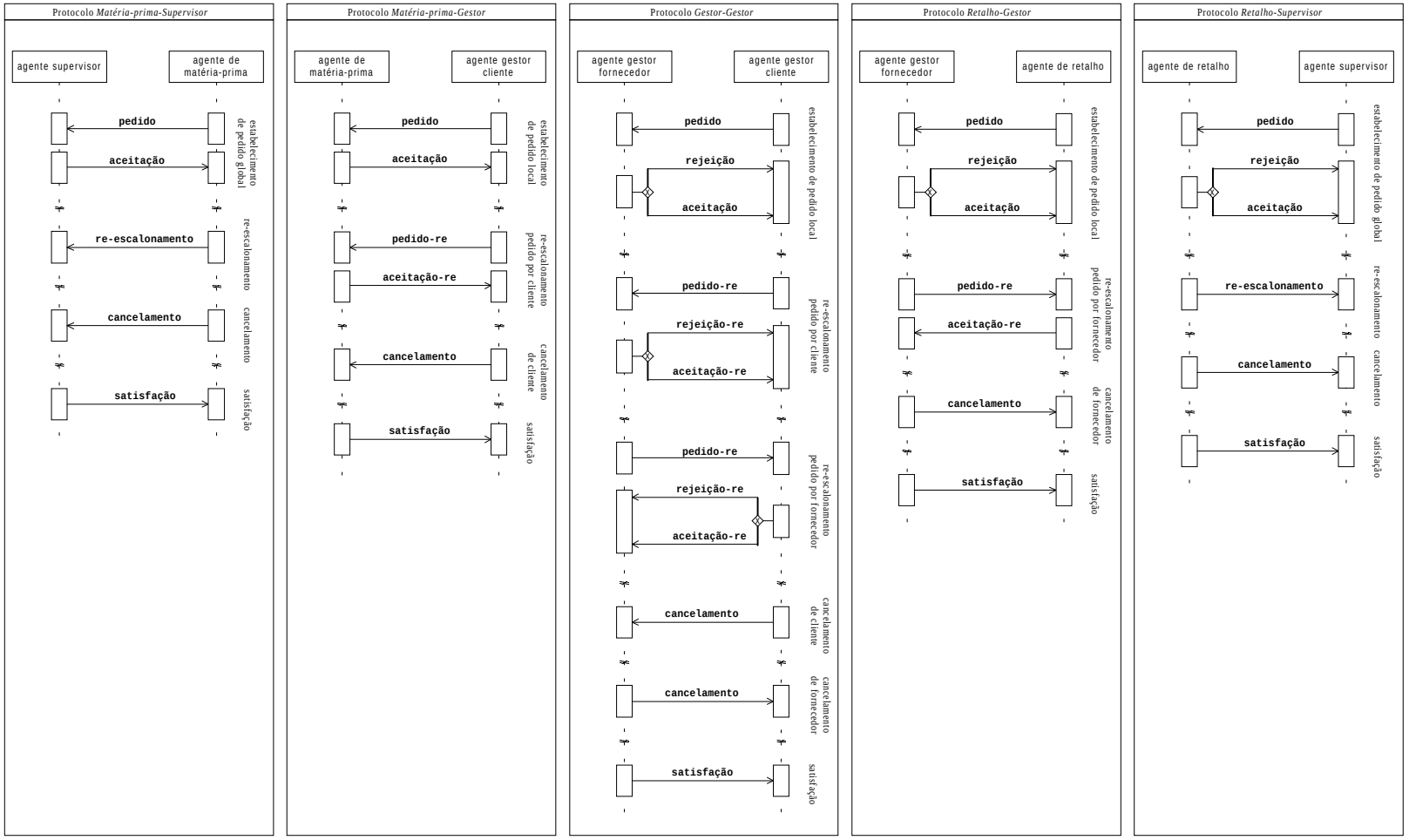
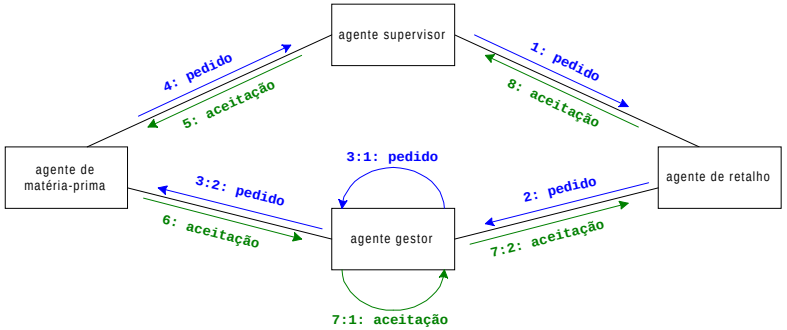
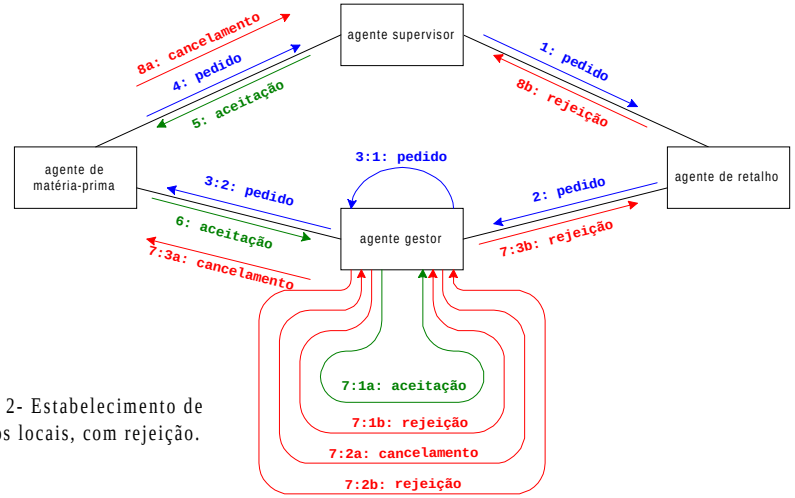


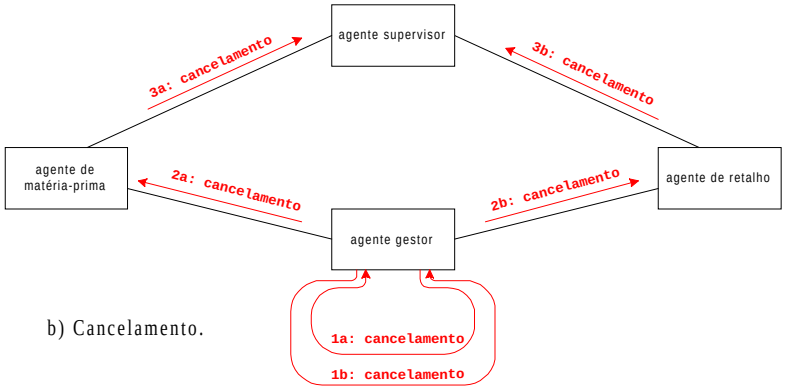
Figura 4-25- Protocolos de interacção entre agentes (diagramas de sequência).



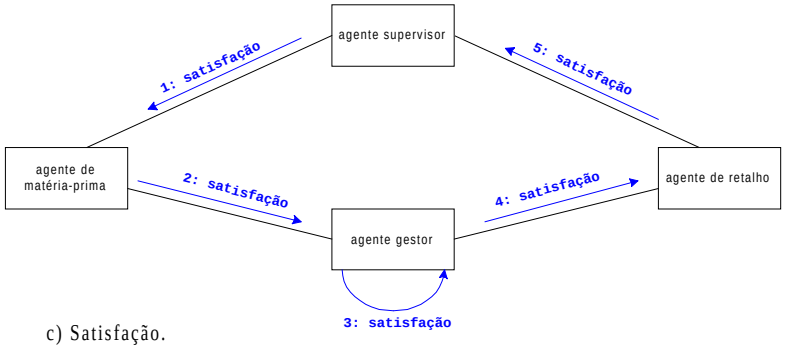
a) Caso 1- Estabelecimento de pedidos locais, sem rejeição.



a) Caso 2- Estabelecimento de pedidos locais, com rejeição.



b) Cancelamento.



c) Satisfação.

Figura 4-26- Interacções no contexto dos protocolos de interacção (diagramas de colaboração).

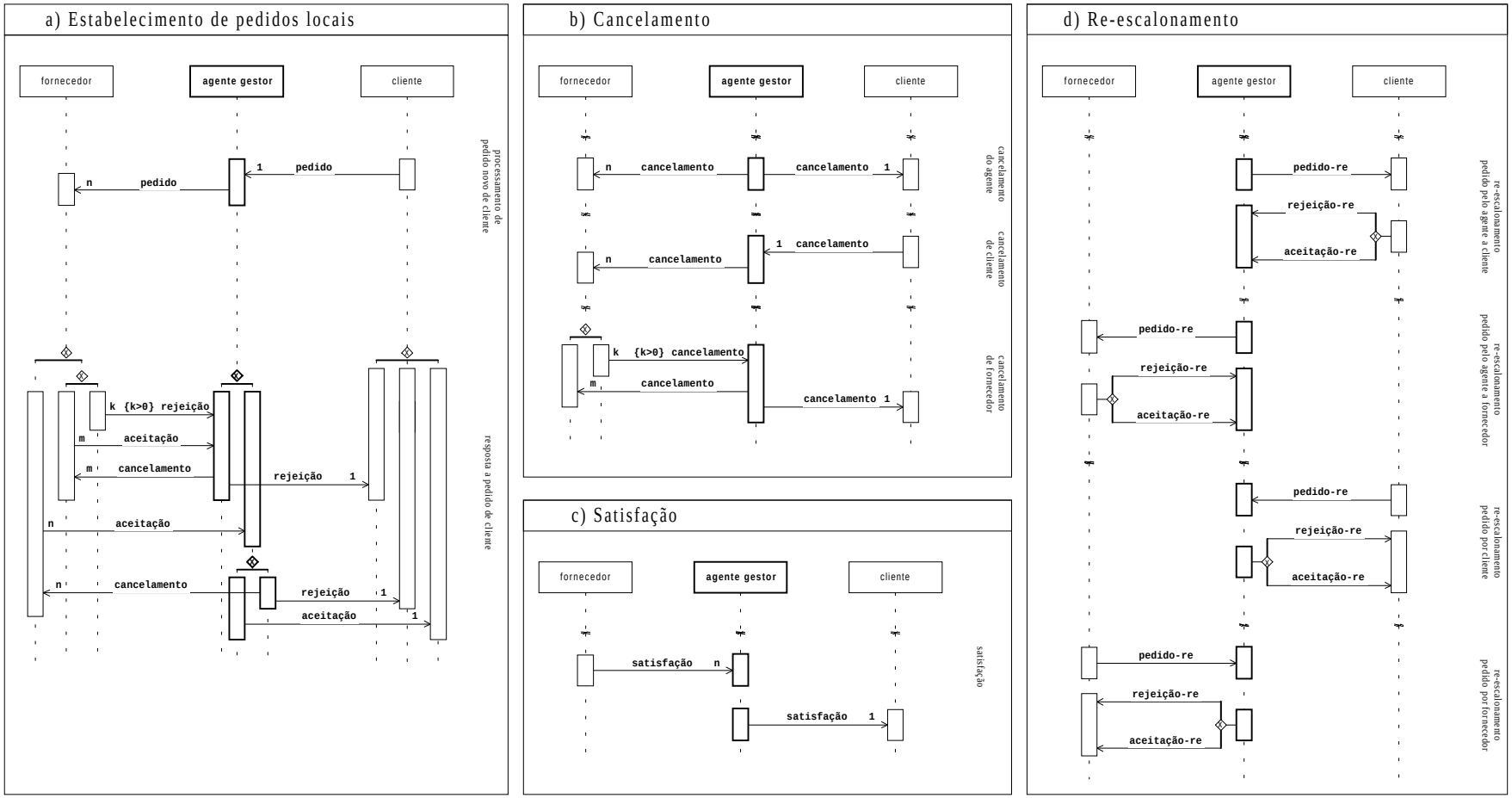


Figura 4-27- Interacção entre um agente gestor, um agente cliente e os agentes fornecedores envolvidos na satisfação de um mesmo pedido global do exterior à rede (diagramas de sequência), no contexto do protocolo *Gestor-Gestor* (modelos de conversação *Pedido-de-Cliente*, para interacção com o cliente e *Pedido-a-Fornecedor*, para interacção com o fornecedor).

Para agentes de retalho (modelos do conjunto `MODELOS-CONV-RETALHO()`):

– *Pedido-para-Montante* - Para interacção de um agente de retalho com um agente fornecedor. Este último é sempre um agente gestor;

– *Pedido-do-Exterior* - Para interacção de um agente de retalho com o agente supervisor.

Tabela 4-10- Protocolos, modelos de conversação associados e tipos de mensagens usadas nos protocolos.

Protocolo	Modelos de Conversação	Tipos de mensagens
Gestor-Gestor	*Pedido-de-Cliente*, *Pedido-a-Fornecedor*	`pedido, aceitação, rejeição, satisfação, cancelamento, pedido-re, aceitação-re, rejeição-re`
Retalho-Gestor	*Pedido-para-Montante*, *Pedido-de-Cliente*	`pedido, aceitação, rejeição, satisfação, cancelamento, pedido-re, aceitação-re`
Matéria-prima-Gestor	*Pedido-de-Jusante*, *Pedido-a-Fornecedor*	`pedido, aceitação, satisfação, cancelamento, pedido-re, aceitação-re`
Retalho-Supervisor	*Pedido-do-Exterior*, *Pedido-Global-à-Saída*	`pedido, aceitação, rejeição, satisfação, cancelamento, re-escalonamento`
Matéria-prima-Supervisor	*Pedido-para-o-Exterior*, *Pedido-Global-à-Entrada*	`pedido, aceitação, satisfação, cancelamento, re-escalonamento`
(nenhum protocolo)	(nenhum modelo)	`pedido-in, pedido-out`

Para agentes de matéria-prima (modelos do conjunto `MODELOS-CONV-MATÉRIA-PRIMA()`):

– *Pedido-de-Jusante* - Para interacção de um agente de matéria-prima com um agente cliente. Este último é sempre um agente gestor;

– *Pedido-para-o-Exterior* - Para interacção de um agente de matéria-prima com o agente supervisor.

Para o agente supervisor (modelos do conjunto `MODELOS-CONV-SUPERVISOR()`):

– *Pedido-Global-à-Saída* - Para interacção do agente supervisor com um agente de retalho;

– *Pedido-Global-à-Entrada* - Para interacção do agente supervisor com um agente de matéria-prima.

Os protocolos em 2 (modelos *Pedido-de-Cliente* e *Pedido-para-Montante*, ver página 267) e 3 (modelos *Pedido-a-Fornecedor* e *Pedido-de-Jusante*) não são simétricos, pois há estados e transições de um dos modelos de conversação que não têm estado e transição homólogos no outro modelo de conversação do mesmo protocolo.

Esta assimetria acontece devido a ter sido convencionado que todos os agentes gestores de uma rede de EE dialoguem usando o mesmo conjunto de modelos de conversação. Qualquer agente gestor pode dialogar com um qualquer dos seus clientes usando o modelo de conversação *Pedido-de-Cliente*, independentemente de o cliente ser um outro agente gestor ou

um agente de retalho; também, qualquer agente gestor pode dialogar com um qualquer dos seus fornecedores usando o modelo de conversação *Pedido-a-Fornecedor*, independentemente de o fornecedor ser um outro agente gestor ou um agente de matéria-prima.[79]

Esta convenção foi usada de modo a que os agentes gestores sejam independentes dos restantes agentes no que respeita aos modelos de conversação. Embora o exterior da rede de EE não seja modelado é, de alguma forma, representado através de uma fronteira, constituída por nós de retalho e nós de matéria-prima no nível físico e por agentes de retalho, agentes de matéria-prima e o agente supervisor, no nível virtual. Convencionou-se que, no nível físico, o exterior da rede opera como um *sumidouro de produtos de saída* e como uma *fonte de produtos de entrada* da rede[80] e que, no nível virtual, o exterior opera como uma *fonte de encomendas do exterior* de produtos de saída da rede (que o agente supervisor introduz na forma de novos pedidos globais do exterior à rede) e como um *sumidouro de encomendas ao exterior* de produtos de entrada da rede (que correspondem aos pedidos globais da rede ao exterior recolhidos pelo agente supervisor dos agentes de matéria-prima).

De acordo com o modelo de conversação *Pedido-para-Montante* (ver o protocolo *Retalho-Gestor* na Figura 4-25), os agentes de retalho não cancelam pedidos enviados a um fornecedor (uma vez que o fornecedor tenha aceite o pedido), nem rejeitam (aceitam sempre) pedidos de re-escalonamento de um fornecedor, nem enviam pedidos de re-escalonamento a um fornecedor, ainda que o modelo de conversação *Pedido-de-Cliente* (ver o protocolo *Gestor-Gestor* na Figura 4-25) inclua a possibilidade de um agente gestor receber um cancelamento de um pedido que aceitou de um cliente, uma rejeição de um pedido de re-escalonamento que enviou a um cliente, ou um pedido de re-escalonamento de um cliente, respectivamente. E também, de acordo com o modelo de conversação *Pedido-de-Jusante* (ver o protocolo *Matéria-prima-Gestor* na Figura 4-25), os agentes de matéria-prima não rejeitam (aceitam sempre) pedidos recebidos de clientes, nem cancelam um pedido de um cliente (após o terem aceite), nem rejeitam (aceitam sempre) um pedido de re-escalonamento de um cliente, nem enviam pedidos de re-escalonamento a um cliente, ainda que o modelo de conversação *Pedido-a-Fornecedor* (ver o protocolo *Gestor-Gestor* na Figura 4-25) inclua a possibilidade de um agente gestor receber uma rejeição de um pedido que enviou a um fornecedor, um cancelamento de um pedido aceite por um fornecedor, uma rejeição de um pedido de re-escalonamento que enviou a um fornecedor, ou um pedido de re-escalonamento de um fornecedor, respectivamente.

Na Tabela 4-11 descrevem-se, resumidamente, os tipos de mensagens usadas e indicam-se os modelos de conversação nos quais as mensagens são usadas. Os tipos de mensagens `pedido-in` e `pedido-out` constituem uma excepção, pois não se incluem em modelo de conversação algum. Os sentidos de circulação possíveis para os vários tipos de mensagens, bem como o seu domínio de circulação na rede de EE por agentes das diferentes classes de agentes, de acordo com os modelos de conversação indicados, são ilustrados na Figura 4-28. Na Figura 4-29 enquadram-se os tipos de mensagens (à excepção de `pedido-in` e `pedido-out`) numa classificação quanto ao tipo básico de acto associado, considerando

[79] *I.e.*, um agente conhece os agentes que são seus clientes e os agentes que são seus fornecedores, mas não distingue clientes que são agentes de retalho, se os tiver, dos que o não são (no primeiro caso), nem fornecedores que são agentes de matéria-prima, se os tiver, dos que o não são. Os agentes não sabem se os nós que gerem estão ou não junto aos extremos jusante ou montante da rede física.

[80] Ver o Capítulo 3.

como tipos básicos de actos *solicitação, comprometimento, informação, recusa* e *acção*. Esta classificação é adaptada (simplificada) de [Parunak 1996b].

Tabela 4-11- Tipos de mensagens, descrição e modelos de conversação em que são empregues.

Tipo de mensagem	Descrição	Modelos de conversação
`pedido`	Pedido novo (introduz uma nova conversação acerca de um novo problema de escalonamento).	*Pedido-de-Cliente*, *Pedido-a-Fornecedor*, *Pedido-para-Montante*, *Pedido-do-Exterior*, *Pedido-de-Jusante*, *Pedido-para-o-Exterior*, *Pedido-Global-à-Saída*, *Pedido-Global-à-Entrada*
`aceitação`	Aceitação do pedido novo.	(todos os modelos)
`satisfação`	Satisfação do pedido.	
`cancelamento`	Cancelamento do pedido.	
`rejeição`	Rejeição do pedido novo.	*Pedido-de-Cliente*, *Pedido-a-Fornecedor*, *Pedido-para-Montante*, *Pedido-do-Exterior*, *Pedido-Global-à-Saída*.
`pedido-re`	Pedido de re-escalonamento do pedido.	*Pedido-de-Cliente*, *Pedido-a-Fornecedor*, *Pedido-para-Montante*, *Pedido-de-Jusante*.
`aceitação-re`	Aceitação do pedido de re-escalonamento anterior.	
`rejeição-re`	Rejeição do pedido de re-escalonamento anterior.	*Pedido-de-Cliente*, *Pedido-a-Fornecedor*.
`re-escalonamento`	Notificação de que houve um re-escalonamento de pedido.	*Pedido-do-Exterior*, *Pedido-para-o-Exterior*, *Pedido-Global-à-Saída*, *Pedido-Global-à-Entrada*.
`pedido-in`	Pedido global do exterior à rede.	nenhum modelo (mensagens usadas só para comunicação do agente supervisor com o exterior da rede de EE)
`pedido-out`	Pedido global da rede ao exterior.	

Os actos *solicitação, comprometimento, informação* e *recusa* são actos de discurso; o acto *acção* corresponde a um acto físico, o acto da entrega de uma encomenda do autor ao interlocutor, no caso de `satisfação`. Os actos de tipo *solicitação* correspondem a uma tentativa do autor conseguir um entendimento com o interlocutor (conseguir uma crença mútua) no sentido de que o autor pretende que o interlocutor realize um certo acto. Os actos de tipo *asserção* (tipos *comprometimento, informação* e *recusa*) correspondem a uma tentativa de o autor fazer entender ao interlocutor que acredita que o que lhe está comunicando é verdadeiro. Os actos de tipo *comprometimento* são uma asserção afirmando que o autor passa a adoptar um objectivo de modo a conseguir qualquer coisa que o interlocutor lhe comunicou pretender. Os actos de tipo *informação* são simultaneamente uma asserção e uma tentativa da parte do autor no sentido de que o interlocutor creia naquilo que o autor lhe comunica. Os actos de tipo *recusa* são uma *asserção* afirmando que o autor não adopta, ou não mais mantém, o objectivo no sentido de conseguir qualquer coisa que o interlocutor pretende.

`pedido` e `pedido-re` são uma *solicitação* de um agente a um interlocutor para que ele se lhe comprometa a um certo acto pretendido pelo agente. Este acto é o da aceitação de uma encomenda (com a consequente satisfação futura), no caso de `pedido` e o da aceitação (ou rejeição) de um re-escalonamento proposto para um pedido, no caso de `pedido-re`. `aceitação` e `aceitação-re` significam *comprometimento* de um agente perante uma solicitação anterior de `pedido` e de `pedido-re`, respectivamente. `re-escalonamento` é um acto de *informação*, através do qual um agente de retalho ou um agente de matéria-prima informa o agente supervisor de um re-escalonamento ocorrido na data associada a um `pedido` global (mensagens do tipo `re-escalonamento` são usadas apenas entre os agentes de retalho e o agente supervisor ou entre os agentes de matéria-prima e o agente supervisor).

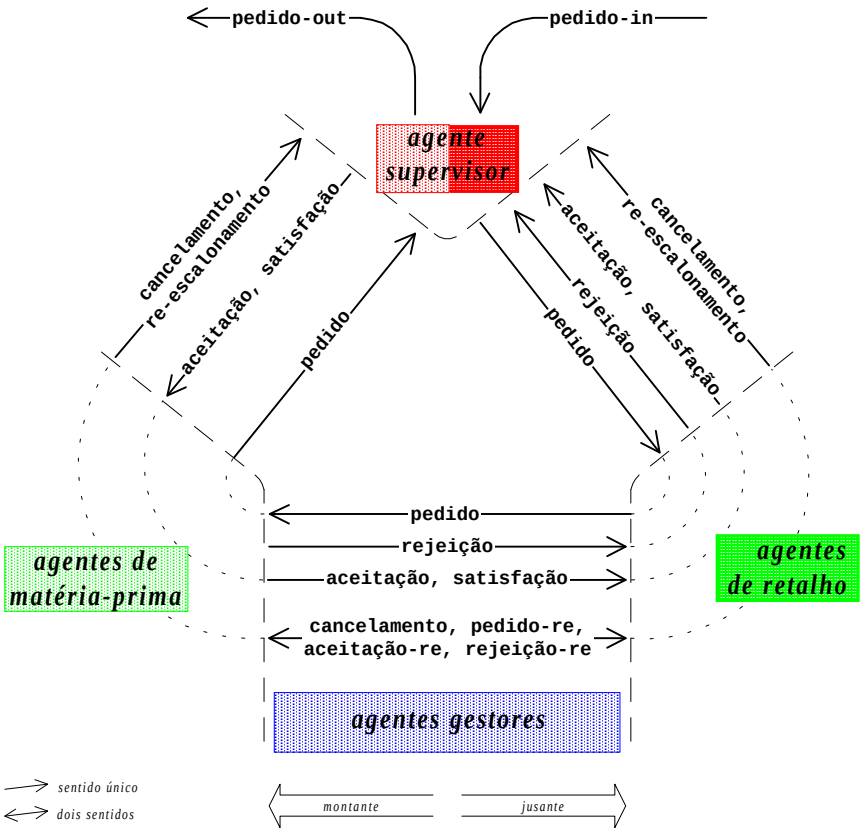


Figura 4-28- Circuito e sentido de circulação das mensagens, por tipo, numa rede de EE.

A razão das linhas a tracejado na Figura 4-29 é a de as mensagens de tipos `pedido`, `pedido-re` e `aceitação` servirem também, para comunicar a informação adicional de um problema de escalonamento necessária para a realização do mecanismo de coordenação para escalonamento temporal (as folgas temporais incluídas no conteúdo das mensagens); por essa razão, os actos associados a `pedido`, `pedido-re` e `aceitação` são, em parte, também actos de *informação*. `rejeição`, `rejeição-re` e `cancelamento` são actos de *recusa* em aceitar um `pedido`, no caso de `rejeição`, em aceitar um `pedido-re`, no caso de

`rejeição-re`, ou que significam desistência relativamente a dar `satisfação` futura um `pedido` para o qual houve anteriormente uma `aceitação`, no caso de `cancelamento`.[81]

Nas quatro secções seguintes, cada modelo de conversação é representado graficamente através de um diagrama de estados. Este diagrama é um grafo cujos nós representam os estados e cujos arcos direccionados representam transições entre estados. Cada arco é etiquetado por símbolos "*tipo-de-mensagem* /" ou "/ *tipo-de-mensagem*". "*tipo-de-mensagem* /" corresponde a uma etiqueta `ETIQUETA(R,tipo-de-mensagem)` na definição do modelo e significa que a transição se dá por recepção (`R`) de uma mensagem de tipo `tipo-de-mensagem`. "/ *tipo-de-mensagem*" corresponde a uma etiqueta `ETIQUETA(S,tipo-de-mensagem)` na definição do modelo e significa que a transição se dá por envio (`S`) de uma mensagem de tipo `tipo-de-mensagem`.

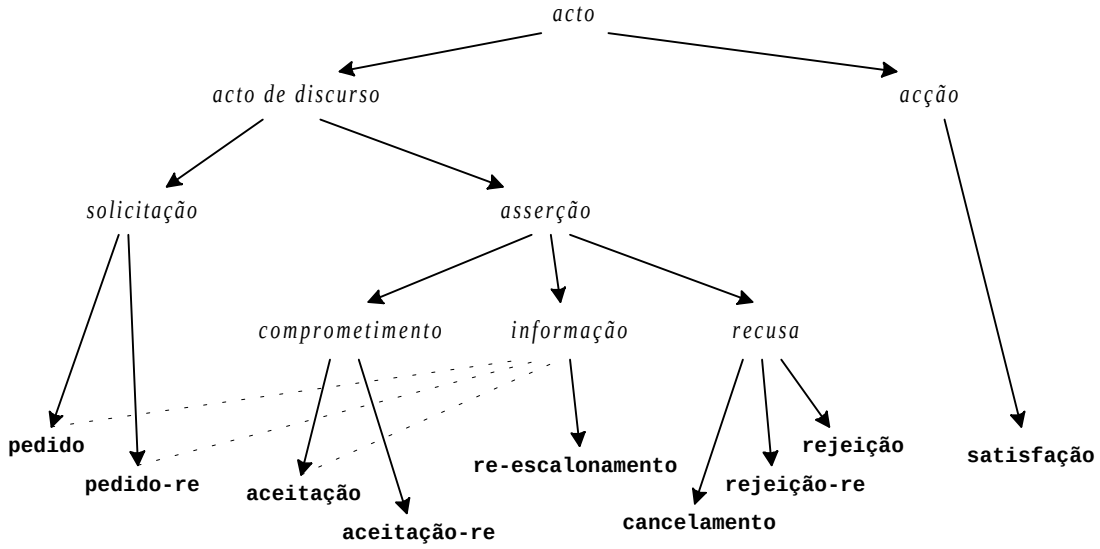


Figura 4-29- Classificação dos tipos de mensagens usadas nos protocolos de acordo com o tipo de acto (adaptado da classificação em [Parunak 1996b]).

Adicionalmente, na descrição do conteúdo de algumas mensagens está incluída uma folga ou um outro qualquer parâmetro temporal. Nesses casos, assume-se que a informação que está associada ao parâmetro folga inclui o tipo de parâmetro (*i.e.*, se se trata de uma folga a jusante `FJ`, uma folga a montante `FM`, uma folga a externa a jusante `FEJ`, uma folga a externa a montante `FEM`, uma folga jusante-montante `FJM`, uma folga montante-jusante `FMJ`, uma folga total `FT`, uma data limite mais tarde `DD` ou uma data de lançamento mais cedo `RD`) e o valor da folga temporal e assume-se que o agente receptor da mensagem é capaz de interpretar

[81] Os actos *solicitação*, *comprometimento*, *informação*, *recusa* e *acção* são designados por *solicit*, *commit*, *inform*, *refuse* e *do*, respectivamente, em [Parunak 1996b]. O acto *do* é apelidado de *non-speech act* "não-acto de discurso"). Em [Parunak 1996b] também, consideram-se adicionalmente como sub-tipos de *solicit question* (pergunta) e *request* (pedido) e como sub-tipos de *do ship* (enviar) e *pay* (pagar). Nesta classificação `pedido` e `pedido-re` correspondem a um acto de tipo *request* e `satisfação` corresponde a um acto de tipo *ship*.

correctamente esta informação. Em cada um dos casos, o tipo de folga é sugerido pelo símbolo usado, por exemplo, FJ significa folga a jusante, FM significa folga a montante, etc..[82]

4.4.7.1 Os Modelos de Conversação de Agentes Gestores

Para os agentes gestores, as conversações podem envolver interacção com um agente cliente ou com um agente fornecedor. Os modelos de conversação para cada um destes casos são:

- *Pedido-de-Cliente* - Uma instância da conversação deste modelo é criada internamente no agente gestor quando ele recebe um pedido novo de um cliente (um outro agente gestor, ou um agente de retalho). O agente assume, na conversação, o papel de fornecedor em diálogo com o agente cliente acerca do pedido local que este último lhe enviou. A conversação termina com a rejeição ou a satisfação do pedido pelo agente, ou ainda com o cancelamento por parte do agente ou do cliente;
- *Pedido-a-Fornecedor* - Uma instância da conversação deste modelo é criada internamente no agente gestor quando ele deve fazer um pedido novo a um fornecedor (um outro agente gestor, ou um agente de matéria-prima). O agente assume, na conversação, o papel de cliente em diálogo com o agente fornecedor acerca do pedido local que envia a este último. A conversação termina com a rejeição ou a satisfação do pedido pelo fornecedor, ou ainda com o cancelamento do pedido pelo agente ou pelo fornecedor.

O modelo de conversação *Pedido-de-Cliente* é definido pela operação `PEDIDO-DE-CLIENTE`, como se segue:[83]

```
PEDIDO-DE-CLIENTE() ::=
MODELO-DE-CONVERSAÇÃO(
  1,                                        ; estado inicial
  4,                                        ; estado final
  {1,2,3,4,5,6},                            ; estados
  { TRANSIÇÃO(1,2,ETIQUETA(R,pedido)),      ; transições
    TRANSIÇÃO(2,3,ETIQUETA(S,aceitação)),
    TRANSIÇÃO(2,4,ETIQUETA(S,rejeição)),
    TRANSIÇÃO(3,4,ETIQUETA(S,satisfação)),
    TRANSIÇÃO(3,4,ETIQUETA(S,cancelamento)),
    TRANSIÇÃO(3,4,ETIQUETA(R,cancelamento)),
    TRANSIÇÃO(3,5,ETIQUETA(S,pedido-re)),
    TRANSIÇÃO(5,3,ETIQUETA(R,aceitação-re)),
    TRANSIÇÃO(5,3,ETIQUETA(R,rejeição-re)),
    TRANSIÇÃO(3,6,ETIQUETA(R,pedido-re)),
    TRANSIÇÃO(6,3,ETIQUETA(S,aceitação-re)),
    TRANSIÇÃO(6,3,ETIQUETA(S,rejeição-re)) } ).
```

O modelo de conversação *Pedido-de-Cliente* é representado no diagrama de estados da Figura 4-30.

---

[82] Pode assumir-se que cada um destes símbolos denota um par tipo-de-parâmetro-valor. Por exemplo, FMJ denotaria o par `<FMJ,x>`, em que `FMJ` seria um símbolo denotando o tipo de folga (folga montante-jusante, no caso) e `x` o respectivo valor numérico.

[83] O conjunto de tipos de mensagens e o conjunto de modelos de conversação que um agente gestor conhece são dados pelas operações `TIPOS-MSG-GESTOR` e `MODELOS-CONV-GESTOR`, respectivamente (ver a secção 4.4.1.1).

Os estados possíveis no contexto de uma conversação de modelo *Pedido-de-Cliente* (ver Figura 4-30), são:

1. Estado inicial (conversação iniciada);
2. Pedido recebido;
3. Pedido aceite;
4. Estado final (conversação terminada): pedido rejeitado, satisfeito, cancelado pelo agente ou cancelado pelo cliente;
5. Pedido pendente de re-escalonamento requerido pelo agente;
6. Pedido pendente de re-escalonamento requerido pelo cliente.

No modelo de conversação *Pedido-de-Cliente*, cada um dos estados é homólogo de um estado com o mesmo número no modelo de conversação *Pedido-a-Fornecedor* (ver adiante) e com transições de estado simétricas. Os tipos de mensagens usados no modelo *Pedido-de-Cliente* são os do conjunto `TIPOS-MSG-GESTOR()` (os mesmos que para o modelo *Pedido-a-Fornecedor*).

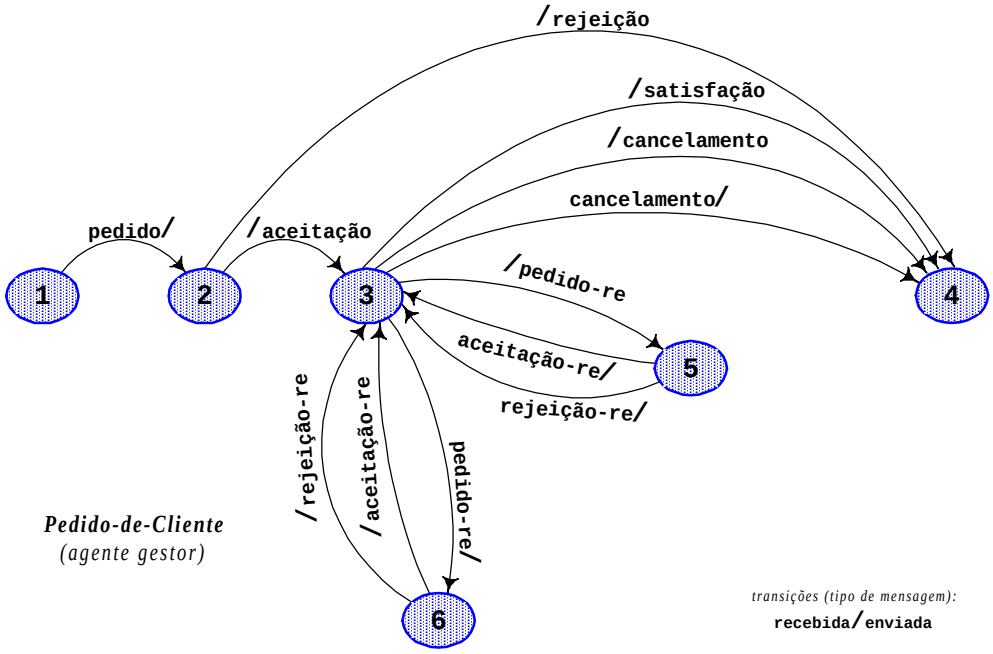


Figura 4-30- Modelo de conversação *Pedido-de-Cliente*, para interacção de um agente gestor com um agente cliente (diagrama de estado).

No modelo de conversação *Pedido-de-Cliente*, cada um dos estados é homólogo de um estado com o mesmo número no modelo de conversação *Pedido-a-Fornecedor* (ver adiante) e com transições de estado simétricas. Os tipos de mensagens usados no modelo *Pedido-de-Cliente* são os do conjunto `TIPOS-MSG-GESTOR()` (os mesmos que para o modelo *Pedido-a-Fornecedor*).

Os tipos de mensagens que podem ser trocadas em cada transição de estado, no contexto de uma conversação de modelo *Pedido-de-Cliente* e o significado do conteúdo, relativamente a um agente gestor que recebe ou envia a mensagem, são descritos a seguir. Os símbolos `R`, `S` e

S/R indicam que a mensagem pode ser, respectivamente, recebida, enviada e ou recebida ou enviada. No que respeita ao conteúdo assume-se que é possível ao agente interpretar correctamente o significado dos elementos do conjunto do conteúdo da mensagem.

- *pedido* (R) - Conteúdo {𝕕, 𝔽𝔼𝕁}, em que 𝕕 é o pedido do cliente e 𝔽𝔼𝕁 a folga externa a jusante para o agente. Uma mensagem deste tipo inicia uma nova conversação entre o agente e o cliente remetente, acerca do pedido 𝕕;
- *aceitação* (S) - Conteúdo {𝔽𝕄𝕁}, em que 𝔽𝕄𝕁 é a folga montante-jusante. Uma mensagem deste tipo significa aceitação, por parte do agente, do pedido 𝕕, do cliente;
- *rejeição* (S) - Conteúdo {}. Uma mensagem deste tipo significa rejeição, por parte do agente, do pedido 𝕕, do cliente;
- *satisfação* (S) - Conteúdo {}. Significa que o pedido 𝕕, do cliente, está sendo satisfeito pelo agente. Esta mensagem é enviada pelo agente quando o tempo actual é igual ao tempo do pedido actualizado (*i.e.*, TEMPO(𝕕), sendo 𝕕 o pedido original, ou o último pedido re-escalonado se houve re-escalonamentos);
- *cancelamento* (S/R) - Conteúdo {}. Uma mensagem deste tipo significa, se enviada, cancelamento por parte do agente ou, se recebida, cancelamento por parte do cliente, do pedido 𝕕;
- *pedido-re* (S/R) - Conteúdo {𝕕', 𝔽𝕁}, {𝕕', 𝔽𝕄}, {𝕕', 𝔽𝔼𝕁}, {𝕕', 𝔽𝔼𝕄} ou {𝕕'}, em que 𝕕' é o pedido re-escalonado (*i.e.*, TEMPO(𝕕')≠TEMPO(𝕕)) proposto e 𝔽𝕁, 𝔽𝕄, 𝔽𝔼𝕁 e 𝔽𝔼𝕄 são folgas. Uma mensagem deste tipo significa, se enviada, um pedido de re-escalonamento por parte do agente ou, se recebida, um pedido de re-escalonamento por parte do cliente, do pedido 𝕕, para 𝕕'. A folga no conteúdo é usada como uma justificação para o pedido de re-escalonamento na fase 2 de resolução do problema de escalonamento associado ao pedido 𝕕. Para dar resposta a possíveis situações de pedidos de re-escalonamento mútuos, usa-se a folga do conteúdo para atribuição de prioridades a cada um dos pedidos de re-escalonamento de cada um dos agentes. Se acontecerem pedidos de re-escalonamento mútuos, a maior prioridade é dada ao pedido de re-escalonamento justificado por uma folga FJ ou FM; se as folgas são do mesmo tipo, a prioridade maior é para o menor valor de folga; se são do mesmo tipo e têm igual valor, a prioridade maior é para o pedido de re-escalonamento enviado pelo agente mais a montante (portanto, neste caso, o próprio agente); o pedido de re-escalonamento não prioritário é rejeitado pelo receptor do pedido;
- *aceitação-re* (S/R) - Conteúdo {}. Uma mensagem deste tipo significa, se enviada, a aceitação do pedido de re-escalonamento anterior por parte do agente ou, se recebida, a aceitação do pedido de re-escalonamento anterior por parte do cliente. Quando um pedido de re-escalonamento é aceite, o pedido 𝕕 a que se refere a conversação, é substituído pelo pedido re-escalonado, 𝕕';
- *rejeição-re* (S/R) - Conteúdo {}. Semelhante ao tipo de mensagem *aceitação-re*, mas significa rejeição do pedido de re-escalonamento.

O modelo para a classe de conversação *Pedido-a-Fornecedor* é representado no diagrama de estados da Figura 4-31. Este modelo é definido pela operação PEDIDO-A-FORNECEDOR, como se segue:

```
PEDIDO-A-FORNECEDOR() ::=
MODELO-DE-CONVERSAÇÃO(
  1,                                          ; estado inicial
  4,                                          ; estado final
  {1,2,3,4,5,6},                              ; estados
  { TRANSIÇÃO(1,2,ETIQUETA(S,pedido)),        ; transições
    TRANSIÇÃO(2,3,ETIQUETA(R,aceitação)),
    TRANSIÇÃO(2,4,ETIQUETA(R,rejeição)),
    TRANSIÇÃO(3,4,ETIQUETA(R,satisfação)),
    TRANSIÇÃO(3,4,ETIQUETA(S,cancelamento)),
    TRANSIÇÃO(3,4,ETIQUETA(R,cancelamento)),
    TRANSIÇÃO(3,5,ETIQUETA(R,pedido-re)),
    TRANSIÇÃO(5,3,ETIQUETA(S,aceitação-re)),
    TRANSIÇÃO(5,3,ETIQUETA(S,rejeição-re)),
    TRANSIÇÃO(3,6,ETIQUETA(S,pedido-re)),
    TRANSIÇÃO(6,3,ETIQUETA(R,aceitação-re)),
    TRANSIÇÃO(6,3,ETIQUETA(R,rejeição-re)) } ).
```

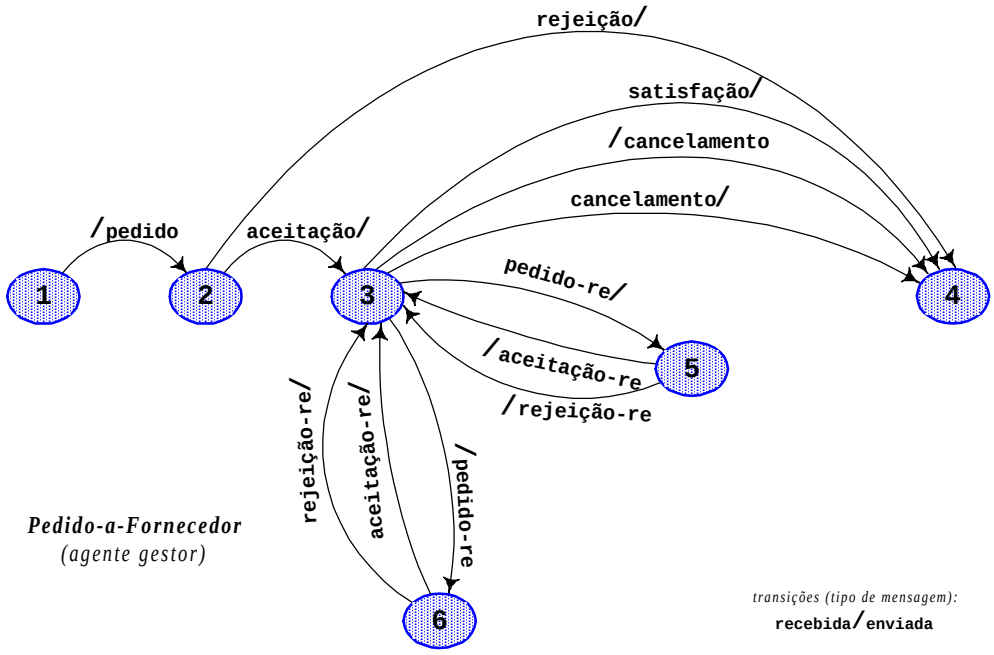


Figura 4-31- Modelo de conversação *Pedido-a-Fornecedor*, para interacção de um agente gestor com um agente fornecedor (diagrama de estado).

Os estados possíveis no contexto de uma conversação de modelo *Pedido-a-Fornecedor* (ver Figura 4-31), são:

1. Estado inicial (conversação iniciada);
2. Pedido enviado;
3. Pedido aceite;

4. Estado final (conversação terminada): pedido rejeitado, satisfeito, cancelado pelo fornecedor ou cancelado pelo agente;
5. Pedido pendente de re-escalonamento requerido pelo fornecedor;
6. Pedido pendente de re-escalonamento requerido pelo agente.

No modelo de conversação *Pedido-a-Fornecedor* cada um dos estados é homólogo de um estado com o mesmo número do modelo de conversação *Pedido-de-Cliente* e com transições de estado simétricas. Os tipos de mensagens usados no modelo *Pedido-a-Fornecedor* são os do conjunto `TIPOS-MSG-GESTOR()` (os mesmos que para o modelo *Pedido-de-Cliente*).

Descrevem-se a seguir, os tipos de mensagens que podem ser trocadas em cada transição de estado, no contexto de uma conversação de modelo *Pedido-a-Fornecedor* e o significado do conteúdo respectivo.

– *pedido* (S) - Conteúdo {𝕕,𝔽𝕁𝕄}, em que 𝕕 é o pedido do agente e 𝔽𝕁𝕄 a folga jusante-montante a transmitir ao fornecedor;

– *aceitação* (R) - Conteúdo {𝔽𝔼𝕄}, em que 𝔽𝔼𝕄 é a folga externa a montante transmitida pelo fornecedor;

– *rejeição* (R), *satisfação* (R), *cancelamento* (S/R), *pedido-re* (S/R), *aceitação-re* (S/R), *rejeição-re* (S/R) - Conteúdos e significados idênticos aos das mensagens do mesmo tipo do modelo de conversação *Pedido-de-Cliente*.

4.4.7.2 Os Modelos de Conversação de Agentes de Retalho

Para os agentes de retalho, as conversações podem envolver interacção com um agente gestor ou com o agente supervisor. Os modelos de conversação para cada um destes casos são:

- *Pedido-para-Montante* - Uma instância da conversação deste modelo é criada internamente no agente de retalho quando ele deve enviar um pedido a um fornecedor (que é sempre um agente gestor). A conversação termina com a rejeição, a satisfação ou o cancelamento do pedido pelo fornecedor;
- *Pedido-do-Exterior* - Uma instância da conversação deste modelo é criada internamente no agente de retalho quando ele recebe um novo pedido global do exterior, do agente supervisor. A conversação termina com a rejeição, a satisfação ou o cancelamento do pedido pelo agente.

O modelo de conversação *Pedido-para-Montante* é representado no diagrama de estados da Figura 4-32 e é definido pela operação `PEDIDO-PARA-MONTANTE`, como se segue:[84]

```
PEDIDO-PARA-MONTANTE() ::=
MODELO-DE-CONVERSAÇÃO(
  1,                                    ; estado inicial
  4,                                    ; estado final
  {1,2,3,4,5},                          ; estados
  { TRANSIÇÃO(1,2,ETIQUETA(S,pedido)),  ; transições
```

[84] O conjunto de tipos de mensagens e o conjunto de modelos de conversação que um agente de retalho conhece são dados pelas operações `TIPOS-MSG-RETALHO` e `MODELOS-CONV-RETALHO`, respectivamente (ver a secção 4.4.2.1).

```
TRANSIÇÃO(2,3,ETIQUETA(R,aceitação)),
TRANSIÇÃO(2,4,ETIQUETA(R,rejeição)),
TRANSIÇÃO(3,4,ETIQUETA(R,satisfação)),
TRANSIÇÃO(3,4,ETIQUETA(R,cancelamento)),
TRANSIÇÃO(3,5,ETIQUETA(R,pedido-re)),
TRANSIÇÃO(5,3,ETIQUETA(S,aceitação-re)) } ).
```

Os estados possíveis no contexto de uma conversação de modelo *Pedido-para-Montante* (ver Figura 4-32), são:

1. Estado inicial (conversação iniciada);
2. Pedido enviado;
3. Pedido aceite;
4. Estado final (conversação terminada): pedido rejeitado, satisfeito ou cancelado pelo fornecedor;
5. Pedido de re-escalonamento recebido.

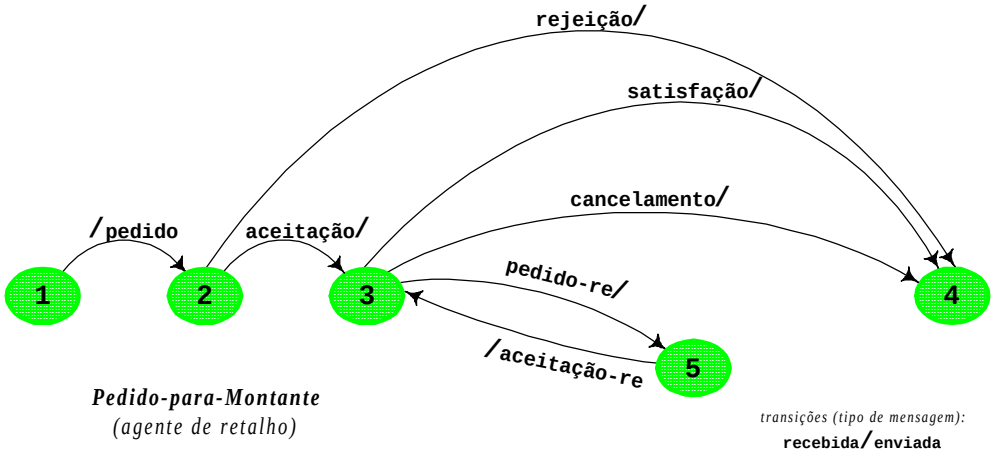


Figura 4-32- Modelo de conversação *Pedido-para-Montante*, para interacção de um agente de retalho com um agente gestor fornecedor (diagrama de estado).

No modelo de conversação *Pedido-para-Montante* os estados 1, 2, 3, 4 e 5 são estados homólogos dos estados com o mesmo número do modelo de conversação *Pedido-de-Cliente* e com transições de estado simétricas.[85] Os tipos de mensagens usados no modelo *Pedido-para-Montante* são os do conjunto `TIPOS-MSG-RETALHO()` ∩ `TIPOS-MSG-GESTOR()` (*i.e.*, `pedido`, `aceitação`, `rejeição`, `satisfação`, `cancelamento`, `pedido-re` e `aceitação-re`).

[85] A transição 3-4 por "*cancelamento*/" (recepção de uma mensagem de cancelamento) de *Pedido-para-Montante* (ver Figura 4-32) é simétrica da transição 3-4 por "/*cancelamento*" (envio de uma mensagem de cancelamento) de *Pedido-de-Cliente* (ver Figura 4-30); a transição 5-3 por "/*aceitação-re*" (envio de uma mensagem de aceitação de pedido de re-escalonamento) de *Pedido-para-Montante* é simétrica da transição 5-3 por "*aceitação-re*/" (recepção de uma mensagem de aceitação de pedido de re-escalonamento) de *Pedido-de-Cliente*.

O conteúdo de cada um dos tipos de mensagens trocadas em cada transição de estado, no contexto de conversações de modelo *Pedido-para-Montante* é descrito a seguir.

– *pedido* (S) - Conteúdo {𝕕,𝔽𝔼𝕁}, em que 𝕕 é o pedido e 𝔽𝔼𝕁 a folga externa a jusante a transmitir ao fornecedor.

– *aceitação* (R) - Conteúdo {𝔽𝔼𝕄}, em que 𝔽𝔼𝕄 é a folga externa a montante transmitida pelo fornecedor;

– *rejeição* (R), *satisfação* (R), *cancelamento* (R) e *aceitação-re* (S) - Conteúdo {};

– *pedido-re* (R) - Conteúdo idêntico ao da mensagem do mesmo tipo do modelo de conversação *Pedido-a-Fornecedor* (a folga de justificação, se existir, é ignorada pelo agente de retalho, por ser desnecessária, já que o agente de retalho aceitará qualquer pedido de re-escalonamento de fornecedor).

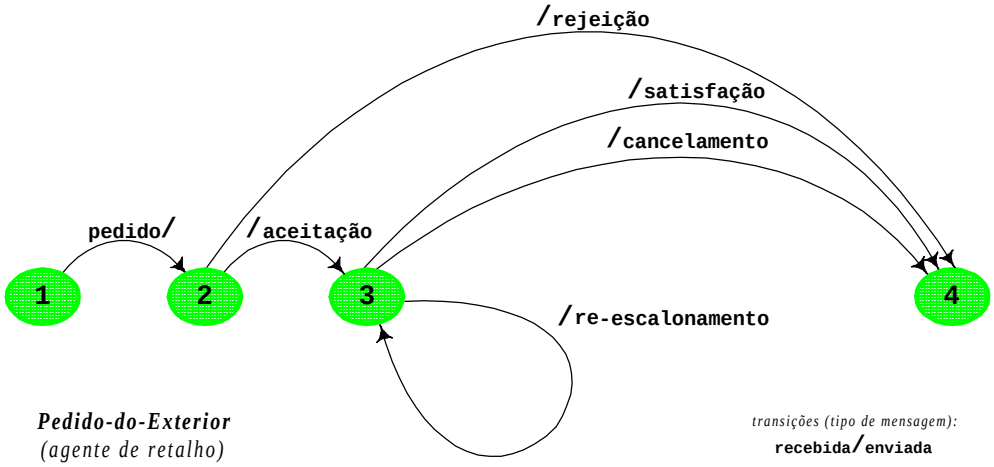


Figura 4-33- Modelo de conversação *Pedido-do-Exterior*, para interacção de um agente de retalho com o agente supervisor (diagrama de estado).

O modelo para a classe de conversação *Pedido-do-Exterior* é representado no diagrama de estados da Figura 4-33. Este modelo é definido pela operação PEDIDO-DO-EXTERIOR, como se segue:

```
PEDIDO-DO-EXTERIOR() ::=
MODELO-DE-CONVERSAÇÃO(
  1,                                          ; estado inicial
  4,                                          ; estado final
  {1,2,3,4},                                  ; estados
  { TRANSIÇÃO(1,2,ETIQUETA(R,pedido)),        ; transições
    TRANSIÇÃO(2,3,ETIQUETA(S,aceitação)),
    TRANSIÇÃO(2,4,ETIQUETA(S,rejeição)),
    TRANSIÇÃO(3,4,ETIQUETA(S,satisfação)),
    TRANSIÇÃO(3,4,ETIQUETA(S,cancelamento)),
    TRANSIÇÃO(3,3,ETIQUETA(S,re-escalonamento)) } ).
```

Os estados possíveis no contexto de uma conversação de modelo *Pedido-do-Exterior* (ver Figura 4-33), são:

1. Estado inicial (conversação iniciada);
2. Pedido recebido;
3. Pedido aceite;
4. Estado final (conversação terminada): pedido rejeitado, satisfeito ou cancelado.

Os tipos de mensagens usados no modelo *Pedido-do-Exterior* são os do conjunto `TIPOS-MSG-SUPERVISOR()` ∩ `TIPOS-MSG-RETALHO()` (*i.e.*, `pedido`, `aceitação`, `rejeição`, `satisfação`, `cancelamento` e `re-escalonamento`).

O conteúdo de cada um dos tipos de mensagens trocadas, em cada transição de estado, no contexto de conversações de modelo *Pedido-do-Exterior* é descrito a seguir.

– *pedido* (R) - Conteúdo {𝕕,𝔻𝔻}, em que 𝕕 é o pedido do agente supervisor e 𝔻𝔻 a data limite mais tarde para o agente;

– *aceitação* (S) - Conteúdo {𝔽𝕋}, em que 𝔽𝕋 é a folga total;

– *rejeição* (S), *satisfação* (S), *cancelamento* (S) - Conteúdo {};

– *re-escalonamento* (S) - Conteúdo {𝕕'}, em que 𝕕' é o pedido re-escalonado.

4.4.7.3 Os Modelos de Conversação de Agentes de Matéria-Prima

Para os agentes de matéria-prima, as conversações podem envolver interacção com um agente gestor ou com o agente supervisor. Os modelos de conversação para cada um destes casos são:

- *Pedido-de-Jusante* - Uma instância da conversação deste modelo é criada internamente no agente de matéria-prima quando ele recebe um pedido de um cliente (que é sempre um agente gestor). A conversação termina com a satisfação do pedido pelo agente ou com o cancelamento do pedido pelo cliente;
- *Pedido-para-o-Exterior* - Uma instância da conversação deste modelo é criada internamente no agente de matéria-prima quando ele deve enviar um novo pedido global ao exterior, ao agente supervisor. A conversação termina com a satisfação do pedido pelo supervisor ou com o cancelamento do pedido pelo agente.

O modelo de conversação *Pedido-de-Jusante* é representado no diagrama de estados da Figura 4-34 e é definido pela operação `PEDIDO-DE-JUSANTE`, como se segue:[86]

```
PEDIDO-DE-JUSANTE() ::=
MODELO-DE-CONVERSAÇÃO(
  1,                                        ; estado inicial
  4,                                        ; estado final
  {1,2,3,4,5},                              ; estados
  { TRANSIÇÃO(1,2,ETIQUETA(R,pedido)),      ; transições
    TRANSIÇÃO(2,3,ETIQUETA(S,aceitação)),
    TRANSIÇÃO(3,4,ETIQUETA(S,satisfação)),
    TRANSIÇÃO(3,4,ETIQUETA(R,cancelamento)),
```

[86] O conjunto de tipos de mensagens e o conjunto de modelos de conversação que um agente de matéria-prima conhece são dados pelas operações `TIPOS-MSG-MATÉRIA-PRIMA` e `MODELOS-CONV-MATÉRIA-PRIMA`, respectivamente (ver a secção 4.4.3.1).

```
TRANSIÇÃO(3,5,ETIQUETA(R,pedido-re)),
TRANSIÇÃO(5,3,ETIQUETA(S,aceitação-re)) } ).
```

Os estados possíveis no contexto de uma conversação de modelo *Pedido-de-Jusante* (ver Figura 4-34), são:

1. Estado inicial (conversação iniciada);
2. Pedido recebido;
3. Pedido aceite;
4. Estado final (conversação terminada): pedido satisfeito, ou cancelado pelo cliente;
5. Pedido de re-escalonamento recebido.

No modelo de conversação *Pedido-de-Jusante* os estados 1, 2, 3, 4 e 5 são estados homólogos dos estados 1, 2, 3, 4 e 6, respectivamente, do modelo de conversação *Pedido-a-Fornecedor* e com transições de estado simétricas.[87] Os tipos de mensagens usados no modelo *Pedido-de-Jusante* são os do conjunto `TIPOS-MSG-MATÉRIA-PRIMA()` ∩ `TIPOS-MSG-GESTOR()` (*i.e.*, `pedido`, `aceitação`, `satisfação`, `cancelamento`, `pedido-re` e `aceitação-re`).

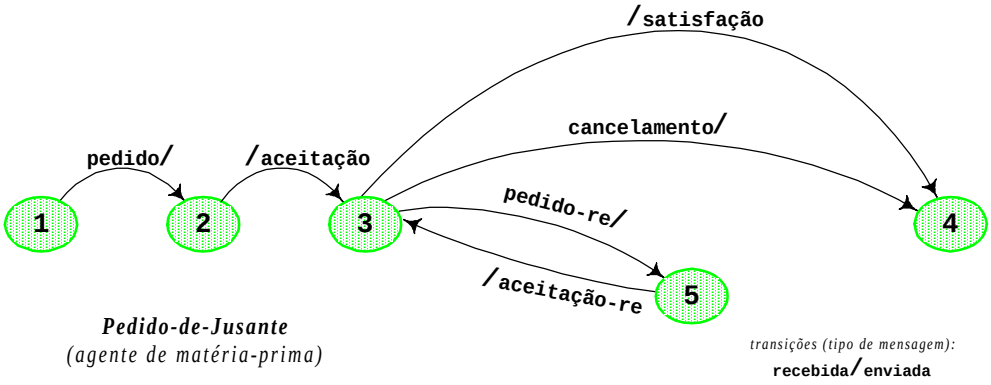


Figura 4-34- Modelo de conversação *Pedido-de-Jusante*, para interacção de um agente de matéria-prima com um agente gestor cliente (diagrama de estado).

O conteúdo de cada um dos tipos de mensagens trocadas, em cada transição de estado, no contexto de conversações de modelo *Pedido-de-Jusante* é descrito a seguir.

– *pedido* (R) - Conteúdo {𝕕, 𝔽𝔼𝕁}, em que 𝕕 é o pedido do cliente e 𝔽𝔼𝕁 a folga externa a jusante transmitida pelo cliente;

– *aceitação* (S) - Conteúdo {𝔽𝔼𝕄}, em que 𝔽𝔼𝕄 é a folga externa a montante a transmitir ao cliente;

[87] A transição 3-4 por "*cancelamento*/" (recepção de uma mensagem de cancelamento) de *Pedido-de-Jusante* (ver Figura 4-34) é simétrica da transição 3-4 por "/*cancelamento*" (envio de uma mensagem de cancelamento) de *Pedido-a-Fornecedor* (ver Figura 4-31); a transição 5-3 por "/*aceitação-re*" (envio de uma mensagem de aceitação de pedido de re-escalonamento) de *Pedido-de-Jusante* é simétrica da transição 6-3 por "*aceitação-re*/" (recepção de uma mensagem de aceitação de pedido de re-escalonamento) de *Pedido-a-Fornecedor*.

– *satisfação* (S), *cancelamento* (R) e *aceitação-re* (S) - Conteúdo {};

– *pedido-re* (R) - Conteúdo idêntico ao da mensagens do mesmo tipo do modelo de conversação *Pedido-de-Cliente* (a folga de justificação, se existir, é ignorada pelo agente de matéria-prima, por ser desnecessária, já que o agente de matéria-prima aceitará qualquer pedido de re-escalonamento de cliente).

O modelo para a classe de conversação *Pedido-para-o-Exterior* é representado no diagrama de estados da Figura 4-35. Este modelo é definido pela operação PEDIDO-PARA-O-EXTERIOR, como se segue:

```
PEDIDO-PARA-O-EXTERIOR() ::=
MODELO-DE-CONVERSAÇÃO(
  1,                                          ; estado inicial
  4,                                          ; estado final
  {1,2,3,4},                                  ; estados
  { TRANSIÇÃO(1,2,ETIQUETA(S,pedido)),        ; transições
    TRANSIÇÃO(2,3,ETIQUETA(R,aceitação)),
    TRANSIÇÃO(3,4,ETIQUETA(R,satisfação)),
    TRANSIÇÃO(3,4,ETIQUETA(S,cancelamento)),
    TRANSIÇÃO(3,3,ETIQUETA(S,re-escalonamento)) } ).
```

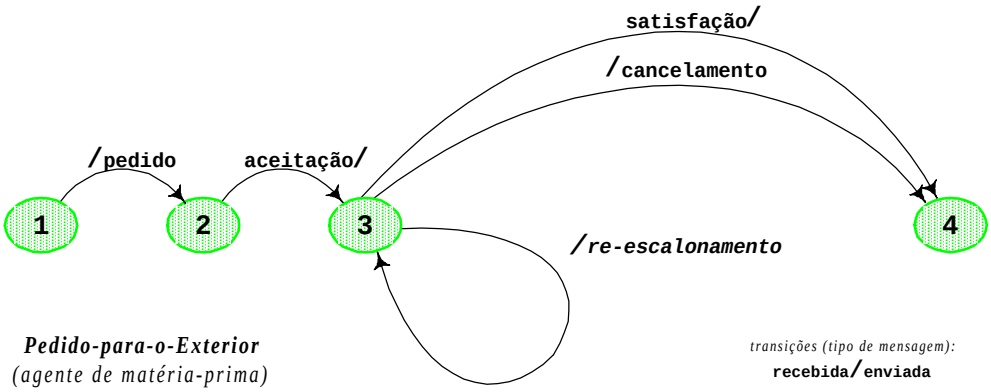


Figura 4-35- Modelo de conversação *Pedido-para-o-Exterior*, para interacção de um agente de matéria-prima com o agente supervisor (diagrama de estado).

Os estados possíveis no contexto de uma conversação de modelo *Pedido-para-o-Exterior* (ver Figura 4-35), são:

1. Estado inicial (conversação iniciada);
2. Pedido enviado;
3. Pedido aceite;
4. Estado final (conversação terminada): pedido satisfeito ou cancelado.

Os tipos de mensagens usados no modelo *Pedido-para-o-Exterior* são os do conjunto TIPOS-MSG-SUPERVISOR() ∩ TIPOS-MSG-MATÉRIA-PRIMA() (*i.e.*, pedido, aceitação, satisfação, cancelamento e re-escalonamento).

O conteúdo de cada um dos tipos de mensagens trocadas, em cada transição de estado, no contexto de conversações de modelo *Pedido-para-o-Exterior* é descrito a seguir.

– *pedido* (S) - Conteúdo {𝕕}, em que 𝕕 é o pedido ao agente supervisor;

– *aceitação* (R) - Conteúdo {ℝ𝔻}, em que ℝ𝔻 é a data de lançamento mais cedo;

– *satisfação* (R), *cancelamento* (S) - Conteúdo {};

– *re-escalonamento* (S) - Conteúdo {𝕕'}, em que 𝕕' é o pedido re-escalonado.

4.4.7.4 Os Modelos de Conversação do Agente Supervisor

Para o agente supervisor, as conversações podem envolver interacção com um agente de retalho ou com um agente de matéria-prima. Os modelos de conversação para cada um destes casos são:

- *Pedido-Global-à-Saída* - Uma instância da conversação deste modelo é criada internamente no agente supervisor quando ele deve enviar um novo pedido global do exterior, para um agente de retalho (o que acontece quando o agente supervisor tem um novo pedido global do exterior à rede). A conversação termina com a rejeição, a satisfação ou o cancelamento do pedido pelo agente de retalho.
- *Pedido-Global-à-Entrada* - Uma instância da conversação deste modelo é criada internamente no agente supervisor quando ele recebe um novo pedido global ao exterior, de um agente de matéria-prima. A conversação termina com a satisfação do pedido pelo supervisor ou com o cancelamento do pedido pelo agente de matéria-prima.

As mensagens do tipo `pedido-in` e do tipo `pedido-out`, do conjunto `TIPOS-MSG-SUPERVISOR()`, são mensagens especiais, com um formato distinto das que o agente pode trocar no contexto da rede de EE. Estas mensagens são usadas apenas para comunicação entre o agente supervisor e o exterior da rede de EE, não são usadas nos modelos de conversação do conjunto `MODELOS-CONV-SUPERVISOR()` e são descritas na secção 4.4.7.5.[88]

O modelo para a classe de conversação *Pedido-Global-à-Saída* é representado no diagrama de estados da Figura 4-36. Este modelo é definido pela operação `PEDIDO-GLOBAL-À-SAÍDA`, como se segue:

```
PEDIDO-GLOBAL-À-SAÍDA() ::=
MODELO-DE-CONVERSAÇÃO(
  1,                                        ; estado inicial
  4,                                        ; estado final
  {1,2,3,4},                                ; estados
  { TRANSIÇÃO(1,2,ETIQUETA(S,pedido)),      ; transições
    TRANSIÇÃO(2,3,ETIQUETA(R,aceitação)),
    TRANSIÇÃO(2,4,ETIQUETA(R,rejeição)),
    TRANSIÇÃO(3,4,ETIQUETA(R,satisfação)),
    TRANSIÇÃO(3,4,ETIQUETA(R,cancelamento)),
```

[88] O conjunto de tipos de mensagens e o conjunto de modelos de conversação que o agente supervisor conhece são dados pelas operações `TIPOS-MSG-SUPERVISOR` e `MODELOS-CONV-SUPERVISOR`, respectivamente (ver a secção 4.4.4.1).

```
TRANSIÇÃO(3,3,ETIQUETA(R,re-escalonamento)) } ).
```

O modelo de conversação *Pedido-Global-à-Saída* é completamente simétrico do modelo *Pedido-do-Exterior* de agente de retalho.[89]

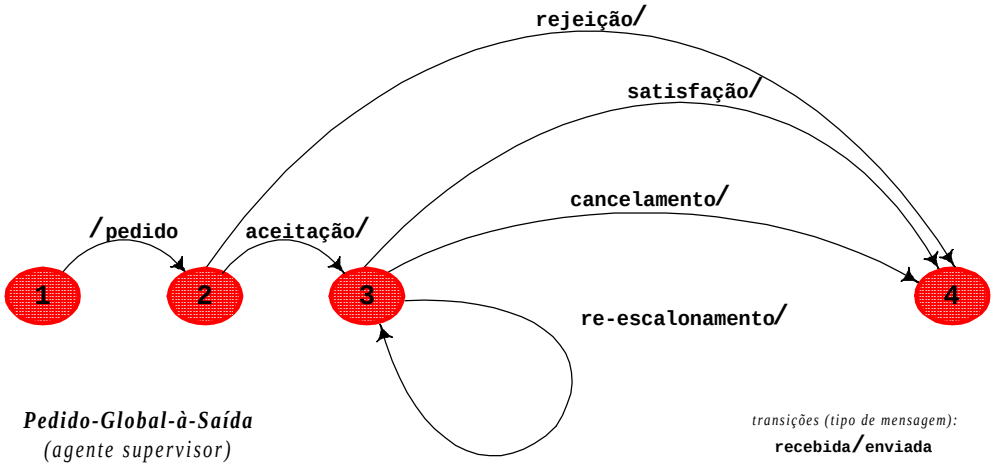


Figura 4-36- Modelo de conversação *Pedido-Global-à-Saída*, para interacção do agente supervisor com um agente de retalho (diagrama de estado).

O modelo para a classe de conversação *Pedido-Global-à-Entrada* é representado no diagrama de estados da Figura 4-37. Este modelo é definido pela operação PEDIDO-GLOBAL-À-ENTRADA, como se segue:

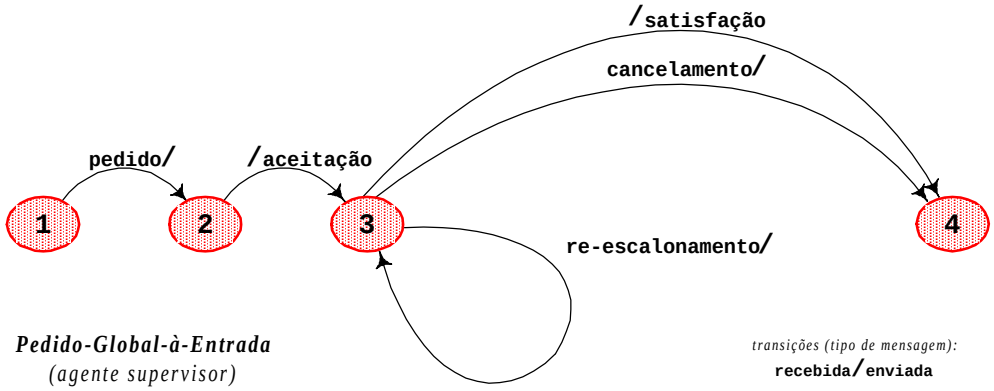


Figura 4-37- Modelo de conversação *Pedido-Global-à-Entrada*, para interacção do agente supervisor com um agente de matéria-prima (diagrama de estado).

```
PEDIDO-GLOBAL-À-ENTRADA() ::=
MODELO-DE-CONVERSAÇÃO(
```

[89] Os dois modelos têm o mesmo número total de estados, os estados com o mesmo número num e noutro modelo são estados homólogos e as transições são simétricas, com o mesmo tipo e conteúdo idêntico das mensagens trocadas.

```
1,                                          ; estado inicial
4,                                          ; estado final
{1,2,3,4},                                  ; estados
{ TRANSIÇÃO(1,2,ETIQUETA(R,pedido)),        ; transições
  TRANSIÇÃO(2,3,ETIQUETA(S,aceitação)),
  TRANSIÇÃO(3,4,ETIQUETA(S,satisfação)),
  TRANSIÇÃO(3,4,ETIQUETA(R,cancelamento)),
  TRANSIÇÃO(3,3,ETIQUETA(R,re-escalonamento)) } ).
```

O modelo de conversação *Pedido-Global-à-Entrada* é completamente simétrico do modelo *Pedido-para-o-Exterior* de agente de matéria-prima.[89]

4.4.7.5 Comunicação com o Exterior da Rede de EE

Nesta secção descrevem-se os tipos de mensagens `pedido-in` e do tipo `pedido-out`, do conjunto `TIPOS-MSG-SUPERVISOR()`, no que respeita ao conteúdo das mensagens. Como foi referido na secção anterior, estas mensagens têm um formato distinto das que o agente pode trocar no contexto da rede de EE (com os agentes de retalho e com os agentes de matéria-prima) e são usadas apenas para comunicação entre o agente supervisor e o exterior da rede de EE. Elas servem para o agente supervisor receber pedidos globais do exterior à rede (mensagens do tipo `pedido-in`) ou enviar pedidos globais da rede ao exterior (mensagens do tipo `pedido-out`).

Como foi anteriormente definido,[90] uma mensagem de tipo `pedido-in` é um par:

<pedido-in, Conteúdo>

Sendo Conteúdo um par $<d^{0,r}_{i_r,r}, \mathbb{H}_{i_r,r}>$, em que $d^{0,r}_{i_r,r}$ é um pedido global do exterior à rede e $\mathbb{H}_{i_r,r}$ é o horizonte temporal global associado a esse pedido (*.e.*, o intervalo $<RD_{i_r,r}, DD_{i_r,r}>$).

Como foi anteriormente definido também, uma mensagem de tipo `pedido-out` é um par:

<pedido-out, Conteúdo>

Em que é $Conteúdo = d^{m,0}_{i_r,r}$, sendo $d^{m,0}_{i_r,r}$ um pedido global da rede ao exterior.

4.4.8 Comportamento dos Agentes Gestores

Esta secção diz respeito à actividade interna dos agentes gestores, mais especificamente ao comportamento dos agentes daquela classe, que se relaciona com o protocolo de interacção *Gestor-Gestor*. Sistematizam-se os componentes de agente relevantes para a representação do *estado interno* de um agente gestor (na secção 4.4.8.1), descrevem-se as *capacidades* de um agente gestor (no sentido de o que um agente é capaz de realizar; na secção 4.4.8.2) e descreve-se a *actividade interna* adequada para um comportamento de agente coerente com o protocolo de interacção *Gestor-Gestor* (nas secções 4.4.8.3 e 4.4.8.4).

O modo como opera o componente de agente *Comportamento* é descrito aqui sem, no entanto, indicar detalhes de realização computacional (não só para manter a descrição

[90] Ver a secção 4.4.6.3.

abstracta, mas também para maior brevidade e por razões de espaço). A forma como até agora este componente foi descrito foi a de um conjunto de regras, do tipo condição-acção, a serem aplicadas por um interpretador de regras,[91] para execução, ou operação, do agente. No entanto, os elementos constituintes de *Comportamento*, bem como a sua organização, poderão ser traduzidos por quaisquer outros, desde que sejam funcionalmente equivalentes. Por exemplo, numa realização computacional, as regras podem ser traduzidas por procedimentos, ou rotinas, da linguagem de programação empregue, a serem executados condicionalmente, dependendo do estado dos restantes componentes do agente.[92]

4.4.8.1 Representação do Estado Interno

De acordo com a arquitectura descrita na secção 4.4.1 os componentes directos de um agente gestor, para além do identificador e do componente relativo ao comportamento (acessíveis pelas operações `LOCAL` e `COMPORTAMENTO`, respectivamente) são:

Componentes agrupados em *interface*:

- Clientes - Conjunto dos identificadores dos agentes clientes (acessível pela operação `CLIENTES`);
- Fornecedores - Conjunto dos identificadores dos agentes fornecedores (acessível pela operação `FORNECEDORES`);
- Produtos e clientes - Conjunto de pares associando cada produto de saída a cada cliente do produto (acessível pela operação `PRODUTOS-CLIENTES`);
- Produtos e fornecedores - Conjunto de pares associando cada produto de entrada a cada fornecedor do produto (acessível pela operação `MATERIAIS-FORNECEDORES`).
- Mensagens recebidas - Caixa de correio de entrada (acessível pela operação `CORREIO-IN`);
- Mensagens enviadas - Caixa de correio de saída (acessível pela operação `CORREIO-OUT`);

Componente agrupados em *estado interno*:

- Estado de interacção - Conjunto de conversações actualmente mantidas com agentes clientes e fornecedores (acessível pela operação `CONVERSAÇÕES`);
- Problemas de escalonamento - Conjunto de problemas de escalonamento locais actuais (acessível pela operação `PROBLEMAS-DE-ESCALONAMENTO`);
- Estado de capacidade - O estado do nó de capacidade gerido pelo agente (o perfil de capacidade utilizada com tarefas escalonadas; acessível pela operação `NÓ`).

Do ponto de vista da mutabilidade, os componentes podem classificar-se em componentes *estáticos* ou componentes *dinâmicos*. Para os componentes estáticos o conteúdo não se altera durante o tempo de vida do agente, uma vez que sejam estabelecidos quando o agente é gerado; para os componentes dinâmicos o conteúdo altera-se com a interacção do agente com os outros agentes na rede de EE. São estes últimos que contribuem directamente para a

---

[91] Ver as secções 4.4.1, 4.4.2, 4.4.3 e 4.4.4.

[92] Neste caso, o componente *Comportamento* poderia ser visto como um *módulo* do agente contendo uma colecção de procedimentos, com instruções que invocam operações definidas que operam sobre os restantes componentes do agente.

componente dinâmica do estado do agente (e indirectamente para componente dinâmica do estado global do sistema multi-agente). Adicionalmente, os componentes podem classificar-se em componentes *relativos ao próprio agente* ou componentes *relativos a outros agentes*. A seguir descrevem-se estes componentes de acordo com esta classificação (incluem-se componentes e sub-componentes, *i.e.*, componentes implícitos, acessíveis através de operações aplicáveis aos componentes de agente acima listados).

Componentes estáticos relativos aos outros agentes:

1. Clientes e fornecedores (operações `CLIENTES` e `FORNECEDORES`, respectivamente);
2. Produtos de saída, produtos de entrada, clientes de cada produto de saída e fornecedores de cada produto de entrada (operações `PRODUTOS`, `MATERIAIS`, `PRODUTOS-CLIENTES` e `MATERIAIS-FORNECEDORES`, respectivamente).

Componentes estáticos relativos ao próprio agente (exceptuando o identificador do agente e o comportamento):

3. Tipo de nó, produtos de entrada necessários para cada produto de saída, proporção entre um produto de entrada e um produto de saída, factores de conversão de capacidade (em *uppp* ou *uppa*, conforme o tipo de nó, por unidade de produto de saída) para cada produto de saída e perfil de capacidade máxima (operações `TIPO`, `MATERIAIS-PARA-PRODUTO`, `RAZÃO-MATERIAL-PRODUTO`, `FACTOR-C` e `PERFIL-C-MAX`, respectivamente, aplicáveis ao nó gerido pelo agente).

Componentes dinâmicos relativos aos outros agentes:

4. Mensagens recebidas e mensagens enviadas (operações `CORREIO-IN` e `CORREIO-OUT`, respectivamente);
5. Conjunto de conversações com outros agentes (operação `CONVERSAÇÕES`);
6. Conjunto de problemas de escalonamento (um por cada problema de escalonamento colocado à rede e por pedido local de cliente aceite, não satisfeito e não cancelado; operação `PROBLEMAS-DE-ESCALONAMENTO`).

Componentes dinâmicos relativos ao próprio agente:

7. Conjunto de perfis de capacidade utilizada e perfil de capacidade utilizada para cada produto de saída, com as tarefas escalonadas (operações `PERFIS-C-UTIL` e `PERFIL-C-UTIL`, respectivamente, aplicáveis ao nó gerido pelo agente).

Os itens incluídos 4, 5, 6 e 7 representam, em conjunto, a componente dinâmica do estado interno actual do agente.

4.4.8.2 Capacidades

As capacidades são as *acções* que o agente *pode* realizar. Dividem-se em *capacidades externas* (ou de interacção/comunicação) e *capacidades internas*. As *capacidades externas* têm a ver a actuação no meio exterior ao agente, um meio que ele apenas pode controlar parcialmente e são expressas por actos (de discurso, ou outros). As *capacidades internas* são capacidades de actuação no meio interior do agente, que ele controla directamente. Estas capacidades são a seguir descritas, de acordo com esta classificação (indicam-se as operações de agente gestor que correspondem a cada capacidade).

As *capacidades externas* são:

1. *Receber comunicação do exterior* - Receber uma mensagem de outro agente, através da caixa de correio de entrada, com a actualização da conversação respectiva (abrange as operações HÁ-CORREIO-IN?, OBTÉM-MSG-IN, RETIRA-CORREIO-IN, INTRODUZ-CONV, TRANSITA-CONV e RETIRA-CONV);
2. *Enviar comunicação para o exterior* - Enviar uma mensagem a outro agente, através da caixa de correio de saída, com a actualização da conversação respectiva (abrange as operações PÕE-CORREIO-OUT, INTRODUZ-CONV, TRANSITA-CONV e RETIRA-CONV).

As *capacidades internas* são:

3. *Escalonamento de tarefas* - Adicionar ou remover problemas de escalonamento (abrange as operações JUNTA-PROBLEMA e RETIRA-PROBLEMA) e escalonar ou desfazer o escalonamento das tarefas no nó gerido pelo agente (abrange as operações TAREFA, ESCALONA-TAREFA e DES-ESCALONA-TAREFA);
4. *Auto-inspecção* - Inspecção dos componentes do próprio agente para acesso ao estado interno, escalonamento e re-escalonamento.[93]

4.4.8.3 Operações Específicas com Conversações do Protocolo Gestor-Gestor

Nesta secção definem-se algumas operações adicionais com conversações, que facilitam o tratamento de conversações de modelo *Pedido-de-Cliente* ou *Pedido-a-Fornecedor* (aplicáveis apenas a este tipo de conversações).

Os predicados definidos a seguir permitem aceder ao estado e à evolução seguida por uma conversação de modelo *Pedido-de-Cliente* ou *Pedido-a-Fornecedor*, do agente com outro agente (o parâmetro conv significa a conversação) no passado imediato (última transição). Os predicados testam, respectivamente, se algum dos agentes enviou um pedido novo ao outro, se o pedido foi aceite, se o pedido foi rejeitado, se o agente recebeu um pedido de re-escalonamento do outro agente, se o agente aceitou o pedido de re-escalonamento, se o agente rejeitou o pedido de re-escalonamento, se o agente enviou ao outro agente um pedido de re-escalonamento, se o outro agente aceitou o pedido de re-escalonamento, se o outro agente rejeitou o pedido de re-escalonamento, se o pedido foi já satisfeito, se o agente recebeu um cancelamento do outro agente e se o agente cancelou o pedido em questão.

---

[93] Incluem-se aqui as operações EXISTE-CONV?, CONVS-AGENTE, CONV-AGENTE, CONVS-CLIENTES, CONVS-FORNECEDORES, CONV-CLIENTE e CONVS-ESTADO, que dizem respeito ao estado comunicacional e as operações EXISTE-PE? e PE, para problemas de escalonamento (ver, no presente capítulo, a secção 4.4.1). Relativamente ao estado de capacidade, incluem-se também as operações para a inspecção dos perfis de capacidade C-MAX, C-UTIL, C-DISP, C-DISP-MIN, C-MAX-TOTAL, C-UTIL-TOTAL; para determinação da capacidade ou duração necessárias a uma tarefa C-NEC e D-NEC; para detecção de conflitos de capacidade com identificadores iniciando-se por C-CONFLITO; para detecção de conflitos temporais com identificadores iniciando-se por T-CONFLITO; as operações para escalonamento e re-escalonamento ESCALONA-PTRÁS?, ESCALONA-PFRENTE?, OPÇÃO-ESCALONA-PTRÁS, OPÇÃO-ESCALONA-PFRENTE, RE-ESCALONA-PTRÁS?, RE-ESCALONA-PFRENTE?, OPÇÃO-RE-ESCALONA-PTRÁS, OPÇÃO-RE-ESCALONA-PFRENTE. Estas operações são aplicáveis ao nó gerido pelo agente (ver o Capítulo 3), à excepção das operações para detecção de conflitos temporais, que se aplicam a problemas de escalonamento de agente gestor (ver no presente capítulo, a secção 4.4.5.2).

```
PEDIDO-NOVO?(conv) ::= (ESTADO(conv)=2).

ACEITE?(conv) ::= (ESTADO(conv)=3).

REJEITADO?(conv) ::=
CONVERSAÇÃO-TERMINADA?(conv) ∧
(TIPO-DE-MENSAGEM(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=rejeição).

PEDIDO-RE-NOVO?(conv) ::=
(SENTIDO(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=R) ∧
(TIPO-DE-MENSAGEM(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=
                                          pedido-re).

ACEITAÇÃO-RE-ENVIADA?(conv) ::=
(SENTIDO(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=S) ∧
(TIPO-DE-MENSAGEM(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=
                                          aceitação-re).

REJEIÇÃO-RE-ENVIADA?(conv) ::=
(SENTIDO(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=S) ∧
(TIPO-DE-MENSAGEM(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=
                                          rejeição-re).

PEDIDO-RE-ENVIADO?(conv) ::=
(SENTIDO(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=S) ∧
(TIPO-DE-MENSAGEM(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=
                                          pedido-re).

RE-ACEITE?(conv) ::=
(SENTIDO(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=R) ∧
(TIPO-DE-MENSAGEM(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=
                                          aceitação-re).

RE-REJEITADO?(conv) ::=
(SENTIDO(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=R) ∧
(TIPO-DE-MENSAGEM(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=
                                          rejeição-re).

PEDIDO-SATISFEITO?(conv) ::=
CONVERSAÇÃO-TERMINADA?(conv) ∧
(TIPO-DE-MENSAGEM(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=
                                        satisfação).

CANCELAMENTO-RECEBIDO?(conv) ::=
CONVERSAÇÃO-TERMINADA?(conv) ∧
```

```
(SENTIDO(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=R) ∧
(TIPO-DE-MENSAGEM(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=
                                              cancelamento).

CANCELAMENTO-ENVIADO?(conv) ::=
CONVERSAÇÃO-TERMINADA?(conv) ∧
(SENTIDO(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=S) ∧
(TIPO-DE-MENSAGEM(ETIQUETA(ÚLTIMA-TRANSIÇÃO(conv)))=
                                              cancelamento).
```

A operação seguinte é um predicado que, para um dado agente, dada uma conversação do modelo *Pedido-de-Cliente*, no contexto da qual o agente deve responder ao pedido de cliente (conversação no estado 2), testa se todos os pedidos de fornecimentos necessários à satisfação do pedido do cliente foram feitos pelo agente e *todos* tiveram já resposta, de aceitação ou de rejeição, por parte dos fornecedores (conv é a conversação com o agente cliente).

```
PEDIDOS-DE-FORNECIMENTO-RESPONDIDOS?(Gk,conv) ::=
(conv∈CONVS-CLIENTES(Gk)) ∧ PEDIDO-NOVO?(conv) ∧
(∀pj  (pj∈MATERIAIS-PARA-PRODUTO(
        NÓ(Gk),
        PRODUTO(PEDIDO(ÚLTIMO(MENSAGENS-RECEBIDAS(conv)))))) :
    EXISTE-CONV?(Gk,FORNECEDOR(Gk,pj),ID-PE(conv)) ∧
    (ACEITE?(CONV-AGENTE( Gk,
                          FORNECEDOR(Gk,pj),
                          ID-PE(conv))) ∨
     REJEITADO?(CONV-AGENTE(  Gk,
                              FORNECEDOR(Gk,pj),
                              ID-PE(conv))))).
```

As operações que a seguir se definem exprimem a realização de transições e o envio de mensagens.

Para cancelamento de um pedido de um cliente, ou de um pedido a um fornecedor, define-se a operação CANCELA, que realiza a transição apropriada na conversação com o outro agente e envia a mensagem de cancelamento (conv é a conversação com o outro agente).

```
CANCELA(Gk,conv) ::=
TRANSITA-CONV(  Gk,
                conv,
                MENSAGEM( cancelamento,
                          LOCAL(Gk),
                          SE( INICIANTE(conv)=LOCAL(Gk),
                              DESTINATÁRIO(conv),
                              INICIANTE(conv)),
                          ID-PE(conv),
                          { } )).
```

As operações ACEITA e REJEITA tratam uma conversação de modelo *Pedido-de-Cliente* e realizam, respectivamente, a aceitação e a rejeição de um pedido de cliente, com a transição apropriada na conversação com o agente cliente e envio da mensagem adequada (conv é a conversação com o cliente e *Conteúdo* o conteúdo da mensagem de aceitação).

```
ACEITA(Gk,conv,Conteúdo) ::=
TRANSITA-CONV(  Gk,
                conv,
                MENSAGEM( aceitação,
                          LOCAL(Gk),
                          INICIANTE(conv),
                          ID-PE(conv),
                          Conteúdo )).

REJEITA(Gk,conv) ::=
TRANSITA-CONV(  Gk,
                conv,
                MENSAGEM( rejeição,
                          LOCAL(Gk),
                          INICIANTE(conv),
                          ID-PE(conv),
                          { } )).
```

A operação FAZ-PEDIDO-RE realiza o envio de um pedido de re-escalonamento a outro agente (conv é a conversação com o outro agente e *Conteúdo* é o conteúdo da mensagem de re-escalonamento):

```
FAZ-PEDIDO-RE(Gk,conv,Conteúdo) ::=
TRANSITA-CONV(  Gk,
                conv,
                MENSAGEM( pedido-re,
                          LOCAL(Gk),
                          SE( INICIANTE(conv)=LOCAL(Gk),
                              DESTINATÁRIO(conv),
                              INICIANTE(conv)),
                          ID-PE(conv),
                          Conteúdo)).
```

As operações ACEITA-RE e REJEITA-RE realizam, respectivamente, a aceitação e a rejeição de um pedido de re-escalonamento de outro agente (conv é a conversação com o outro agente):

```
ACEITA-RE(Gk,conv) ::=
TRANSITA-CONV(  Gk,
                conv,
                MENSAGEM( aceitação-re,
```

```
                        LOCAL(Gk),
                        SE( INICIANTE(conv)=LOCAL(Gk),
                            DESTINATÁRIO(conv),
                            INICIANTE(conv)),
                        ID-PE(conv),
                        {} )).

REJEITA-RE(Gk,conv) ::=
TRANSITA-CONV(  Gk,
                conv,
                MENSAGEM( rejeição-re,
                        LOCAL(Gk),
                        SE( INICIANTE(conv)=LOCAL(Gk),
                            DESTINATÁRIO(conv),
                            INICIANTE(conv)),
                        ID-PE(conv),
                        {} )).
```

Por fim, a operação SATISFAZ realiza a satisfação de um pedido de cliente (esta operação é usada quando TEMPO-ACTUAL()=TEMPO(d), em que d é o pedido actualizado do cliente; conv é a conversação com o agente cliente).

```
SATISFAZ(Gk,conv) ::=
TRANSITA-CONV(  Gk,
                conv,
                MENSAGEM( satisfação,
                        LOCAL(Gk),
                        INICIANTE(conv),
                        ID-PE(conv),
                        { } )).
```

4.4.8.4 O Ciclo de Operação do Agente Gestor

Foram definidos, na secção 4.4.7, quatro casos de interacção no contexto do protocolo *Gestor-Gestor*. Estes quatro casos, ilustrados pelos diagramas de sequência na Figura 4-27, centram-se na interacção de um agente gestor com um cliente e com fornecedores envolvidos no mesmo problema de escalonamento (e, portanto, envolvidos na satisfação do mesmo pedido global do exterior à rede). Nas figuras que se seguem, ilustra-se de forma sintetizada a actividade interna do agente gestor, para cada um daqueles casos de interacção, através de diagramas de sequência envolvendo componentes do agente.

A Figura 4-38 diz respeito ao caso de interacção durante o estabelecimento de um pedido local de um problema de escalonamento e abrange os casos com e sem rejeição do pedido do cliente (confronte-se o diagrama da Figura 4-38 com o do caso a) da Figura 4-27). A interacção referida ocorre durante a fase 1 de resolução de um problema de escalonamento, de acordo com a secção 4.3.6.3. A rejeição do pedido do cliente pode ocorrer tanto pela razão de um ou mais dos pedidos a fornecedores serem rejeitados, como pela razão de o agente concluir (após todos os pedidos a fornecedores terem sido aceites) que o problema de escalonamento é

temporalmente super-constrangido.[94] A primeira e a segunda mensagem de rejeição para o cliente, no diagrama de sequência da Figura 4-38, dizem respeito ao primeiro e ao segundo daqueles dois casos de rejeição, respectivamente. Como se pode ver, o estabelecimento de pedidos locais pode envolver cancelamentos de pedidos, no caso de ter de existir rejeição do pedido do cliente. No primeiro caso de rejeição acima descrito (quando há rejeição de um ou mais dos fornecedores), o agente deve enviar mensagens de cancelamento aos fornecedores que aceitaram os pedidos de fornecimento, se os houver; no segundo caso (problema de escalonamento temporalmente super-constrangido), o agente deve enviar mensagens de cancelamento a todos os fornecedores, pois todos os pedidos de fornecimento foram aceites.

A Figura 4-39 mostra os três casos possíveis de interacção de cancelamento, relacionados com conflitos de capacidade (incluídos em b), na Figura 4-27), que ocorrem durante a fase 3 de resolução de um problema de escalonamento. Estes casos são originados pelo facto de existir pelo menos um agente gestor na rede, envolvido no mesmo problema de escalonamento, que não consegue resolver conflitos de capacidade e opta por desfazer o escalonamento da tarefa do problema e cancelar os respectivos pedidos a clientes e a fornecedores. O caso b1) da Figura 4-39 (cancelamento por iniciativa do agente), ilustra a situação em que o próprio agente (o agente gestor de referência na figura) se encontra na condição de conflito de capacidade; os casos b2 e b3 da Figura 4-39 (cancelamento do cliente e cancelamento de fornecedor) ilustram situações em que aquela condição ocorre para agentes a jusante, ou para agentes a montante, respectivamente, do próprio agente.

A Figura 4-40 ilustra o caso de interacção relativo à satisfação (o caso c) na Figura 4-27), que termina a fase 3 de resolução de um problema de escalonamento. Nesta interacção decorrem a satisfação de todos os pedidos a fornecedores e, posteriormente, a satisfação do pedido do cliente.

A Figura 4-41 e a Figura 4-42 ilustram os casos de interacção no re-escalonamento (os casos incluídos em d) na Figura 4-27), que podem ocorrer nas fases 2 e 3 de resolução de um problema de escalonamento. São representados quatro casos possíveis de interacção de re-escalonamento: d1) re-escalonamento pedido pelo agente ao cliente, d2) re-escalonamento pedido pelo agente a um fornecedor, d3) re-escalonamento pedido pelo cliente e d4) re-escalonamento pedido por um fornecedor. Os dois primeiros, d1) e d2), são casos de pedido de re-escalonamento pela iniciativa do agente e envolvem duas fases: a fase do envio de pedido de re-escalonamento e, posteriormente, a fase de recepção da resposta ao pedido de re-escalonamento.

A actividade interna de um agente gestor, decorre através de um ciclo de operação que se repete ao longo de um horizonte temporal de escalonamento, o *ciclo do agente gestor*. Assume-se que um ciclo tem a duração de um período temporal e que em cada ciclo o agente processa todas as mensagens de outros agentes, actualiza as respectivas conversações, executa acções internas de acordo com o estado das conversações com os outros agentes e o estado de capacidade e envia mensagens a outros agentes.

---

[94] Não se considera o caso de rejeição que deveria suceder se o pedido de cliente tivesse uma data limite tão próxima do tempo actual que o agente não pudesse aceitá-lo sem incorrer num conflito temporal absoluto.

a) Estabelecimento de pedido local:

a1) processamento de pedido novo de cliente;
a2) resposta a pedido de cliente.

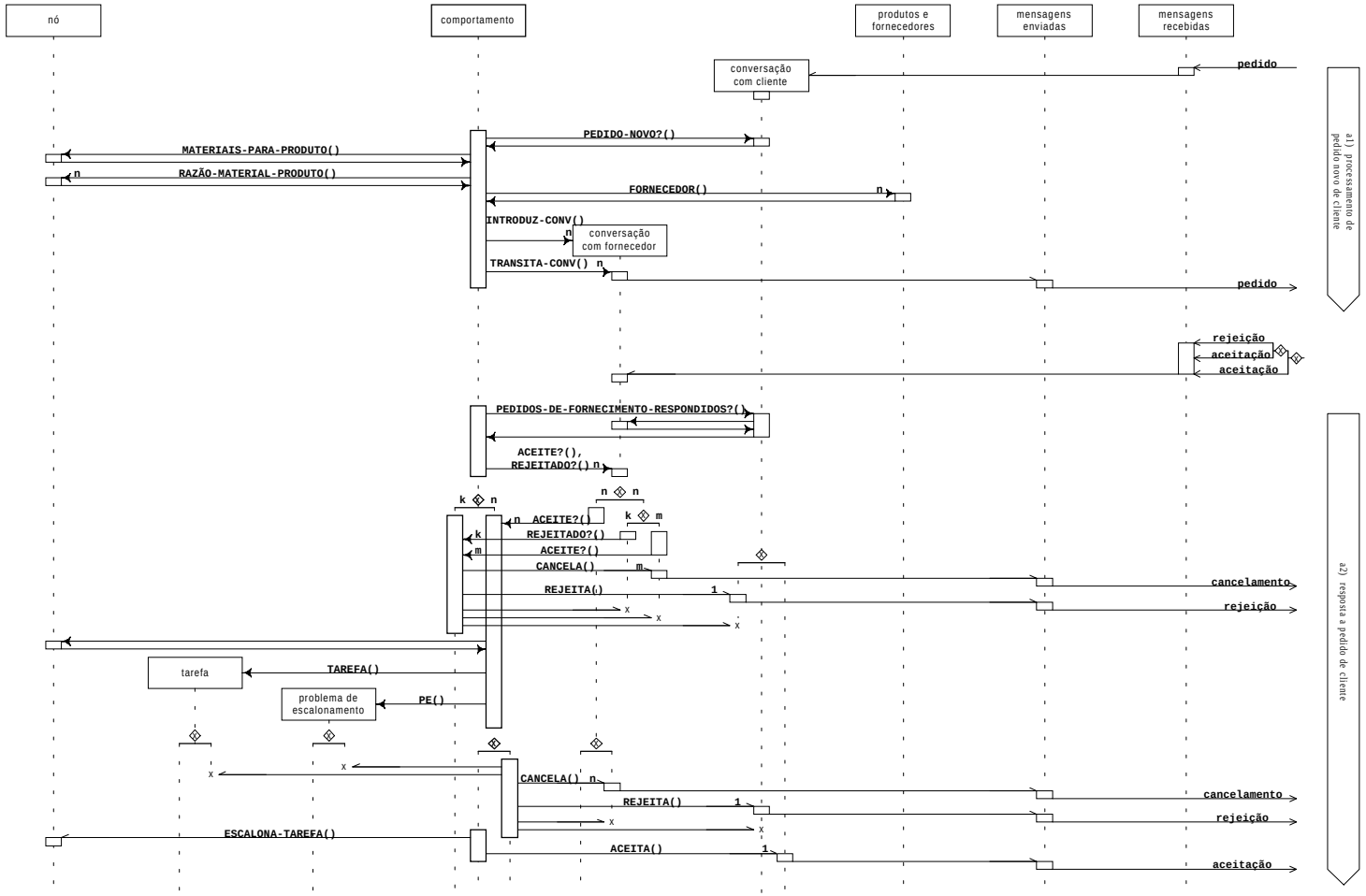


Figura 4-38- Estabelecimento de um pedido local (diagrama de sequência para a actividade interna de um agente gestor).

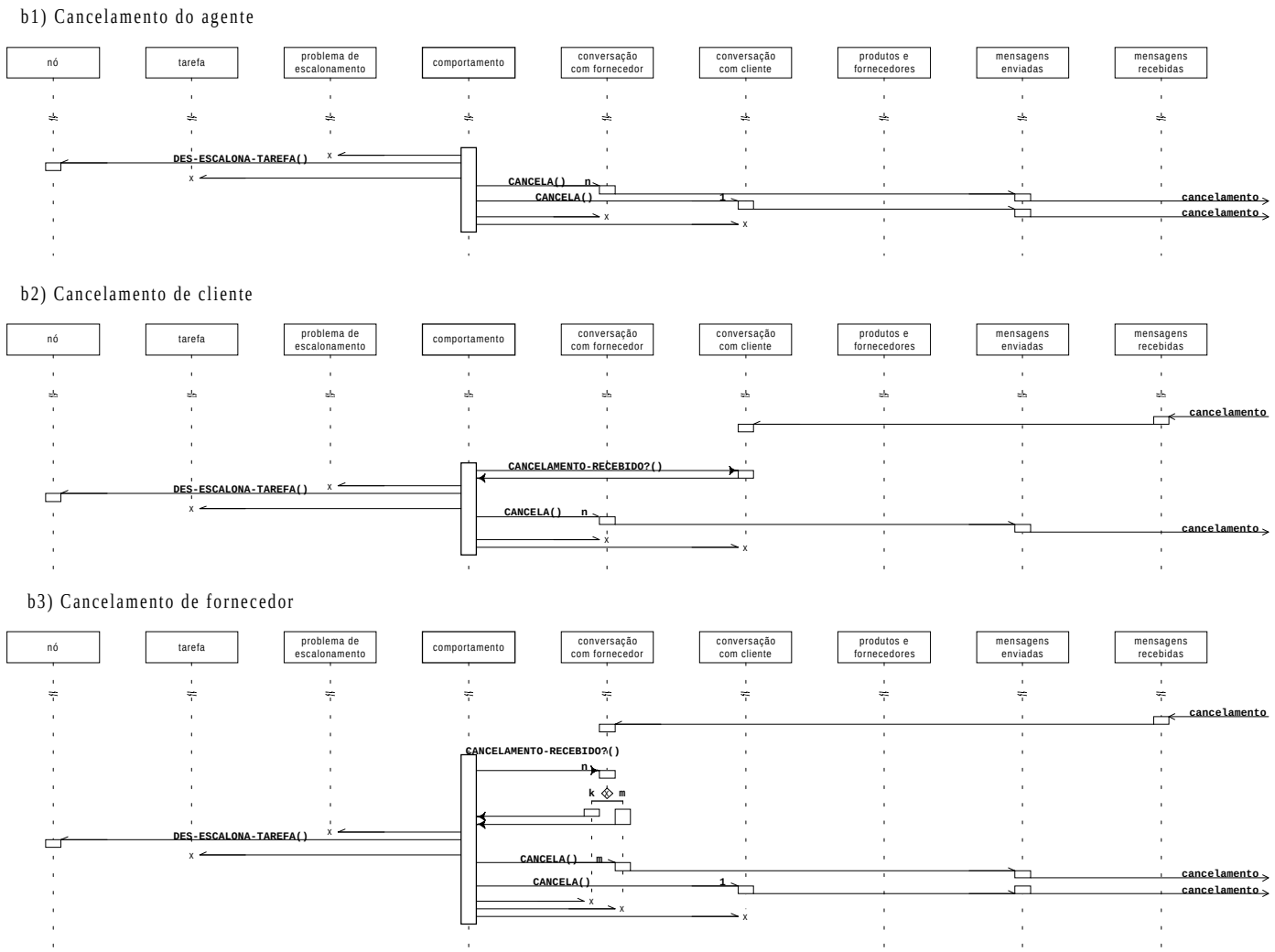


Figura 4-39- Cancelamentos (diagramas de sequência para a actividade interna de um agente gestor).

c) Satisfação:

c1) satisfação de pedidos de fornecedores;
c2) satisfação de pedido de cliente.

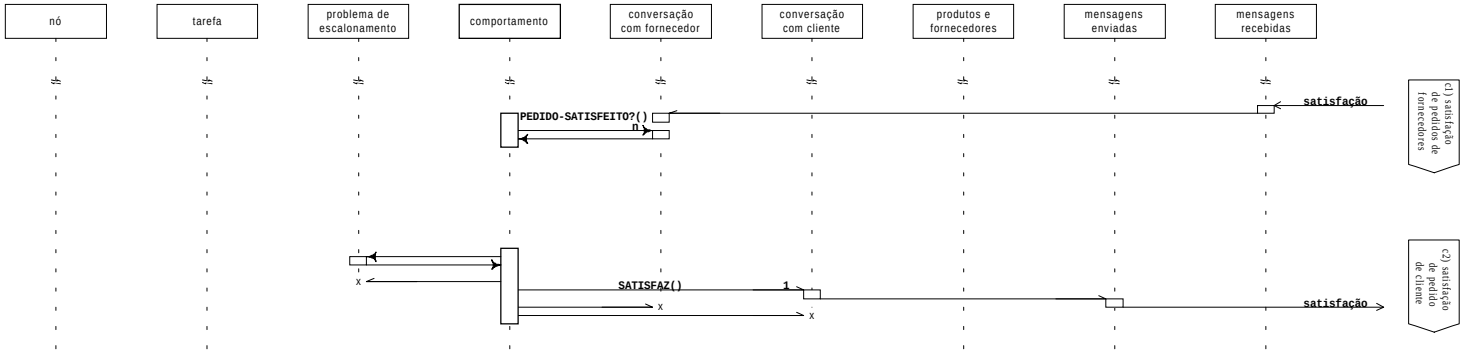
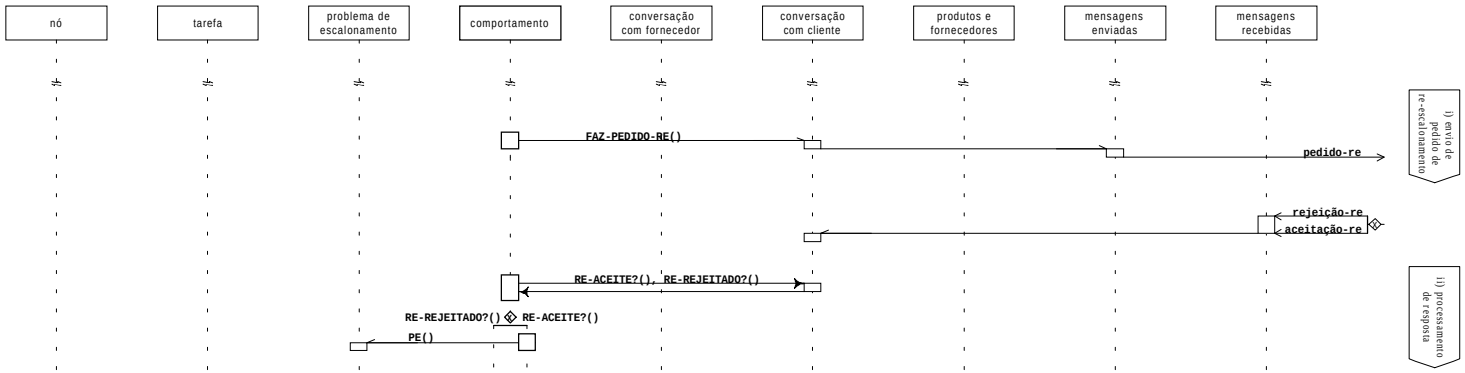


Figura 4-40- Satisfação (diagrama de sequência para a actividade interna de um agente gestor).

d1) Re-escalonamento pedido pelo agente a cliente:
i) envio de pedido de re-escalonamento;
ii) processamento de resposta.



d2) Re-escalonamento pedido pelo agente a fornecedor:
i) envio de pedido de re-escalonamento;
ii) processamento de resposta.

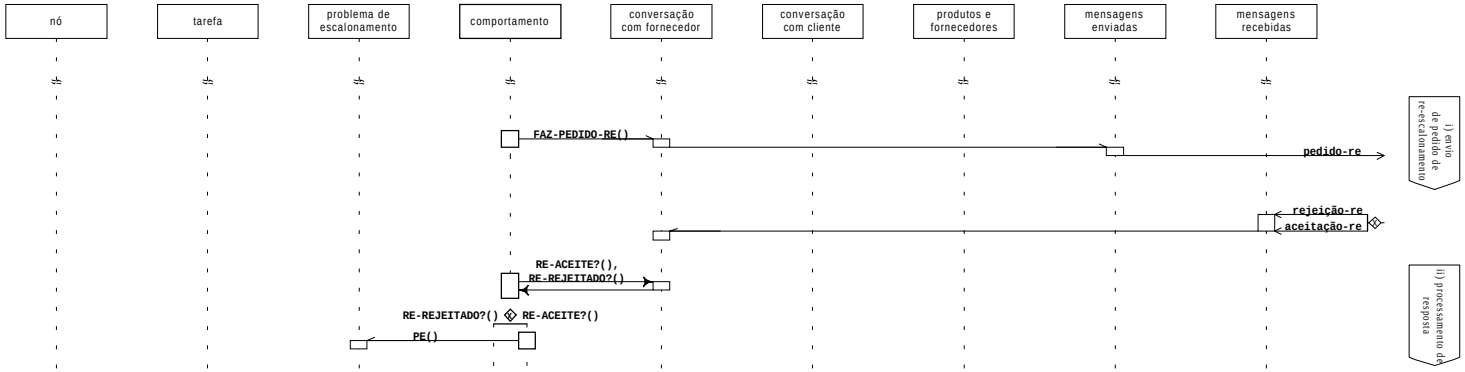
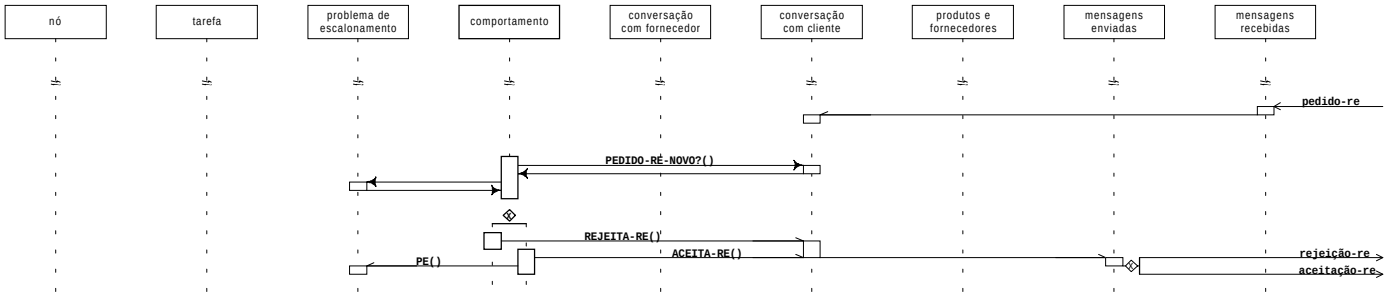


Figura 4-41- Re-escalonamentos pedidos pelo agente (diagramas de sequência para a actividade interna de um agente gestor).

d3) Re-escalonamento pedido por cliente



d4) Re-escalonamento pedido por fornecedor

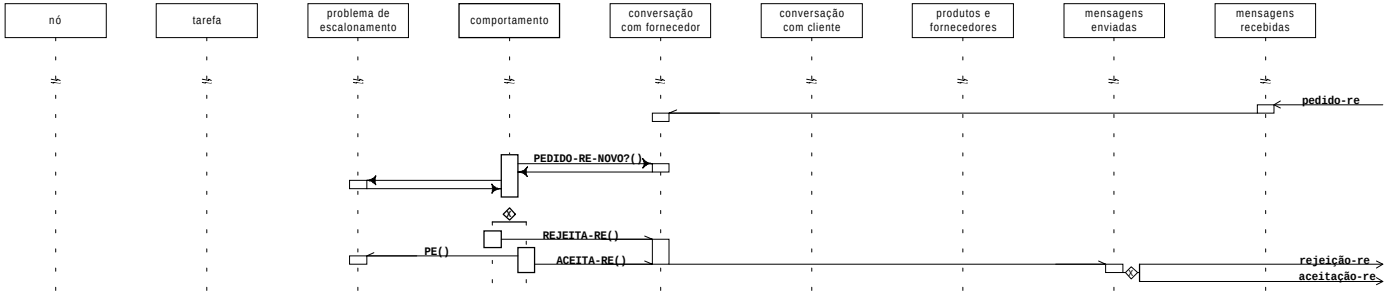


Figura 4-42- Re-escalonamentos pedidos por outros agentes (diagramas de sequência para a actividade interna de um agente gestor).

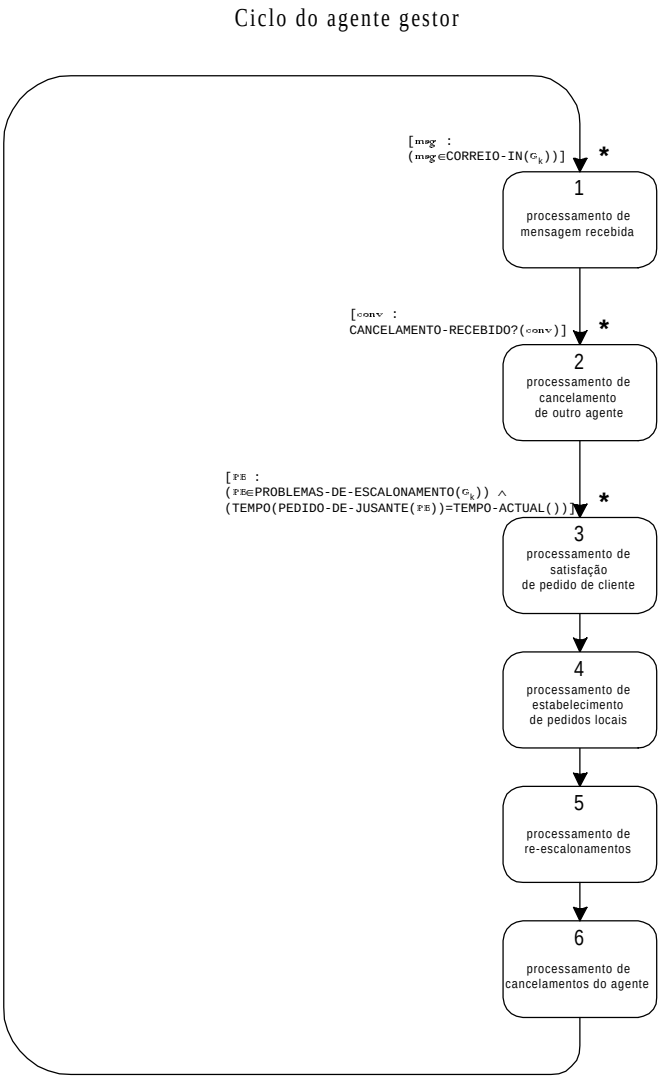


Figura 4-43- O ciclo de operação de agente gestor (diagrama de actividades).

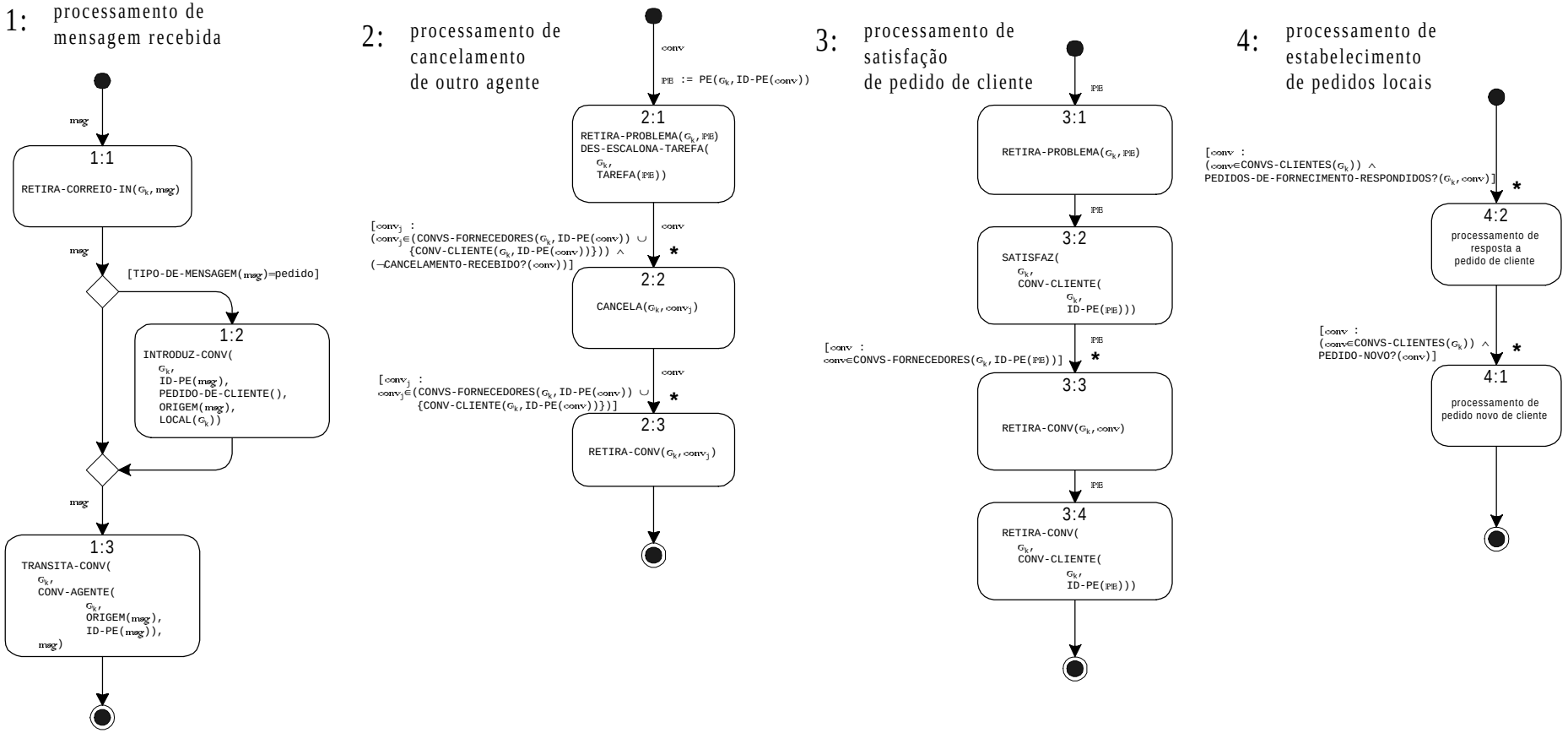


Figura 4-44- Passos do ciclo de operação de agente gestor (diagramas de actividades).

Estas actividades são representadas no diagrama de actividades da Figura 4-43. Os diagramas da Figura 4-44 correspondem a um maior detalhe de alguns dos passos do ciclo de operação do agente gestor.

Num ambiente de escalonamento dinâmico e reactivo, em que o tempo actual e a execução de um escalonamento devem ser tidos em conta, podem ocorrer, na fase 3 de resolução de problemas de escalonamento, situações problemáticas relacionadas com a sincronização das interacções, que são adversas a uma boa coordenação inter-agente na resolução cooperativa de problemas de escalonamento. A análise e o tratamento detalhado dessas situações foram relegados para trabalho futuro. Dois tipos de situações problemáticas são, no entanto, descritas a seguir, sugerindo-se possíveis formas de as evitar no contexto do método de resolução de problemas com três fases, exposto na secção 4.3.6.3.

A primeira situação problemática tem a ver com a interferência entre satisfação e cancelamento de pedidos locais do mesmo problema de escalonamento. Poderá acontecer que, após a satisfação de alguns dos pedidos de fornecimento, um agente gestor receba uma mensagem de cancelamento de um pedido a um fornecedor, ou do pedido do cliente. Então, as restrições de continuidade são violadas[95] e o fluxo de produtos do processo fica interrompido no nó gerido pelo agente. Uma solução possível para evitar que isto aconteça é impor um tempo limite, suficientemente anterior à data limite de cada pedido local, para cancelamento de pedidos locais *por iniciativa de cada agente gestor*; a partir do tempo limite cada agente só pode *propagar* cancelamentos de outros agentes, clientes ou fornecedores (*i.e.*, pode enviar mensagens de cancelamento de pedidos locais associados a um problema de escalonamento se, e só se, recebeu alguma mensagem de cancelamento, de um fornecedor ou do cliente, de um pedido local associado ao mesmo problema de escalonamento). Este tempo limite poderá ser fixo, local e igual para todos os agentes (por exemplo, um agente gestor só poderá tomar a iniciativa de cancelar pedidos locais até um certo tempo predeterminado antes da data limite do pedido) ou estabelecido dinâmica e globalmente (o que pode ser realizado tomando o tempo de determinado evento na rede como referência, por exemplo, estabelecendo-se que, a partir do momento em que ocorre a primeira satisfação de pedido local num processo de rede — o que significa que o processo está já em execução — não é mais possível a um agente, por sua iniciativa, cancelar pedidos locais do processo). Esta última opção, no entanto, teria de envolver ampliação dos protocolos de modo a acomodarem mensagens adicionais, trocadas entre agentes gestores ou, alternativamente, entre o agente supervisor e os agentes gestores; as mensagens indicariam a expiração do prazo para cancelamentos para cada problema de escalonamento.

A segunda situação problemática tem a ver com uma interferência semelhante à primeira mas ocorre entre satisfação e re-escalonamento de pedidos locais do mesmo problema de escalonamento. Na fase 3, para cada agente gestor, tarefas de um mesmo agente mas de problemas diferentes competem pela capacidade do mesmo nó. Para um número mais elevado de problemas de escalonamento introduzidos na rede por unidade de tempo os agentes recorrerão mais intensivamente ao re-escalonamento, de modo a resolver conflitos de capacidade, que serão mais frequentes. Este cenário de dinamismo elevado pode levar a uma instabilidade das soluções estabelecidas para problemas de escalonamento que já ultrapassaram a fase 1. A instabilidade advém do facto de, para se escalonarem sem conflitos os processos de rede de problemas mais recentemente introduzidos poderem ocorrer re-escalonamentos frequentes de processos de rede de problemas anteriormente introduzidos. Neste caso,

[95] Ver o Capítulo 3.

localmente para cada agente, o que se pretende é que as tarefas mais antigas deixem alguma vez de poder ser re-escalonadas "congelando-se", para cada tarefa, o seu escalonamento em alguma data anterior à data limite do pedido local. Para esta estabilização de solução mecanismos semelhantes aos que foram sugeridos para a primeira situação problemática podem ser usados.

Aspectos que também não são considerados no presente trabalho e que foram relegados para trabalho futuro, são os que têm a ver com restrições de capacidade, incluindo a eliminação de conflitos de capacidade, e com preferências de escalonamento individuais dos agentes. As preferências de escalonamento individuais têm a ver com a forma como cada agente *prefere* afectar capacidade disponível às suas tarefas ao longo do tempo (a inclusão de preferências passa pela definição prévia de critérios de preferência para cada agente gestor). Um agente pode preferir certos intervalos de tempo para escalonamento de uma mesma tarefa em detrimento de outros (quando há vários intervalos possíveis em que dispõe de capacidade disponível suficiente) dentro do horizonte temporal do problema de escalonamento respectivo. Por exemplo, o agente pode preferir escalonar uma tarefa o mais cedo possível (iniciando-se o mais próximo possível da data de lançamento mais cedo), ou o mais tarde possível (terminando o mais próximo possível da data limite mais tarde). Outras formas de preferência, puramente temporais, podem ainda fazer-se intervir, por exemplo, preferência por uma determinada folga local a jusante e/ou a montante. Ainda outras formas de preferência são preferir a decomposição de uma tarefa em sub-tarefas escalonadas de determinada forma distribuída no tempo e o emprego de maior ou menor quantidade de capacidade por unidade de tempo numa tarefa de processamento (só para agentes processadores), originando uma menor ou maior duração para a tarefa, respectivamente (estas duas formas de preferência são ambas excluídas pelo pressuposto 2 da secção 4.3.4).

Relativamente aos aspectos da capacidade, foi indicada, na secção 4.3.6.3, uma direcção possível para a evolução deste trabalho, no sentido da concepção de um mecanismo de coordenação adicional para escalonamento baseado na capacidade disponível nos nós que permita, na fase 3 do processo de resolução de problemas de escalonamento de rede, conduzir cada agente para escolhas de soluções de escalonamento boas para o agente e para a rede, de uma forma rápida. A ampliação do modelo exposto no presente trabalho de modo a acomodar os aspectos relativos às restrições de capacidade (e eliminação de conflitos de capacidade) e preferências individuais passará por um maior detalhe e complexidade do modelo (por exemplo, quais os algoritmos, ou heurísticas, ou regras de despacho que, localmente, cada agente emprega — se são diferentes, ou iguais, para todos os agentes, o que é melhor para um melhor desempenho colectivo). A conjugação de preferências de escalonamento com cooperação entre agentes antevê-se complexa, pois leva a considerar uma vasta gama de possibilidades, entre o caso extremo de agentes sem preferências e totalmente cooperativos (como o que se considera no presente trabalho) e o extremo oposto de agentes com preferências conflituosas e não cooperativos.

4.5 Rede de EE

Um *rede de EE* $\mathbb{R}$, é um triplo:

$<\mathbb{RF}, \mathbb{G}_0, \mathcal{AG}>$

Em que $\mathbb{RF}$ representa o nível físico e $\mathbb{G}_0$ e $\mathcal{AG}$ representam o nível virtual de $\mathbb{R}$ e em que:

- $\mathbb{RF}$ é uma rede física;
- $\mathbb{G}_0$ é o agente supervisor de $\mathbb{R}$;
- $\mathcal{AG}=\{\mathbb{G}_1, \ldots, \mathbb{G}_{Nn}\}$ (Nn agentes, com Nn>2) é o conjunto de agentes de rede (*i.e.*, $\mathcal{AG}$ contém todos os agentes de $\mathbb{R}$ à excepção do agente supervisor) de $\mathbb{R}$, cada um dos quais está associado a um único nó da rede física $\mathbb{RF}$, e vice-versa.

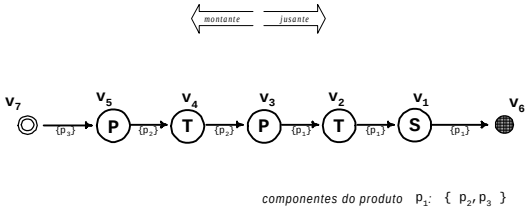
Cada um dos agentes de $\mathcal{AG}$ toma decisões no que respeita ao escalonamento de tarefas para satisfação de pedidos comunicados por outros agentes da rede de EE. O agente supervisor é um agente distinto dos restantes, em que recebe pedidos do exterior que comunica para os agentes de rede e recebe pedidos de agentes de rede dirigidos para o exterior. O subconjunto dos agentes de rede que comunica directamente com o agente supervisor é formado pelo conjunto dos agentes de retalho (agentes de rede que recebem pedidos do agente supervisor) e o conjunto dos agentes de matéria-prima (agentes de rede que enviam pedidos ao agente supervisor). Os restantes agentes de $\mathcal{AG}$, denominados os agentes gestores, estão associados a nós de capacidade, nos quais escalonam tarefas necessárias à satisfação dos pedidos de produtos recebidos de agentes clientes, enviando adicionalmente os pedidos de produtos necessários às tarefas a agentes fornecedores. O conjunto $\mathcal{AG}$ contém, pelo menos, um agente gestor, um agente de retalho e um agente de matéria-prima (logo, é Nn>2).

Assume-se que a comunicação inter-agente numa rede de EE é realizada através de mensagens trocadas entre pares de agentes comunicantes através de *canais de comunicação* (implícitos) bidireccionais. Estes canais de comunicação estabelecem-se entre o agente supervisor e cada agente de retalho, entre o agente supervisor e cada agente de matéria-prima e entre cada par cliente-fornecedor de agentes de rede. Convenciona-se representar o nível virtual de uma rede de EE através de um grafo não direccionado que tem como nós os agentes da rede de EE e como arcos os canais de comunicação entre os agentes, designando a rede assim formada por *rede virtual*.

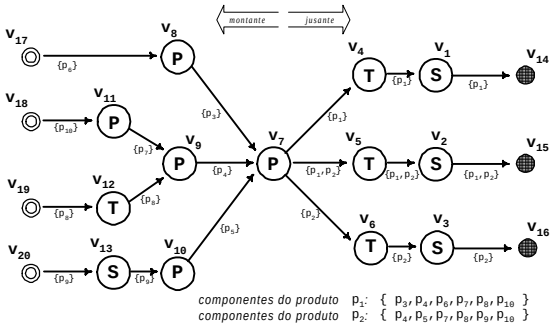
Note-se que, dada uma rede física $\mathbb{RF}$ e dois agentes de rede $\mathbb{G}_x$ e $\mathbb{G}_y$, estes apenas deverão ser agentes comunicantes entre si na rede de EE correspondente a $\mathbb{RF}$, se se verificar que:

$$(\text{NÓ}(\mathbb{G}_x) \in \text{NÓS}(\mathbb{RF})) \wedge (\text{NÓ}(\mathbb{G}_y) \in \text{NÓS}(\mathbb{RF})) \wedge$$
$$((\text{NÓ}(\mathbb{G}_y) \in \text{NÓS-CLIENTES}(\mathbb{RF}, \text{NÓ}(\mathbb{G}_x))) \vee$$
$$(\text{NÓ}(\mathbb{G}_x) \in \text{NÓS-CLIENTES}(\mathbb{RF}, \text{NÓ}(\mathbb{G}_y))))$$

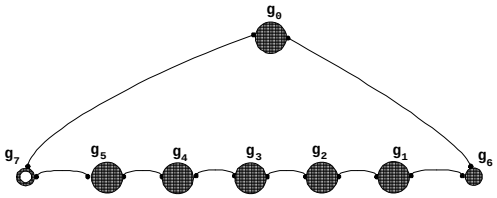
A título de exemplo, representam-se graficamente duas redes de EE na Figura 4-45-a) e na Figura 4-45-b) (com cinco e treze nós de capacidade/agentes gestores, respectivamente). O caso da Figura 4-45-a) exemplifica uma rede linear, ou cadeia. A Figura 4-45-b) representa o caso geral, típico das redes de produção e distribuição, que é o de uma rede convergente-divergente.



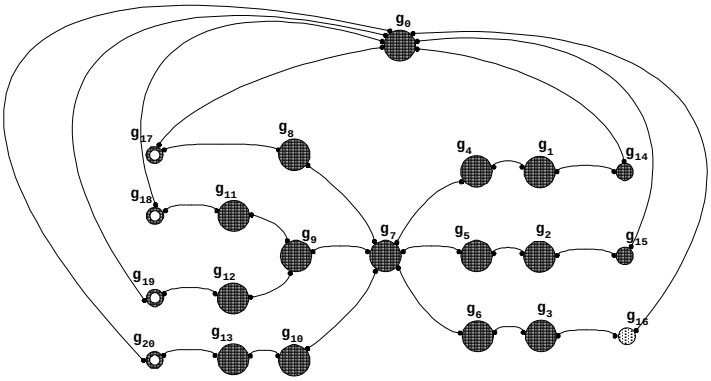
a1) Rede física (nível físico).



b1) Rede física (nível físico).



a2) Rede virtual (nível virtual) para a rede física em a1).



b2) Rede virtual (nível virtual) para a rede física em b1).

a) b)

Figura 4-45- Dois exemplos de redes de EE.

4.5.1 Operações com Redes de EE

Definem-se as seguintes operações selectoras para redes de EE:

REDE(<$\mathbb{RF}$, $\mathbb{G}_0$, $\mathbb{AG}$>) ::= $\mathbb{RF}$.

SUPERVISOR(<$\mathbb{RF}$, $\mathbb{G}_0$, $\mathbb{AG}$>) ::= $\mathbb{G}_0$.

AGENTES-DE-REDE(<$\mathbb{RF}$, $\mathbb{G}_0$, $\mathbb{AG}$ >) ::= $\mathbb{AG}$.

A operação construtora para rede de EE é:

REDE-EE($\mathbb{RF}$, $\mathbb{G}_0$, $\mathbb{AG}$) ::= <$\mathbb{RF}$, $\mathbb{G}_0$, $\mathbb{AG}$>.

A operação AGENTES produz o conjunto de todos os agentes de uma rede de EE dada $\mathbb{R}$:

AGENTES($\mathbb{R}$) ::= { SUPERVISOR($\mathbb{R}$) } $\cup$ AGENTES-DE-REDE($\mathbb{R}$).

A operação AGENTE produz um agente de uma rede de EE $\mathbb{R}$, dado o identificador do agente g_x:

AGENTE($\mathbb{R}$,g_x) ::= $\mathbb{G}_x$: ($\mathbb{G}_x \in$ AGENTES($\mathbb{R}$)) $\wedge$ (LOCAL($\mathbb{G}_x$)=g_x).

As operações definidas a seguir permitem obter os conjuntos de identificadores de agentes de retalho, de agentes de matéria-prima, dos produtos de saída e dos produtos de entrada de uma rede de EE:

LOCAIS-RETALHO($\mathbb{R}$) ::= AGENTES-DE-RETALHO(SUPERVISOR($\mathbb{R}$)).

LOCAIS-MATÉRIA-PRIMA($\mathbb{R}$) ::=
AGENTES-DE-MATÉRIA-PRIMA(SUPERVISOR($\mathbb{R}$)).

PRODUTOS($\mathbb{R}$) ::= PRODUTOS(SUPERVISOR($\mathbb{R}$)).

MATERIAIS($\mathbb{R}$) ::= MATERIAIS(SUPERVISOR($\mathbb{R}$)).

As duas operações que se seguem produzem os conjuntos de pares que associam, respectivamente, cada produto de saída da rede a cada identificador de agente de retalho que é cliente desse produto na rede e cada produto de entrada da rede a cada identificador de agente de matéria-prima fornecedor desse produto na rede:

PRODUTOS-LOCAIS($\mathbb{R}$) ::= PRODUTOS-AGENTES(SUPERVISOR($\mathbb{R}$)).

MATERIAIS-LOCAIS($\mathbb{R}$) ::= MATERIAIS-AGENTES(SUPERVISOR($\mathbb{R}$)).

4.6 Conclusão

Neste capítulo descreveu-se o *nível virtual* do modelo de escalonamento multi-agente. Enquanto que o *nível físico*, descrito no Capítulo 3, diz respeito aos objectos manipulados no escalonamento, o nível virtual diz respeito às entidades decisoras envolvidas na actividade de escalonamento multi-agente — os agentes — e à interacção entre os agentes necessária ao apoio e à coordenação da mesma actividade.

No Capítulo 4 definiram-se classes de objectos como *pedido*, várias classes de *agente* e *rede de EE*. Propôs-se um método de resolução cooperativa de problemas de escalonamento multi-agente que pressupõe o uso de um mecanismo de coordenação para escalonamento temporal, também proposto. Resumidamente, este mecanismo permite, a cada agente, percepcionar localmente as restrições temporais globais rígidas de um problema de escalonamento, identificar problemas de escalonamento temporalmente super-constrangidos e guiar os agentes de uma rede de EE na procura cooperativa de soluções para os problemas de escalonamento. Foi ainda proposto um protocolo de interacção de alto nível adequado à interacção entre os agentes na actividade de escalonamento, no qual se enquadra o método de resolução e o mecanismo de coordenação referidos.

4.7 Lista Resumida de Símbolos Usados no Capítulo 4

Resumem-se na seguinte lista os símbolos usados no presente capítulo.

	Dimensões
$\mathscr{g}$	Conjunto de identificadores de agentes (locais virtuais)
$\mathtt{g}$	Identificador de agente (local virtual, elemento de $\mathscr{g}$)
$\mathtt{g}_0$	Identificador de agente supervisor
	Pedidos
$\mathscr{d}$	Conjunto de pedidos
$\mathbb{d}$	Pedido
	Parâmetros de Escalonamento Temporal
$\mathbb{h}$	Intervalo de horizonte temporal local
$\mathbb{H}$	Intervalo de horizonte temporal
$\mathtt{dd}$	Data limite local
$\mathtt{rd}$	Data de lançamento local
$\mathtt{DD}$	Data limite mais tarde
$\mathtt{RD}$	Data de lançamento mais cedo
$\mathscr{RD}$	Conjunto de datas de lançamento mais cedo
$\mathtt{fij}$	Folga interna a jusante
$\mathtt{fim}$	Folga interna a montante
$\mathtt{FEJ}$	Folga externa a jusante
$\mathtt{FEM}$	Folga externa a montante
$\mathtt{FJ}$	Folga a jusante
$\mathtt{FM}$	Folga a montante
$\mathtt{FT}$	Folga total
$\mathtt{FJM}$	Folga jusante-montante
$\mathtt{FMJ}$	Folga montante-jusante
	Agentes, Rede Virtual, Rede de EE
$\mathbb{G}$	Agente (nó virtual)
$\mathbb{G}_0$	Agente supervisor
$\mathscr{Cl}$	Conjunto de identificadores de agentes clientes de um agente

Fr	Conjunto de identificadores de agentes fornecedores de um agente
Pcl	Conjunto de pares que associam um produto de saída a um identificador de agente cliente
Pfr	Conjunto de pares que associam um produto de entrada a um identificador de agente fornecedor
pg	Par que associa um produto a um identificador de agente
Msg-in	Conjunto de mensagens recebidas, caixa de correio de entrada
Msg-out	Conjunto de mensagens enviadas, caixa de correio de saída
msg	Mensagem
Conteúdo	Conteúdo de mensagem
Convs	Conjunto de conversações
conv	Conversação
Modelo	Modelo de conversação
Estados	Conjunto de estados (de modelo de conversação)
Estado	Estado (de modelo de conversação)
Transições	Conjunto de transições (de modelo de conversação ou de conversação)
transição	Transição (de modelo de conversação ou de conversação)
etiqueta	Etiqueta de transição
PE-Conj	Conjunto de problemas de escalonamento
PE	Problema de escalonamento
Id-PE	Identificador de problema de escalonamento
Comportamento	Conjunto de regras de comportamento
regra	Regra de comportamento
AG	Rede do nível virtual (conjunto dos agentes de rede)
$\mathbb{R}$	Rede de EE
Nk	Número total de agentes gestores numa rede de EE
Nr	Número total de agentes de retalho numa rede de EE
Nm	Número total de agentes de matéria-prima numa rede de EE
$Nf^{k}_{i_r,r}$	Número de pedidos locais de aprovisionamento a fornecedores associados ao pedido local $d^{i,k}_{i_r,r}$, recebido pelo agente gestor g_k de um cliente g_i e originado pelo pedido global do exterior à rede $d^{0,r}_{i_r,r}$
$Nm_{i_r,r}$	Número de pedidos globais da rede ao exterior originados pelo pedido global do exterior à rede $d^{0,r}_{i_r,r}$

5 Exemplos

Neste capítulo mostram-se resultados de aplicação de determinados aspectos do modelo desenvolvido no presente trabalho, com certas restrições.

Através de exemplos, demonstra-se a validade do método de resolução cooperativa de problemas de escalonamento multi-agente, em particular no que respeita ao mecanismo de coordenação para escalonamento temporal e às regras para identificação de conflitos temporais para os agentes (ambos expostos na secção 4.3.6) na perspectiva do sistema multi-agente de uma rede de EE, bem como o uso do protocolo de interacção entre os agentes (exposto na secção 4.4.7).

5.1 Introdução

Este capítulo apresenta resultados de exemplos de aplicação do modelo desenvolvido a um caso particular de rede de EE. Estes resultados foram obtidos através de simulações feitas num programa de folha de cálculo.

Um sistema que realiza computacionalmente as simulações do nível físico e do nível virtual do modelo (ver os capítulos 3 e 4) tem vindo a ser desenvolvido. Este sistema é realizado na linguagem CLOS (Common Lisp Object System, ver [Steele 1990]) e permite construir redes físicas (de acordo com o Capítulo 3), construir uma rede de EE a partir de uma rede física e fazer simulações de escalonamento multi-agente no contexto de uma rede de EE (de acordo com o Capítulo 4). Este sistema está, à data de conclusão deste trabalho, ainda inacabado. Por esta razão se recorreu à simulação de casos restritos usando uma folha de cálculo.

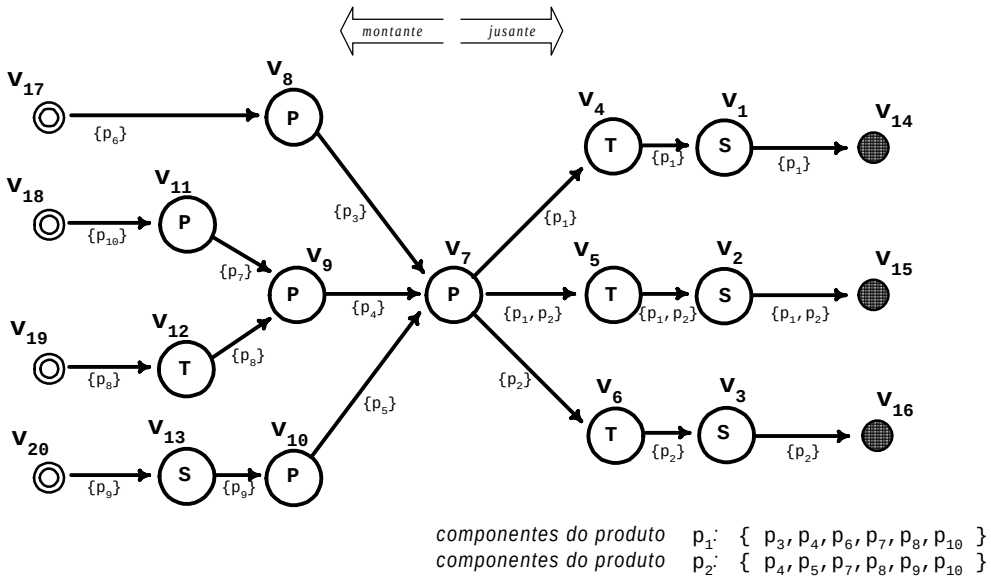
Os exemplos abrangem as fases 1 (estabelecimento de pedidos locais) e 2 (eliminação de conflitos temporais) de resolução cooperativa de problemas de escalonamento e demonstram a utilização e a validade do mecanismo de coordenação para escalonamento temporal e das regras para identificação de conflitos temporais propostos na secção 4.3.6. Assume-se o uso do protocolo de interacção entre os agentes exposto na secção 4.4.7.

As simplificações assumidas nos exemplos são:

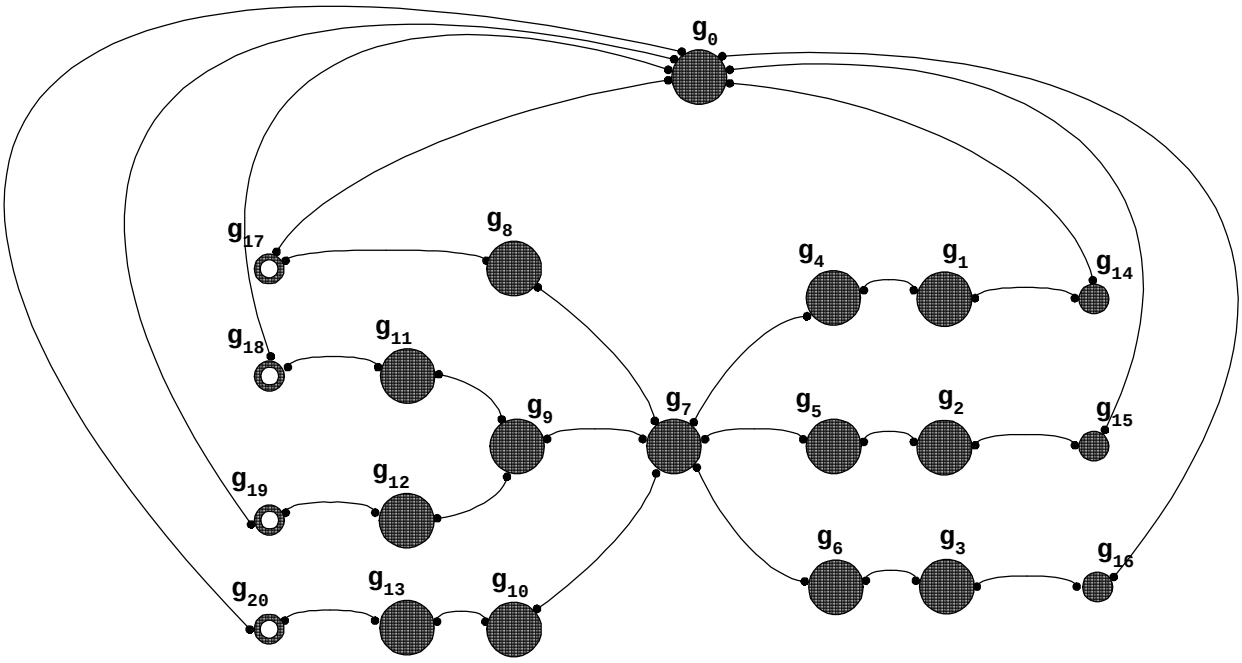
1. As referidas nos pressupostos descritos na secção 4.3.4;
2. As restrições de capacidade[1] não foram consideradas, nem para obedecer a limites de capacidade máxima nos nós nem, para o caso de tarefas de processamento, para determinação da duração das tarefas mediante um valor escolhido de capacidade a dedicar-lhes em cada unidade de tempo.[2] A duração das tarefas de cada agente é fixa;
3. As quantidades de produto dos pedidos entre agentes são irrelevantes, dado que não se consideram restrições de capacidade. O escalonamento multi-agente é executado atendendo apenas a restrições temporais de precedência entre tarefas do mesmo processo de rede e restrições temporais de datas limite (data limite e data de lançamento);
4. Apenas se tomam em consideração conflitos temporais relativos globais e conflitos temporais relativos locais externos. Assume-se que os agentes mantêm sempre folgas temporais internas não negativas e que, portanto, não existem conflitos temporais relativos internos. Para casos de conflito temporal relativo global total o sistema multi-agente desiste do problema; para os restantes casos de conflitos considerados, assume-se que os agentes recorrem ao *tratamento minimal* apropriado conforme descrito na secção 4.3.6.

A rede de EE usada nos exemplos é representada na Figura 5-1-a) (nível físico) e na Figura 5-1-b) (nível virtual).

[1] Ver o Capítulo 3.
[2] O valor de `cval`, ver o Capítulo 3.



a) Rede física (nível físico).

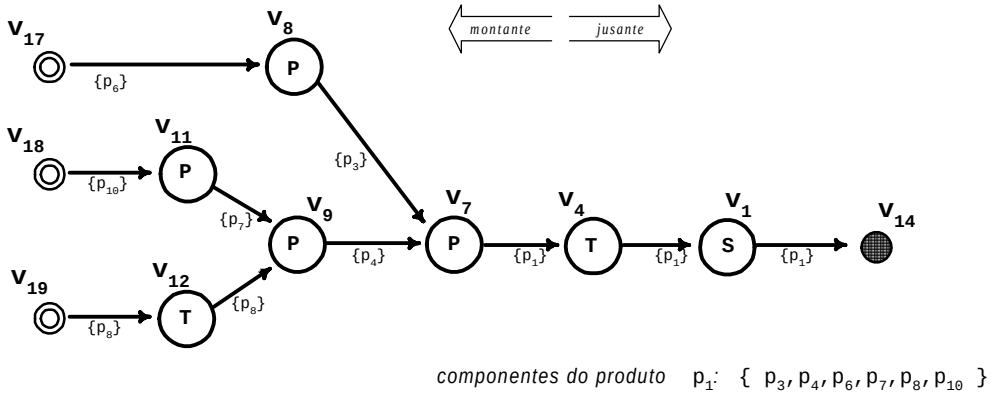


b) Rede virtual (nível virtual).

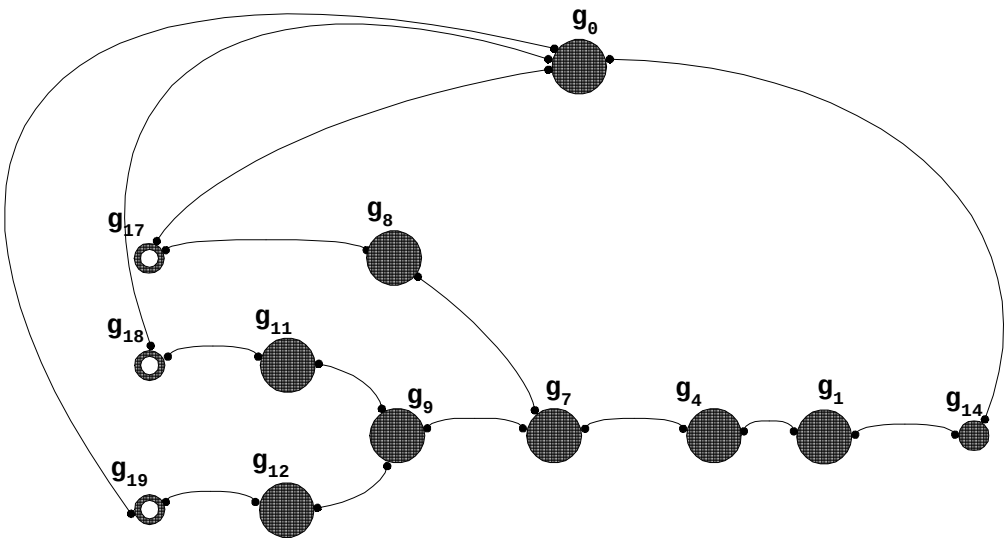
Figura 5-1- Rede de EE usada nos exemplos.

Esta rede de EE é igual à que foi usada como exemplo na secção 4.3.5 (ver a Figura 4-2) e na secção 4.5 (ver a Figura 4-45-b)).

Adicionalmente, nos exemplos, os pedidos globais do exterior à rede são pedidos de produto p_1 destinados ao agente de retalho g_{14}, pelo que apenas uma fracção da rede representada na Figura 5-1 é utilizada. Esta fracção da rede de EE utilizada nos exemplos é representada na Figura 5-2-a) (nível físico) e na Figura 5-2-b) (nível virtual).



a) Rede física (nível físico).



b) Rede virtual (nível virtual).

Figura 5-2- Rede de EE utilizada nos exemplos (esta rede é uma parte da rede de EE da Figura 5-1).

Algumas indicações adicionais são dadas nas duas secções seguintes com vista à compreensão das simulações.

5.1.1 Estabelecimento de Pedidos Locais

A resolução de um problema de escalonamento multi-agente numa rede de EE inicia-se com a fase 1 (estabelecimento de pedidos locais). Para preparar uma simulação desta fase, certos parâmetros são previamente estabelecidos. Estes parâmetros são, para cada agente:

1. A duração das tarefas do agente. Para agentes com nós acumuladores, a duração é unitária.[3]
2. A folga interna a jusante `fij` e as folgas internas a montante $\texttt{fij}^{j}$ com cada fornecedor g_j (valores sempre não negativos).

Estes dados são introduzidos como parâmetros dos agentes gestores. Veja-se o exemplo de área de entrada de dados na Figura 5-3. Os valores prefixados para as folgas `fij` e `fim` em 2 podem ser entendidos como uma forma rudimentar de preferências de escalonamento. Se a fase 1 termina com sucesso mas subsistem conflitos temporais, o re-escalonamento efectuado posteriormente pelos agentes na fase 2 (eliminação de conflitos temporais) inclui reajustamentos destas folgas, de modo a eliminar os conflitos temporais.

Para cada conjunto de valores prefixados das durações e das folgas internas, realizam-se então várias simulações para diferentes problemas de escalonamento multi-agente. Para cada problema de escalonamento os dados de entrada são os valores dos seguintes parâmetros:

a) A data limite do pedido global do exterior à rede, $\texttt{dd}_{i_r,r}$ ($\texttt{dd}_{i_r,r}$=TEMPO($\text{d}^{0,r}_{i_r,r}$), em que $\text{d}^{0,r}_{i_r,r}$ é o pedido global do exterior à rede);

b) A data limite global, $\texttt{DD}_{i_r,r}$;

c) A data de lançamento global, $\texttt{RD}_{i_r,r}$.

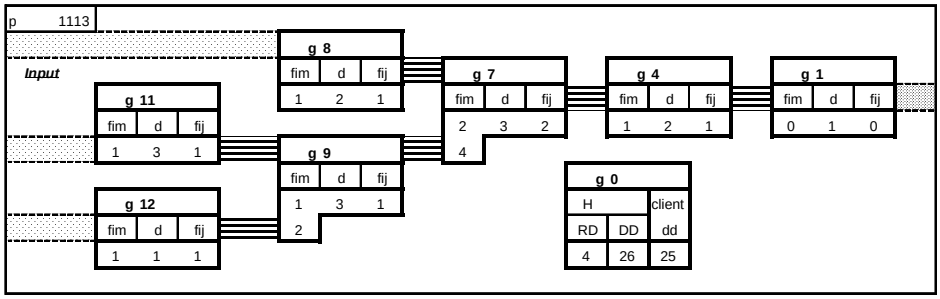


Figura 5-3- Exemplo de área de entrada de dados de simulação de um problema de escalonamento multi-agente para a rede de EE na Figura 5-2. São indicados os dados globais do problema, associados ao agente supervisor g_0 e os dados inicialmente fixados para cada agente gestor g_1, g_4, g_7, g_8, g_9, g_{11} e g_{12} (d significa duração de tarefa; zonas riscadas na horizontal entre cada área de agente indicam as relações de cliente-fornecedor).

Os valores destes parâmetros são introduzidos para o agente supervisor. Veja-se o exemplo de área de entrada de dados na Figura 5-3.

Para um problema de escalonamento não temporalmente super-constrangido, todos os agentes percepcionam localmente folgas totais (FT) não negativas e a fase 1 termina com sucesso, *i.e.*,

[3] Com duração unitária (e fixa) para as tarefas de acumulação, as folgas `fij` e `fim` serão sempre, consequentemente, nulas, de acordo com o definido na secção 4.3.5.

todos os pedidos locais decorrentes dum pedido global do exterior à rede são aceites. Num caso destes, os agentes representados na Figura 5-2-b), trocam entre eles, até ao final da fase 1, os tipos de mensagens representados no diagrama de colaboração na Figura 5-4. Para um problema de escalonamento temporalmente super-constrangido, existe pelo menos um agente que detecta folga total negativa e rejeita o pedido do cliente. Num caso destes, a fase 1 termina sem sucesso. O diagrama de colaboração da Figura 5-5 representa os tipos de mensagens trocadas num exemplo de insucesso da fase 1 (no caso, com rejeição de pedido de cliente por parte do agente g_{11}). Os pedidos e as folgas propagados na rede, para o caso de uma fase 1 com sucesso, são ilustrados na Figura 5-6-a) (processamento de pedido novo de cliente) e Figura 5-6-b) (resposta a pedido de cliente). Na Figura 5-6-c) representa-se o processo de rede resultante desta fase.

A Figura 5-3 representa um exemplo de área de entrada de dados da folha de cálculo para uma simulação de um problema de escalonamento para a rede na Figura 5-2-b). Para cada agente gestor g_1, g_4, g_7, g_8, g_9, g_{11} e g_{12}, são mostradas as folgas internas e as durações das tarefas (os valores assinalados por d, na figura). Zonas riscadas entre as áreas de dados de cada agente gestor indicam as relações de cliente fornecedor. Adicionalmente, são representados os dados de entrada do problema de escalonamento, associados ao agente supervisor g_0. Estes últimos são a data de lançamento global RD, a data limite global DD e a data limite do pedido global do exterior à rede dd.

A Figura 5-7 representa um exemplo dos dados de saída de uma simulação, relativos à fase de 1, para os dados na área de entrada de dados da Figura 5-3. As mensagens trocadas entre os agentes e os dados dos problemas de escalonamento resultantes para cada agente gestor são representados em quadros etiquetados por Quadro 1, na Figura 5-7-a), e Quadro 2, na Figura 5-7-b), respectivamente. Destacam-se, nos títulos deste último quadro, os parâmetros prefixados (d, fim e fij) para distingui-los dos calculados pelos agentes na fase 1.

O escalonamento resultante é mostrado no gráfico de Gantt etiquetado por *network schedule 1*, na Figura 5-7-c). Este gráfico de Gantt representa, para cada agente gestor, o intervalo de escalonamento da tarefa (intervalo $<t_s, t_e>$, da tarefa do agente) a cor azul, cada intervalo de horizonte temporal local com cada fornecedor (os intervalos $\mathbb{h}^j=<rd^j, dd>$, do agente com cada fornecedor g_j) a cor verde e os intervalos de horizonte temporal com cada fornecedor (os intervalos $\mathbb{H}^j=<RD^j, DD>$, do agente com cada fornecedor g_j) a cor vermelha. Associados ao agente supervisor (agente g_0), o gráfico representa o intervalo de horizonte temporal global (o intervalo $\mathbb{H}=<RD, DD>$, para toda a rede) a cor vermelha e ainda intervalos $<rd^8, dd>$, $<rd^{11}, dd>$ e $<rd^{12}, dd>$ a cor verde. Para estes últimos intervalos, dd é a data limite do pedido global do exterior à rede e rd^8, rd^{11} e rd^{12} são as datas de lançamento local dos agentes g_8, g_{11} e g_{12} (iguais às datas limite dos pedidos globais da rede ao exterior dos mesmos agentes). Datas limite locais e datas limite mais tarde são representadas pelos valores temporais dos quadrados a cheio dos extremos finais dos intervalos $\mathbb{h}$ e $\mathbb{H}$, respectivamente; datas de lançamento locais e datas de lançamento mais cedo são representadas pelos valores temporais dos losangos a cheio dos extremos iniciais dos intervalos $\mathbb{h}$ e $\mathbb{H}$, respectivamente.

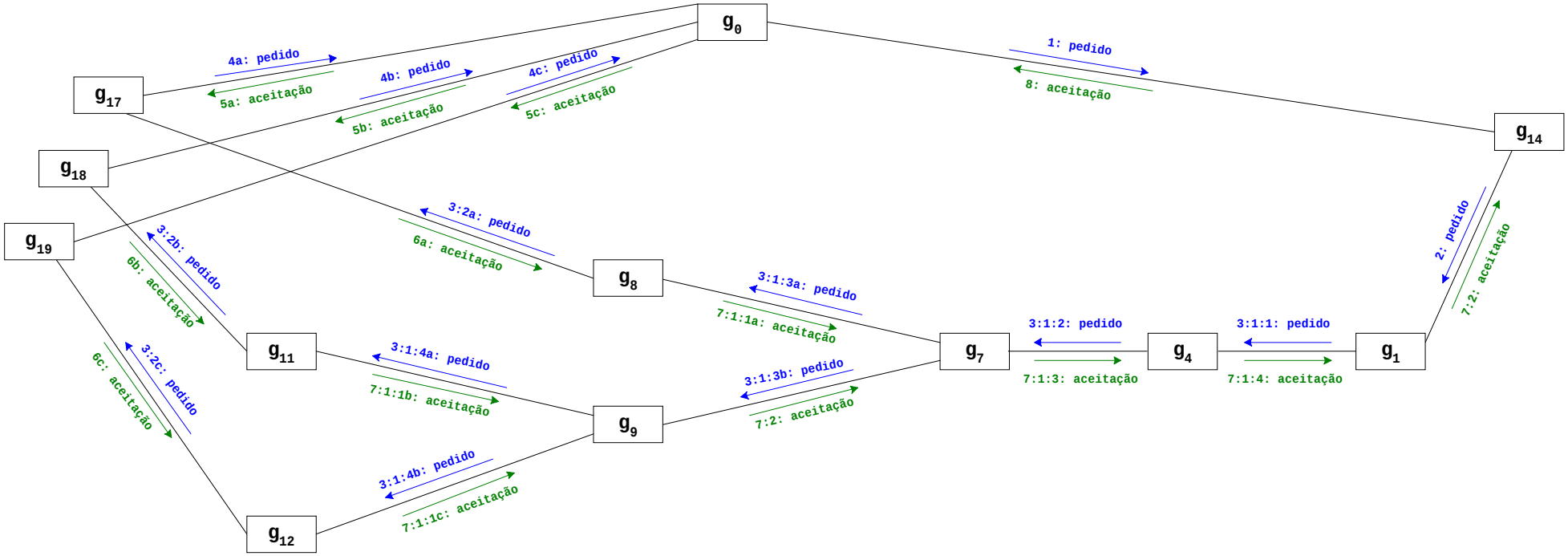


Figura 5-4- Tipos de mensagens trocadas entre os agentes num caso de sucesso da fase 1 (estabelecimento de pedidos locais), na resolução de um problema de escalonamento multi-agente para a rede de EE da Figura 5-2.

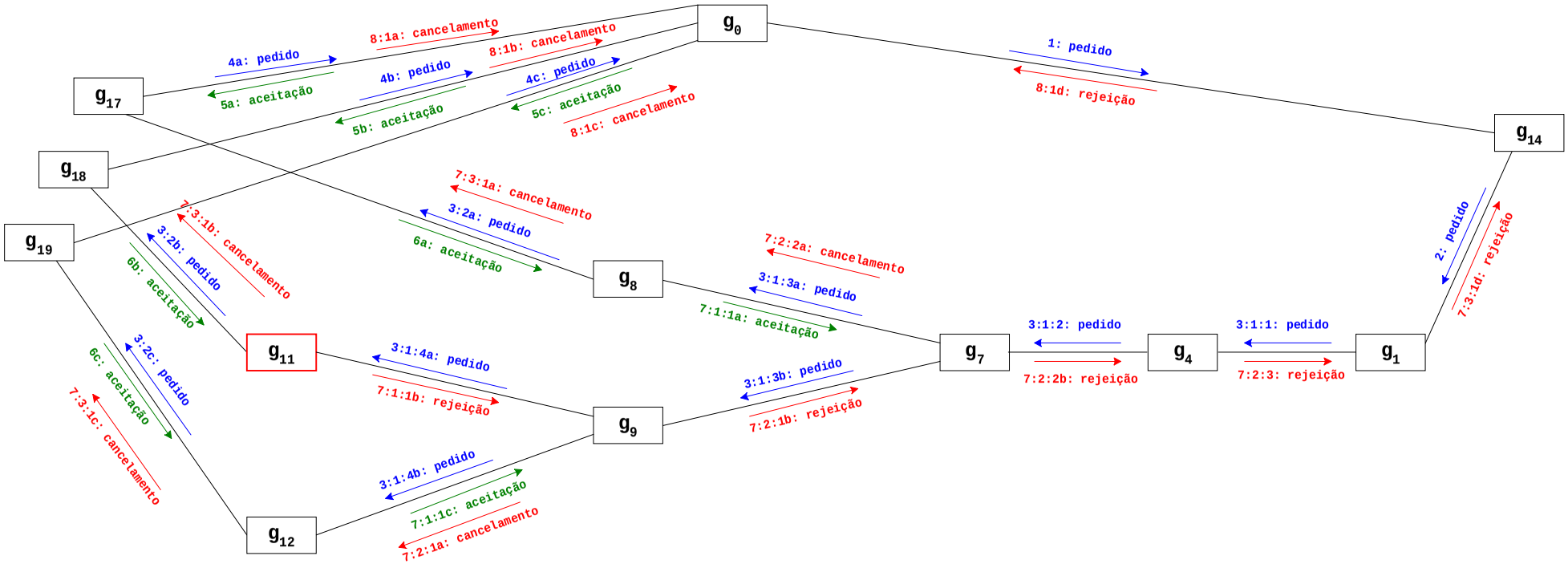
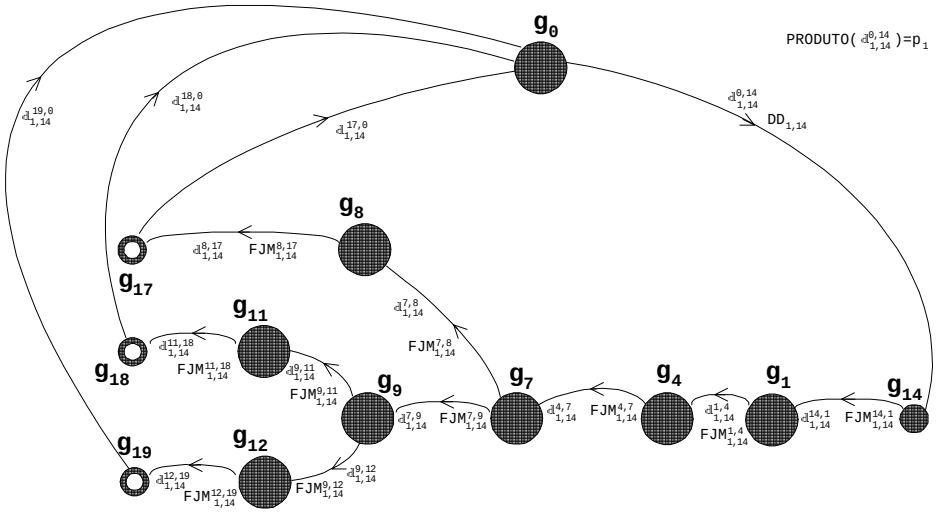
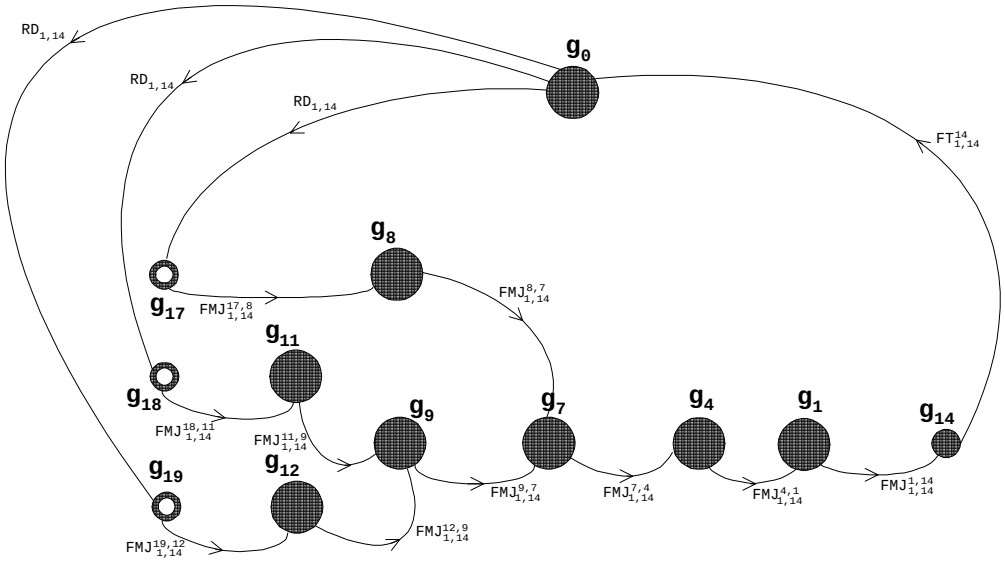


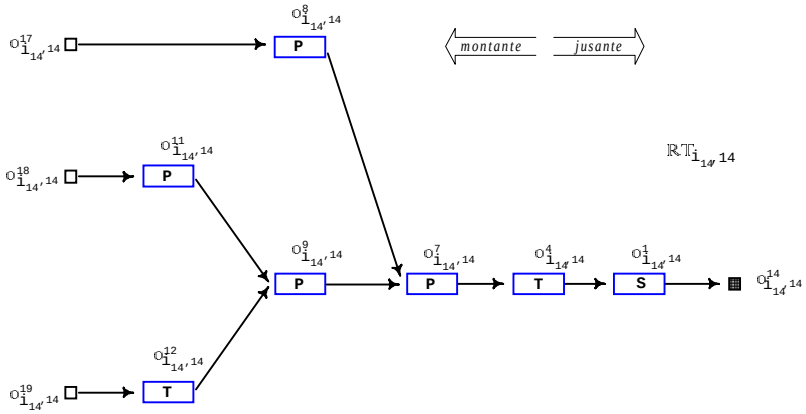
Figura 5-5- Tipos de mensagens trocadas entre os agentes num caso de insucesso da fase 1 (estabelecimento de pedidos locais), na resolução de um problema de escalonamento multi-agente para a rede de EE da Figura 5-2. No caso representado, existiu rejeição de pedido de cliente por parte do agente g_{11}.



a) Processamento de pedido novo de cliente: pedidos (𝕕) e folgas jusante-montante (FJM) são propagados para montante na rede (através das mensagens do tipo pedido).



b) Resposta a pedido de cliente: folgas montante-jusante (FMJ) são propagadas para jusante na rede (através das mensagens do tipo aceitação).



c) Processo de rede.

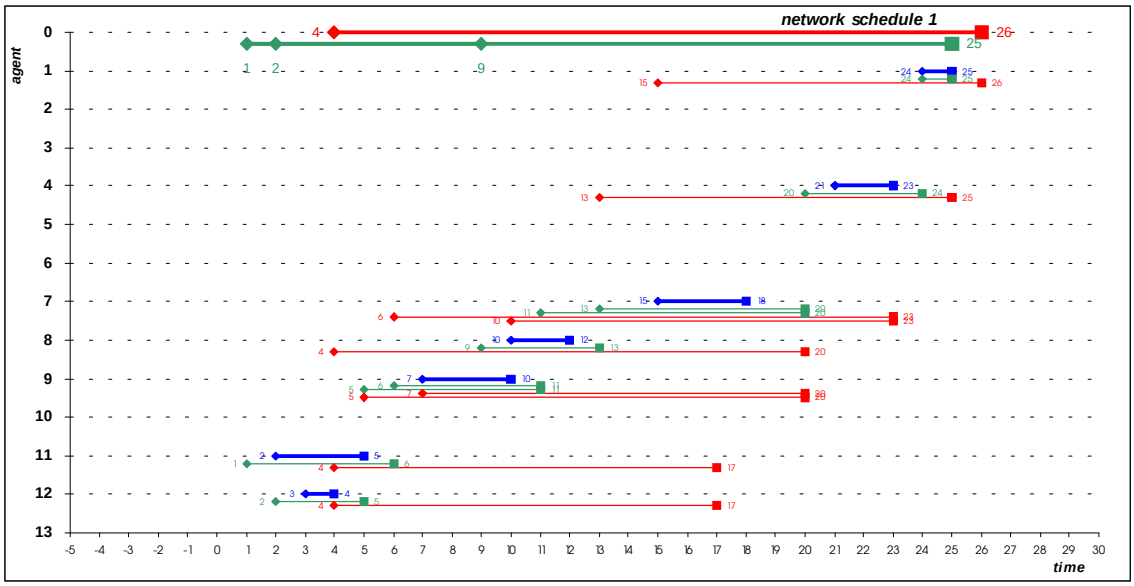
Figura 5-6- Processamento de pedido novo de cliente, em a), e resposta a pedido de cliente, em b), numa fase 1 (estabelecimento de pedidos locais) com sucesso, na resolução de um problema de escalonamento multi-agente para a rede de EE da Figura 5-2. Em c) representa-se o processo de rede resultante.

Quadro 1	Estabelecimento de pedidos locais				p
dd = 25	Mensagens trocadas entre agentes				1113
H = <4,26>					
tipo	origem	destino	conteúdo		
pedido-in	exterior		dd = 25	<4,26>	
pedido	g0	g14	dd = 25	DD =	26
pedido	g14	g1	dd = 25	FJM / FEJ =	1
pedido	g1	g4	dd = 24	FJM / FEJ =	1
pedido	g4	g7	dd = 20	FJM / FEJ =	3
pedido	g7	g8	dd = 13	FJM / FEJ =	7
pedido	g7	g9	dd = 11	FJM / FEJ =	9
pedido	g9	g11	dd = 6	FJM / FEJ =	11
pedido	g9	g12	dd = 5	FJM / FEJ =	12
pedido	g8	g17	dd = 9	FJM / FEJ =	9
pedido	g11	g18	dd = 1	FJM / FEJ =	13
pedido	g12	g19	dd = 2	FJM / FEJ =	14
pedido	g17	g0	dd = 9		
pedido	g18	g0	dd = 1		
pedido	g19	g0	dd = 2		
pedido-out	g0	exterior	dd = 9		
pedido-out	g0	exterior	dd = 1		
pedido-out	g0	exterior	dd = 2		
aceitação	g0	g17	RD = 4		
aceitação	g0	g18	RD = 4		
aceitação	g0	g19	RD = 4		
aceitação	g17	g8		FMJ / FEM =	5
aceitação	g18	g11		FMJ / FEM =	-3
aceitação	g19	g12		FMJ / FEM =	-2
aceitação	g11	g9		FMJ / FEM =	-1
aceitação	g12	g9		FMJ / FEM =	0
aceitação	g8	g7		FMJ / FEM =	7
aceitação	g9	g7		FMJ / FEM =	1
aceitação	g7	g4		FMJ / FEM =	7
aceitação	g4	g1		FMJ / FEM =	9
aceitação	g1	g14		FMJ / FEM =	9
aceitação	g14	g0		FT =	10

a) Mensagens trocadas entre os agentes.

Quadro 2	Estabelecimento de pedidos locais													p
dd = 25	Problema de escalonamento													1113
H = <4,26>	de agente gestor							Folgas						
Agente gestor	fornece-dores	cliente	Datas				Tarefa							
			RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	4	9	13	20	10	2	12	6	5	1	1	7	8	14
g11	g18	g9	4	1	6	17	2	3	5	-2	-3	1	1	11	12	10
g12	g19	g9	4	2	5	17	3	1	4	-1	-2	1	1	12	13	12
g9	g11 g12	g7	7 5	6 5	11	20	7	3	10	0 2	-1 0	1 2	1	9	10	10
g7	g8 g9	g4	6 10	13 11	20	23	15	3	18	9 5	7 1	2 4	2	3	5	10
g4	g7	g1	13	20	24	25	21	2	23	8	7	1	1	1	2	10
g1	g4	g14	15	24	25	26	24	1	25	9	9	0	0	1	1	10

b) Problemas de escalonamento dos agentes gestores.



c) Escalonamento resultante.

Figura 5-7- Exemplo dos dados de saída resultantes de uma simulação, relativos à fase 1 de resolução de um problema de escalonamento multi-agente para a rede de EE da Figura 5-2.

No caso usado como exemplo para a Figura 5-7, pode ver-se que a fase 1 terminou com sucesso (não houve nenhuma rejeição), devido ao facto de todos os agentes gestores envolvidos percepcionam localmente uma folga total não negativa para o problema de escalonamento (no caso, a folga total da rede é de 10 unidades de tempo).

5.1.2 Re-Escalonamento

Uma vez que um problema de escalonamento seja reconhecido pelos agentes envolvidos como não sendo temporalmente super-constrangido, a fase 1 termina com sucesso e obtém-se uma solução inicial. Segue-se, então, a fase 2. O objectivo da fase 2 é eliminar todos os conflitos temporais que existam na solução inicial, estabelecida na fase 1. Após a conclusão da fase 2, o escalonamento resultante não exibirá conflitos temporais.

Os conflitos temporais que subsistem após a conclusão de uma fase 1 com sucesso são detectáveis, nas simulações, no Quadro 2, pela presença de valores negativos de folgas `FJ`, `FM`, `FEJ` ou `FEM` para algum problema de escalonamento de agente gestor. Na fase 2, qualquer situação de conflito temporal se enquadra num dos quatro casos de conflito temporal identificáveis pelas quatro regras descritas na secção 4.3.6.2 (regras 1, 2, 3 e 4). Para reparação da solução, os conflitos temporais são eliminados através da aplicação do tratamento minimal apropriado a cada caso identificado por cada agente gestor, conforme descrito na secção 4.3.6.5.

Os conflitos temporais globais (identificados pelas regras 1, se a jusante e 2, se a montante) são assinalados, no gráfico de Gantt do escalonamento *network schedule 1*, pelo facto de um intervalo de tarefa (a azul) não estar contido no intervalo de horizonte temporal (a vermelho) mais restrito (quando existe mais que um), para algum agente gestor. Por exemplo, pode ver-se na Figura 5-7-c) que ambos os agentes g_{11} e g_{12} têm, cada um, um conflito temporal global a montante (que cada um deles identificaria pela regra 2). No caso de g_{11}, o conflito temporal é causado pelo facto de o tempo de início da tarefa do agente estar fora do intervalo de horizonte temporal de g_{11}, com avanço temporal relativamente ao tempo de início deste intervalo (também, a data do pedido ao fornecedor g_{18} está fora do intervalo de horizonte temporal de g_{11} com o fornecedor g_{18}, com avanço temporal relativamente ao tempo de início deste intervalo); no caso do agente g_{12}, a situação é semelhante.

Os conflitos temporais locais externos (identificados pelas regras 3, se a jusante, e 4, se a montante) são assinalados, no gráfico de Gantt *network schedule 1*, pelo facto de, pelo menos um intervalo de horizonte temporal local com um fornecedor (a verde) não estar contido no intervalo de horizonte temporal com o mesmo fornecedor (a vermelho), para algum agente gestor. Por exemplo, pode ver-se na Figura 5-7-c) que o agente g_9 tem um conflito temporal local externo a montante (que o agente identificaria pela regra 4). O conflito temporal é causado pelo facto de a data do pedido ao fornecedor g_{11} estar fora do intervalo de horizonte temporal de g_9 com o fornecedor g_{11}, com avanço temporal relativamente ao tempo de início deste intervalo.

Para a simulação da actividade de re-escalonamento que ocorre na fase 2, serão mostrados casos em que a fase 1 termina com sucesso, mas em que há conflitos temporais.

A Figura 5-8 representa um exemplo dos dados de saída de uma simulação, relativos ao re-escalonamento que ocorre na fase 2, para os dados de saída da fase 1 da Figura 5-7. As mensagens trocadas entre os agentes e os dados dos problemas de escalonamento para cada agente gestor que se obtêm com o re-escalonamento efectuado são representados em quadros etiquetados por Quadro 3, na Figura 5-8-a), e Quadro 4, na Figura 5-8-b), respectivamente. Destacam-se, nos títulos deste último quadro, os parâmetros prefixados (`d`, `fim` e `fij`) e os que foram calculados pelos agentes na fase 1 e cujo valor permanecerá fixo (`RD` e `DD`), para distingui-los dos restantes.

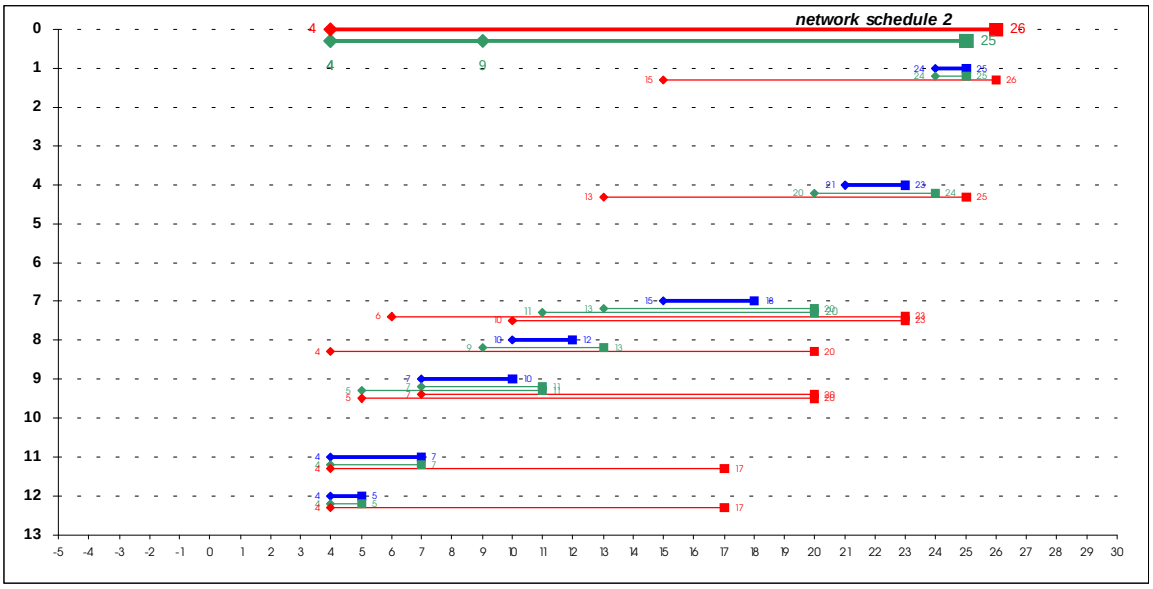
O escalonamento resultante da fase 2 é mostrado no gráfico de Gantt etiquetado por *network schedule 2*, na Figura 5-8-c). Como se pode ver tanto pelo Quadro 4, na Figura 5-8-b), como pelo gráfico de Gantt *network schedule 2*, todos os conflitos temporais foram resolvidos.

Quadro 3	Re-escalonamento			p
dd = 25 H = <4,26>	Mensagens trocadas entre agentes			1113
tipo	origem	destino	conteúdo	
pedido-re	g9	g11	rd = 7	FEM = -1
pedido-re	g11	g9	dd = 7	FM = -2
pedido-re	g11	g18	rd = 4	FM = -2
pedido-re	g12	g19	rd = 4	FM = -1
re-escalonamento	g18	g0	dd = 4	
re-escalonamento	g19	g0	dd = 4	

a) Mensagens trocadas entre os agentes.

Quadro 4	Re-escalonamento													p
dd = 25 H = <4,26>	Problema de escalonamento de agente gestor							Folgas						1113
Agente gestor	fornece-dores	cliente	Datas				Tarefa							
			RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	4	9	13	20	10	2	12	6	5	1	1	7	8	14
g11	g18	g9	4	4	7	17	4	3	7	0	0	1	1	10	10	10
g12	g19	g9	4	4	5	17	4	1	5	0	0	1	1	12	12	12
g9	g11 g12	g7	7 5	7 5	11	20	7	3	10	0 2	0 0	1 2	1	9	10	10
g7	g8 g9	g4	6 10	13 11	20	23	15	3	18	9 5	7 1	1 4	1	3	5	10
g4	g7	g1	13	20	24	25	21	2	23	8	7	1	1	1	2	10
g1	g4	g14	15	24	25	26	24	1	25	9	9	1	1	1	1	10

b) Problemas de escalonamento dos agentes gestores.



c) Escalonamento resultante.

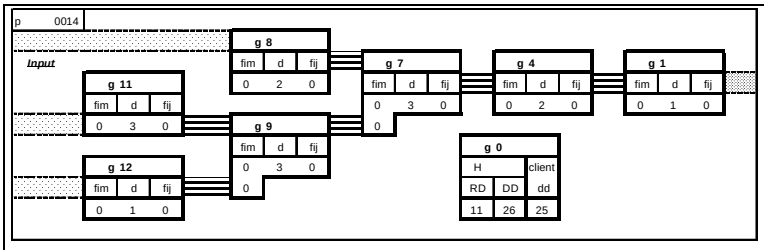
Figura 5-8- Exemplo dos dados de saída resultantes de uma simulação, relativos à fase 2 de resolução de um problema de escalonamento multi-agente para a rede de EE da Figura 5-2.

5.2 Exemplos

Os vários exemplos são organizados em casos sem re-escalonamento e casos com re-escalonamento. Os casos sem re-escalonamento são aqueles em que a fase 1, ou termina com sucesso e com uma solução possível (sem conflitos temporais) para o problema de escalonamento, ou termina sem sucesso pelo facto de o problema de escalonamento ser temporalmente super-constrangido. Os casos com re-escalonamento são aqueles em que a fase 1 termina com sucesso mas com uma solução contendo conflitos temporais.

5.2.1 Casos sem Re-Escalonamento

Nos exemplos apresentados nesta secção o re-escalonamento não é necessário. Podem ser casos em que a fase 1 termina com sucesso, ou sem sucesso. Mesmo nos casos em que não há sucesso é mostrado o gráfico de Gantt (*network schedule 1*) do escalonamento multi-agente não válido resultante.



Quadro 1	Estabelecimento de pedidos locais			p
dd = 25	Mensagens trocadas entre agentes			14
H = <11,26>				
tipo	origem	destino	conteúdo	
pedido-in	exterior		dd = 25	<11,26>
pedido	g0	g14	dd = 25	DD = 26
pedido	g14	g1	dd = 25	FJM / FEJ = 1
pedido	g1	g4	dd = 24	FJM / FEJ = 1
pedido	g4	g7	dd = 22	FJM / FEJ = 1
pedido	g7	g8	dd = 19	FJM / FEJ = 1
pedido	g7	g9	dd = 19	FJM / FEJ = 1
pedido	g9	g11	dd = 16	FJM / FEJ = 1
pedido	g9	g12	dd = 16	FJM / FEJ = 1
pedido	g8	g17	dd = 17	FJM / FEJ = 1
pedido	g11	g18	dd = 13	FJM / FEJ = 1
pedido	g12	g19	dd = 15	FJM / FEJ = 1
pedido	g17	g0	dd = 17	
pedido	g18	g0	dd = 13	
pedido	g19	g0	dd = 15	
pedido-out	g0	exterior	dd = 17	
pedido-out	g0	exterior	dd = 13	
pedido-out	g0	exterior	dd = 15	
aceitação	g0	g17	RD = 11	
aceitação	g0	g18	RD = 11	
aceitação	g0	g19	RD = 11	
aceitação	g17	g8		FMJ / FEM = 6
aceitação	g18	g11		FMJ / FEM = 2
aceitação	g19	g12		FMJ / FEM = 4
aceitação	g11	g9		FMJ / FEM = 2
aceitação	g12	g9		FMJ / FEM = 4
aceitação	g8	g7		FMJ / FEM = 6
aceitação	g9	g7		FMJ / FEM = 2
aceitação	g7	g4		FMJ / FEM = 2
aceitação	g4	g1		FMJ / FEM = 2
aceitação	g1	g14		FMJ / FEM = 2
aceitação	g14	g0		FT = 3

Quadro 2	Estabelecimento de pedidos locais																p
dd = 25	Problema de escalonamento																14
H = <11,26>	de agente gestor									Folgas							
Agente	fornece-	cliente	Datas				Tarefa										
gestor	dores		RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT	
g8	g17	g7	11	17	19	20	17	2	19	6	6	0	0	1	1	7	
g11	g18	g9	11	13	16	17	13	3	16	2	2	0	0	1	1	3	
g12	g19	g9	11	15	16	17	15	1	16	4	4	0	0	1	1	5	
g9	g11	g7	14	16	19	20	16	3	19	2	2	0	0	1	1	3	
	g12		12	16						4	4	0					
g7	g8	g4	13	19	22	23	19	3	22	6	6	0	0	1	1	3	
	g9		17	19						2	2	0					
g4	g7	g1	20	22	24	25	22	2	24	2	2	0	0	1	1	3	
g1	g4	g14	22	24	25	26	24	1	25	2	2	0	0	1	1	3	

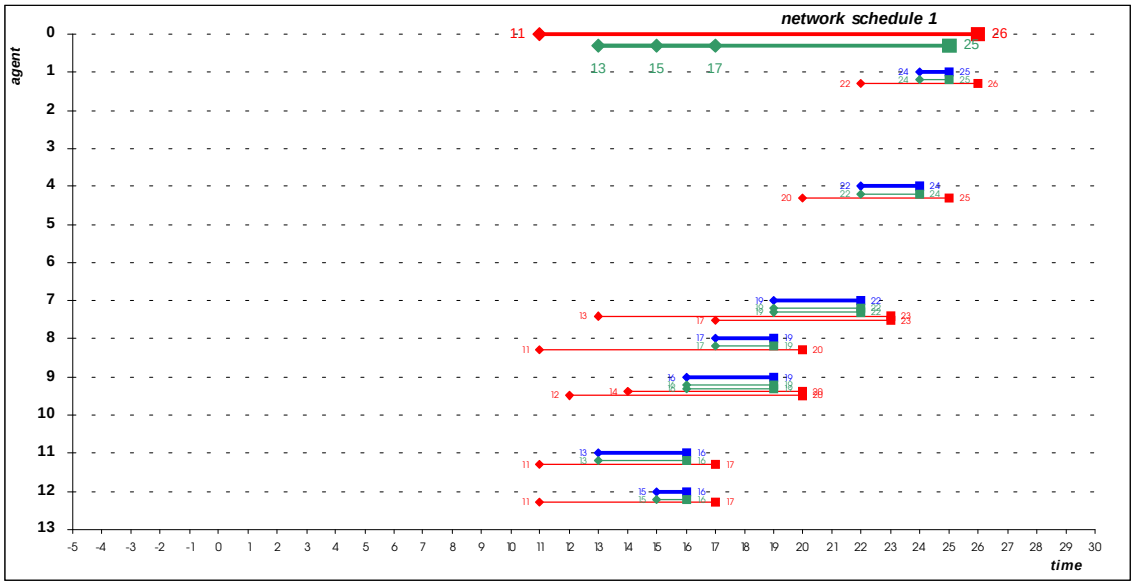
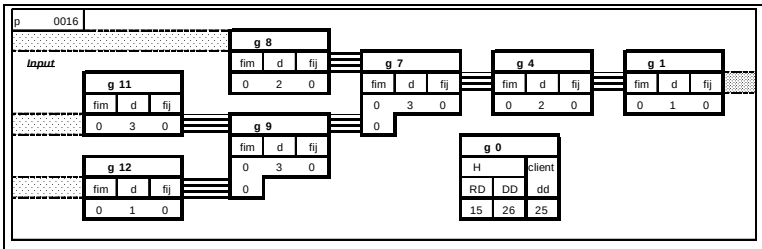


Figura 5-9- Exemplo 0014: resultados da fase 1.



Quadro 1	Estabelecimento de pedidos locais				p
dd = 25	Mensagens trocadas entre agentes				16
H = <15,26>					
tipo	origem	destino	conteúdo		
pedido-in	exterior		dd = 25	<15,26>	
pedido	g0	g14	dd = 25	DD =	26
pedido	g14	g1	dd = 25	FJM / FEJ =	1
pedido	g1	g4	dd = 24	FJM / FEJ =	1
pedido	g4	g7	dd = 22	FJM / FEJ =	1
pedido	g7	g8	dd = 19	FJM / FEJ =	1
pedido	g7	g9	dd = 19	FJM / FEJ =	1
pedido	g9	g11	dd = 16	FJM / FEJ =	1
pedido	g9	g12	dd = 16	FJM / FEJ =	1
pedido	g8	g17	dd = 17	FJM / FEJ =	1
pedido	g11	g18	dd = 13	FJM / FEJ =	1
pedido	g12	g19	dd = 15	FJM / FEJ =	1
pedido	g17	g0	dd = 17		
pedido	g18	g0	dd = 13		
pedido	g19	g0	dd = 15		
pedido-out	g0	exterior	dd = 17		
pedido-out	g0	exterior	dd = 13		
pedido-out	g0	exterior	dd = 15		
aceitação	g0	g17	RD = 15		
aceitação	g0	g18	RD = 15		
aceitação	g0	g19	RD = 15		
aceitação	g17	g8		FMJ / FEM =	2
aceitação	g18	g11		FMJ / FEM =	-2
aceitação	g19	g12		FMJ / FEM =	0
rejeição	g11	g9			
aceitação	g12	g9		FMJ / FEM =	0
aceitação	g8	g7		FMJ / FEM =	2
rejeição	g9	g7			
rejeição	g7	g4			
rejeição	g4	g1			
rejeição	g1	g14			
rejeição	g14	g0			
cancelamento	g7	g8			
cancelamento	g8	g17			
cancelamento	g9	g12			
cancelamento	g11	g18			
cancelamento	g12	g19			
cancelamento	g17	g0			
cancelamento	g18	g0			
cancelamento	g19	g0			

Quadro 2	Estabelecimento de pedidos locais														p
dd = 25	Problema de escalonamento														16
H = <15,26>	de agente gestor							Folgas							
Agente gestor	fornece-dores	cliente	Datas				Tarefa								
			RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	15	17	19	20	17	2	19	2	2	0	0	1	1	3
g11	g18	g9	15	13	16	17	13	3	16	-2	-2	0	0	1	1	-1
g12	g19	g9	15	15	16	17	15	1	16	0	0	0	0	1	1	1
g9	g11	g7	--	16	19	20	16	3	19	--	--	0	0	1	1	--
	g12		16	16						0	0	0				
g7	g8	g4	17	19	22	23	19	3	22	2	2	0	0	1	1	--
	g9		--	19						--	--	0				
g4	g7	g1	--	22	24	25	22	2	24	--	--	0	0	1	1	--
g1	g4	g14	--	24	25	26	24	1	25	--	--	0	0	1	1	--

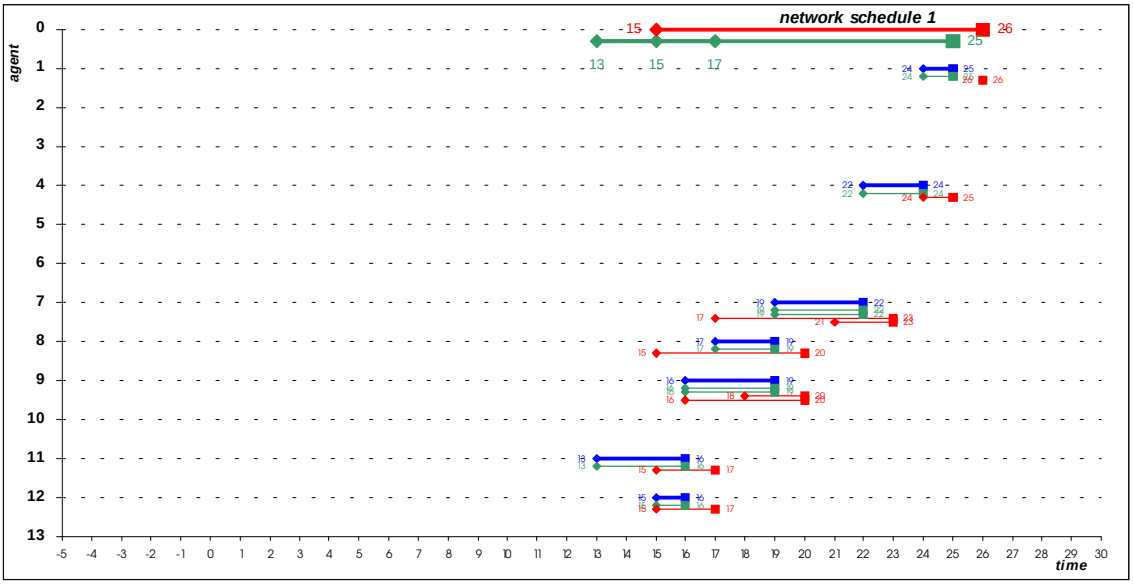
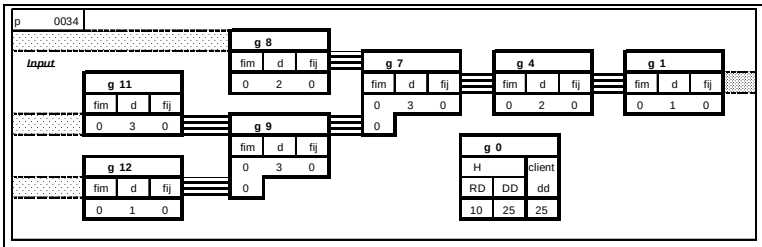


Figura 5-10- Exemplo 0016: resultados da fase 1 (sem sucesso).



Quadro 1	Estabelecimento de pedidos locais			p
dd = 25	Mensagens trocadas entre agentes			34
H = <10,25>				
tipo	origem	destino	conteúdo	
pedido-in	exterior		dd = 25	<10,25>
pedido	g0	g14	dd = 25	DD = 25
pedido	g14	g1	dd = 25	FJM / FEJ = 0
pedido	g1	g4	dd = 24	FJM / FEJ = 0
pedido	g4	g7	dd = 22	FJM / FEJ = 0
pedido	g7	g8	dd = 19	FJM / FEJ = 0
pedido	g7	g9	dd = 19	FJM / FEJ = 0
pedido	g9	g11	dd = 16	FJM / FEJ = 0
pedido	g9	g12	dd = 16	FJM / FEJ = 0
pedido	g8	g17	dd = 17	FJM / FEJ = 0
pedido	g11	g18	dd = 13	FJM / FEJ = 0
pedido	g12	g19	dd = 15	FJM / FEJ = 0
pedido	g17	g0	dd = 17	
pedido	g18	g0	dd = 13	
pedido	g19	g0	dd = 15	
pedido-out	g0	exterior	dd = 17	
pedido-out	g0	exterior	dd = 13	
pedido-out	g0	exterior	dd = 15	
aceitação	g0	g17	RD = 10	
aceitação	g0	g18	RD = 10	
aceitação	g0	g19	RD = 10	
aceitação	g17	g8		FMJ / FEM = 7
aceitação	g18	g11		FMJ / FEM = 3
aceitação	g19	g12		FMJ / FEM = 5
aceitação	g11	g9		FMJ / FEM = 3
aceitação	g12	g9		FMJ / FEM = 5
aceitação	g8	g7		FMJ / FEM = 7
aceitação	g9	g7		FMJ / FEM = 3
aceitação	g7	g4		FMJ / FEM = 3
aceitação	g4	g1		FMJ / FEM = 3
aceitação	g1	g14		FMJ / FEM = 3
aceitação	g14	g0		FT = 3

Quadro 2	Estabelecimento de pedidos locais													p
dd = 25	Problema de escalonamento													34
H = <10,25>	de agente gestor								Folgas					
Agente	fornece-	cliente	Datas				Tarefa							
gestor	dores		RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	10	17	19	19	17	2	19	7	7	0	0	0	0	7
g11	g18	g9	10	13	16	16	13	3	16	3	3	0	0	0	0	3
g12	g19	g9	10	15	16	16	15	1	16	5	5	0	0	0	0	5
g9	g11	g7	13	16	19	19	16	3	19	3	3	0	0	0	0	3
	g12		11	16						5	5	0				
g7	g8	g4	12	19	22	22	19	3	22	7	7	0	0	0	0	3
	g9		16	19						3	3	0				
g4	g7	g1	19	22	24	24	22	2	24	3	3	0	0	0	0	3
g1	g4	g14	21	24	25	25	24	1	25	3	3	0	0	0	0	3

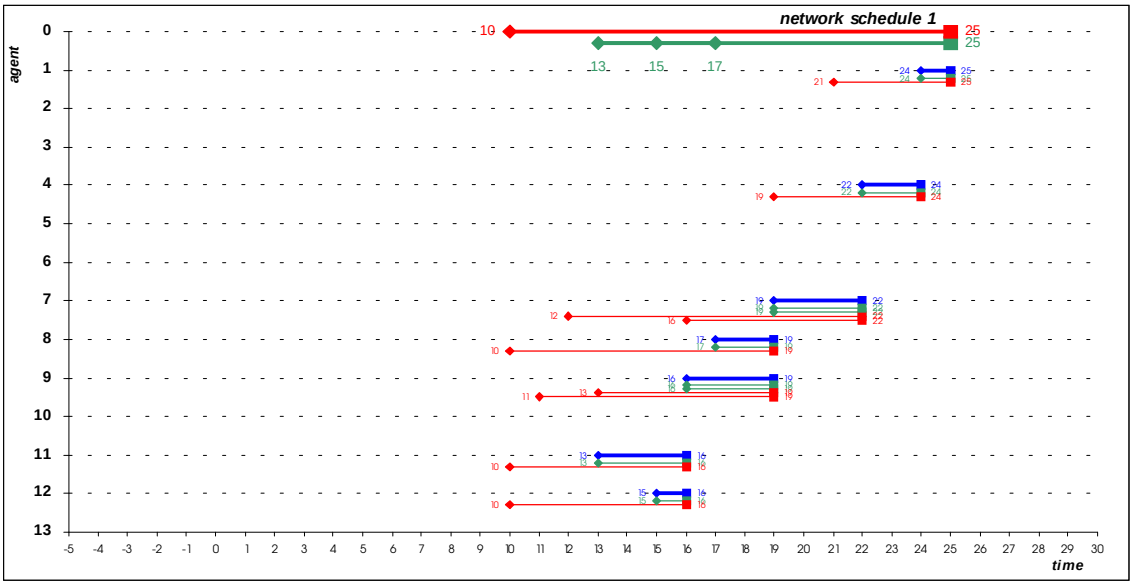
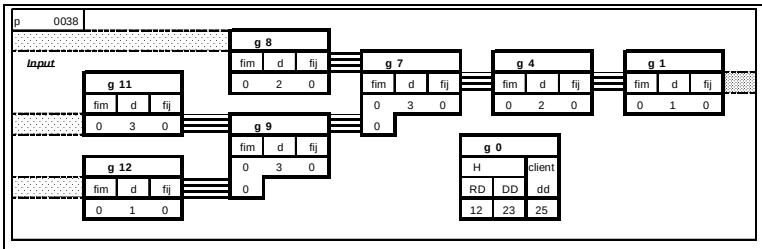


Figura 5-11- Exemplo 0034: resultados da fase 1.



Quadro 1	Estabelecimento de pedidos locais				p
dd = 25	Mensagens trocadas entre agentes				38
H = <12,23>					
tipo	origem	destino	conteúdo		
pedido-in	exterior		dd = 25	<12,23>	
pedido	g0	g14	dd = 25	DD = 23	
pedido	g14	g1	dd = 25	FJM / FEJ = -2	
pedido	g1	g4	dd = 24	FJM / FEJ = -2	
pedido	g4	g7	dd = 22	FJM / FEJ = -2	
pedido	g7	g8	dd = 19	FJM / FEJ = -2	
pedido	g7	g9	dd = 19	FJM / FEJ = -2	
pedido	g9	g11	dd = 16	FJM / FEJ = -2	
pedido	g9	g12	dd = 16	FJM / FEJ = -2	
pedido	g8	g17	dd = 17	FJM / FEJ = -2	
pedido	g11	g18	dd = 13	FJM / FEJ = -2	
pedido	g12	g19	dd = 15	FJM / FEJ = -2	
pedido	g17	g0	dd = 17		
pedido	g18	g0	dd = 13		
pedido	g19	g0	dd = 15		
pedido-out	g0	exterior	dd = 17		
pedido-out	g0	exterior	dd = 13		
pedido-out	g0	exterior	dd = 15		
aceitação	g0	g17	RD = 12		
aceitação	g0	g18	RD = 12		
aceitação	g0	g19	RD = 12		
aceitação	g17	g8		FMJ / FEM = 5	
aceitação	g18	g11		FMJ / FEM = 1	
aceitação	g19	g12		FMJ / FEM = 3	
rejeição	g11	g9			
aceitação	g12	g9		FMJ / FEM = 3	
aceitação	g8	g7		FMJ / FEM = 5	
rejeição	g9	g7			
rejeição	g7	g4			
rejeição	g4	g1			
rejeição	g1	g14			
rejeição	g14	g0			
cancelamento	g7	g8			
cancelamento	g8	g17			
cancelamento	g9	g12			
cancelamento	g11	g18			
cancelamento	g12	g19			
cancelamento	g17	g0			
cancelamento	g18	g0			
cancelamento	g19	g0			

Quadro 2	Estabelecimento de pedidos locais														p
dd = 25	Problema de escalonamento														38
H = <12,23>	de agente gestor														
			Datas				Tarefa			Folgas							
Agente gestor	fornece-dores	cliente	RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	12	17	19	17	17	2	19	5	5	0	0	-2	-2	3
g11	g18	g9	12	13	16	14	13	3	16	1	1	0	0	-2	-2	-1
g12	g19	g9	12	15	16	14	15	1	16	3	3	0	0	-2	-2	1
g9	g11	g7	--	16	19	17	16	3	19	--	--	0	0	-2	-2	--
	g12		13	16						3	3	0				
g7	g8	g4	14	19	22	20	19	3	22	5	5	0	0	-2	-2	--
	g9		--	19						--	--	0				
g4	g7	g1	--	22	24	22	22	2	24	--	--	0	0	-2	-2	--
g1	g4	g14	--	24	25	23	24	1	25	--	--	0	0	-2	-2	--

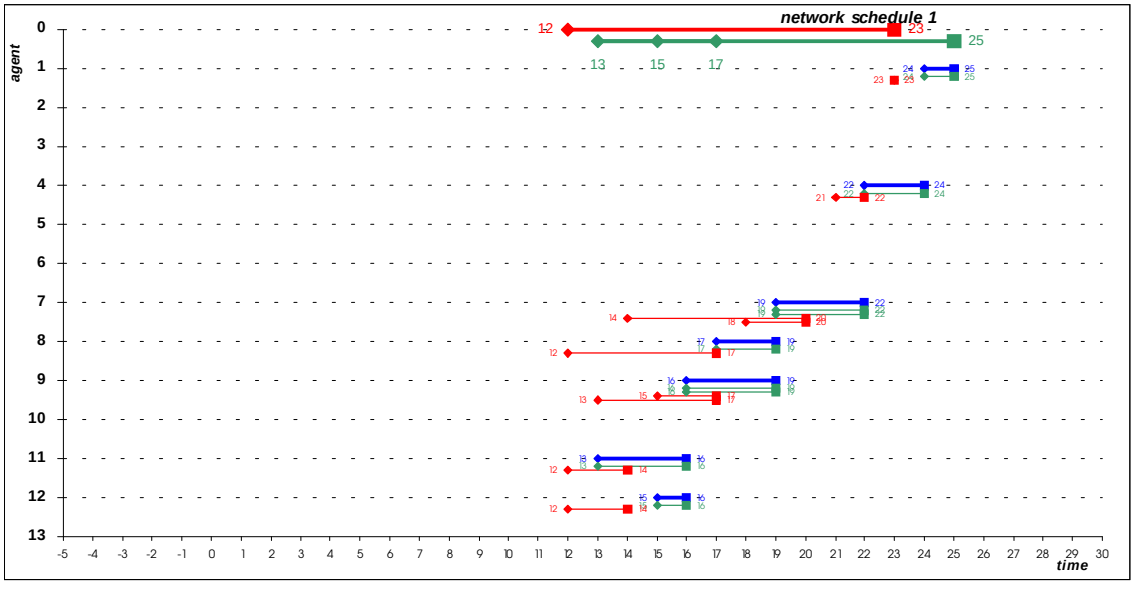
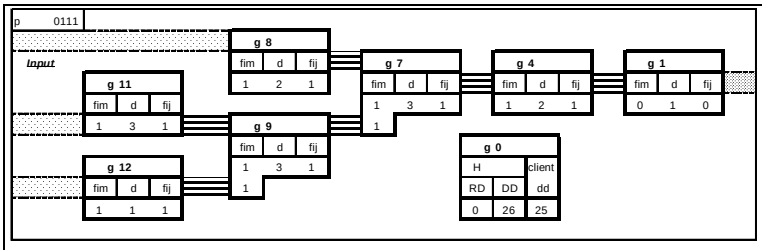


Figura 5-12- Exemplo 0038: resultados da fase 1 (sem sucesso).



Quadro 1	Estabelecimento de pedidos locais			p
dd = 25	Mensagens trocadas entre agentes			111
H = <0,26>				
tipo	origem	destino	conteúdo	
pedido-in	exterior		dd = 25	<0,26>
pedido	g0	g14	dd = 25	DD = 26
pedido	g14	g1	dd = 25	FJM / FEJ = 1
pedido	g1	g4	dd = 24	FJM / FEJ = 1
pedido	g4	g7	dd = 20	FJM / FEJ = 3
pedido	g7	g8	dd = 15	FJM / FEJ = 5
pedido	g7	g9	dd = 15	FJM / FEJ = 5
pedido	g9	g11	dd = 10	FJM / FEJ = 7
pedido	g9	g12	dd = 10	FJM / FEJ = 7
pedido	g8	g17	dd = 11	FJM / FEJ = 7
pedido	g11	g18	dd = 5	FJM / FEJ = 9
pedido	g12	g19	dd = 7	FJM / FEJ = 9
pedido	g17	g0	dd = 11	
pedido	g18	g0	dd = 5	
pedido	g19	g0	dd = 7	
pedido-out	g0	exterior	dd = 11	
pedido-out	g0	exterior	dd = 5	
pedido-out	g0	exterior	dd = 7	
aceitação	g0	g17	RD = 0	
aceitação	g0	g18	RD = 0	
aceitação	g0	g19	RD = 0	
aceitação	g17	g8		FMJ / FEM = 11
aceitação	g18	g11		FMJ / FEM = 5
aceitação	g19	g12		FMJ / FEM = 7
aceitação	g11	g9		FMJ / FEM = 7
aceitação	g12	g9		FMJ / FEM = 9
aceitação	g8	g7		FMJ / FEM = 13
aceitação	g9	g7		FMJ / FEM = 9
aceitação	g7	g4		FMJ / FEM = 11
aceitação	g4	g1		FMJ / FEM = 13
aceitação	g1	g14		FMJ / FEM = 13
aceitação	g14	g0		FT = 14

Quadro 2	Estabelecimento de pedidos locais														p
dd = 25	Problema de escalonamento														111
H = <0,26>	de agente gestor											Folgas			
Agente gestor	fornece-dores	cliente	Datas				Tarefa								
			RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	0	11	15	20	12	2	14	12	11	1	1	5	6	18
g11	g18	g9	0	5	10	17	6	3	9	6	5	1	1	7	8	14
g12	g19	g9	0	7	10	17	8	1	9	8	7	1	1	7	8	16
g9	g11 g12	g7	3 1	10 10	15	20	11	3	14	8 10	7 9	1 1	1	5	6	14
g7	g8 g9	g4	2 6	15 15	20	23	16	3	19	14 10	13 9	1 1	1	3	4	14
g4	g7	g1	9	20	24	25	21	2	23	12	11	1	1	1	2	14
g1	g4	g14	11	24	25	26	24	1	25	13	13	0	0	1	1	14

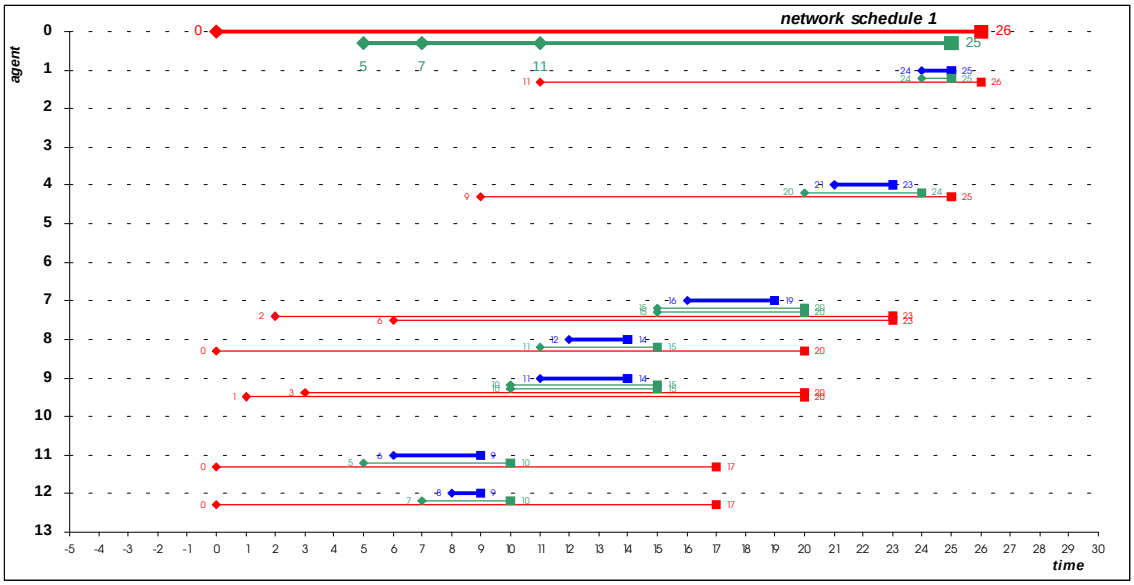
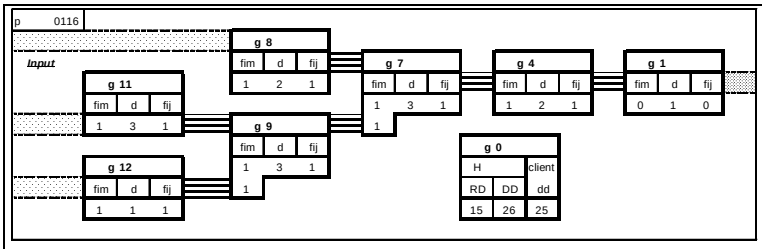


Figura 5-13- Exemplo 0111: resultados da fase 1.



Quadro 1	Estabelecimento de pedidos locais			p
dd = 25	Mensagens trocadas entre agentes			0116
H = <15,26>				
tipo	origem	destino	conteúdo	
pedido-in	exterior		dd = 25	<15,26>
pedido	g0	g14	dd = 25	DD = 26
pedido	g14	g1	dd = 25	FJM / FEJ = 1
pedido	g1	g4	dd = 24	FJM / FEJ = 1
pedido	g4	g7	dd = 20	FJM / FEJ = 3
pedido	g7	g8	dd = 15	FJM / FEJ = 5
pedido	g7	g9	dd = 15	FJM / FEJ = 5
pedido	g9	g11	dd = 10	FJM / FEJ = 7
pedido	g9	g12	dd = 10	FJM / FEJ = 7
pedido	g8	g17	dd = 11	FJM / FEJ = 7
pedido	g11	g18	dd = 5	FJM / FEJ = 9
pedido	g12	g19	dd = 7	FJM / FEJ = 9
pedido	g17	g0	dd = 11	
pedido	g18	g0	dd = 5	
pedido	g19	g0	dd = 7	
pedido-out	g0	exterior	dd = 11	
pedido-out	g0	exterior	dd = 5	
pedido-out	g0	exterior	dd = 7	
aceitação	g0	g17	RD = 15	
aceitação	g0	g18	RD = 15	
aceitação	g0	g19	RD = 15	
aceitação	g17	g8		FMJ / FEM = -4
aceitação	g18	g11		FMJ / FEM = -10
aceitação	g19	g12		FMJ / FEM = -8
rejeição	g11	g9		
aceitação	g12	g9		FMJ / FEM = -6
aceitação	g8	g7		FMJ / FEM = -2
rejeição	g9	g7		
rejeição	g7	g4		
rejeição	g4	g1		
rejeição	g1	g14		
rejeição	g14	g0		
cancelamento	g7	g8		
cancelamento	g8	g17		
cancelamento	g9	g12		
cancelamento	g11	g18		
cancelamento	g12	g19		
cancelamento	g17	g0		
cancelamento	g18	g0		
cancelamento	g19	g0		

Quadro 2	Estabelecimento de pedidos locais													p
dd = 25	Problema de escalonamento													0116
H = <15,26>	de agente gestor								Folgas					
Agente gestor	fornece-dores	cliente	Datas				Tarefa							
			RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	15	11	15	20	12	2	14	-3	-4	1	1	5	6	3
g11	g18	g9	15	5	10	17	6	3	9	-9	-10	1	1	7	8	-1
g12	g19	g9	15	7	10	17	8	1	9	-7	-8	1	1	7	8	1
g9	g11	g7	--	10	15	20	11	3	14	--	--	1	1	5	6	--
	g12		16	10						-5	-6	1				
g7	g8	g4	17	15	20	23	16	3	19	-1	-2	1	1	3	4	--
	g9		--	15						--	--	1				
g4	g7	g1	--	20	24	25	21	2	23	--	--	1	1	1	2	--
g1	g4	g14	--	24	25	26	24	1	25	--	--	0	0	1	1	--

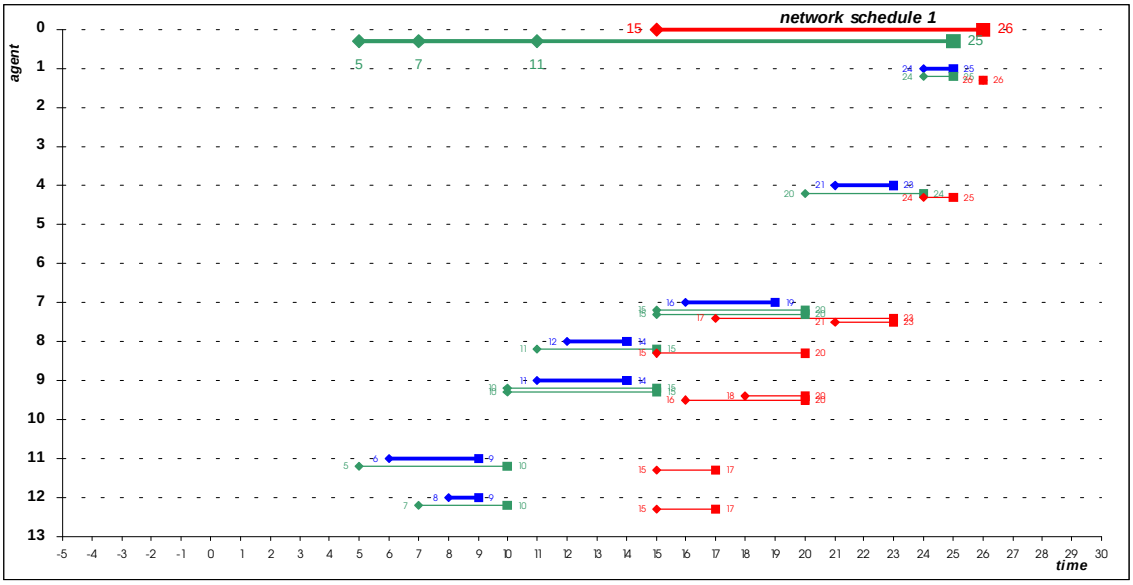
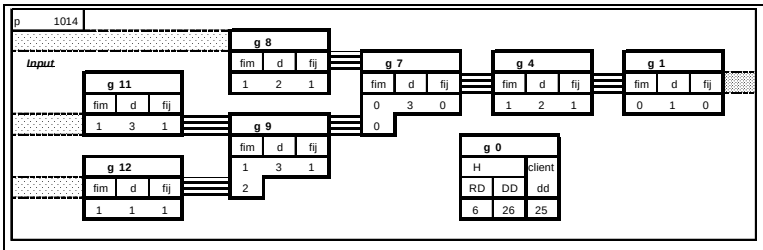


Figura 5-14- Exemplo 0116: resultados da fase 1 (sem sucesso).



Quadro 1	Estabelecimento de pedidos locais			p
dd = 25	Mensagens trocadas entre agentes			1014
H = <6,26>				
tipo	origem	destino	conteúdo	
pedido-in	exterior		dd = 25	<6,26>
pedido	g0	g14	dd = 25	DD = 26
pedido	g14	g1	dd = 25	FJM / FEJ = 1
pedido	g1	g4	dd = 24	FJM / FEJ = 1
pedido	g4	g7	dd = 20	FJM / FEJ = 3
pedido	g7	g8	dd = 17	FJM / FEJ = 3
pedido	g7	g9	dd = 17	FJM / FEJ = 3
pedido	g9	g11	dd = 12	FJM / FEJ = 5
pedido	g9	g12	dd = 11	FJM / FEJ = 6
pedido	g8	g17	dd = 13	FJM / FEJ = 5
pedido	g11	g18	dd = 7	FJM / FEJ = 7
pedido	g12	g19	dd = 8	FJM / FEJ = 8
pedido	g17	g0	dd = 13	
pedido	g18	g0	dd = 7	
pedido	g19	g0	dd = 8	
pedido-out	g0	exterior	dd = 13	
pedido-out	g0	exterior	dd = 7	
pedido-out	g0	exterior	dd = 8	
aceitação	g0	g17	RD = 6	
aceitação	g0	g18	RD = 6	
aceitação	g0	g19	RD = 6	
aceitação	g17	g8		FMJ / FEM = 7
aceitação	g18	g11		FMJ / FEM = 1
aceitação	g19	g12		FMJ / FEM = 2
aceitação	g11	g9		FMJ / FEM = 3
aceitação	g12	g9		FMJ / FEM = 4
aceitação	g8	g7		FMJ / FEM = 9
aceitação	g9	g7		FMJ / FEM = 5
aceitação	g7	g4		FMJ / FEM = 5
aceitação	g4	g1		FMJ / FEM = 7
aceitação	g1	g14		FMJ / FEM = 7
aceitação	g14	g0		FT = 8

Quadro 2	Estabelecimento de pedidos locais														p
dd = 25	Problema de escalonamento														1014
H = <6,26>	de agente gestor									Folgas					
Agente gestor	fornece-dores	cliente	Datas				Tarefa								
			RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	6	13	17	20	14	2	16	8	7	1	1	3	4	12
g11	g18	g9	6	7	12	17	8	3	11	2	1	1	1	5	6	8
g12	g19	g9	6	8	11	17	9	1	10	3	2	1	1	6	7	10
g9	g11	g7	9	12	17	20	13	3	16	4	3	1	1	3	4	8
	g12		7	11						6	4	2				
g7	g8	g4	8	17	20	23	17	3	20	9	9	0	0	3	3	8
	g9		12	17						5	5	0				
g4	g7	g1	15	20	24	25	21	2	23	6	5	1	1	1	2	8
g1	g4	g14	17	24	25	26	24	1	25	7	7	0	0	1	1	8

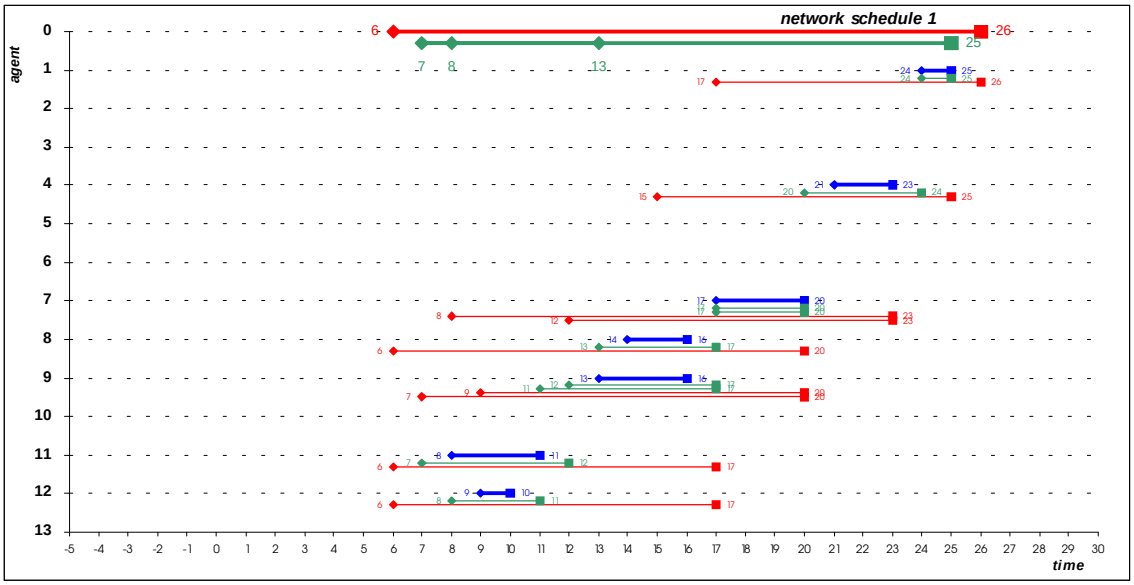
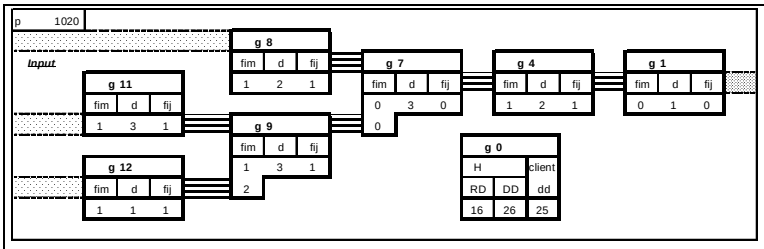


Figura 5-15- Exemplo 1014: resultados da fase 1.



Quadro 1	Estabelecimento de pedidos locais			p
dd = 25	Mensagens trocadas entre agentes			1020
H = <16,26>				
tipo	origem	destino	conteúdo	
pedido-in	exterior		dd = 25	<16,26>
pedido	g0	g14	dd = 25	DD = 26
pedido	g14	g1	dd = 25	FJM / FEJ = 1
pedido	g1	g4	dd = 24	FJM / FEJ = 1
pedido	g4	g7	dd = 20	FJM / FEJ = 3
pedido	g7	g8	dd = 17	FJM / FEJ = 3
pedido	g7	g9	dd = 17	FJM / FEJ = 3
pedido	g9	g11	dd = 12	FJM / FEJ = 5
pedido	g9	g12	dd = 11	FJM / FEJ = 6
pedido	g8	g17	dd = 13	FJM / FEJ = 5
pedido	g11	g18	dd = 7	FJM / FEJ = 7
pedido	g12	g19	dd = 8	FJM / FEJ = 8
pedido	g17	g0	dd = 13	
pedido	g18	g0	dd = 7	
pedido	g19	g0	dd = 8	
pedido-out	g0	exterior	dd = 13	
pedido-out	g0	exterior	dd = 7	
pedido-out	g0	exterior	dd = 8	
aceitação	g0	g17	RD = 16	
aceitação	g0	g18	RD = 16	
aceitação	g0	g19	RD = 16	
aceitação	g17	g8		FMJ / FEM = -3
aceitação	g18	g11		FMJ / FEM = -9
aceitação	g19	g12		FMJ / FEM = -8
rejeição	g11	g9		
aceitação	g12	g9		FMJ / FEM = -6
aceitação	g8	g7		FMJ / FEM = -1
rejeição	g9	g7		
rejeição	g7	g4		
rejeição	g4	g1		
rejeição	g1	g14		
rejeição	g14	g0		
cancelamento	g7	g8		
cancelamento	g8	g17		
cancelamento	g9	g12		
cancelamento	g11	g18		
cancelamento	g12	g19		
cancelamento	g17	g0		
cancelamento	g18	g0		
cancelamento	g19	g0		

Quadro 2	Estabelecimento de pedidos locais														p
dd = 25	Problema de escalonamento														1020
H = <16,26>	de agente gestor									Folgas					
Agente	fornece-	cliente	Datas				Tarefa								
gestor	dores		RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	16	13	17	20	14	2	16	-2	-3	1	1	3	4	2
g11	g18	g9	16	7	12	17	8	3	11	-8	-9	1	1	5	6	-2
g12	g19	g9	16	8	11	17	9	1	10	-7	-8	1	1	6	7	0
g9	g11	g7	--	12	17	20	13	3	16	--	--	1	1	3	4	--
	g12		17	11						-4	-6	2				
g7	g8	g4	18	17	20	23	17	3	20	-1	-1	0	0	3	3	--
	g9		--	17						--	--	0				
g4	g7	g1	--	20	24	25	21	2	23	--	--	1	1	1	2	--
g1	g4	g14	--	24	25	26	24	1	25	--	--	0	0	1	1	--

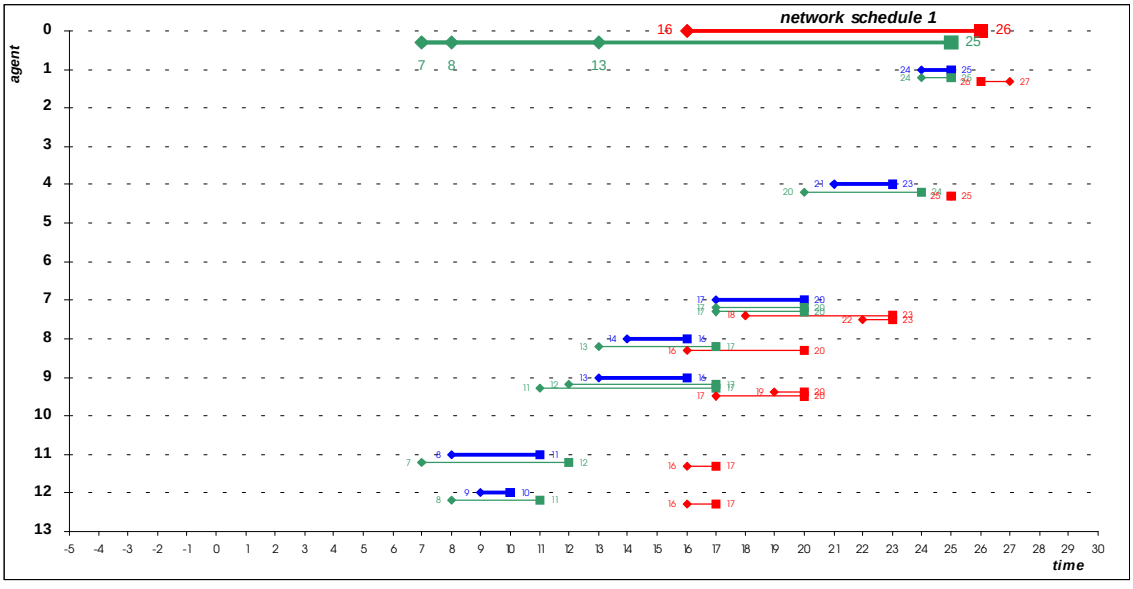
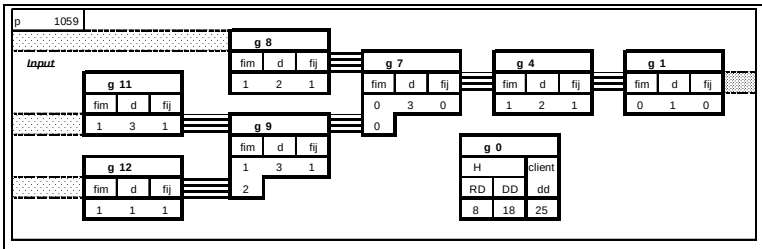


Figura 5-16- Exemplo 1020: resultados da fase 1 (sem sucesso).



Quadro 1	Estabelecimento de pedidos locais			p
dd = 25	Mensagens trocadas entre agentes			1059
H = <8,18>				
tipo	origem	destino	conteúdo	
pedido-in	exterior		dd = 25	<8,18> DD = 18
pedido	g0	g14	dd = 25	
pedido	g14	g1	dd = 25	FJM / FEJ = -7
pedido	g1	g4	dd = 24	FJM / FEJ = -7
pedido	g4	g7	dd = 20	FJM / FEJ = -5
pedido	g7	g8	dd = 17	FJM / FEJ = -5
pedido	g7	g9	dd = 17	FJM / FEJ = -5
pedido	g9	g11	dd = 12	FJM / FEJ = -3
pedido	g9	g12	dd = 11	FJM / FEJ = -2
pedido	g8	g17	dd = 13	FJM / FEJ = -3
pedido	g11	g18	dd = 7	FJM / FEJ = -1
pedido	g12	g19	dd = 8	FJM / FEJ = 0
pedido	g17	g0	dd = 13	
pedido	g18	g0	dd = 7	
pedido	g19	g0	dd = 8	
pedido-out	g0	exterior	dd = 13	
pedido-out	g0	exterior	dd = 7	
pedido-out	g0	exterior	dd = 8	
aceitação	g0	g17	RD = 8	
aceitação	g0	g18	RD = 8	
aceitação	g0	g19	RD = 8	
aceitação	g17	g8		FMJ / FEM = 5
aceitação	g18	g11		FMJ / FEM = -1
aceitação	g19	g12		FMJ / FEM = 0
rejeição	g11	g9		
aceitação	g12	g9		FMJ / FEM = 2
aceitação	g8	g7		FMJ / FEM = 7
rejeição	g9	g7		
rejeição	g7	g4		
rejeição	g4	g1		
rejeição	g1	g14		
rejeição	g14	g0		
cancelamento	g7	g8		
cancelamento	g8	g17		
cancelamento	g9	g12		
cancelamento	g11	g18		
cancelamento	g12	g19		
cancelamento	g17	g0		
cancelamento	g18	g0		
cancelamento	g19	g0		

Quadro 2	Estabelecimento de pedidos locais													p
dd = 25	Problema de escalonamento													1059
H = <8,18>	de agente gestor													
								Folgas						
Agente gestor	fornece-dores	cliente	Datas RD	rd	dd	DD	Tarefa s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	8	13	17	12	14	2	16	6	5	1	1	-5	-4	2
g11	g18	g9	8	7	12	9	8	3	11	0	-1	1	1	-3	-2	-2
g12	g19	g9	8	8	11	9	9	1	10	1	0	1	1	-2	-1	0
g9	g11	g7	--	12	17	12	13	3	16	--	--	1	1	-5	-4	--
	g12		9	11						4	2	2				
g7	g8	g4	10	17	20	15	17	3	20	7	7	0	0	-5	-5	--
	g9		--	17						--	--	0				
g4	g7	g1	--	20	24	17	21	2	23	--	--	1	1	-7	-6	--
g1	g4	g14	--	24	25	18	24	1	25	--	--	0	0	-7	-7	--

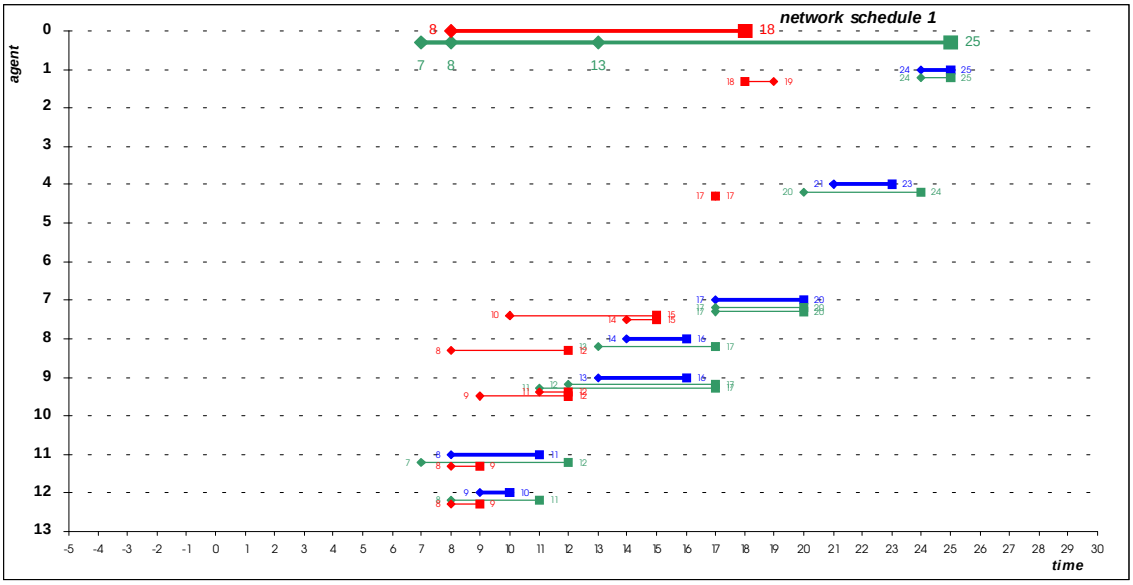
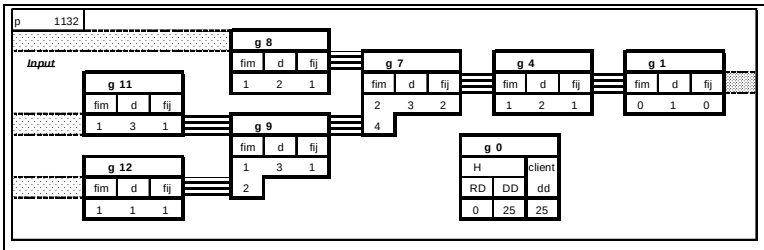


Figura 5-17- Exemplo 1059: resultados da fase 1 (sem sucesso).



Quadro 1	Estabelecimento de pedidos locais			p
dd = 25	Mensagens trocadas entre agentes			1132
H = <0,25>				
tipo	origem	destino	conteúdo	
pedido-in	exterior		dd = 25	<0,25>
pedido	g0	g14	dd = 25	DD = 25
pedido	g14	g1	dd = 25	FJM / FEJ = 0
pedido	g1	g4	dd = 24	FJM / FEJ = 0
pedido	g4	g7	dd = 20	FJM / FEJ = 2
pedido	g7	g8	dd = 13	FJM / FEJ = 6
pedido	g7	g9	dd = 11	FJM / FEJ = 8
pedido	g9	g11	dd = 6	FJM / FEJ = 10
pedido	g9	g12	dd = 5	FJM / FEJ = 11
pedido	g8	g17	dd = 9	FJM / FEJ = 8
pedido	g11	g18	dd = 1	FJM / FEJ = 12
pedido	g12	g19	dd = 2	FJM / FEJ = 13
pedido	g17	g0	dd = 9	
pedido	g18	g0	dd = 1	
pedido	g19	g0	dd = 2	
pedido-out	g0	exterior	dd = 9	
pedido-out	g0	exterior	dd = 1	
pedido-out	g0	exterior	dd = 2	
aceitação	g0	g17	RD = 0	
aceitação	g0	g18	RD = 0	
aceitação	g0	g19	RD = 0	
aceitação	g17	g8		FMJ / FEM = 9
aceitação	g18	g11		FMJ / FEM = 1
aceitação	g19	g12		FMJ / FEM = 2
aceitação	g11	g9		FMJ / FEM = 3
aceitação	g12	g9		FMJ / FEM = 4
aceitação	g8	g7		FMJ / FEM = 11
aceitação	g9	g7		FMJ / FEM = 5
aceitação	g7	g4		FMJ / FEM = 11
aceitação	g4	g1		FMJ / FEM = 13
aceitação	g1	g14		FMJ / FEM = 13
aceitação	g14	g0		FT = 13

Quadro 2	Estabelecimento de pedidos locais													p
dd = 25	Problema de escalonamento													1132
H = <0,25>	de agente gestor									Folgas				
Agente	fornece-	cliente	Datas				Tarefa							
gestor	dores		RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	0	9	13	19	10	2	12	10	9	1	1	6	7	17
g11	g18	g9	0	1	6	16	2	3	5	2	1	1	1	10	11	13
g12	g19	g9	0	2	5	16	3	1	4	3	2	1	1	11	12	15
g9	g11	g7	3	6	11	19	7	3	10	4	3	1	1	8	9	13
	g12		1	5						6	4	2				
g7	g8	g4	2	13	20	22	15	3	18	13	11	2	2	2	4	13
	g9		6	11						9	5	4				
g4	g7	g1	9	20	24	24	21	2	23	12	11	1	1	0	1	13
g1	g4	g14	11	24	25	25	24	1	25	13	13	0	0	0	0	13

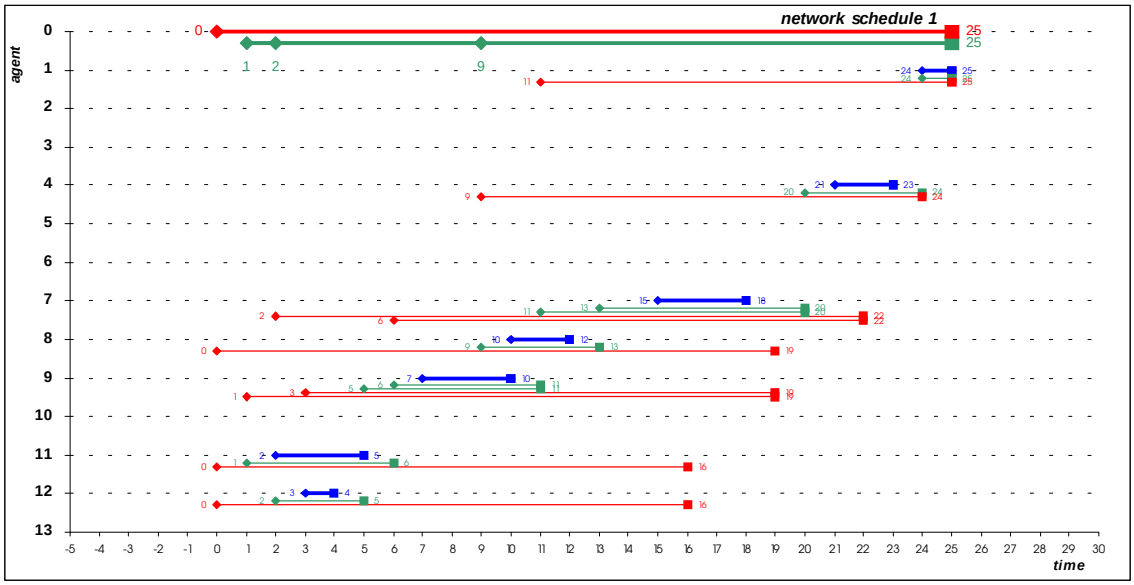
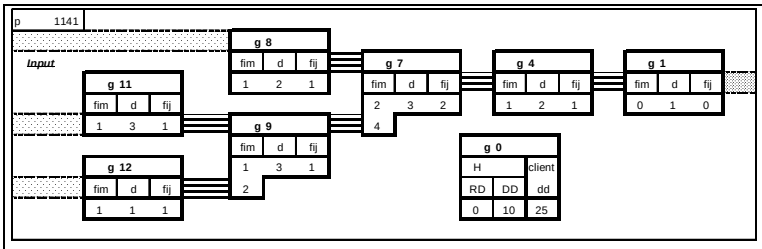


Figura 5-18- Exemplo 1132: resultados da fase 1.



Quadro 1	Estabelecimento de pedidos locais				p
dd = 25	Mensagens trocadas entre agentes				1141
H = <0,10>					
tipo	origem	destino	conteúdo		
pedido-in	exterior		dd = 25	<0,10>	
pedido	g0	g14	dd = 25	DD = 10	
pedido	g14	g1	dd = 25	FJM / FEJ = -15	
pedido	g1	g4	dd = 24	FJM / FEJ = -15	
pedido	g4	g7	dd = 20	FJM / FEJ = -13	
pedido	g7	g8	dd = 13	FJM / FEJ = -9	
pedido	g7	g9	dd = 11	FJM / FEJ = -7	
pedido	g9	g11	dd = 6	FJM / FEJ = -5	
pedido	g9	g12	dd = 5	FJM / FEJ = -4	
pedido	g8	g17	dd = 9	FJM / FEJ = -7	
pedido	g11	g18	dd = 1	FJM / FEJ = -3	
pedido	g12	g19	dd = 2	FJM / FEJ = -2	
pedido	g17	g0	dd = 9		
pedido	g18	g0	dd = 1		
pedido	g19	g0	dd = 2		
pedido-out	g0	exterior	dd = 9		
pedido-out	g0	exterior	dd = 1		
pedido-out	g0	exterior	dd = 2		
aceitação	g0	g17	RD = 0		
aceitação	g0	g18	RD = 0		
aceitação	g0	g19	RD = 0		
aceitação	g17	g8		FMJ / FEM = 9	
aceitação	g18	g11		FMJ / FEM = 1	
aceitação	g19	g12		FMJ / FEM = 2	
rejeição	g11	g9			
aceitação	g12	g9		FMJ / FEM = 4	
aceitação	g8	g7		FMJ / FEM = 11	
rejeição	g9	g7			
rejeição	g7	g4			
rejeição	g4	g1			
rejeição	g1	g14			
rejeição	g14	g0			
cancelamento	g7	g8			
cancelamento	g8	g17			
cancelamento	g9	g12			
cancelamento	g11	g18			
cancelamento	g12	g19			
cancelamento	g17	g0			
cancelamento	g18	g0			
cancelamento	g19	g0			

Quadro 2	Estabelecimento de pedidos locais													p
dd = 25	Problema de escalonamento													1141
H = <0,10>	de agente gestor								Folgas					
Agente gestor	fornece-dores	cliente	Datas				Tarefa			FM	FEM	fim	fij	FEJ	FJ	FT
			RD	rd	dd	DD	s	d	e							
g8	g17	g7	0	9	13	4	10	2	12	10	9	1	1	-9	-8	2
g11	g18	g9	0	1	6	1	2	3	5	2	1	1	1	-5	-4	-2
g12	g19	g9	0	2	5	1	3	1	4	3	2	1	1	-4	-3	0
g9	g11 g12	g7	-- 1	6 5	11	4	7	3	10	-- 6	-- 4	1 2	1	-7	-6	--
g7	g8 g9	g4	2 --	13 11	20	7	15	3	18	13 --	11 --	2 4	2	-13	-11	--
g4	g7	g1	--	20	24	9	21	2	23	--	--	1	1	-15	-14	--
g1	g4	g14	--	24	25	10	24	1	25	--	--	0	0	-15	-15	--

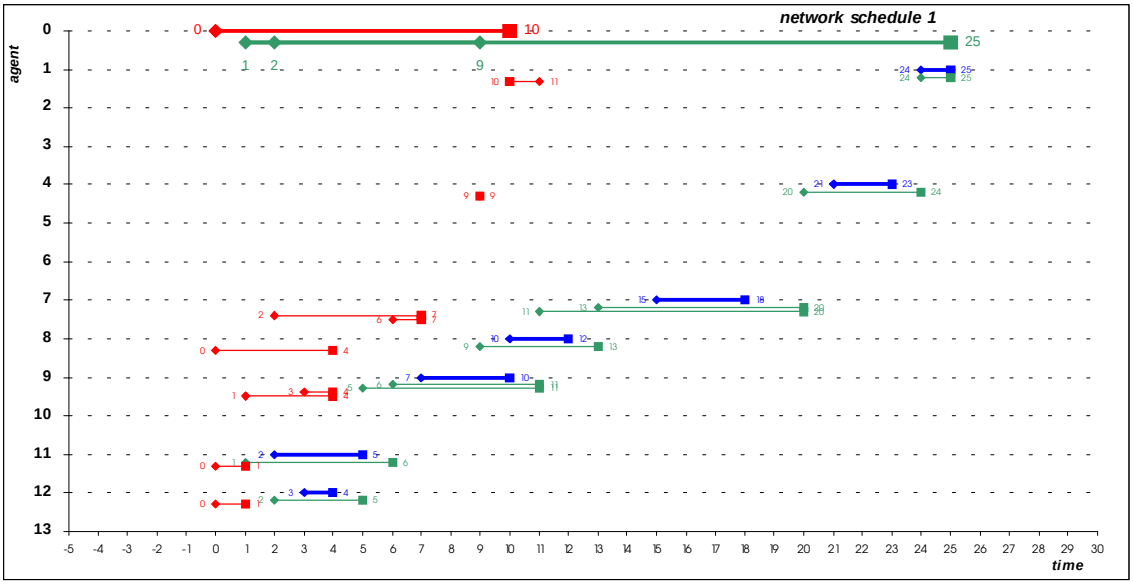
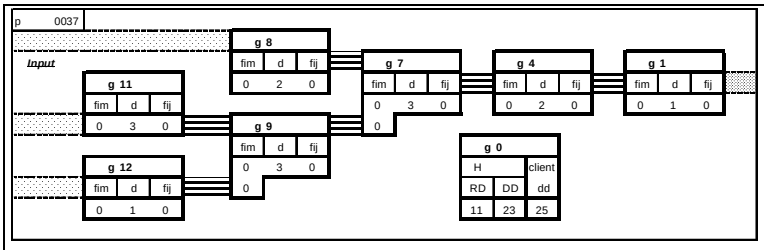


Figura 5-19- Exemplo 1141: resultados da fase 1 (sem sucesso).

5.2.2 Casos com Re-Escalonamento

Nos exemplos apresentados nesta secção, a fase 1 termina com sucesso, mas existem conflitos temporais no escalonamento multi-agente obtido. Estes conflitos temporais são removidos pelo re-escalonamento operado na fase 2. Ambos os resultados, da fase 1 e da fase 2, são mostrados.



Quadro 1	Estabelecimento de pedidos locais			p
dd = 25	Mensagens trocadas entre agentes			0037
H = <11,23>				
tipo	origem	destino	conteúdo	
pedido-in	exterior		dd = 25	<11,23>
pedido	g0	g14	dd = 25	DD = 23
pedido	g14	g1	dd = 25	FJM / FEJ = -2
pedido	g1	g4	dd = 24	FJM / FEJ = -2
pedido	g4	g7	dd = 22	FJM / FEJ = -2
pedido	g7	g8	dd = 19	FJM / FEJ = -2
pedido	g7	g9	dd = 19	FJM / FEJ = -2
pedido	g9	g11	dd = 16	FJM / FEJ = -2
pedido	g9	g12	dd = 16	FJM / FEJ = -2
pedido	g8	g17	dd = 17	FJM / FEJ = -2
pedido	g11	g18	dd = 13	FJM / FEJ = -2
pedido	g12	g19	dd = 15	FJM / FEJ = -2
pedido	g17	g0	dd = 17	
pedido	g18	g0	dd = 13	
pedido	g19	g0	dd = 15	
pedido-out	g0	exterior	dd = 17	
pedido-out	g0	exterior	dd = 13	
pedido-out	g0	exterior	dd = 15	
aceitação	g0	g17	RD = 11	
aceitação	g0	g18	RD = 11	
aceitação	g0	g19	RD = 11	
aceitação	g17	g8		FMJ / FEM = 6
aceitação	g18	g11		FMJ / FEM = 2
aceitação	g19	g12		FMJ / FEM = 4
aceitação	g11	g9		FMJ / FEM = 2
aceitação	g12	g9		FMJ / FEM = 4
aceitação	g8	g7		FMJ / FEM = 6
aceitação	g9	g7		FMJ / FEM = 2
aceitação	g7	g4		FMJ / FEM = 2
aceitação	g4	g1		FMJ / FEM = 2
aceitação	g1	g14		FMJ / FEM = 2
aceitação	g14	g0		FT = 0

Quadro 2	Estabelecimento de pedidos locais														p
dd = 25	Problema de escalonamento														0037
H = <11,23>	de agente gestor							Folgas							
Agente	fornece-	cliente	Datas				Tarefa								
gestor	dores		RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	11	17	19	17	17	2	19	6	6	0	0	-2	-2	4
g11	g18	g9	11	13	16	14	13	3	16	2	2	0	0	-2	-2	0
g12	g19	g9	11	15	16	14	15	1	16	4	4	0	0	-2	-2	2
g9	g11	g7	14	16	19	17	16	3	19	2	2	0	0	-2	-2	0
	g12		12	16						4	4	0				
g7	g8	g4	13	19	22	20	19	3	22	6	6	0	0	-2	-2	0
	g9		17	19						2	2	0				
g4	g7	g1	20	22	24	22	22	2	24	2	2	0	0	-2	-2	0
g1	g4	g14	22	24	25	23	24	1	25	2	2	0	0	-2	-2	0

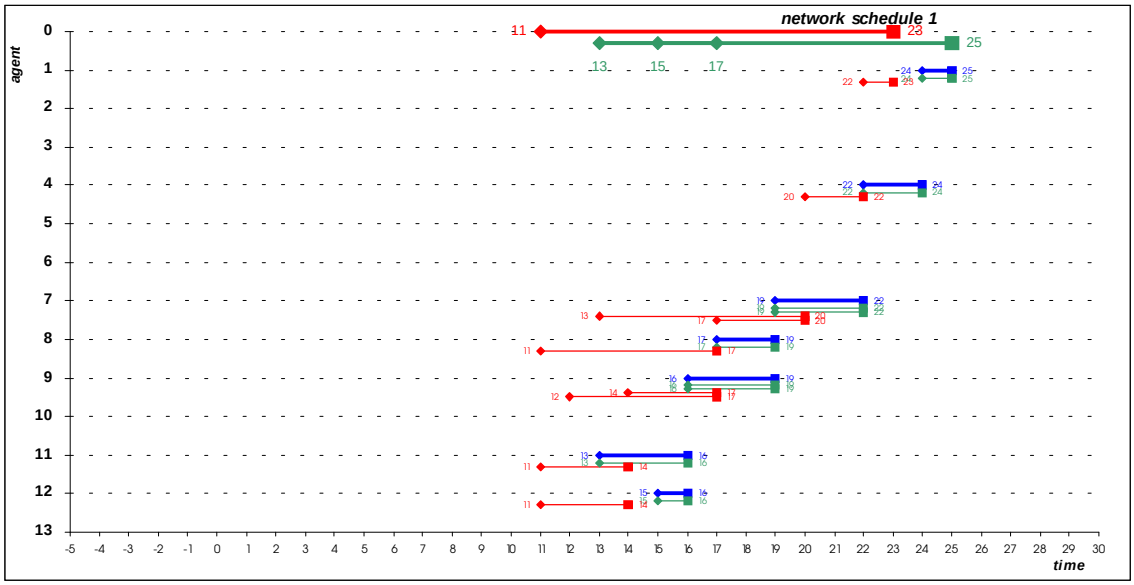


Figura 5-20- Exemplo 0037: resultados da fase 1.

Quadro 3	Re-escalonamento			p
dd = 25	Mensagens trocadas entre agentes			0037
H = <11,23>				
tipo	origem	destino	conteúdo	
re-escalonamento	g14	g0	dd = 23	
pedido-re	g1	g14	dd = 23	FJ = -2
pedido-re	g1	g4	rd = 22	FJ = -2
pedido-re	g4	g1	dd = 22	FJ = -2
pedido-re	g4	g7	rd = 20	FJ = -2
pedido-re	g7	g4	dd = 20	FJ = -2
pedido-re	g7	g8	rd = 17	FJ = -2
pedido-re	g7	g9	rd = 17	FJ = -2
pedido-re	g8	g7	dd = 17	FJ = -2
pedido-re	g8	g17	rd = 15	FJ = -2
pedido-re	g9	g7	dd = 17	FJ = -2
pedido-re	g9	g11	rd = 14	FJ = -2
pedido-re	g9	g12	rd = 14	FJ = -2
pedido-re	g11	g9	dd = 14	FJ = -2
pedido-re	g11	g18	rd = 11	FJ = -2
pedido-re	g12	g9	dd = 14	FJ = -2
pedido-re	g12	g19	rd = 13	FJ = -2
re-escalonamento	g17	g0	dd = 15	
re-escalonamento	g18	g0	dd = 11	
re-escalonamento	g19	g0	dd = 13	

Quadro 4	Re-escalonamento													p
dd = 25	Problema de escalonamento													0037
H = <11,23>	de agente gestor							Folgas						
Agente gestor	fornece-dores	cliente	Datas				Tarefa							
			RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	11	15	17	17	15	2	17	4	4	0	0	0	0	4
g11	g18	g9	11	11	14	14	11	3	14	0	0	0	0	0	0	0
g12	g19	g9	11	13	14	14	13	1	14	2	2	0	0	0	0	2
g9	g11	g7	14	14	17	17	14	3	17	0	0	0	0	0	0	0
	g12		12	14						2	2	0				
g7	g8	g4	13	17	20	20	17	3	20	4	4	0	0	0	0	0
	g9		17	17						0	0	0				
g4	g7	g1	20	20	22	22	20	2	22	0	0	0	0	0	0	0
g1	g4	g14	22	22	23	23	22	1	23	0	0	0	0	0	0	0

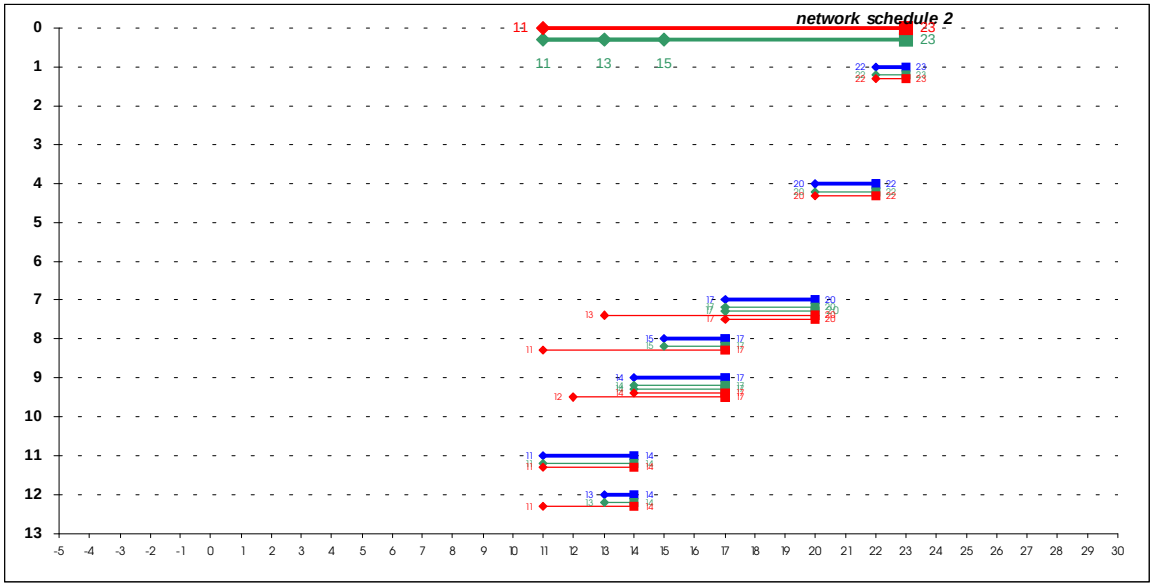
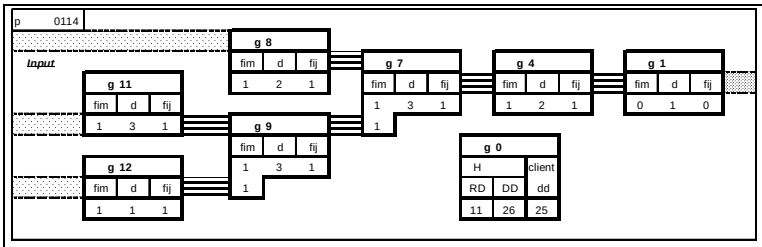


Figura 5-21- Exemplo 0037: resultados da fase 2.



Quadro 1	Estabelecimento de pedidos locais			p
dd = 25	Mensagens trocadas entre agentes			0114
H = <11,26>				
tipo	origem	destino	conteúdo	
pedido-in	exterior		dd = 25	<11,26>
pedido	g0	g14	dd = 25	DD = 26
pedido	g14	g1	dd = 25	FJM / FEJ = 1
pedido	g1	g4	dd = 24	FJM / FEJ = 1
pedido	g4	g7	dd = 20	FJM / FEJ = 3
pedido	g7	g8	dd = 15	FJM / FEJ = 5
pedido	g7	g9	dd = 15	FJM / FEJ = 5
pedido	g9	g11	dd = 10	FJM / FEJ = 7
pedido	g9	g12	dd = 10	FJM / FEJ = 7
pedido	g8	g17	dd = 11	FJM / FEJ = 7
pedido	g11	g18	dd = 5	FJM / FEJ = 9
pedido	g12	g19	dd = 7	FJM / FEJ = 9
pedido	g17	g0	dd = 11	
pedido	g18	g0	dd = 5	
pedido	g19	g0	dd = 7	
pedido-out	g0	exterior	dd = 11	
pedido-out	g0	exterior	dd = 5	
pedido-out	g0	exterior	dd = 7	
aceitação	g0	g17	RD = 11	
aceitação	g0	g18	RD = 11	
aceitação	g0	g19	RD = 11	
aceitação	g17	g8		FMJ / FEM = 0
aceitação	g18	g11		FMJ / FEM = -6
aceitação	g19	g12		FMJ / FEM = -4
aceitação	g11	g9		FMJ / FEM = -4
aceitação	g12	g9		FMJ / FEM = -2
aceitação	g8	g7		FMJ / FEM = 2
aceitação	g9	g7		FMJ / FEM = -2
aceitação	g7	g4		FMJ / FEM = 0
aceitação	g4	g1		FMJ / FEM = 2
aceitação	g1	g14		FMJ / FEM = 2
aceitação	g14	g0		FT = 3

Quadro 2	Estabelecimento de pedidos locais													p
dd = 25	Problema de escalonamento													0114
H = <11,26>	de agente gestor								Folgas					
Agente	fornece-	cliente	Datas				Tarefa							
gestor	dores		RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	11	11	15	20	12	2	14	1	0	1	1	5	6	7
g11	g18	g9	11	5	10	17	6	3	9	-5	-6	1	1	7	8	3
g12	g19	g9	11	7	10	17	8	1	9	-3	-4	1	1	7	8	5
g9	g11	g7	14	10	15	20	11	3	14	-3	-4	1	1	5	6	3
	g12		12	10						-1	-2	1				
g7	g8	g4	13	15	20	23	16	3	19	3	2	1	1	3	4	3
	g9		17	15						-1	-2	1				
g4	g7	g1	20	20	24	25	21	2	23	1	0	1	1	1	2	3
g1	g4	g14	22	24	25	26	24	1	25	2	2	0	0	1	1	3

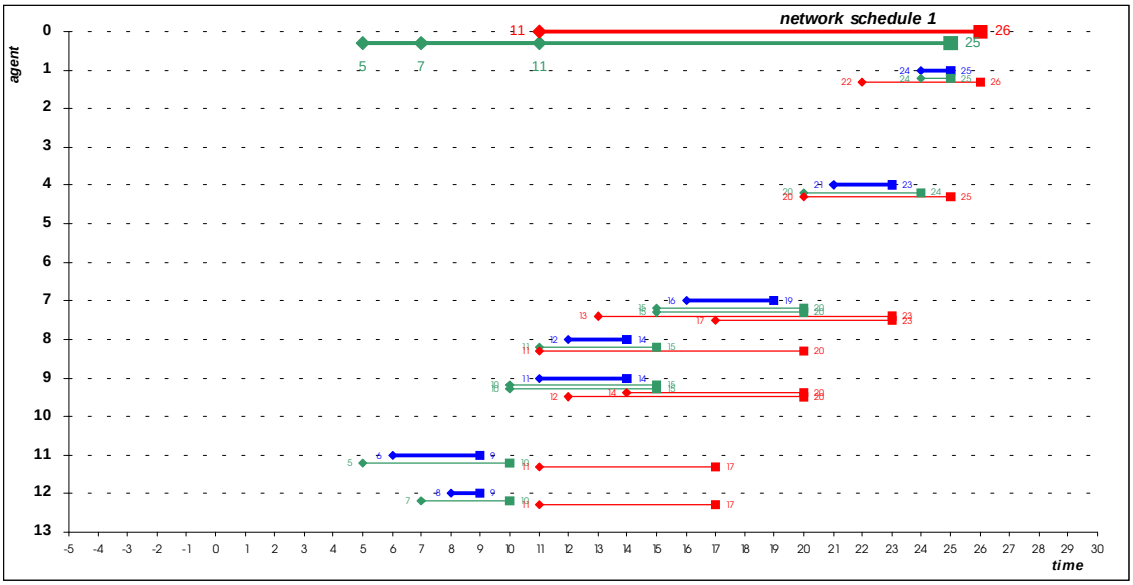


Figura 5-22- Exemplo 0114: resultados da fase 1.

Quadro 3	Re-escalonamento			p
dd = 25	Mensagens trocadas entre agentes			0114
H = <11,26>				
tipo	origem	destino	conteúdo	
pedido-re	g7	g9	rd = 17	FM = -1
pedido-re	g9	g7	dd = 17	FM = -3
pedido-re	g9	g11	rd = 14	FM = -3
pedido-re	g9	g12	rd = 12	FM = -1
pedido-re	g11	g9	dd = 14	FM = -5
pedido-re	g11	g18	rd = 11	FM = -5
pedido-re	g12	g9	dd = 12	FM = -3
pedido-re	g12	g19	rd = 11	FM = -3
re-escalonamento	g18	g0	dd = 11	
re-escalonamento	g19	g0	dd = 11	

Quadro 4	Re-escalonamento													p
dd = 25	Problema de escalonamento													0114
H = <11,26>	de agente gestor								Folgas					
Agente gestor	fornece-dores	cliente	Datas				Tarefa							
			RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	11	11	15	20	12	2	14	1	0	1	1	5	6	7
g11	g18	g9	11	11	14	17	11	3	14	0	0	1	1	3	3	3
g12	g19	g9	11	11	12	17	11	1	12	0	0	1	1	5	5	5
g9	g11	g7	14	14	17	20	14	3	17	0	0	1	1	3	3	3
	g12		12	12						2	0	1				
g7	g8	g4	13	15	20	23	17	3	20	4	2	1	1	3	3	3
	g9		17	17						0	0	1				
g4	g7	g1	20	20	24	25	21	2	23	1	0	1	1	1	2	3
g1	g4	g14	22	24	25	26	24	1	25	2	2	1	1	1	1	3

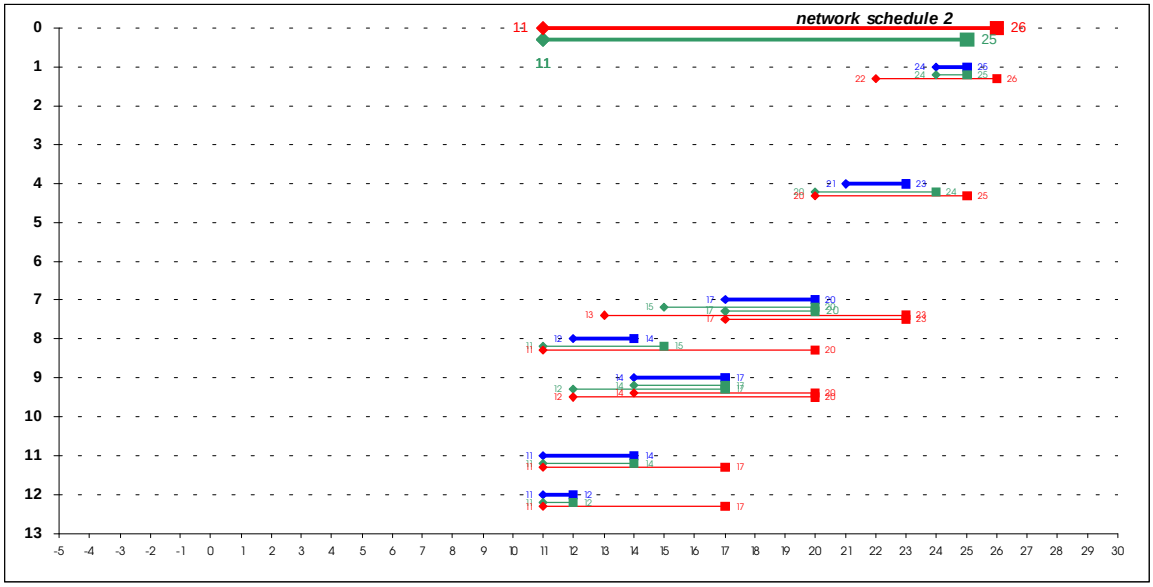
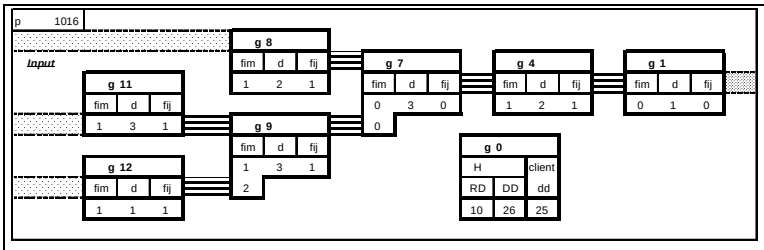


Figura 5-23- Exemplo 0114: resultados da fase 2.



Quadro 1	Estabelecimento de pedidos locais			p
dd = 25	Mensagens trocadas entre agentes			1016
H = <10,26>				
tipo	origem	destino	conteúdo	
pedido-in	exterior		dd = 25	<10,26>
pedido	g0	g14	dd = 25	DD = 26
pedido	g14	g1	dd = 25	FJM / FEJ = 1
pedido	g1	g4	dd = 24	FJM / FEJ = 1
pedido	g4	g7	dd = 20	FJM / FEJ = 3
pedido	g7	g8	dd = 17	FJM / FEJ = 3
pedido	g7	g9	dd = 17	FJM / FEJ = 3
pedido	g9	g11	dd = 12	FJM / FEJ = 5
pedido	g9	g12	dd = 11	FJM / FEJ = 6
pedido	g8	g17	dd = 13	FJM / FEJ = 5
pedido	g11	g18	dd = 7	FJM / FEJ = 7
pedido	g12	g19	dd = 8	FJM / FEJ = 8
pedido	g17	g0	dd = 13	
pedido	g18	g0	dd = 7	
pedido	g19	g0	dd = 8	
pedido-out	g0	exterior	dd = 13	
pedido-out	g0	exterior	dd = 7	
pedido-out	g0	exterior	dd = 8	
aceitação	g0	g17	RD = 10	
aceitação	g0	g18	RD = 10	
aceitação	g0	g19	RD = 10	
aceitação	g17	g8		FMJ / FEM = 3
aceitação	g18	g11		FMJ / FEM = -3
aceitação	g19	g12		FMJ / FEM = -2
aceitação	g11	g9		FMJ / FEM = -1
aceitação	g12	g9		FMJ / FEM = 0
aceitação	g8	g7		FMJ / FEM = 5
aceitação	g9	g7		FMJ / FEM = 1
aceitação	g7	g4		FMJ / FEM = 1
aceitação	g4	g1		FMJ / FEM = 3
aceitação	g1	g14		FMJ / FEM = 3
aceitação	g14	g0		FT = 4

Quadro 2	Estabelecimento de pedidos locais													p
dd = 25	Problema de escalonamento													1016
H = <10,26>	de agente gestor							Folgas						
Agente	fornece-	cliente	Datas				Tarefa							
gestor	dores		RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	10	13	17	20	14	2	16	4	3	1	1	3	4	8
g11	g18	g9	10	7	12	17	8	3	11	-2	-3	1	1	5	6	4
g12	g19	g9	10	8	11	17	9	1	10	-1	-2	1	1	6	7	6
g9	g11	g7	13	12	17	20	13	3	16	0	-1	1	1	3	4	4
	g12		11	11						2	0	2				
g7	g8	g4	12	17	20	23	17	3	20	5	5	0	0	3	3	4
	g9		16	17						1	1	0				
g4	g7	g1	19	20	24	25	21	2	23	2	1	1	1	1	2	4
g1	g4	g14	21	24	25	26	24	1	25	3	3	0	0	1	1	4

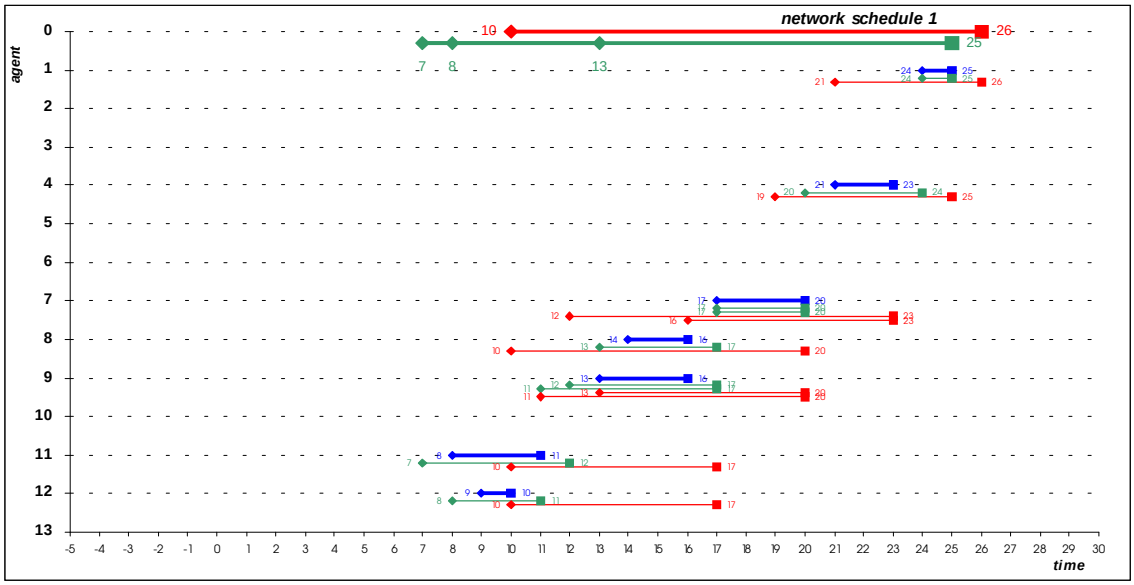


Figura 5-24- Exemplo 1016: resultados da fase 1.

Quadro 3	Re-escalonamento			p
dd = 25	Mensagens trocadas entre agentes			1016
H = <10,26>				
tipo	origem	destino	conteúdo	
pedido-re	g9	g11	rd = 13	FEM = -1
pedido-re	g11	g9	dd = 13	FM = -2
pedido-re	g11	g18	rd = 10	FM = -2
pedido-re	g12	g19	rd = 10	FM = -1
re-escalonamento	g18	g0	dd = 10	
re-escalonamento	g19	g0	dd = 10	

Quadro 4	Re-escalonamento														p
dd = 25	Problema de escalonamento														1016
H = <10,26>	de agente gestor								Folgas						
Agente gestor	fornece-dores	cliente	Datas				Tarefa			FM	FEM	fim	fij	FEJ	FJ	FT
			RD	rd	dd	DD	s	d	e							
g8	g17	g7	10	13	17	20	14	2	16	4	3	1	1	3	4	8
g11	g18	g9	10	10	13	17	10	3	13	0	0	1	1	4	4	4
g12	g19	g9	10	10	11	17	10	1	11	0	0	1	1	6	6	6
g9	g11	g7	13	13	17	20	13	3	16	0	0	1	1	3	4	4
	g12		11	11						2	0	2				
g7	g8	g4	12	17	20	23	17	3	20	5	5	1	1	3	3	4
	g9		16	17						1	1	0				
g4	g7	g1	19	20	24	25	21	2	23	2	1	1	1	1	2	4
g1	g4	g14	21	24	25	26	24	1	25	3	3	1	1	1	1	4

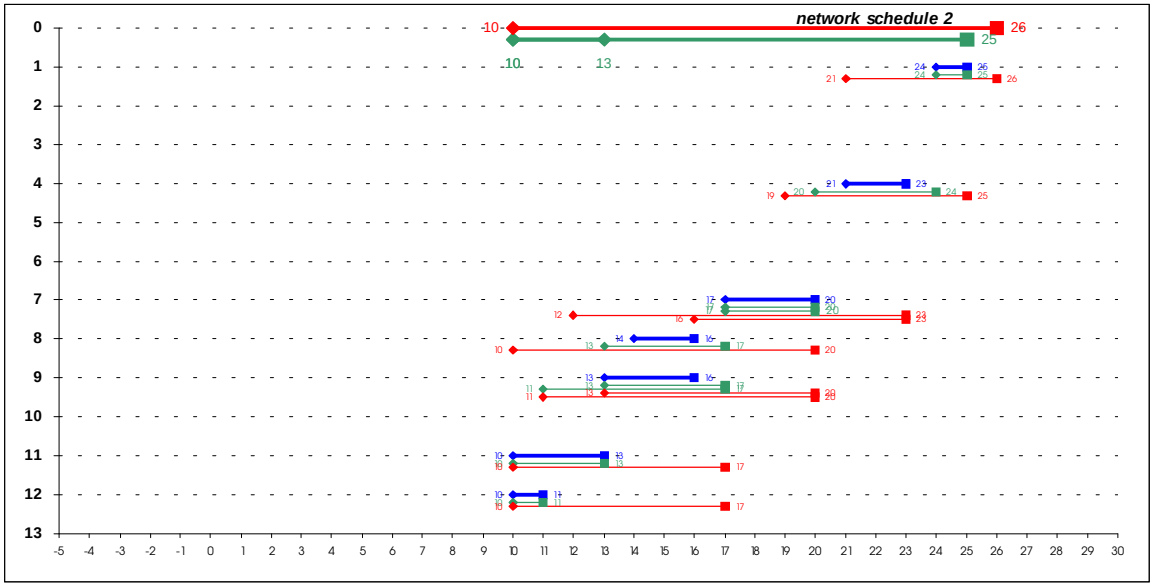
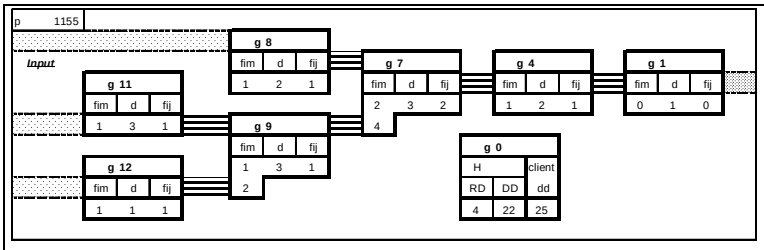


Figura 5-25- Exemplo 1016: resultados da fase 2.



Quadro 1	Estabelecimento de pedidos locais			p
dd = 25	Mensagens trocadas entre agentes			1155
H = <4,22>				
tipo	origem	destino	conteúdo	
pedido-in	exterior		dd = 25	<4,22>
pedido	g0	g14	dd = 25	DD = 22
pedido	g14	g1	dd = 25	FJM / FEJ = -3
pedido	g1	g4	dd = 24	FJM / FEJ = -3
pedido	g4	g7	dd = 20	FJM / FEJ = -1
pedido	g7	g8	dd = 13	FJM / FEJ = 3
pedido	g7	g9	dd = 11	FJM / FEJ = 5
pedido	g9	g11	dd = 6	FJM / FEJ = 7
pedido	g9	g12	dd = 5	FJM / FEJ = 8
pedido	g8	g17	dd = 9	FJM / FEJ = 5
pedido	g11	g18	dd = 1	FJM / FEJ = 9
pedido	g12	g19	dd = 2	FJM / FEJ = 10
pedido	g17	g0	dd = 9	
pedido	g18	g0	dd = 1	
pedido	g19	g0	dd = 2	
pedido-out	g0	exterior	dd = 9	
pedido-out	g0	exterior	dd = 1	
pedido-out	g0	exterior	dd = 2	
aceitação	g0	g17	RD = 4	
aceitação	g0	g18	RD = 4	
aceitação	g0	g19	RD = 4	
aceitação	g17	g8		FMJ / FEM = 5
aceitação	g18	g11		FMJ / FEM = -3
aceitação	g19	g12		FMJ / FEM = -2
aceitação	g11	g9		FMJ / FEM = -1
aceitação	g12	g9		FMJ / FEM = 0
aceitação	g8	g7		FMJ / FEM = 7
aceitação	g9	g7		FMJ / FEM = 1
aceitação	g7	g4		FMJ / FEM = 7
aceitação	g4	g1		FMJ / FEM = 9
aceitação	g1	g14		FMJ / FEM = 9
aceitação	g14	g0		FT = 6

Quadro 2	Estabelecimento de pedidos locais													p
dd = 25	Problema de escalonamento													1155
H = <4,22>	de agente gestor								Folgas					
Agente gestor	fornece-dores	cliente	Datas				Tarefa							
			RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	4	9	13	16	10	2	12	6	5	1	1	3	4	10
g11	g18	g9	4	1	6	13	2	3	5	-2	-3	1	1	7	8	6
g12	g19	g9	4	2	5	13	3	1	4	-1	-2	1	1	8	9	8
g9	g11	g7	7	6	11	16	7	3	10	0	-1	1	1	5	6	6
	g12		5	5						2	0	2				
g7	g8	g4	6	13	20	19	15	3	18	9	7	2	2	-1	1	6
	g9		10	11						5	1	4				
g4	g7	g1	13	20	24	21	21	2	23	8	7	1	1	-3	-2	6
g1	g4	g14	15	24	25	22	24	1	25	9	9	0	0	-3	-3	6

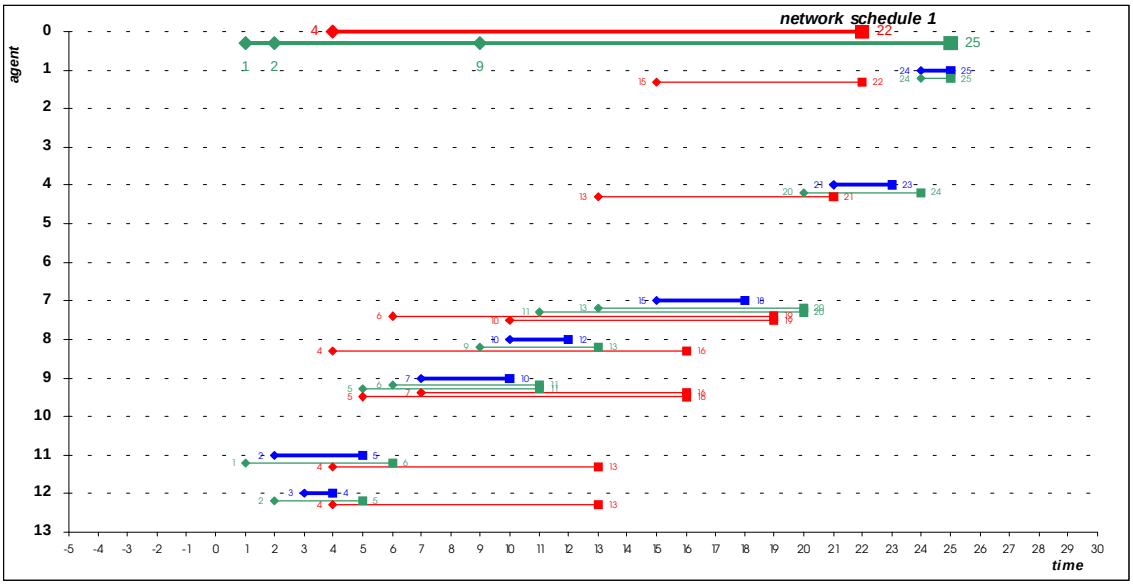


Figura 5-26- Exemplo 1155: resultados da fase 1.

Quadro 3	Re-escalonamento			p
dd = 25	Mensagens trocadas entre agentes			1155
H = <4,22>				
tipo	origem	destino	conteúdo	
re-escalonamento	g14	g0	dd = 22	
pedido-re	g1	g14	dd = 22	FJ = -3
pedido-re	g1	g4	rd = 21	FJ = -3
pedido-re	g4	g1	dd = 21	FJ = -2
pedido-re	g4	g7	rd = 19	FJ = -2
pedido-re	g7	g4	dd = 19	FEJ = -1
pedido-re	g9	g11	rd = 7	FEM = -1
pedido-re	g11	g9	dd = 7	FM = -2
pedido-re	g11	g18	rd = 4	FM = -2
pedido-re	g12	g19	rd = 4	FM = -1
re-escalonamento	g18	g0	dd = 4	
re-escalonamento	g19	g0	dd = 4	

Quadro 4	Re-escalonamento														p
dd = 25	Problema de escalonamento														1155
H = <4,22>	de agente gestor								Folgas						
Agente gestor	fornece-dores	cliente	Datas				Tarefa								
			RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	4	9	13	16	10	2	12	6	5	1	1	3	4	10
g11	g18	g9	4	4	7	13	4	3	7	0	0	1	1	6	6	6
g12	g19	g9	4	4	5	13	4	1	5	0	0	1	1	8	8	8
g9	g11	g7	7	7	11	16	7	3	10	0	0	1	1	5	6	6
	g12		5	5						2	0	2				
g7	g8	g4	6	13	19	19	15	3	18	9	7	1	1	0	1	6
	g9		10	11						5	1	4				
g4	g7	g1	13	19	21	21	19	2	21	6	6	1	1	0	0	6
g1	g4	g14	15	21	22	22	21	1	22	6	6	1	1	0	0	6

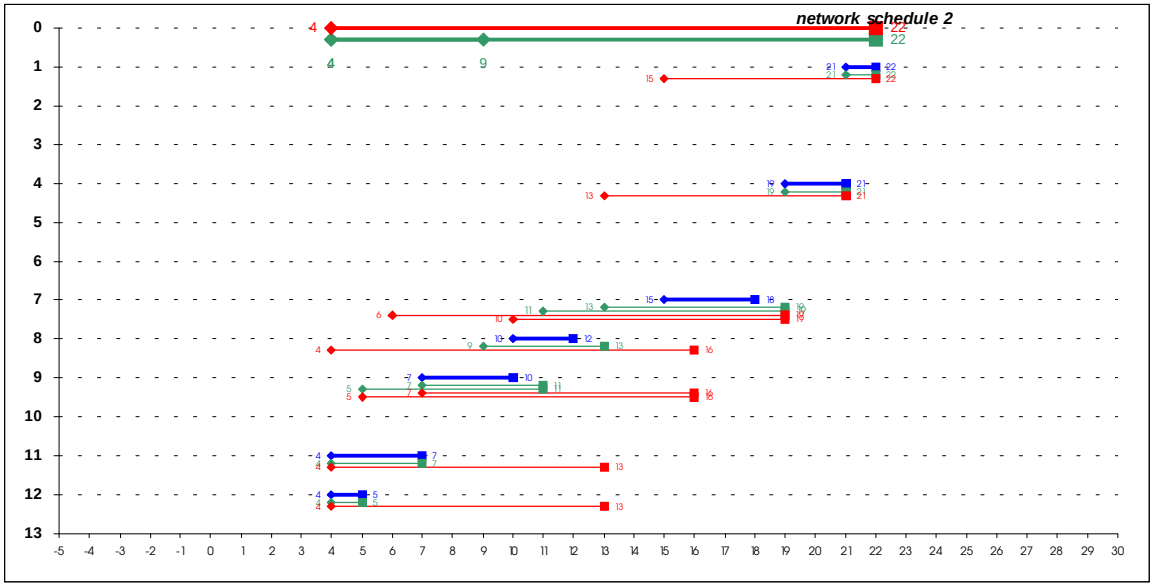
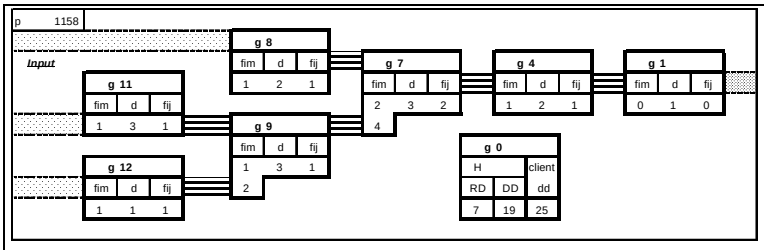


Figura 5-27- Exemplo 1155: resultados da fase 2.



Quadro 1	Estabelecimento de pedidos locais			p
dd = 25	Mensagens trocadas entre agentes			1158
H = <7,19>				
tipo	origem	destino	conteúdo	
pedido-in	exterior		dd = 25	<7,19>
pedido	g0	g14	dd = 25	DD = 19
pedido	g14	g1	dd = 25	FJM / FEJ = -6
pedido	g1	g4	dd = 24	FJM / FEJ = -6
pedido	g4	g7	dd = 20	FJM / FEJ = -4
pedido	g7	g8	dd = 13	FJM / FEJ = 0
pedido	g7	g9	dd = 11	FJM / FEJ = 2
pedido	g9	g11	dd = 6	FJM / FEJ = 4
pedido	g9	g12	dd = 5	FJM / FEJ = 5
pedido	g8	g17	dd = 9	FJM / FEJ = 2
pedido	g11	g18	dd = 1	FJM / FEJ = 6
pedido	g12	g19	dd = 2	FJM / FEJ = 7
pedido	g17	g0	dd = 9	
pedido	g18	g0	dd = 1	
pedido	g19	g0	dd = 2	
pedido-out	g0	exterior	dd = 9	
pedido-out	g0	exterior	dd = 1	
pedido-out	g0	exterior	dd = 2	
aceitação	g0	g17	RD = 7	
aceitação	g0	g18	RD = 7	
aceitação	g0	g19	RD = 7	
aceitação	g17	g8		FMJ / FEM = 2
aceitação	g18	g11		FMJ / FEM = -6
aceitação	g19	g12		FMJ / FEM = -5
aceitação	g11	g9		FMJ / FEM = -4
aceitação	g12	g9		FMJ / FEM = -3
aceitação	g8	g7		FMJ / FEM = 4
aceitação	g9	g7		FMJ / FEM = -2
aceitação	g7	g4		FMJ / FEM = 4
aceitação	g4	g1		FMJ / FEM = 6
aceitação	g1	g14		FMJ / FEM = 6
aceitação	g14	g0		FT = 0

Quadro 2	Estabelecimento de pedidos locais													p
dd = 25	Problema de escalonamento													1158
H = <7,19>	de agente gestor							Folgas						
Agente	fornece-	cliente	Datas				Tarefa							
gestor	dores		RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	7	9	13	13	10	2	12	3	2	1	1	0	1	4
g11	g18	g9	7	1	6	10	2	3	5	-5	-6	1	1	4	5	0
g12	g19	g9	7	2	5	10	3	1	4	-4	-5	1	1	5	6	2
g9	g11	g7	10	6	11	13	7	3	10	-3	-4	1	1	2	3	0
	g12		8	5						-1	-3	2				
g7	g8	g4	9	13	20	16	15	3	18	6	4	2	2	-4	-2	0
	g9		13	11						2	-2	4				
g4	g7	g1	16	20	24	18	21	2	23	5	4	1	1	-6	-5	0
g1	g4	g14	18	24	25	19	24	1	25	6	6	0	0	-6	-6	0

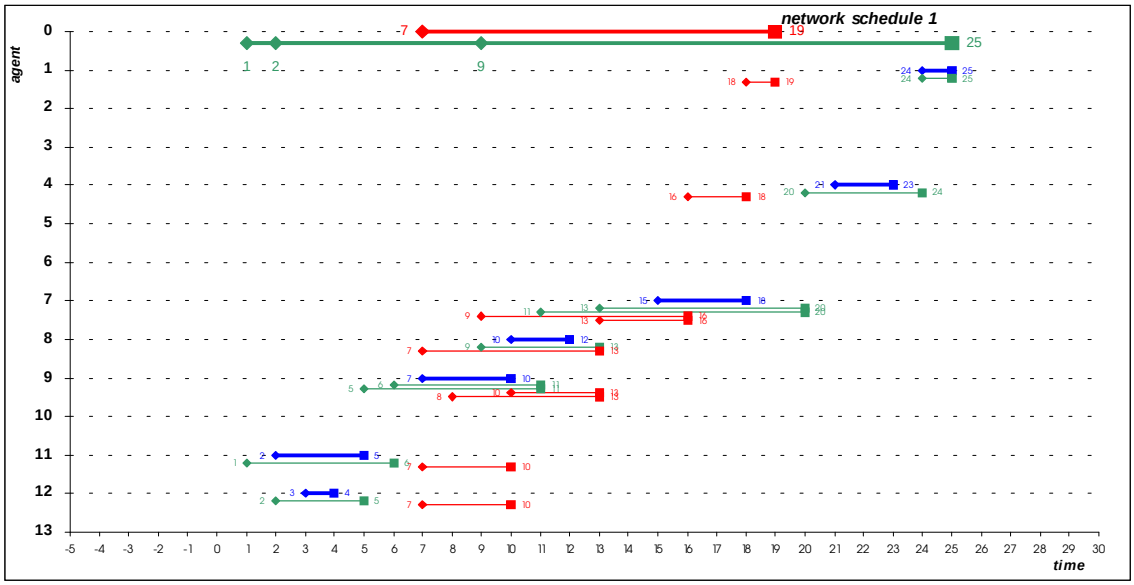


Figura 5-28- Exemplo 1158: resultados da fase 1.

Quadro 3	Re-escalonamento			p
dd = 25	Mensagens trocadas entre agentes			1158
H = <7,19>				
tipo	origem	destino	conteúdo	
re-escalonamento	g14	g0	dd = 19	
pedido-re	g1	g14	dd = 19	FJ = -6
pedido-re	g1	g4	rd = 18	FJ = -6
pedido-re	g4	g1	dd = 18	FJ = -5
pedido-re	g4	g7	rd = 16	FJ = -5
pedido-re	g7	g4	dd = 16	FJ = -2
pedido-re	g7	g9	rd = 13	FEM = -2
pedido-re	g9	g7	dd = 13	FM = -3
pedido-re	g9	g11	rd = 10	FM = -3
pedido-re	g9	g12	rd = 8	FM = -1
pedido-re	g11	g9	dd = 10	FM = -5
pedido-re	g11	g18	rd = 7	FM = -5
pedido-re	g12	g9	dd = 8	FM = -4
pedido-re	g12	g19	rd = 7	FM = -4
re-escalonamento	g18	g0	dd = 7	
re-escalonamento	g19	g0	dd = 7	

Quadro 4	Re-escalonamento														p
dd = 25	Problema de escalonamento														1158
H = <7,19>	de agente gestor								Folgas						
Agente	fornece-	cliente	Datas				Tarefa								
gestor	dores		RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	7	9	13	13	10	2	12	3	2	1	1	0	1	4
g11	g18	g9	7	7	10	10	7	3	10	0	0	1	1	0	0	0
g12	g19	g9	7	7	8	10	7	1	8	0	0	1	1	2	2	2
g9	g11	g7	10	10	13	13	10	3	13	0	0	1	1	0	0	0
	g12		8	8						2	0	2				
g7	g8	g4	9	13	16	16	13	3	16	4	4	1	1	0	0	0
	g9		13	13						0	0	4				
g4	g7	g1	16	16	18	18	16	2	18	0	0	1	1	0	0	0
g1	g4	g14	18	18	19	19	18	1	19	0	0	1	1	0	0	0

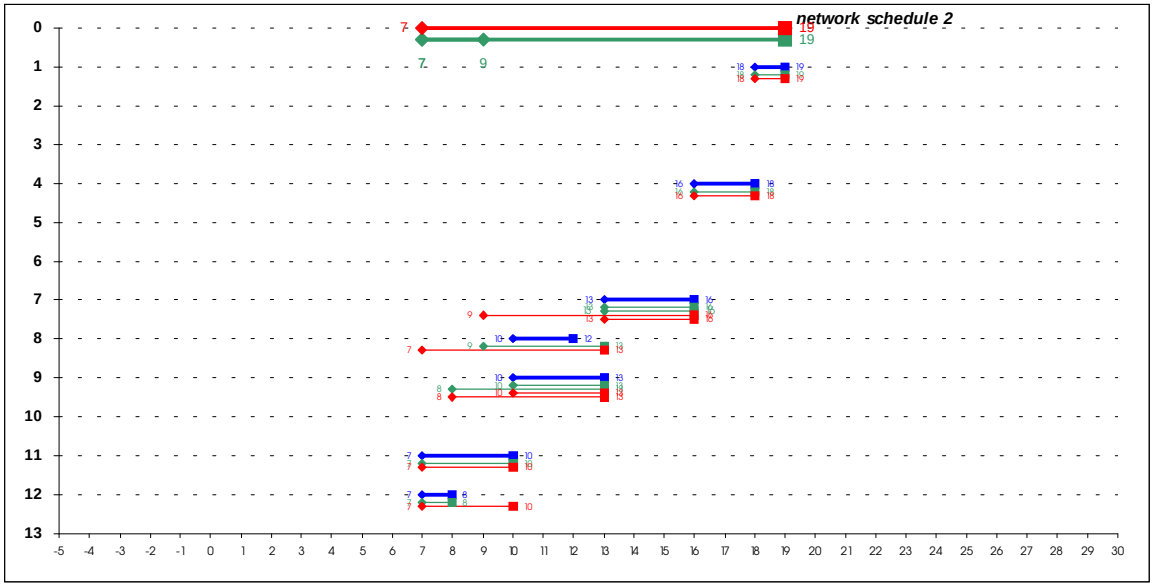


Figura 5-29- Exemplo 1158: resultados da fase 2.

6 Conclusões e Trabalho Futuro

Neste capítulo apresentam-se conclusões e contribuições do presente trabalho e descreve-se o trabalho futuro a desenvolver.

6.1 Conclusões e Contribuições

O objectivo geral que norteou o presente trabalho foi o de abordar o problema de escalonamento numa perspectiva que vai além da do escalonamento convencional, em que as decisões de escalonamento são tomadas de forma centralizada, e encarar o escalonamento como uma actividade em que a decisão é distribuída por vários agentes, num contexto multi-agente e cooperativo. Deste modo, nos objectivos gerais incluem-se, também, a modelação dos agentes decisores de escalonamento e da comunicação necessária para coordenação da actividade de escalonamento, num contexto de escalonamento multi-agente cooperativo. O trabalho desenvolvido pode ser, de um modo muito geral, resumido nas seguintes palavras chave:

- Escalonamento multi-agente, escalonamento cooperativo;
- Comunicação inter-agente;
- Gestão da cadeia de fornecimento, Empresa Estendida.

Na perspectiva típica do escalonamento convencional, existe um conjunto de recursos instalados cuja capacidade deve ser afectada ao longo do tempo a tarefas com várias necessidades de recursos; a afectação dos recursos às tarefas deve respeitar um conjunto de restrições temporais de precedência e de datas limite sobre as tarefas e de restrições de capacidade dos recursos. No trabalho desenvolvido estes aspectos são enquadrados numa perspectiva multi-agente. Como a tomada de decisão é distribuída por mais de um agente, o que não acontece no escalonamento convencional, há a necessidade de coordenar as decisões dos vários agentes na resolução de problemas e, para isso, tem existir comunicação entre os agentes. Adicionalmente, aplicou-se a perspectiva multi-agente à gestão da cadeia de fornecimento (*supply chain*), usando o paradigma da Empresa Estendida (EE). Os seguintes objectivos gerais foram adoptados:

1. Abordar o problema de escalonamento numa perspectiva que envolve a comunicação entre agentes de escalonamento, num contexto de escalonamento multi-agente cooperativo;
2. Desenvolver um modelo de escalonamento, de acordo com a perspectiva em 1, para escalonamento de tarefas logísticas de produção e distribuição, aplicável num contexto de gestão da cadeia de fornecimento e segundo a perspectiva do paradigma da Empresa Estendida de gestão integrada da cadeia de fornecimento.

Os seguintes objectivos específicos foram adoptados:

1. Modelar os aspectos interessantes para o escalonamento convencional, nomeadamente tempo, capacidade, recursos e tarefas, enquadrados no contexto de rede de produção e distribuição;
2. Modelar os aspectos interessantes do escalonamento multi-agente e da comunicação inter-agente para coordenação da actividade de escalonamento multi-agente no contexto da gestão da cadeia de fornecimento, nomeadamente agentes intervenientes, informação trocada entre os agentes e protocolo de interacção entre eles;
3. Desenvolver um sistema computacional no qual os aspectos dos objectivos específicos 1 e 2 possam ser representados e testados e baseado no qual seja possível desenvolver trabalho futuro, dando continuidade ao trabalho actual.

O trabalho desenvolvido consistiu na elaboração de um modelo de escalonamento multi-agente e uma realização computacional da maior parte do modelo. Relativamente ao modelo propôs-se uma organização do mesmo em dois níveis:

- O *nível físico* que modela a parte física do ambiente de escalonamento, dizendo respeito a tempo, quantidades, produtos, recursos e capacidade, redes de recursos e relações cliente-fornecedor entre recursos, tarefas e escalonamento de tarefas nos recursos, fluxos de produtos que podem ocorrer entre os recursos de uma rede, processos de rede;
- O *nível virtual* que modela a parte não física do ambiente de escalonamento, dizendo respeito aos agentes que gerem as capacidades dos recursos de uma rede e à interacção entre os agentes quando na presença de problemas de escalonamento multi-agente para os quais uma solução deve ser encontrada.

Os objectivos específicos 1 e 2 foram satisfeitos pela elaboração dos níveis físico e virtual do modelo, descritos nos capítulos 3 e 4. O objectivo específico 3 apenas é parcialmente satisfeito já que a realização computacional do modelo está, na data de conclusão do trabalho, incompleta. Apesar de incompleta, esta realização computacional é referida por ter ajudado a refinar, testar e consolidar ideias durante o desenvolvimento do modelo. Para a demonstração da validade de um aspecto importante do modelo — um método proposto para a resolução de problemas de escalonamento multi-agente cooperativo, que inclui um mecanismo de coordenação inter-agente — recorreu-se à simulação de casos exemplificativos (Capítulo 5).

A seguir salientam-se os aspectos mais relevantes do trabalho, que se entendem ser contribuições do mesmo.

6.1.1 O Nível Físico

No nível físico são modelados os componentes físicos de escalonamento multi-agente. Os aspectos básicos relevantes do nível físico do modelo são:

- Acontecimentos logísticos são representados por eventos, que associam pares produto-quantidade, um valor de tempo e um local (identificador de nó de rede). Os eventos servem basicamente para definir tarefas e caracterizam o que acontece no início ou no fim de uma tarefa escalonada num nó de rede (*i.e.*, entrega ou consumo de uma quantidade de um produto num instante, num determinado nó). A representação sugerida para o tempo é discreta e baseada predominantemente em pontos temporais (instantes são número inteiros).

O componentes principais do nível físico do modelo são a rede de produção e distribuição e tarefas. A rede de produção e distribuição inclui nós, que representam os recursos com capacidades limitadas da rede e arcos, que representam as relações de cliente-fornecedor entre os nós e, portanto, os possíveis fluxos físicos de produtos entre os nós. Nós podem classificar-se em nós de capacidade, nós de retalho e nós de matéria-prima. Estas duas últimas classes de nós são artificiais e apenas servem para representar uma fronteira física da rede com o exterior; numa instância de modelo apenas os nós de capacidade correspondem a recursos reais.

As características mais relevantes dos nós de capacidade do modelo são:

- Terem um conjunto de produtos de saída, que podem entregar (são nós multi-produto, em geral) e um conjunto de produtos de entrada, que podem consumir; a entrega e o consumo estão associados a tarefas do nó. Para cada nó individual existe uma relação entre estes dois

conjuntos, fixa no momento de criação do nó, que estabelece que produtos e que quantidades de produtos são necessárias para que o nó possa entregar cada unidade de determinado produto de saída;

- Terem capacidade, afectável a tarefas a escalonar, limitada, que pode ser partilhada por tarefas que entregam produtos diferentes (quando o nó é multi-produto). A capacidade é classificada em capacidade de acumulação, definida com base numa quantidade de produto, e capacidade de processamento, definida com base numa taxa de processamento, equivalente a uma quantidade de produto por unidade de tempo. A quantidade de capacidade necessária para o escalonamento de uma tarefa num nó (constante durante o intervalo de tempo da tarefa) depende do nó e do produto de saída da tarefa, sendo proporcional à quantidade de produto. As constantes de proporcionalidade entre capacidade e quantidade de cada produto de saída são fixas no momento de criação do nó. A capacidade máxima e a capacidade utilizada em cada produto de saída ao longo do tempo são representadas através de perfis de capacidade do nó (perfil de capacidade máxima e perfis de capacidade utilizada, respectivamente). O perfil de capacidade máxima é fixo no momento de criação do nó e não muda; os perfis de capacidade utilizada mudam conforme a capacidade é afectada ou desafectada a tarefas e representam o estado de capacidade do nó. As condições em que, em qualquer instante, a soma da capacidade utilizada aplicada em tarefas escalonadas excede a capacidade máxima traduzem violação de restrições de capacidade e são denominadas conflitos de capacidade;
- Os nós de capacidade são classificados em acumuladores, que dispõem de capacidade de acumulação, e processadores, que dispõem de capacidade de processamento. Os processadores classificam-se ainda em produtores e transportadores. Nos acumuladores, nos produtores e nos transportadores são escalonáveis tarefas de acumulação, de produção e de transporte, respectivamente. Para efeitos de escalonamento de tarefas nos acumuladores e nos transportadores, a cada produto de saída corresponde o mesmo produto de entrada; nos produtores cada produto de saída fica disponível à custa do consumo de vários (um, ou mais) produtos de entrada, em geral.

As tarefas são criadas para serem escalonadas em nós de rede. Podem existir tarefas de capacidade, tarefas de retalho e tarefas de matéria-prima, que se destinam a escalonamento em nós de capacidade, nós de retalho e nós de matéria-prima, respectivamente. As duas últimas classes de tarefas são artificiais e apenas servem para representar fronteiras temporais de processos de rede; numa instância de modelo apenas as tarefas de capacidade correspondem a tarefas reais, representando actividades logísticas, nomeadamente de produção, armazenamento e transporte. Tarefas de capacidade são classificadas em tarefas de acumulação, de produção e de transporte. Ao ser criada, uma tarefa herda certas características do nó a que se destina para escalonamento como o produto de saída, o produto ou produtos de entrada e a razão entre a quantidade de cada produto de entrada e o produto de saída.

As características mais relevantes das tarefas de capacidade são:

- Uma tarefa é criada para ser escalonada num determinado nó de capacidade, com a finalidade de entregar um produto de saída do nó, consumindo certos produtos de entrada do nó em quantidades preestabelecidas no nó e utilizando capacidade do nó durante o intervalo de tempo em que é escalonada;
- Uma tarefa pode ser decomposta em sub-tarefas. As sub-tarefas tornam disponível o mesmo produto de saída e consomem os mesmos produtos de entrada (e nas mesmas proporções)

que a tarefa mãe e têm intervalos temporalmente contidos no intervalo desta. A decomposição de tarefas, ao permitir escalonar uma tarefa em intervalos distribuídos ao longo do tempo, pode servir para uniformizar o nível de utilização da capacidade (melhor distribuição da capacidade utilizada no tempo, maior flexibilidade da capacidade para acomodar escalonamento de tarefas futuras) ou, por coordenação com tarefas clientes e fornecedoras, para reduzir o tempo de fluxo dos processos de rede.

6.1.2 O Nível Virtual

O nível virtual é o nível da decisão. Neste nível modelam-se os agentes intervenientes na actividade de escalonamento e os fluxos de informação necessários à coordenação desta actividade. Neste nível, que pressupõe o nível físico, definiram-se pedidos e agentes. Adicionalmente, foi definida rede de EE, que engloba o nível físico e o nível virtual.

Numa rede de EE existem, no nível virtual, agentes de rede, cada um associado a um nó de rede do nível físico. Os agentes de rede classificam-se em agentes gestores (ou de capacidade), agentes de retalho e agentes de matéria-prima, que têm associados nós de capacidade, nós de retalho e nós de matéria-prima, respectivamente. Os agentes gestores podem ainda ser classificados de acordo com os nós que gerem em agentes acumuladores e agentes processadores e estes últimos em agentes produtores ou agentes transportadores. No nível virtual, as relações de cliente-fornecedor entre os nós do nível físico são estendidas aos agentes. Para além dos agentes de rede existe um agente adicional, o agente supervisor, que serve de interface com o exterior e tem a função de introduzir trabalho na rede de EE, *i.e.*, novos problemas de escalonamento multi-agente.

Os aspectos mais relevantes no nível virtual do modelo são:

- A definição de uma arquitectura de agente apropriada à actividade de escalonamento dinâmico multi-agente e à comunicação entre agentes para coordenação daquela actividade. Esta arquitectura acomoda representações internas actualizáveis de cada problema de escalonamento posto ao agente, do estado de comunicação entre o agente e cada um dos agentes com que ele comunica e, para agentes gestores, do estado de capacidade do nó gerido;
- A coordenação da actividade de escalonamento multi-agente assenta na interacção por comunicação de mensagens entre agentes. Protocolos de interacção foram definidos para as classes de agentes comunicantes do modelo, com base em sequências padronizadas de mensagens (conversações) a trocar entre um par de agentes e dizendo respeito a um mesmo problema de escalonamento multi-agente que envolva ambos os agentes. Estes protocolos acomodam o estabelecimento de problemas de escalonamento novos e a modificação, desistência ou terminação de problemas de escalonamentos preexistentes por parte dos agentes;
- A unidade de troca de informação relevante para escalonamento (e re-escalonamento) multi-agente é o pedido. Pares de agentes gestores cliente-fornecedor transmitem entre si a informação associada a novos problemas de escalonamento, através da transmissão de mensagens contendo pedidos novos, do agente cliente para o agente fornecedor. O tratamento de um pedido novo recebido de um cliente, por um agente gestor, envolve emissão de um pedido ou pedidos novos a fornecedores (os necessários para satisfazer as necessidades de fornecimento de uma tarefa) e, se todos estes forem aceites e se houver tempo e capacidade disponível para a execução da tarefa, aceitar o pedido do cliente, criar

internamente a tarefa e o problema de escalonamento correspondentes e escalonar a tarefa. Do ponto de vista de estrutura de informação, pedidos são eventos;

- O modelo contempla re-escalonamento (modificação de uma solução de escalonamento) multi-agente. Quando o estabelecimento de um novo problema de escalonamento tem sucesso, todos os pedidos trocados entre os agentes gestores envolvidos juntamente com as tarefas escalonadas por eles constituem o escalonamento multi-agente resultante. Este escalonamento é uma solução inicialmente proposta para o problema e pode conter conflitos temporais, ou conflitos de capacidade. Para remover conflitos detectados, os agentes recorrem dinamicamente a ajustamentos locais do escalonamento, nomeadamente re-escalonamento das suas tarefas e/ou re-escalonamento dos pedidos inter-agente através de comunicação de mensagens com propostas de re-escalonamento. O re-escalonamento é útil tanto em situações de conflito, como em situações em que há que ajustar o escalonamento actual das tarefas dos agentes de modo a acomodar novas tarefas de problemas de escalonamento novos. É portanto, uma característica adequada a escalonamento dinâmico e reactivo;
- O modelo contempla a desistência relativamente a um problema de escalonamento multi-agente. Durante o estabelecimento de um problema de escalonamento novo, quando um agente gestor decide rejeitar um pedido de cliente, seja porque alguns dos pedidos de fornecimento foram rejeitados, ou porque (tendo os pedidos de fornecimento sido todos aceites) o novo problema de escalonamento origina um conflito temporal fatal (irresolúvel), o agente recorre ao cancelamento dos pedidos de fornecimento aceites. Outra situação é a do cancelamento dos pedidos de um problema de escalonamento estabelecido (bastante mais drástica por originar, como consequência, o cancelamento de um pedido global do exterior) devido a capacidade insuficiente. Num ambiente de escalonamento dinâmico, pode sempre acontecer que um agente gestor passe a ter conflitos de capacidade em problemas de escalonamento, inexistentes no momento do estabelecimento dos problemas, que não possam ser removidos recorrendo apenas ao re-escalonamento. Quando não consegue eliminar estes conflitos, um agente gestor pode sempre recorrer ao cancelamento para um ou alguns desses problemas e, possivelmente, ao re-escalonamento para outros. As decisões sobre a que problemas aplicar cancelamento e/ou re-escalonamento poderão depender de preferências do agente, mas se o tempo actual é importante, essas decisões serão tanto mais imperativas para um problema quanto a proximidade entre o tempo actual e as datas do problema (escalonamento reactivo).

Adicionalmente, foram propostos:

- Uma taxonomia e uma caracterização dos conflitos temporais possíveis em problemas de escalonamento multi-agente;
- Um conjunto de regras, para uso dos agentes gestores, para detecção e identificação de conflitos temporais relativos. Estas regras foram apelidadas de regras 0, 1, 2, 3 e 4 e fazem uso de informação sobre folgas temporais partilhada entre os agentes. A regra 0 detecta conflitos temporais fatais e as restantes detectam conflitos temporais não fatais (resolúveis por meio de re-escalonamento). A regra 0 permite detectar, localmente, um conflito global total, o que revela que o problema é temporalmente super-constrangido. As regras 1 e 2 servem para detectar conflitos globais, respectivamente a jusante e a montante, e as regras 3 e 4 servem para detectar conflitos locais externos, respectivamente a jusante e a montante;
- Um conjunto de tratamentos minimais diferenciados para cada tipo de conflito temporal detectado pelas regras acima referidas, para agentes gestores processadores e para agentes

gestores acumuladores. Estes tratamentos foram, para os conflitos detectados pelas regras 1, 2, 3 e 4, respectivamente apelidados de casos 1, 2, 3 e 4 de tratamento de conflitos temporais. Para um conflito detectado pela regra 0 o tratamento consiste em o agente desistir do problema de escalonamento (deste modo, o conflito desaparece); os tratamentos dos casos 1 e 2 envolvem o re-escalonamento da tarefa e de um pedido, ou pedidos, do problema de escalonamento do agente; os tratamentos dos casos 3 e 4 envolvem apenas o re-escalonamento de um pedido, ou pedidos, do problema de escalonamento do agente;

- Um mecanismo de coordenação para escalonamento temporal, baseado em restrições temporais relativas (folgas temporais), que permite coordenar a actividade de escalonamento e re-escalonamento multi-agente. Este mecanismo opera, para cada problema novo, por propagação de informação de folgas temporais a jusante e a montante dos agentes, para trás e para a frente na rede, respectivamente, servindo-se das mensagens do protocolo de interacção entre agentes. A propagação destas folgas corresponde a uma partilha de informação entre os agentes, informação esta que não se pode considerar privada relativamente a algum agente em particular (mantêm-se a privacidade da informação dos agentes). O mecanismo permite que cada agente gestor disponha da informação necessária para desistir de problemas de escalonamento, quando estes são temporalmente super-constrangidos, o que é detectável através da regra 0 (acima referida). A desistência evita que os agentes se envolvam em buscas inúteis de soluções inexistentes. Adicionalmente, o mecanismo permite que os agentes mantenham informação sobre limites temporais de escalonamento (para a tarefa, para o pedido do cliente e para os pedidos aos fornecedores) de um problema estabelecido. Em caso de ser necessário re-escalonamento, o que é detectável pelas regras 1, 2, 3 e 4 (acima referidas), esta informação permite que os agentes ponham de parte valores temporais (para a tarefa, para o pedido do cliente ou para os pedidos aos fornecedores) que inviabilizam soluções possíveis;
- Um método de resolução cooperativa de problemas de escalonamento multi-agente. Neste método são enquadrados o mecanismo de coordenação para escalonamento temporal, as regras para detecção e identificação de conflitos temporais e os tratamentos associados a cada caso, acima referidos. O método prescreve um comportamento individual cooperativo para os agentes gestores, com três fases de resolução de problema de escalonamento apelidadas de fases 1, 2 e 3. Na fase 1 o agente determina se o problema é ou não temporalmente super-constrangido, através da regra 0. Se o problema é temporalmente super-constrangido, o agente desiste do problema, rejeitando o pedido do cliente e cancelando os pedidos aos fornecedores; caso contrário, estabelece localmente uma solução inicial. Na fase 2, se a solução inicial tiver conflitos temporais, estes são eliminados recorrendo ao re-escalonamento. A detecção e identificação de conflitos temporais é realizada através das regras 1, 2, 3 e 4 e sua eliminação realizada recorrendo aos tratamentos apelidados de casos de tratamento 1, 2, 3 e 4, respectivamente. Na fase 3, o agente deve eliminar os conflitos de capacidade que subsistam na solução, sem originar conflitos temporais, recorrendo ao re-escalonamento e, em último lugar, se necessário, recorrendo à desistência de um ou mais problemas de escalonamento (através do cancelamento do pedido do cliente e de todos os pedidos a fornecedores do problema).

6.1.3 Realização Computacional

Um sistema que realiza computacionalmente o modelo tem vindo a ser desenvolvido. Para este sistema, as classes de objectos descritas no Capítulo 3 e no Capítulo 4, juntamente com a maior parte das operações descritas, foram realizadas. O sistema é realizado na linguagem CLOS (Common Lisp Object System [Steele 1990]) e permite construir redes físicas, construir uma rede de EE a partir de uma rede física e deverá permitir fazer simulações de escalonamento multi-agente tendo em conta restrições temporais (relativas e absolutas) e restrições de capacidade. CLOS é uma extensão de Common Lisp, uma linguagem de programação de alto nível flexível e extensível que permite várias formas de abstracção e a construção rápida de programas (ver, por exemplo, [Norvig 1992], [Graham 1994], [Graham 1996]). As razões da opção pelo uso de CLOS tiveram a ver com o facto de ser uma linguagem orientada por objectos (permitindo a modularização de programas, a reutilização de componentes de programa, herança e polimorfismo) que retém todas as vantagens de Common Lisp, a experiência prévia de programação na linguagem, a disponibilidade de um sistema de desenvolvimento com uma realização completa do padrão aceite da linguagem e com a possibilidade de multi-programação.

O sistema referido está, à data de conclusão deste trabalho, inacabado (por esta razão se recorreu à simulação de casos restritos com folha de cálculo no Capítulo 5), carecendo de realização de algumas operações, refinamentos e testes de integração de componentes. No entanto, ele é aqui referido pela razão de que o seu desenvolvimento ajudou, de forma prática, a concretizar certos aspectos do modelo.

Entre aqueles aspectos referem-se, em particular, os de representação de capacidade limitada e de escalonamento de tarefas nos nós de capacidade. A realização computacional destes aspectos, descritos na secção 3.4.1 do Capítulo 3 (incluindo os aspectos descritos na secção 3.4.1.12 do mesmo capítulo), foi efectuada e é detalhada em [Reis 1998], onde se descreve uma representação compacta de perfis de capacidade e algoritmos para as operações de carga e descarga de tarefas em perfis de capacidade, que mantêm o perfil isento de fragmentação.

Outros aspectos que a realização computacional ajudou a concretizar têm a ver com a representação de redes físicas e construção de redes físicas bem-formadas (de acordo com o pressuposto 3 na secção 4.3.4), a decomposição de tarefas, a geração dos agentes de uma rede de EE, dada uma rede física e os protocolos de interacção de entre agentes e o processamento de conversações e de mensagens.

6.2 Trabalho Futuro

Indica-se, a seguir, quais os aspectos que se pretendem desenvolver como trabalho futuro.

6.2.1 Realização Computacional e Experimentação

Uma vez que se conclua a realização computacional será possível realizar simulações mais gerais do que as apresentadas no Capítulo 5, usando modelos de redes de EE diversas e tendo em conta todas restrições temporais e de capacidade. Cenários de actividade de escalonamento essenciais para a realização de simulações com modelos de redes de EE dados serão:

Escalonamento estático - Os pedidos são introduzidos na rede de EE em lote (no mesmo instante de simulação) e o tempo actual (simulado) é considerado irrelevante. Este é um cenário de escalonamento em *off-line*, em que se poderá investir em experimentação com comportamentos de escalonamento dos agentes gestores, diferentes ou não de agente para agente, mais orientados para optimização das soluções de escalonamento (com ou sem re-escalonamento) dado que, durante a resolução de problemas (a partir do momento inicial de introdução dos pedidos), todos os problemas de escalonamento são conhecidos.

Escalonamento dinâmico - Os pedidos são introduzidos na rede de EE ao longo do tempo, o tempo actual é considerado irrelevante, mas a ordem de introdução de pedidos diferentes é relevante (possivelmente originando soluções de escalonamento multi-agente diferentes para ordens diferentes). Este cenário foi o mais sugerido no presente trabalho. Em princípio, os agentes recorrerão intensivamente ao re-escalonamento, já que, para acomodar soluções para problemas novos, soluções para problemas anteriores poderão ter de ser alteradas. Neste cenário deverá investir-se em experimentação com comportamentos de escalonamento dos agentes gestores, diferentes ou não de agente para agente, empregando métodos heurísticos apropriados ao escalonamento dinâmico como os divulgados na literatura de escalonamento,[1] adaptados a um contexto multi-agente.

Escalonamento reactivo - Os pedidos são introduzidos na rede em lote ou ao longo do tempo (como descrito acima), mas o tempo actual é considerado relevante. Este é o cenário mais geral (e mais realista) possível, em que as decisões de escalonamento dos agentes são afectadas por restrições temporais absolutas (tempo actual simulado) e devem acomodar escalonamentos com tarefas em execução. Prevê-se o uso intensivo de re-escalonamento, cancelamento e rejeição de pedidos pelos agentes, principalmente em situações em que os agentes já se encontram com pouco tempo e/ou capacidade disponível. O "congelamento" de problemas de escalonamento, por imposição de limites temporais a partir dos quais não é possível re-escalonar ou cancelar pedidos de um problema, poderá ser adequado para controlar a instabilidade das soluções actuais (aspecto referido na secção 4.4.8.4).

Adicionalmente, incluem-se também no trabalho futuro, a simulação e o estudo de casos em que a decomposição de tarefas (*task splitting*) e a agregação de tarefas (*task batching*) são empregues (aspectos que foram excluídos no pressuposto 2 da secção 4.3.4) e em que existem preferências diversas de escalonamento dos agentes (no extremo agentes não cooperantes com preferências conflituosas).

A decomposição de tarefas permite modelar tarefas cuja execução é distribuída em vários intervalos ao longo do tempo, por exemplo, para nivelar no tempo a capacidade disponível. A agregação de tarefas permite modelar conjuntos de tarefas que estão sujeitas às mesmas restrições (nomeadamente restrições temporais) e que são representadas por uma mesma tarefa agregada, ou modelar tarefas com mais de um produto de saída.

Um comportamento de escalonamento com preferências é realizado incluindo, num agente, regras ou procedimentos de escalonamento com determinadas tendências na afectação de capacidade às tarefas (por exemplo, no caso dos agentes processadores, afectar capacidade disponível o mais cedo possível, ou o mais tarde possível, ou decompor uma tarefa em sub-tarefas ao longo do tempo, empregar maior ou menor quantidade de capacidade) e nas datas de pedidos (extensão maior ou menor das folgas internas a jusante e a montante).

---

[1] Ver, por exemplo, [Morton 1993].

Assumiu-se a existência de nós acumuladores implícitos, com capacidade de acumulação infinita — portanto, sem restrições de capacidade — colocados a jusante e a montante de cada nó processador (pressuposto 1 da secção 4.3.4). Em rigor, estes nós acumuladores deveriam ser representados explicitamente, com capacidade de acumulação limitada gerida pelo agente processador, passando a considerar-se as respectivas restrições de capacidade. A realização deste aspecto permite a experimentação com cenários de escalonamento em que o armazenamento temporário é limitado, restringindo soluções possíveis de escalonamento (por exemplo, cenários de escalonamento *just-in-time*, em que se preferem manter níveis baixos de existências).

Outros aspectos ainda a realizar são os de reservas de capacidade e pedidos complexos (ver as secções 3.4.1.11 e 4.3.3, respectivamente).

6.2.2 Mecanismos de Coordenação para Escalonamento Multi-Agente

No que respeita à coordenação da actividade de escalonamento e re-escalonamento multi-agente numa rede de EE, há que desenvolver um mecanismo adicional de coordenação baseado na capacidade. A coordenação da actividade de escalonamento e re-escalonamento multi-agente deverá então, basear-se em ambos os mecanismos.

Este mecanismo tomará em conta a informação sobre a capacidade disponível nos nós em cada momento, durante a resolução de problemas.[2] Esta informação deverá ser, de algum modo, partilhada entre os agentes e deverá influenciar as decisões de escalonamento deles.

Formas de coordenação baseadas na capacidade deverão apoiar uma actividade de escalonamento e re-escalonamento baseada na perspectiva dos congestionamentos (por exemplo, com atribuição de maiores prioridades aos pedidos de re-escalonamento de agentes com capacidade disponível mais restringida). As expectativas são as de ter espaços de soluções possíveis a explorar de menor dimensão (mais ainda do que empregando apenas o mecanismo puramente temporal) e conseguir mais cedo, na resolução de problemas, evidenciar os aspectos mais críticos para a solução final, permitindo estratégias oportunistas de resolução de problemas de escalonamento.

6.2.3 Estruturas de Rede Alternativas, Negociação entre Agentes e com o Exterior

Ampliações possíveis do modelo, a explorar, são as que acomodam estruturas alternativas de rede física contendo nós com múltiplos nós fornecedores, directos ou indirectos, para um mesmo nó (excluídas nos pressupostos 3 e 5 da secção 4.3.4).

Estruturas de rede alternativas em que existem múltiplos fornecedores directos do mesmo produto de entrada para um nó permitem a existência de caminhos alternativos nos processos de rede (o que corresponde, no escalonamento convencional, à possibilidade de existirem recursos alternativos para execução das mesmas tarefas). Estas estruturas permitem redes de EE em que as soluções de escalonamento podem envolver formas primitivas de negociação entre os agentes, a explorar. Por exemplo, um agente faz pedidos de fornecimento a

[2] Ver as operações `C-MAX-TOTAL`, `C-UTIL-TOTAL` e `C-UTIL-FRACÇÃO`, na secção 3.4.1.14.

fornecedores alternativos e, dos que forem aceites, escolhe o que mais satisfaz as suas preferências, cancelando os restantes; ou em caso de cancelamento de um pedido de fornecimento por um fornecedor, fazer um pedido a um fornecedor alternativo (em vez de cancelar o pedido do cliente), sendo isso possível.

Outro tipo de estruturas de rede alternativas são as que permitem a possibilidade de existir mais de uma tarefa para o mesmo agente no mesmo processo de rede (o agente recebe vários pedidos locais distintos que correspondem ao mesmo pedido global do exterior à rede). Isto pode acontecer tanto pelo facto de se terem estruturas de produtos com componentes repetidos em vários nós da árvore de estrutura de produto como pelo facto de o mesmo agente ser fornecedor de produtos diferentes que são componentes de um mesmo produto final da rede. Para acomodar estes casos, haveria que ampliar o modelo de modo a acomodar problemas de escalonamento de agente gestor com mais de uma tarefa ou, alternativamente, permitir mais de um problema de escalonamento de agente associado a um mesmo problema de escalonamento de rede.

Outro tipo de ampliações do modelo, a explorar, é o de permitir uma interacção mais alargada entre o exterior e a rede de EE. No presente trabalho, o exterior é apenas um "local" de onde surgem pedidos globais do exterior à rede de EE e para onde são enviados pedidos globais da rede ao exterior. Foi assumido (no pressuposto 4 da secção 4.3.4) que o exterior apenas se manifesta na rede de EE através de pedidos globais do exterior — excluindo-se assim qualquer outra forma de interacção entre o exterior e a rede (ver também pressuposto 6 da secção 4.3.4) — e que os agentes gestores dedicam potencialmente toda a capacidade dos nós geridos a tarefas necessárias à satisfação de pedidos dos seus agentes clientes (a capacidade máxima de cada nó não se altera durante a resolução de problemas).

A possibilidade de a capacidade máxima dos nós de rede física se alterar durante a resolução de problemas permite modelar situações em que, por exemplo, existem agentes na rede de EE que podem afectar a sua capacidade também a tarefas estranhas à rede[3] e também situações de falta inesperada de capacidade.

A possibilidade de alterações em pedidos globais por iniciativa do exterior (pedidos de re-escalonamento do exterior, ou alteração do horizonte temporal global) e a possibilidade de pedidos de re-escalonamento ao exterior permite que se possam abranger aspectos de negociação entre a rede e o exterior (tanto relativos a pedidos globais do exterior à rede como a pedidos globais da rede ao exterior). Isto envolverá, naturalmente, a ampliação do conjunto de protocolos de interacção definidos, de modo a acomodar interacção com o exterior de uma natureza mais elaborada.

6.2.4 Escalonamento Multi-Agente e Actividades de Médio-Longo Prazo

No modelo desenvolvido no presente trabalho, apenas se considera a actividade de escalonamento a curto prazo. A actividade resolução de problemas de escalonamento de uma rede de EE é desencadeada apenas por pedidos do exterior (pode dizer-se que a modelação

[3] E também considerar, por exemplo, a possibilidade de várias redes com nós partilhados pelas redes. Nesse caso, a competição pelas capacidades dos nós que se verifica entre processos de uma mesma rede passaria a verificar-se também entre processos da rede e o exterior (*i.e.*, processos de outras redes).

possível se limita a ambientes do tipo produção por encomenda) e o escalonamento não é enquadrado numa perspectiva de médio ou longo prazo.[4]

A adequação do modelo de forma a enquadrar o escalonamento multi-agente numa perspectiva mais geral, que abranja actividades de previsão e planeamento de tarefas a médio-longo prazo, constitui um trabalho futuro que parece ser interessante. O envolvimento do agente supervisor nas actividades de previsão e planeamento, a difusão de informação de previsão da procura pelos agentes e o planeamento coordenado de tarefas futuras da rede serão necessários.

O aspecto mais interessante é o da influência daquelas actividades na actividade de curto prazo de escalonamento. Por exemplo, dispondo de informação da previsão da procura e do planeamento actualizados, os agentes poderão fazer reservas de capacidade nos nós para tarefas futuras possíveis (esta extensão ao modelo é sugerida na secção 3.4.1.11). Também, nós acumuladores poderão armazenar produtos de consumo sazonal em quantidade necessária para satisfazer a procura em períodos de pico de procura.

6.2.5 Modelos Mentais dos Agentes

O aperfeiçoamento e a ampliação da arquitectura dos agentes do modelo constituem uma via de trabalho futuro importante.

Na literatura de IAD existem modelos mentais de agentes que podem servir de suporte a uma arquitectura de agente apropriada para actuar em ambientes dinâmicos e imprevisíveis, como o ambiente de escalonamento multi-agente numa rede de EE (por exemplo, os modelos *Belief-Desire-Intention*, ver [Georgeff 1998]). O uso deste tipo de arquitecturas poderá permitir um comportamento mais adequado dos agentes e melhorar o desempenho global da actividade de escalonamento multi-agente.

Também, no presente trabalho, os agentes de uma rede de EE têm um comportamento "inato" e fixo. Ampliar a arquitecturas dos agentes gestores de modo a acomodar a capacidade de aprender, *i.e.*, de adaptar o seu comportamento baseando-se na experiência prévia de escalonamento de tarefas constituem uma via de trabalho futuro importante.

Um outro aspecto interessante, e possível de ser explorado em trabalho futuro, é o da inclusão de um componente emocional na arquitectura dos agentes. Este componente emocional poderia influenciar o comportamento individual dos agentes, fazendo-se depender as decisões individuais de escalonamento de uma apreciação do contexto de tomada de decisão (*e.g.*, número de pedidos recebidos elevado ou baixo, número de conflitos elevado ou baixo, soluções para os problemas de escalonamento satisfazendo em maior ou menor grau as preferências do agente). Acomodar emoções e estado emocional na arquitectura dos agentes permite simular situações realistas, de influência das emoções na tomada de decisão[5] e estudar que soluções são obtidas, para os mesmos problemas de escalonamento, com diferentes perfis emocionais e em diferentes estados emocionais dos agentes.

---

[4] Por exemplo, considerou-se sempre que a quantidade de capacidade utilizada em produtos armazenados em cada nó acumulador, em qualquer instante, é a que é estritamente necessária para satisfazer os pedidos que foram realizados e ainda não foram satisfeitos no instante.

[5] Na área de IAD existe trabalho desenvolvido relativo à contribuição de componentes emocionais no comportamento dos agentes (veja-se [Botelho 2000] ou [Elliott 1994], a título de exemplo).

6.2.6 Outros Aspectos

Por fim, pretende-se desenvolver uma realização computacional *realmente distribuída* (*i.e.*, fisicamente distribuída), em que os agentes intervenientes numa rede de EE estão localizados em computadores diferentes, comunicando através de rede electrónica e em que, possivelmente, alguns são agentes humanos.

Apêndice A — Diagramas de Classes

Neste apêndice mostram-se diagramas de classes para os objectos do modelo desenvolvido e descrito nos capítulos 3 e 4 do presente trabalho.

Nos diagramas de classes que se seguem, excluem-se protocolos de interacção entre os agentes e operações relacionadas com protocolos (como as que foram definidas no contexto das secções 4.4.7 e 4.4.8, no presente trabalho).

Na representação gráfica dos diagramas é empregue uma notação muito semelhante à usada na linguagem de modelação visual UML (ver [Rumbaugh 1999]), que é um padrão praticamente aceite para modelação de informação. Algumas diferenças relativamente à linguagem UML justificam, no entanto, as breves explicações dos diagramas que se expõem a seguir.

Para cada classe de objecto distinguem-se operações primitivas de operações não primitivas. O conjunto de operações primitivas inclui, em geral, operações selectoras, cada uma permitindo acesso a cada atributo dos objectos da classe e uma operação construtora, que produz um objecto da classe; as operações não primitivas são operações de nível mais alto que as operações primitivas e que podem ser definidas em termos delas. Operações primitivas e operações não primitivas são representadas, dentro do rectângulo da classe a que estão associadas, através do identificador da operação seguido de um par de parêntesis, em secções separadas por uma linha a traço leve, com as operações primitivas em primeiro lugar. Os atributos dos objectos de cada classe estão representados, de forma abstracta, pelas operações primitivas selectoras e não são incluídos.

Para uma qualquer classe, quando todas as suas subclasses têm operações associadas com o mesmo identificador, é representada explicitamente apenas uma única operação com esse identificador, associada à classe.

Adicionalmente, quando uma qualquer classe tem associadas várias (mais de uma) operações com o mesmo identificador, é representada explicitamente, associada à classe, apenas uma única operação com o mesmo identificador. No entanto, o número de operações adicionais que ficam implícitas é representado pelo número de caracteres "." (ponto), descontando o número de operações com o mesmo identificador que já estejam representadas, explicita ou implicitamente, em super-classes da referida classe.

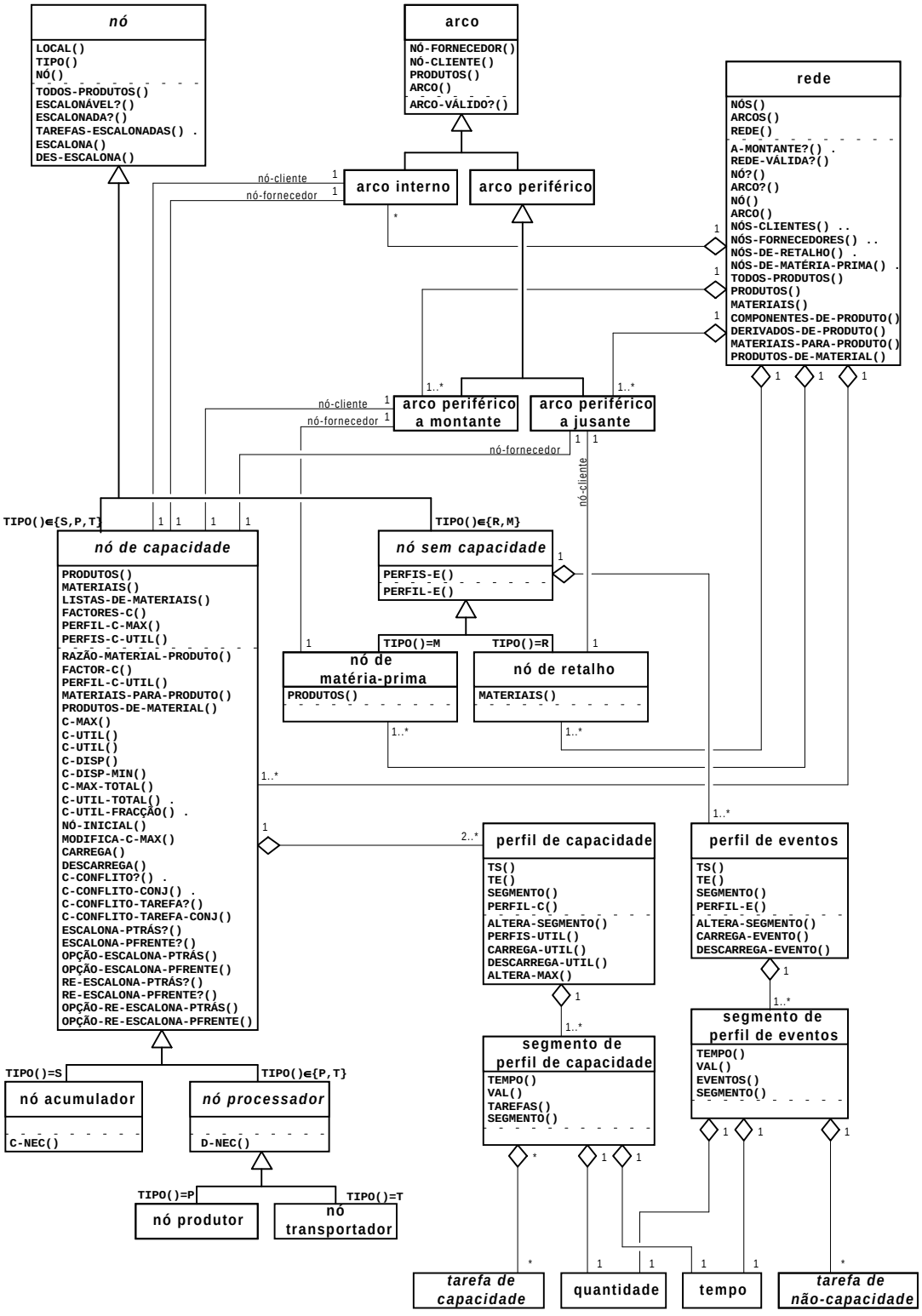


Figura A-1- Diagrama de classes para rede, nó, arco, perfil de capacidade, segmento de perfil de capacidade, perfil de eventos e segmento de perfil de eventos.

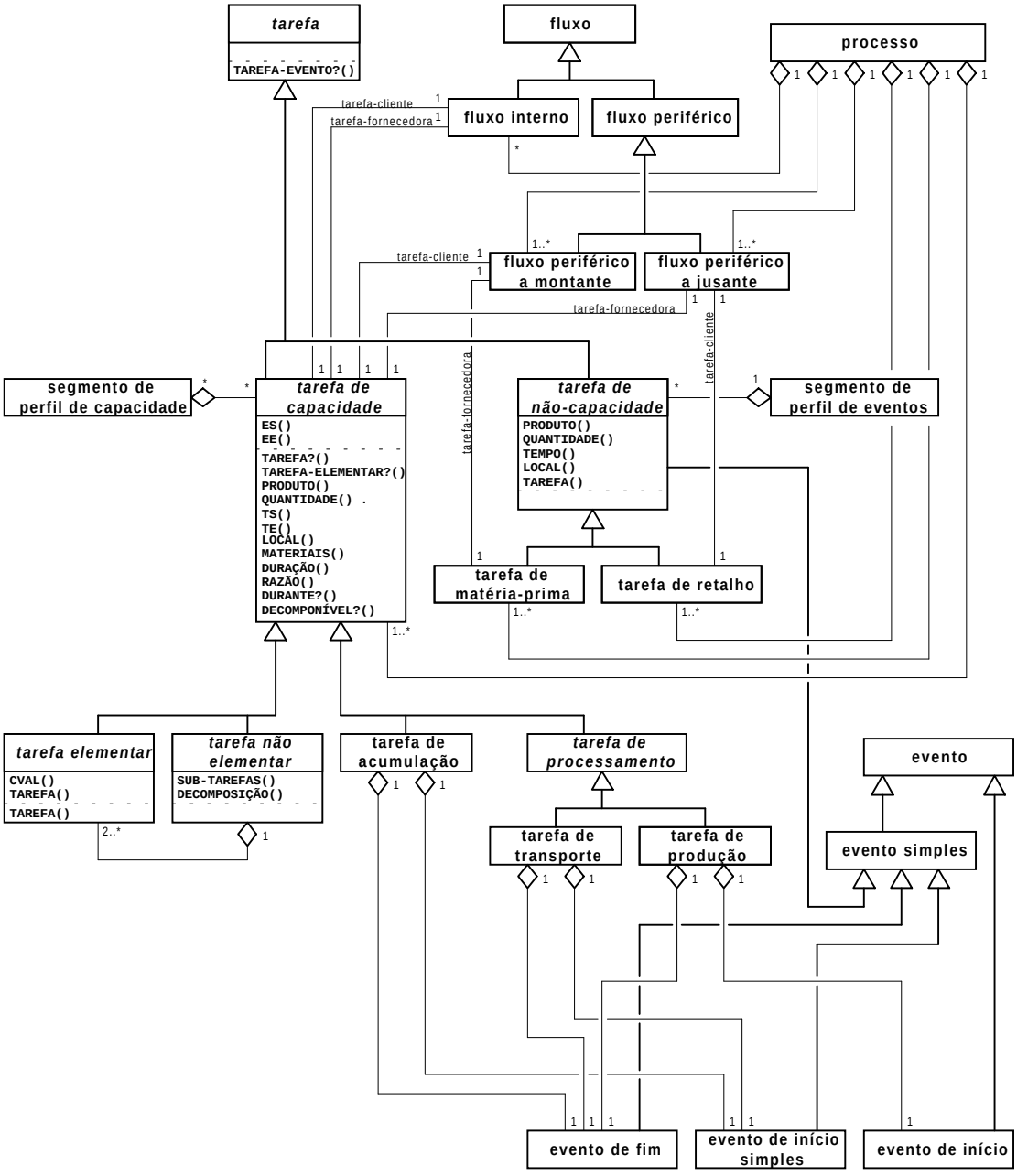


Figura A-2- Diagrama de classes para processo, tarefa e fluxo.

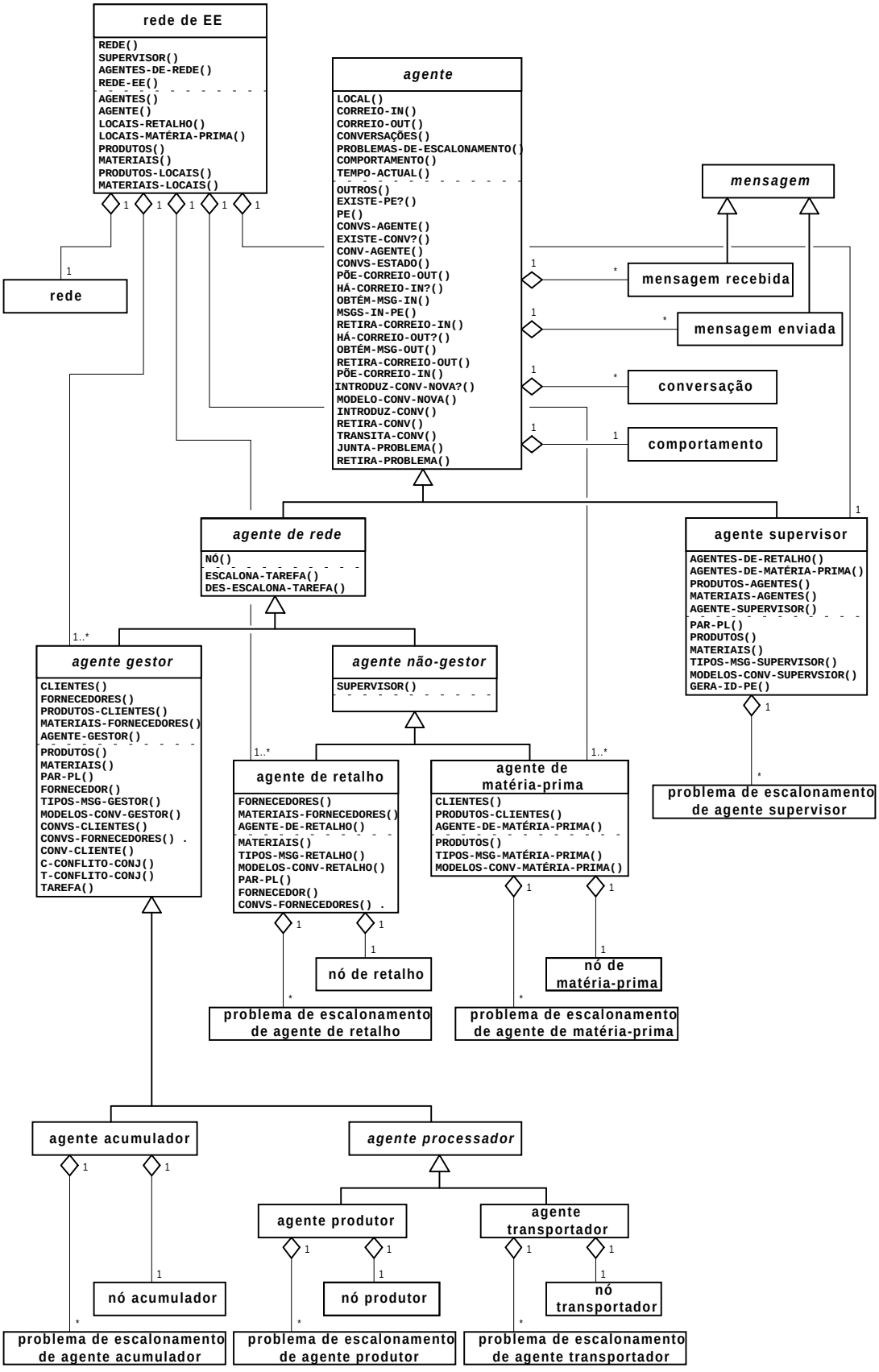


Figura A-3- Diagrama de classes para rede de EE e agente.

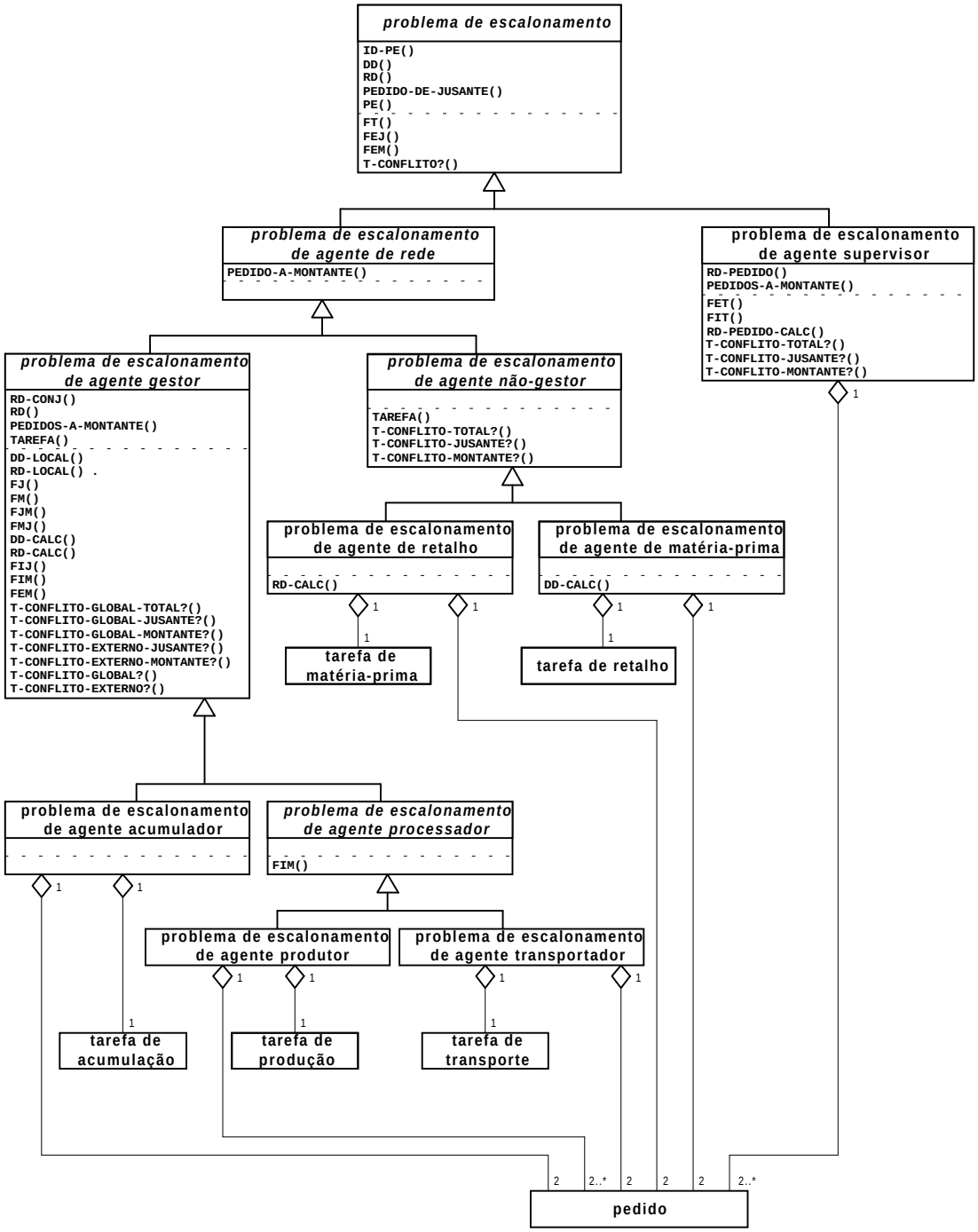


Figura A-4- Diagrama de classes para problema de escalonamento.

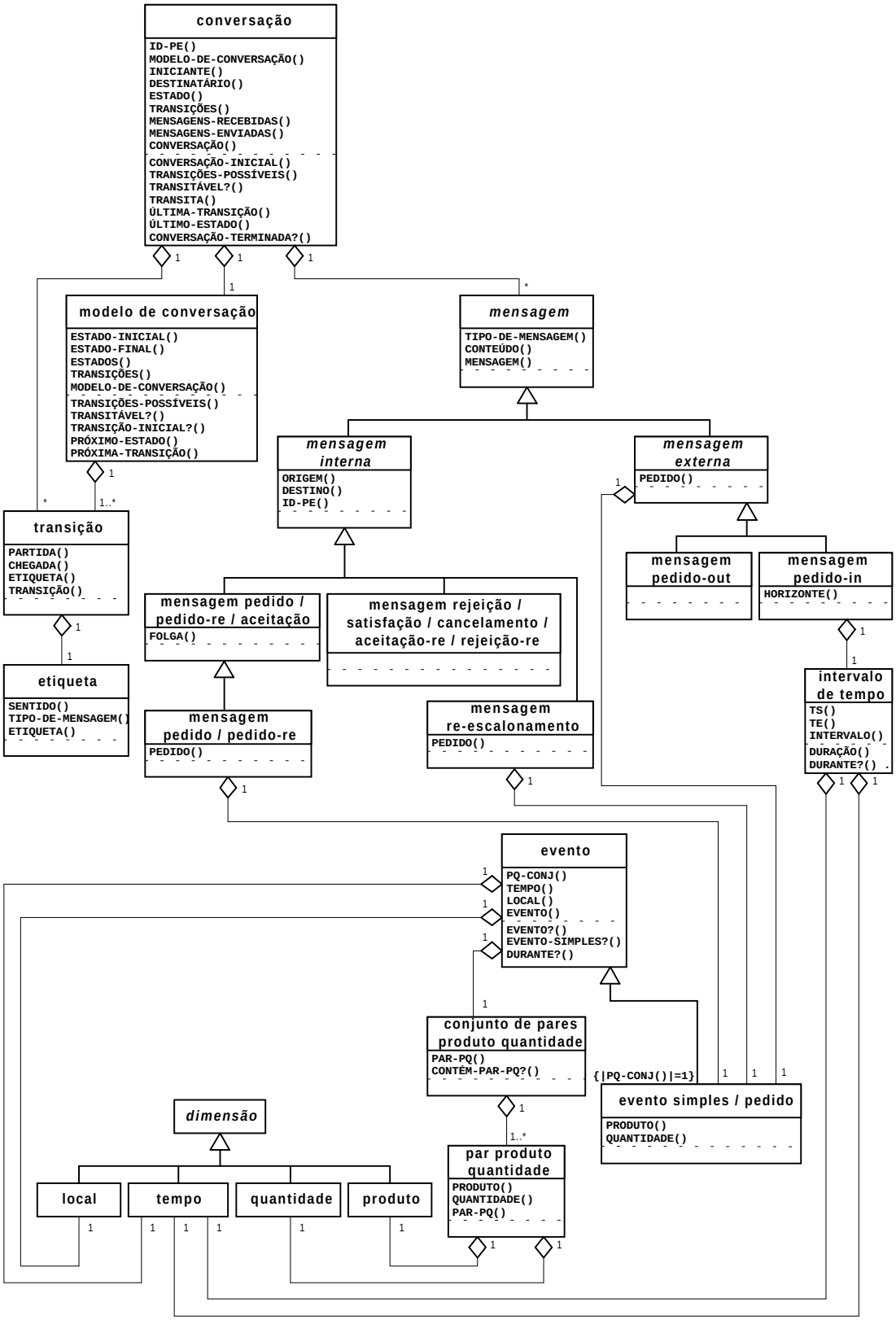


Figura A-5- Diagrama de classes para conversação, modelo de conversação, transição, etiqueta, mensagem, evento, conjunto de pares produto-quantidade, par produto-quantidade, dimensão e intervalo de tempo.

Apêndice B — Inteligência Artificial e Resolução de Problemas através de Procura

Neste apêndice, faz-se uma breve introdução à resolução de problemas através de procura, um método frequentemente combinado com o uso de heurísticas e empregue na resolução de problemas difíceis.

Introdução

A Inteligência Artificial (IA) é uma área de investigação que procura estudar e reproduzir o comportamento inteligente e que usa tipicamente o computador como ferramenta para testar e realizar teorias. Tendo surgido nos anos 50 e abordando inicialmente problemas relativamente simples (jogos, problemas brinquedo) a sua aplicabilidade prática foi fértil a partir dos anos 70 como o aparecimento dos *sistemas periciais* (*expert systems*), que empregam o conhecimento de um perito num domínio específico para automatizar, ou assistir e apoiar, o desempenho de uma tarefa altamente especializada do domínio.[1] Factores importantes, que formaram uma base de desenvolvimento desta área, foram o processamento simbólico e o raciocínio heurístico, muito popularizado por investigadores como Herbert Simon, nos anos 60. Na IA é típica a noção de procura heurística em espaço de estados, o emprego de heurísticas e de conhecimento do domínio dos problemas. Alguns campos de investigação da área de IA que merecem destaque são a Resolução de Problemas e a Procura Heurística, Representação de Conhecimento, Raciocínio (incluindo raciocínio com conhecimento incompleto e raciocínio baseado em casos), Planeamento, Aprendizagem, Compreensão e Geração de Língua Natural, Sistemas Periciais, Visão e Robótica. Introduções à área de IA podem ser encontradas em [Rich 1991], [Luger 1993], [Dean 1995], [Russell 1995] e [Nilsson 1998]. Outros textos, mais especializados são, por exemplo, [Kerr 1991] (aplicação de sistemas de IA baseados em conhecimento à gestão da produção), [Famili 1992] (contém vários artigos sobre aplicações da IA à produção industrial), [Norvig 1992] (técnicas de programação aplicáveis à IA), [Stefik 1995] (sistemas baseados em conhecimento) e [Cohen 1995] (uso de métodos empíricos na IA).

Em muitos problemas abordados pela IA, o número de possíveis configurações dos objectos relevantes para o problema é tipicamente grande e não existe um método directo para obter a solução para o problema. Planeamento de acções (*e.g.*, de um robô),[2] visão (*e.g.*, interpretação de cenas), reconhecimento de linguagem falada, compreensão, geração e tradução de língua natural, raciocínio, jogos (e.g., jogar xadrez), são alguns exemplos destes tipos de problemas. A resolução deste tipo de problemas envolve normalmente o uso de *conhecimento do domínio específico do problema*, em conjunção com um processo de *procura da solução* do problema, que pode empregar uma estratégia geral de procura mais ou menos inteligente.

Na perspectiva de IA, um *problema* define-se através de (ver, por exemplo, [Rich 1991]):

---

[1] A arquitectura de um *sistema pericial* é composta, tipicamente, por: a) uma *base de regras*, contendo regras do tipo condição-acção, que representam o conhecimento de um perito relativo à resolução do tipo de problemas particular, b) um *motor de inferência*, que decide a aplicação repetida das regras para a resolução de um problema, e c) uma *interface*, que permite a junção, remoção ou modificação de regras por um *engenheiro do conhecimento* e a consulta do sistema por um utilizador. Uma outra arquitectura similar que foi frequentemente usada também em sistemas periciais, divulgada nos anos 60 por Alan Newell e Herbert Simon, é a do *sistema de produção*. Nesta arquitectura, a aplicação repetida de regras de modo a modificar a representação do estado da resolução de um problema, com vista a realizar a procura da solução, tem os seguintes componentes: a) um *conjunto de regras* do tipo condição-acção, b) uma *memória de trabalho*, que contém informação estruturada apropriada à resolução do problema particular, seja informação de carácter permanente ou de carácter transitório, c) uma *estratégia de controlo*, que impõe prioridades para a aplicação das regras e impõe o modo de resolução de conflitos entre regras, em situações em que várias regras são aplicáveis, e d) um *aplicador de regras*. Ver por exemplo, [Rich 1991].

[2] Um modo de ver os problemas de escalonamento é como sendo problemas planeamento muito específicos, nos quais há que sequenciar e definir tempos de início para acções de um plano, sujeito a certas restrições.

1. Definição de um *espaço de estados* que contém todos os *estados* possíveis do problema, isto é, definição (não necessariamente explícita) de todas as possíveis configurações dos objectos relevantes para o problema;
2. Especificação do *estado inicial*, ou dos *estados iniciais*, que descrevem possíveis situações a partir das quais o processo de resolução pode ser iniciado;
3. Especificação do *estado objectivo*, ou dos *estados objectivo*, que são estados aceites como soluções para o problema;
4. Especificação do conjunto dos *operadores*, isto é, de um conjunto de regras que permitem, dado um estado, obter *estados sucessores*, ou estados seguintes, desse estado.

Gerar-e-Testar

Uma estratégia básica de abordagem à resolução de um problema é a que emprega um paradigma apelidado de *gerar-e-testar* (*generate-and-test*). Consiste em aplicar um processo em que cada estado possível é sistematicamente gerado e testado para verificar se é um estado objectivo; ao encontrar o primeiro estado reconhecido como estado objectivo o processo pára devolvendo a solução[3] [Rich 1991]. Por si só, sem se especificar uma estratégia mais específica, este processo pode ser muito ineficiente e, por isso, o processo típico empregue em muitos sistemas de IA para resolver problemas é o da *procura em árvore*. A procura em árvore também se enquadra no paradigma gerar-e-testar, mas é mais específica e elaborada, pois permite manter uma história do percurso da procura e o controlo sobre a ordem pela qual os estados são percorridos para chegar a uma solução.

Numa procura em árvore uma *árvore de procura* é progressivamente gerada, total ou parcialmente. Os nós da árvore de procura representam estados da procura. Inicialmente existe um estado inicial (dado), representado pelo nó de raiz da árvore. Primeiro um nó é *expandido*, isto é, são gerados os seus *nós sucessores*, que representam estados seguintes possíveis da procura. A seguir um ou vários dos nós sucessores são escolhidos para serem processados. Depois o nó ou nós escolhidos são testados para determinar se representam estados objectivo. Isto constitui um passo de procura e é repetidamente aplicado, ampliando progressivamente caminhos parciais na árvore de procura, até que se tenha descoberto um caminho completo que tenha chegado ao fim da árvore, por terminar num nó que representa um estado objectivo; alternativamente, um caminho pode terminar num nó que representa um estado sem estados possíveis seguintes. No primeiro caso foi encontrada uma solução do problema de procura; no segundo caso foi encontrada uma solução incompleta inviável (impossível de completar) e a procura poderá, então, terminar sem sucesso, ou continuar, explorando um caminho alternativo da árvore de procura. Se o problema é tal que é possível um mesmo estado ser sucessor de outros estados diferentes, ou um estado ter sucessores que são seus antepassados na árvore de procura, o procedimento de procura deve ter mecanismos para evitar nós repetidos e ciclos.

Dois tipos de estratégias muito gerais de controlo da procura em árvore, que podem ser consideradas dois extremos da filosofia de procura empregue, podem ser usadas: a *procura em largura* (*breath-first search*) e a *procura em profundidade* (*depth-first search*).

---

[3] O processo pode ser continuado, se o objectivo é encontrar mais de uma solução.

Procura em Largura

Numa procura em largura são explorados todos os caminhos possíveis da árvore de procura. São mantidos caminhos parciais com o mesmo número de nós (*i.e.*, caminhos até à mesma profundidade da árvore de procura) que, em cada passo da procura, são ramificados e ampliados com mais um nó, por expansão de cada nó extremo de cada caminho, nos seus nós sucessores. Caminhos terminando em nós sem sucessores são abandonados. Assim que um dos caminhos atinge um nó que representa um estado objectivo a procura termina com sucesso, pois foi encontrada uma solução.[4]

Procura em Profundidade

Numa procura em profundidade, um só caminho da árvore de procura é explorado desde o nó de raiz até ao fundo da árvore. A procura termina com sucesso se o último nó representa um estado objectivo, ou sem sucesso se o último nó representa um estado sem estados seguintes. Neste último caso, a procura em profundidade falha, pois não foi encontrada uma solução para o problema. A técnica de *retrocesso* (*backtracking*) pode ser usada em combinação com a procura em profundidade de modo a que, em caso de falha, a procura possa ser continuada a por um caminho alternativo ao que foi seguido.[5] Em caso de falha, o retrocesso é usado de uma forma sistemática para explorar um caminho alternativo, a partir de um nó antepassado, que não tenha ainda sido explorado. A forma mais simples de retrocesso é a do *retrocesso cronológico* (*chronological backtracking*), em que o nó antepassado para que se retrocede é o nó antecessor do actual. Neste tipo de retrocesso, a procura pode ser forçada a retroceder várias vezes no caminho actual até que encontre um caminho alternativo que possa ser prosseguido.[6] Neste caso poderão ocorrer sucessivas falhas e retrocessos, o que pode tornar o processo bastante ineficiente. O problema da falha repetida da procura com retrocesso em diferentes pontos do espaço de procura pelas mesmas razões é conhecido como fenómeno do *repisar*, ou *thrashing*[7].

Procura Heurística

A procura em largura é pouco económica, relativamente ao espaço de memória necessário pois, em cada passo, todos os nós da parte gerada da árvore de procura são mantidos em memória. No entanto, se existir uma solução, a procura em largura garante encontrá-la (o que, na procura em profundidade só acontece à custa de retrocesso, a não ser que, por sorte, a solução esteja no primeiro caminho completo gerado). Se existir mais de uma solução, a procura em largura garante encontrar uma pelo caminho mais curto, pois os caminhos maiores nunca são explorados antes de os mais curtos o serem. A procura em profundidade é económica, em termos de espaço de memória ocupado pois, em cada passo, o número de nós

---

[4] Se o objectivo é encontrar mais de uma solução, o caminho de solução encontrado pode ser memorizado e posto de parte e a procura continuada.

[5] No caso de o objectivo ser o de encontrar mais de uma solução, o retrocesso também pode ser usado mesmo que a procura já tenha tido sucesso, para permitir que seja continuada.

[6] No limite, a procura poderá terminar por não ter mais caminhos alternativos para explorar (quando o último nó sucessor do nó de raiz da árvore de procura já foi explorado sem sucesso) e nesse caso conclui-se que o problema de procura não tem solução.

[7] Adopta-se para tradução de *thrashing* para o português o termo *repisar*, como adoptado em [Soares 1998].

da árvore de procura mantidos em memória nunca excede o número de nós de um caminho completo. No entanto, se não for combinada com o retrocesso pode falhar, se enveredar por um caminho inviável. Mesmo com retrocesso pode ser muito pouco económica, relativamente ao tempo gasto para encontrar uma solução, dependendo isso muito da ordem pela qual, em cada passo, os nós sucessores são escolhidos: no pior caso, a solução, se existir, pode só ser encontrada com retrocessos sucessivos, após a exploração completa de todos os caminhos alternativos da árvore de procura. Mesmo assim não garante encontrar a solução do caminho mais curto. O problema do fenómeno do repisar no retrocesso cronológico pode ser melhorado realizando, quando há uma falha, um retrocesso selectivo para o nó que é responsável pela falha. Esta técnica é designada por *retrocesso inteligente*. Uma técnica de retrocesso inteligente, designada por *retrocesso guiado por dependências* (*dependency directed backtracking*) [Stallman 1977], pode ser usada para evitar virtualmente todo o trabalho redundante numa procura em árvore, à custa de progressivamente construir e manter um registo de dependências entre estados da procura. No entanto, os procedimentos para determinar a razão da falha podem ser complexos e a utilização desta técnica pode resultar muito mais demorada e consumir muito mais espaço de memória do que uma forma simples de retrocesso.

O problema da ineficiência de certas técnicas de procura advém do facto de procurarem às cegas, quer dizer, são técnicas de *procura não informada*, ou *procura de força bruta*. Como os recursos computacionais são limitados, em tempo de processamento e em espaço de memória, é conveniente, na resolução de um problema de procura, empregar uma *procura informada*, ou *procura inteligente*, ou *procura heurística*, que permita ao procedimento de procura tomar decisões apropriadas acerca de por onde se encaminhar no passo seguinte da procura, evitando considerar todos os caminhos (como na procura em largura) ou explorar caminhos dos quais viria, fatalmente, a retroceder (como na procura em profundidade com retrocesso). Para a tomada destas decisões são empregues *heurísticas*, ou regras de polegar, que servem para guiar a procura a cada passo, na escolha dos ramos da árvore que mais prometem levar a uma solução do problema. Estas regras não garantem, em todos os casos, chegar a uma solução do problema (assumindo que existe solução), e poderão não levar à solução de caminho mais curto, mas são regras que, pela experiência ou intuição humana, se sabe levarem a uma solução aceitável de um problema na maior parte dos casos.

O uso de heurísticas é frequente na resolução de problemas em IA basicamente por duas razões [Luger 1993]: a) o problema a resolver pode não ter uma solução exacta devido a ambiguidades inerentes na formulação do problema ou dos dados do mesmo,[8] e b) o problema tem uma solução exacta mas o custo computacional a despender para a encontrar é proibitivo.[9] A procura informada será tanto menos ineficiente quanto mais informada for, quer dizer,

---

[8] Por exemplo, no diagnóstico médico, para um certo conjunto de sintomas poderá haver várias causas possíveis (os médicos usam heurísticas para escolher o diagnóstico mais provável para prescrever um tratamento); na visão, cenas visuais contêm, em geral, muitas ambiguidades no que respeita à interpretação das ligações, extensão e orientação dos objectos da cena (um sistema de visão usa heurísticas para escolher, de várias interpretações possíveis, as mais plausíveis).

[9] Em muitos problemas de procura o número de caminhos possíveis que é originado a partir de cada nó da árvore de procura é muito grande. Certos problemas, apelidados de *computacionalmente intratáveis*, o crescimento dos caminhos possíveis, à medida que se percorre a árvore de procura, é exponencial relativamente à profundidade percorrida na árvore. Para este tipo de problemas a procura informada é adequada pois, em geral, as técnicas de procura não informadas falham em encontrar uma solução num intervalo de tempo aceitável e/ou dispondo de um espaço de memória aceitável. As heurísticas permitem pôr de parte, numa procura informada, caminhos possíveis que não levam a uma solução, contrariando o crescimento exponencial desses caminhos.

quanto mais precisa for a heurística usada. Se fosse possível ter uma heurística perfeita, que em cada passo levaria o procedimento de procura a escolher um nó no caminho que leva directamente à solução, o percurso do procedimento pela árvore de procura seria idêntico ao de uma procura em profundidade em que não haveria necessidade de retrocesso, pois se encaminha directamente para o nó do estado objectivo.

Procura pelo Melhor

Uma forma de combinar as vantagens da procura em largura e da procura em profundidade e reduzir as suas desvantagens é utilizar a técnica de *procura pelo melhor* (*best first search*). Na procura pelo melhor dispõe-se de uma *função de avaliação* `f` dos nós, que serve para comparar nós da árvore de procura, no sentido de informar em quanto um nó está mais próximo de um nó de estado objectivo. Esta função de avaliação é heurística ou, pelo menos, incorpora algum conhecimento heurístico. A procura pelo melhor segue o mesmo princípio que a estratégia geral da procura de *subida do monte*, no sentido em que toma em consideração o melhor nó sucessor para expandir. Na procura de subida do monte o nó actual é expandido, os seus sucessores são heuristicamente avaliados e o melhor dos sucessores escolhido para ser expandido, repetindo-se o processo até que se obtenha um nó que é melhor que qualquer dos seus sucessores. Apenas um nó é mantido em memória, não se mantendo qualquer "história" (isto inclui não manter o caminho até ao nó actual) da resolução do problema. Ciclos e terminação em estados que são óptimos locais da função de avaliação são problemas que o método da subida do monte pode não conseguir evitar, se a função heurística de avaliação não for boa.[10] Ao contrário do método da subida do monte, o método da procura pelo melhor mantém, além do caminho actualmente em exploração, vários caminhos parciais possíveis e a função de avaliação permite-lhe escolher o caminho mais promissor, ou abdicar do caminho actual em favor de outro que pareça mais promissor. Cada nó, ao ser gerado, é avaliado com a função de avaliação e é colocado numa fila de nós por expandir, ordenados pelo valor de avaliação, juntamente com o caminho até ao nó. Esta fila contém todos os nós que constituem a fronteira da parte gerada da árvore de procura, que são nós extremos de caminhos parciais que poderão vir a ser ampliados. Em cada passo, o procedimento de procura retira o nó no início da fila de nós por expandir (o nó com melhor valor de avaliação) e testa se ele representa um estado objectivo. Em caso afirmativo foi encontrada uma solução; caso contrário expande-o, avalia os seus sucessores e coloca-os na fila de nós, mantendo-a ordenada; se o nó não tem sucessores o seu caminho é abandonado. O processo é, então, repetido.[11] Se não há nós na fila de nós por expandir a procura termina sem sucesso (caso em que se conclui que o problema não tem solução). Nesta forma de procura, qualquer caminho parcial que se revele

---

[10] Note-se, no entanto, que o método da subida do monte tem vantagens óbvias de economia de espaço de memória. Relativamente ao problema com óptimos locais é possível perturbar aleatoriamente a função de avaliação de modo a que o procedimento de procura escape de óptimos locais mas, em geral, não há nenhuma maneira de garantir que o método da subida do monte encontre a solução globalmente óptima.

[11] A lista de nós referida é frequentemente apelidada de lista de *nós abertos*. Se o problema é tal que pode haver ciclos na procura, mantém-se uma lista de nós já processados. Esta lista é frequentemente apelidada de lista de *nós fechados*. Quando um nó é gerado, é verificado se ele está presente nesta lista, ou na lista de nós abertos e, nesse caso, não será colocado nesta última. Numa abordagem mais sofisticada, se se pretende encontrar o caminho mais curto para a solução, quando o nó gerado também se encontra na lista de nós fechados, ou na lista de nós abertos, mas tem um caminho mais curto, no primeiro caso é retirado da lista de nós fechados e colocado na lista de nós abertos com esse caminho mais curto associado, no segundo caso é-lhe associado esse caminho mais curto e ele é mantido na lista de nós abertos.

mais promissor que o actualmente seguido será retomado, o que pode considerar-se uma forma generalizada de retrocesso.

Procura A*

Um procedimento de procura designado por *algoritmo A* corresponde a uma procura pelo melhor em que a função de avaliação `f` é uma função tal que, dado um nó, devolve um valor de custo associado a esse nó. `f` é dada por (ver, por exemplo, [Rich 1991], [Luger 1993] ou [Russell 1995]) `f=g+h`, em que `g` é uma função que, dado um nó, devolve o comprimento do caminho do nó de raiz até esse nó e `h` é uma função heurística que, dado um nó, fornece uma estimativa da distância desse nó ao nó de solução. O melhor nó, em cada passo, será então aquele que tem menor valor de `f`, quer dizer, será escolhido, em cada passo, o nó que minimiza uma estimativa do comprimento do caminho que passa pelo nó e vai até uma solução. Dependendo das funções `g` e `h`, a procura pelo melhor assume características particulares. Se a componente `h` for nula, a procura será guiada por `g`; em particular se se considera que o comprimento do caminho entre cada nó e o seu sucessor é de `1` (*i.e.*, `g` aumenta de `1` de um nó para um qualquer seu sucessor) a procura pelo melhor é igual à procura em largura (e encontrará sempre a solução de caminho mais curto). Se a componente `g` for nula, a procura será guiada por `h` e assemelhar-se-á à procura em profundidade com retrocesso; em particular, com uma boa função heurística `h`, a procura é bastante focada e convergirá rapidamente para uma solução, mesmo que não seja a solução de caminho mais curto.[12] Se ambas as funções, `g` e `h`, são nulas a procura é aleatória. Em muitos problemas o espaço de memória ocupado pelos nós da parte gerada da árvore numa procura pelo melhor resulta, mesmo assim, bastante grande. Uma técnica de procura que tenta obviar este problema, apelidada de *procura em faixa* (*beam search*) resulta de uma simplificação da procura pelo melhor que memoriza, na fila de nós por expandir, apenas um certo número dos melhores nós, em cada passo. O procedimento de procura assim obtido é mais eficiente em termos de espaço de memória utilizado e também, em princípio, mais rápido, pelo facto de o número de nós gerados ser limitado. No entanto, corre o risco de não encontrar uma solução, devido a poder excluir certas porções da árvore de procura muito cedo.

Um procedimento de procura é dito *admissível* se termina sempre com a solução de caminho mais curto (o caminho óptimo de solução). A procura em largura é admissível. Um procedimento do mesmo tipo do algoritmo A em que a função de avaliação `f` é tal que `h` nunca sobrestima a distância até ao nó de solução (que dizer, o valor de `h` para um nó é sempre menor, ou igual, que o valor real da distância do nó ao nó de solução) é dito *algoritmo A** (A asterisco) e é admissível (ver, por exemplo, [Rich 1991], [Luger 1993] ou [Russell 1995]). O algoritmo A* pretende ser admissível, tal como a procura em largura, mas ser também eficiente, tal como uma procura em profundidade, de modo a encontrar a solução sem retrocessos.

[12] Uma procura guiada apenas pela estimativa da distância a uma solução é frequentemente apelidada de *procura glutona* (*greedy search*).

Apêndice C — O Problema de Satisfação de Restrições

Neste apêndice faz-se uma breve introdução ao Problema de Satisfação de Restrições (PSR).

Introdução

Um grande número de problemas pode ser visto como sendo casos especiais do *problema de satisfação de restrições* (PSR, *constraint satisfaction problem* ou CSP). São exemplos destes problemas os de visão artificial, de manutenção de verdade (*truth maintenance*), de raciocínio temporal, os problemas com grafos, certos problemas de planeamento (*e.g.*, de ocupação de superfícies, planeamento de experiências genéticas), o problema da satisfiabilidade, problemas que surgem em projecto de circuitos eléctricos, de máquinas e de processos produtivos, raciocínio para diagnóstico, quebra-cabeças, planeamento da produção, escalonamento [Kumar 1992], [Freuder 1994], [Russell 1995], [Barahona 1997].

O PSR é o problema de encontrar valores para as variáveis de um conjunto de variáveis $\mathcal{X}=\{x_1,\ldots,x_i,\ldots,x_n\}$ de tal modo que um conjunto de restrições $\mathcal{C}=\{c_1,\ldots,c_j,\ldots,c_m\}$ seja satisfeito. A cada variável x_i está associado um domínio d_i que indica os valores possíveis para x_i. Cada restrição c_j é um subconjunto do produto cartesiano $d_1*\ldots,d_i*\ldots*d_n$, que especifica que combinações de valores das variáveis $x_1,\ldots,x_i,\ldots,x_n$ são permitidos. Uma atribuição de valores a todas as variáveis que satisfaça todas as restrições é uma solução do PSR. Em geral, o objectivo do PSR é o de encontrar uma solução, mas pode também ser encontrar todas as soluções possíveis [Kumar 1992].[13] É frequente cada restrição ser definida apenas sobre um subconjunto das variáveis do problema, por exemplo, sobre duas variáveis — ditas restrições binárias — e sobre uma variável — ditas restrições unárias — [Prosser 1991], [Kumar 1992] e é possível converter um PSR com restrições n-árias num PSR com restrições binárias equivalente [Kumar 1992]. Um PSR binário pode ser representado num grafo de restrições em que cada nó representa uma variável e cada arco representa uma restrição entre as duas variáveis representadas nos extremos do arco (restrições unárias representam-se por arcos partindo e terminando num mesmo nó).

A técnica de procura em árvore tipicamente aplicada ao PSR é a da *procura em profundidade com retrocesso* (ver o Apêndice B) em que a procura é, de alguma forma, guiada pelas restrições. O processo consiste basicamente num percurso pelos nós do grafo do PSR, em que se instanciam, uma a uma, as variáveis correspondentes. Os nós da parte gerada da árvore de procura representam variáveis do PSR já instanciadas (a parte do grafo do PSR já percorrido) e constituem uma *solução parcial* do PSR. Em cada passo de procura, uma variável por instanciar é instanciada com um valor do seu domínio e este valor excluído do domínio. A seguir testa-se se todas as restrições envolvendo essa variável e as variáveis previamente instanciadas são respeitadas, isto é, verifica-se se o valor atribuído à variável é *consistente* com o valor de cada uma das outras variáveis instanciadas. Se isso acontece repete-se o processo,

---

[13] Na realidade a definição dada é a de PSR *discreto*, quer dizer, um PSR em que os domínios das variáveis são conjuntos discretos. A resolução de um PSR com variáveis contínuas envolve álgebra sofisticada (ver, por exemplo, [Faltings 1994]). Tipicamente, problemas de escalonamento são formulados através de PSRs discretos. Relativamente às soluções procuradas num PSR, pode também ser procurada uma solução óptima, ou pelo menos, uma considerada de boa qualidade, dada uma qualquer função objectivo definida em termos (dos valores) de algumas, ou todas as variáveis do problema [Smith 1995]. Em [Fox 1989], [Fox 1990] e [Sadeh 1994], este tipo de problemas (um PSR com uma função objectivo a ser optimizada) é referido como *problema de optimização com restrições* (POR, *constrained optimization problem* ou COP) ou, na versão distribuída do problema, um *problema de optimização com restrições distribuído* (PORD, *distributed constrained optimization problem* ou DCOP).

instanciando uma outra variável, das que permaneceram por instanciar. Quando não houver mais variáveis para instanciar, foi encontrada uma solução do PSR.[14] Ao instanciar uma variável, se o valor atribuído à variável está em *conflito* com o valor de alguma outra, tenta-se uma instanciação com outro valor do domínio da variável, com o qual não se tenha ainda tentado; em caso de não existirem mais valores do domínio da variável para tentar (falha da procura no caminho actual), a variável é deixada não instanciada, o seu domínio inicial é reposto e executa-se um retrocesso para um passo anterior, para tentar valores alternativos do domínio de uma variável anteriormente instanciada. Se o retrocesso não pode ser realizado, pelo facto de não haver nenhuma variável instanciada, é porque o PSR não tem solução.

Esta forma de abordar o problema, através de ampliação progressiva de uma solução parcial, é apelidada de *abordagem construtiva* e é a mais tradicional para PSRs. Contrasta com a abordagem por *reparação iterativa*, em que se parte de, e mantém, uma *solução imperfeita*, (*i.e.*, com todas as variáveis do problema com valores atribuídos mas, possivelmente, violando algumas restrições). Nesta última abordagem, em cada passo, uma variável envolvida em conflitos é escolhida e o seu valor alterado de modo a reduzir o número de restrições não respeitadas,[15] seguindo uma estratégia do tipo subida do monte. Neste apêndice assume-se uma abordagem construtiva.

O Retrocesso e o Fenómeno do Repisar

No problema do retrocesso cronológico apelidado de fenómeno do repisar (ver o Apêndice B), a procura falha repetidas vezes em diferentes pontos do espaço de procura pelas mesmas razões, originando retrocessos excessivos [Mackworth 1977], [Kumar 1992]. O *retrocesso guiado por dependências* (*dependency directed backtracking*) [Stallman 1977] é uma forma de *retrocesso inteligente* (retrocesso selectivo para a variável responsável pela falha) que tenta evitar o fenómeno do repisar. Aquela técnica baseia-se em construir progressivamente, e manter, um registo de dependências entre valores permitidos e não permitidos de variáveis e poderia virtualmente eliminar todo o trabalho redundante numa procura em árvore. No entanto, os procedimentos para determinar as razões de uma falha podem ser complexos e a utilização da técnica pode resultar muito mais demorada e consumir muito mais espaço de memória do que o retrocesso simples [Kumar 1992].

Propagação de Restrições

Uma forma de melhorar, ou mesmo eliminar, o problema do retrocesso é usar um processo de *imposição de consistência* (*consistency enforcement*) [Sadeh 1991], [Kumar 1992], [Terpstra 1996]. A imposição de consistência não resolve um PSR completamente mas elimina inconsistências locais, *i.e.*, elimina valores de variáveis que não participam nas soluções do PSR [Mackworth 1985]. Pode ser usada antes da procura (como fase de pré-processamento)

[14] A profundidade máxima da árvore de procura atingida pelo procedimento de procura é igual ao número de variáveis do PSR.

[15] Ver, por exemplo, [Johnson 1991], ou aplicado ao escalonamento, [Zweben 1994b] ou [Zweben 1992] e [Johnston 1994a] ou [Minton 1992]. Um vantagem da reparação iterativa é a de o procedimento de procura poder ser parado em qualquer passo fornecendo sempre uma solução completa (este tipo de abordagens são, por isso, frequentemente apelidadas de *a-qualquer-tempo*); uma outra vantagem é a de no passo seguinte se ter uma solução melhor que a do passo anterior.

ou durante a procura. A técnica usada para impor consistência é denominada *propagação de restrições* e baseia-se em retirar dos domínios das variáveis os valores que não satisfazem restrições do PSR.[16]

A propagação de restrições pode ser usada para impor um maior ou menor *grau de consistência*, garantindo uma procura com menor ou maior retrocesso, mas à custa de um maior ou menor custo computacional, respectivamente. Em geral, é desejável um compromisso entre número de retrocessos e grau de propagação de restrições [Haralick 1980], [Mackworth 1985], [Sadeh 1991], [Kumar 1992]. *Consistência de nó*, ou *consistência de grau 1*, garante que todos as restrições unárias sobre as variáveis do PSR são satisfeitas. *Consistência de arco*, ou *consistência de grau 2*, garante que, para cada par de variáveis ligadas por uma restrição, para qualquer valor do domínio de uma das variáveis existe um valor no domínio da outra que satisfaz a restrição entre elas. *Consistência de grau k* garante que, dados valores de quaisquer $k-1$ variáveis que satisfaçam todas as restrições entre essas variáveis, para qualquer outra variável do PSR existirá um valor que satisfaz todas as restrições entre as k variáveis. Define-se *consistência forte de grau k* como sendo consistência de grau j, para qualquer $j \leq k$[17] [Kumar 1992].

Se num PSR com n variáveis se impuser consistência de grau n a solução do PSR é obtida sem procura. Existem algoritmos para impor consistência forte de grau k, com $k>2$, mas podem ser muito ineficientes em termos do número de computações realizadas (em geral têm complexidade exponencial em k) [Sadeh 1991], [Kumar 1992]. Devido ao custo computacional elevado destes algoritmos, um processo que imponha consistência de arco é, muitas vezes, preferível [Barahona 1997].[18] Frequentemente, adopta-se um processo de procura em profundidade com retrocesso, combinado com propagação de restrições em que, previamente à procura se impõe consistência de nó e depois, em cada passo da procura, se realiza uma propagação de restrições que impõe uma qualquer forma de consistência de arco, ainda que limitada [Haralick 1980], [Prosser 1991], [Kumar 1992], [Dechter 1994]. A ideia básica é a de podar o espaço de procura antes de serem geradas combinações de valores de variáveis que levam a soluções não realizáveis [Kumar 1992] ou, por outras palavras, tentar evitar o mais cedo possível que a procura enverede por caminhos que não levam a soluções e que iriam fatalmente originar retrocesso.[19]

Pode considerar-se que a procura em profundidade com retrocesso aplicada ao PSR, como descrita inicialmente, por si só, já incorpora uma forma muito limitada de consistência de arco: quando uma variável é considerada para instanciação, qualquer dos seus possíveis valores que sejam inconsistentes com instanciações anteriores de variáveis serão postos de parte [Kumar 1992]. Neste caso, a verificação de consistência é feita com variáveis já instanciadas, *i.e.*, é uma *verificação para trás*. Uma forma de propagação de restrições eficiente e que impõe consistência de arco em maior grau, realiza verificações de consistência para a frente e é designada por *verificação para a frente* (*forward checking*) [Haralick 1980], [Prosser 1991],

---

[16] Também apelidada de *propagação local* (de restrições), já que é levada a cabo propagando alterações ocorridas no valor, ou no domínio, de uma variável do PSR para os domínios as outras variáveis *localmente*, *i.e.*, através dos arcos do grafo de restrições do PSR.

[17] Consistência forte de grau 1 é consistência de nó; consistência forte de grau 2 é consistência de arco e de nó.

[18] Um algoritmo que impõe consistência de arco muito conhecido é o *algoritmo de Waltz* (ver em [Mackworth 1977], em que o algoritmo designado por AC-2 é equivalente ao algoritmo de Waltz).

[19] Procedimentos de resolução de PSRs que não procuram garantir consistência completa, mas procuram reduzir os valores possíveis das variáveis, por propagação de restrições, de modo a limitar a procura de soluções, são apelidados de solucionadores de restrições *incompletos* [Barahona 1997].

[Kumar 1992]. Na *verificação para a frente*, quando uma variável é instanciada a um valor, o valor é retirado do domínio da variável e são eliminados (filtrados), dos domínios das variáveis ainda não instanciadas, todos valores não permitidos pelas restrições existentes entre cada uma destas variáveis e a primeira. Se em algum caso esta filtragem de valores dos domínios resultar num domínio vazio, os domínios anteriores à filtragem são repostos e é tentado um valor diferente do domínio da variável actual. Em caso de não haver mais valores para tentar (domínio da variável actual vazio), a variável é deixada não instanciada, o seu domínio anterior é reposto e é realizado um retrocesso para tentar um valor alternativo de uma das variáveis anteriormente instanciadas.

Embora a *verificação para a frente* não imponha consistência de arco completa (apenas impõe consistência de arco entre a variável actualmente considerada e as que não estão ainda instanciadas e que estão ligadas a ela por restrições; não impõe consistência de arco entre as variáveis não instanciadas), é considerada um procedimento eficiente para encontrar uma solução para um PSR, em especial se combinada com uma forma elaborada de retrocesso. Em [Prosser 1991], os algoritmos de procura para o PSR binário são classificados em dois grandes grupos: a) os que dão ênfase à maneira como a procura realiza retrocesso, e b) os que dão ênfase à maneira como a procura avança (neste último grupo está incluído um algoritmo que realiza a *verificação para a frente*, acima referida). Propõem-se então algoritmos *híbridos* (nove, no total) que combinam as vantagens de ambos os grupos e mostram-se resultados de testes experimentais realizados. Uma conclusão extraída destes testes é a de que o algoritmo de *verificação para a frente* combinado com uma forma de retrocesso apelidada de *salto-para-trás guiado por grafo* (*graph based backjumping*) tem, em média, um desempenho muito superior ao dos outros algoritmos testados, tanto em termos de menor número de verificações de consistência como de número de nós visitados até encontrar uma solução. O *salto-para-trás guiado por grafo* realiza uma forma primitiva de *retrocesso guiado por dependências* em que, para cada variável, é memorizado um *conjunto conflito*. Este conjunto contém as variáveis anteriormente consideradas que estiveram alguma vez em conflito com a variável actual;[20] o retrocesso, quando deve ocorrer, faz-se para a última variável com a qual a variável actual esteve em conflito. Esta forma de retrocesso associa informação (os conjuntos conflito) a zonas do espaço de procura (variáveis do PSR), para ser utilizada futuramente em caso de necessidade de retrocesso, quando as mesmas variáveis tiverem de ser consideradas.[21]

Ordenação de Variáveis e de Valores

Tal como para a procura em árvore em geral (ver o Apêndice B), também para o caso da procura de solução de um PSR, a ordem pela qual se tomam em consideração as variáveis e os seus valores possíveis tem um grande impacto no tempo gasto até encontrar a solução. Escolhas incorrectas levam a retrocessos para exploração de caminhos alternativos até que um caminho apropriado seja seguido. Especificamente para o PSR, um estado de procura é definido pelas variáveis já instanciadas e os respectivos valores de instanciação; um estado sucessor é, então, definido pela escolha da próxima variável a instanciar e pelo valor de instanciação da variável. O processo de procura para resolução do PSR contém assim, em cada

---

[20] Um par de variáveis estão em conflito quando os valores delas são inconsistentes, *i.e.*, não respeitam a restrição entre as duas variáveis.

[21] Pode dizer-se que permite uma forma de procura que vai *aprendendo enquanto procura* (*learning while searching*) [Kumar 1992]; em [Dechter 1988] esta técnica é designada *registo de restrições* (*constraint recording*).

passo, dois pontos de decisão: a) escolha da próxima variável a instanciar, e b) escolha do valor para essa variável [Fox 1990], [Kumar 1992], [Sadeh 1995b], [Terpstra 1996], [Barahona 1997].[22] A optimização dinâmica da escolha da variável é apelidada de *ordenação das variáveis* e da escolha do valor apelidada de *ordenação dos valores*. Tipicamente, usam-se heurísticas nestes pontos de decisão que definem uma ordem de prioridade para escolha das variáveis e dos valores. Boas heurísticas para ordenação de variáveis e de valores ajudam a evitar retrocesso excessivo na procura da solução do PSR.

A ordem pela qual as variáveis são tomadas em consideração pode ser *estática*, se é predeterminada no início da procura, ou *dinâmica*, se é revista em cada estado de procura e tendo em consideração instanciações de variáveis anteriores. Visto ser predeterminada, a ordenação estática tem a vantagem de exigir menos computações. No entanto, a ordenação dinâmica é vantajosa por permitir identificar variáveis mais críticas em determinados estados de procura. Estudos analíticos e empíricos mostraram que a ordenação dinâmica pode ser usada para reduzir de forma significativa a quantidade de procura necessária para encontrar uma solução, em particular em certos PSR mais difíceis [Haralick 1980], [Purdom 1983], [Sadeh 1995b]. Uma heurística simples de ordenação de variáveis, denominada *re-arranjo dinâmico da procura* (*dynamic search rearrangement*),[23] é a que ordena as variáveis por ordem crescente do número de elementos do domínio (que leva à escolha da variável com o menor domínio) [Purdom 1983], [Dechter 1994]. Outra heurística é a que ordena as variáveis por ordem decrescente de número de restrições em que estão envolvidas (que leva à escolha da variável sobre a qual incidem mais restrições[24]) [Kumar 1992], [Sadeh 1995b].

Após a escolha da variável a instanciar, pode haver vários valores para a instanciação e a ordem pela qual eles são tomados em consideração tem um impacto substancial no tempo gasto até encontrar a solução. Note-se que, se o PSR tem uma solução e um valor correcto é escolhido para cada variável, então a solução será encontrada sem retrocesso [Kumar 1992]. As heurísticas de ordenação de valores procuram reduzir a probabilidade de retrocesso quando não pode ser evitado. Um exemplo de uma boa heurística de ordenação de valores é heurística do *valor menos limitativo*, ou *least constraining value* (LCV), que põe em primeiro lugar os valores que limitam menos (*i.e.*, maximizam) o número de opções que ficam disponíveis para futuras atribuições de variável (escolher o valor que limita menos) [Haralick 1980], [Kumar 1992], [Sadeh 1995b]. Um valor menos limitativo é um que se espera que participe em muitas soluções do PSR, ou um que se espera que participe em muitas soluções compatíveis com o estado actual da procura. Ao tentar, para a variável actual, os valores menos limitativos em primeiro lugar, está-se a tentar maximizar o número de valores possíveis para as variáveis que ainda estão por instanciar e, desse modo, a tentar evitar enveredar por soluções parciais que não poderão ser completadas.

---

[22] Assume-se aqui uma abordagem construtiva ao PSR. Para uma abordagem de reparação iterativa seria: a) escolha da variável a reparar, e b) escolha do valor para essa variável.

[23] Frequentemente usada associada à *verificação para a frente* [Kumar 1992].

[24] A variável mais constrangida. Estas heurísticas tentam pôr de parte o mais cedo possível, na procura, ramos da árvore de procura que conduzirão a um insucesso. Num certo sentido, tentam que o processo de procura a ter de falhar, falhe o mais cedo possível, de modo a se realizarem menos passos de procura (tentando completar soluções parciais que não podem ser completadas) e menos retrocessos mais tarde.

O Problema de Satisfação de Restrições Distribuído

Existem várias abordagens apropriadas ao PSR na sua versão distribuída (PSRD, ou *Distributed Constraint Satisfaction Problem*, DCSP). Este tipo de problema enquadra-se na área de IAD (ver o Apêndice D).

De um modo geral, no PSRD, as varáveis e as restrições estão distribuídas por diversos agentes, devendo estes coordenar decisões de modo a atribuírem valores às suas variáveis respeitando as restrições [Yokoo 1991], [Yokoo 1990], [Luo 1993a], [Luo 1993b], [Luo 1993c], [Yokoo 1993], [Armstrong 1997], [Parunak 1998a], [Tel 1999], [Yokoo 1999]. Esta distribuição da resolução do problema por vários agentes pode ser originada por imposições do próprio problema (por exemplo, se se trata de um problema de escalonamento e cada variável corresponde a uma tarefa atribuída a um agente) ou de decisões tomadas no método de resolução do problema (o problema é dividido em subproblemas por variáveis, por valores do domínio das variáveis ou por subrotinas do algoritmo em vários processos computacionais; ver [Luo 1993a], por exemplo).

Apêndice D — Inteligência Artificial Distribuída

Neste apêndice faz-se uma breve introdução à área de Inteligência Artificial Distribuída (IAD).

Introdução

Inteligência Artificial Distribuída (IAD, ou DAI, de Distributed Artificial Intelligence) é uma área de investigação relativamente recente,[25] que propõe uma abordagem à resolução de problemas considerando grupos de agentes inteligentes em vez de, como é tradicional na área de Inteligência Artificial (IA), sistemas inteligentes isolados. A IAD é uma área multi-disciplinar que recorre a várias outras, como a Computação Distribuída, o Processamento de Língua Natural, as Ciências Cognitivas, a Filosofia, a Robótica, a Biologia, a Neurofisiologia, as Ciências Sociais [Moulin 1996], [Coelho 1999]. A IAD pode definir-se como o estudo, construção e aplicação de sistemas em que vários agentes inteligentes e inter-actuantes procuram atingir um conjunto de objectivos ou realizar um conjunto de tarefas [Weiss 1999]. Obras que podem introduzir e dar uma panorâmica da IAD são, por exemplo, [Bond 1988], [Gasser 1989], [Moulin 1996], [O'Hare 1996], [Müller 1997], [Huhns 1998], [Ferber 1999], [Weiss 1999]. Outras obras de são [Russell 1995] (a IA na perspectiva dos agentes inteligentes), [Jennings 1998] (tecnologia dos agentes inteligentes e suas aplicações) e [Sichman 1998] (contém artigos sobre simulação de agentes e sistemas multi-agente).

Segundo [Ferber 1999], as motivações que levam a uma abordagem à resolução de problemas distribuída por um grupo de agentes pode vir de vários factores:

- Os problemas são distribuídos do ponto de vista físico. Por exemplo, a gestão das actividades de uma rede de transportes, do tráfego de veículo automóveis, de uma rede de produção e distribuição, de um sistema de produção industrial, envolvem actividades que ocorrem em localizações diferentes.
- Os problemas são distribuídos e heterogéneos em termos funcionais. Por exemplo, o projecto de um produto industrial complexo (um automóvel, uma aeronave), envolve vários aspectos e requer a intervenção de vasto número de especialistas, que têm apenas uma perspectiva limitada do problema.
- As redes de computadores conduzem a uma perspectiva distribuída. Actualmente, dados e capacidade de processamento estão distribuídos por um considerável número de locais (na *Internet* há já várias dezenas de milhar). Isto obriga a uma forma de pensar em termos de sistemas abertos e de inter-operabilidade de sistemas de computadores, no sentido oposto ao do computador como uma máquina puramente sequencial, estática e gerindo todos os dados de que necessita isoladamente. Arquitecturas abertas, distribuídas, heterogéneas e flexíveis, sem uma estrutura predefinida imposta são, então, adequadas.
- A complexidade dos problemas reais impõe perspectivas locais. Quando um problema é demasiado extenso para ser analisado como um todo (por exemplo, por estarem envolvidos um elevado número de parâmetros e de restrições), soluções distribuídas baseadas em perspectivas locais podem permitir resolvê-lo de forma mais rápida.[26]

---

[25] Década de 90, muito embora a investigação se tenha desenvolvido já a partir do final da década de 70.

[26] Muitos dos problemas reais são problemas difíceis, no sentido em que não é conhecida uma forma algorítmica eficiente de encontrar a solução, recorrendo-se, por essa razão, à procura heurística (ver o Apêndice B); o PSR enquadra-se também nesta classe de problemas (ver o Apêndice C). Resultados experimentais (ver [Clearwater 1991]) comprovam que a cooperação entre múltiplos agentes solucionadores de problemas, com múltiplas perspectivas, num contexto de resolução de problemas distribuído, pode originar uma "implosão combinatória" que permite dominar a complexidade na resolução de problemas difíceis.

- Existe a necessidade de os sistemas de computadores serem capazes de se adaptar a mudanças na estrutura (do próprio sistema) ou no ambiente. Não basta projectar sistemas de computadores eficientes, fiáveis e precisos. São necessários sistemas adaptáveis a mudanças (do sistema operativo, da base de dados, da interface gráfica, inclusão de novos programas no sistema) e que possam evoluir para acomodar necessidades crescentes (funcionalidades adicionais, alterações no uso, integração com outros tipos de programas). Adicionalmente, a engenharia de programas encaminhou-se no sentido do projecto de programas usando conceitos de unidades autónomas, ou módulos, inter-actuantes, desenvolvidos independentemente, fáceis de testar e reutilizáveis (veja-se a tecnologia das linguagens de programação orientada por objectos).

Poderia juntar-se, a estes factores motivacionais, um factor de outra natureza: os sistemas baseados em múltiplos agentes podem proporcionar uma forma natural de caracterizar e estudar sistemas inteligentes e ajudar a compreender a relação entre a inteligência e a interacção entre agentes inteligentes naturais, como os seres humanos, organizados em grupos, sociedades, economias [Weiss 1999].

É usual distinguir, na IAD, duas áreas principais de investigação: a Resolução Distribuída de Problemas (RDP, ou *Distributed Problem Solving*, DPS) e os Sistemas Multi-Agente (SMA, ou *Multi-Agent Systems*, MAS). Em RDP, a resolução de um problema é distribuída por um número de módulos, nós, ou agentes cooperantes, que partilham conhecimento do problema e da solução em desenvolvimento. A decomposição do problema, a resolução de subproblemas e a síntese de soluções dos subproblemas são fases da resolução distribuída [Smith 1988a].[27] Em SMA, a preocupação é o comportamento de uma colecção de agentes (em geral preexistente) na resolução de um problema dado. Tipicamente, um *sistema multi-agente* é entendido como uma rede de solucionadores de problemas fracamente acoplada. Estes solucionadores de problemas, apelidados de agentes, são autónomos e trabalham em conjunto para resolver problemas que ultrapassam as suas capacidades individuais [Moulin 1996]. Cada agente tem informação incompleta e capacidades limitadas, os dados e o controlo do sistema estão distribuídos e os agentes operam de modo assíncrono. Os agentes de um sistema multi-agente podem ser heterogéneos, *i.e.*, podem ter capacidades diferentes de resolução de problemas e mesmo diferir nos objectivos individuais.

O trabalho de investigação em IAD pode enquadrado, de acordo com os organizadores dos *workshops* ATAL (*Agent Theories, Architectures, and Languages*),[28] em três grandes blocos da IAD: as teorias, as arquitecturas e as linguagens [Müller 1997], [Wooldridge 1995]. No contexto das teorias discute-se o significado de um agente e o uso de formalismos para os representar e raciocinar acerca das suas propriedades; o contexto das arquitecturas tem a ver com os modelos computacionais dos agentes do ponto de vista da engenharia da programação (projecto de sistemas, computadores e programas, de acordo com as propriedades especificadas nas teorias); o contexto das linguagens diz respeito às ferramentas de programação necessárias para realizar agentes e realizar experimentação com agentes.

---

[27] A RDP é distinta do Processamento Distribuído, em que o objectivo é projectar redes de computadores capazes de executar várias tarefas diferentes, tirando partido dos recursos de computação. Nesta última área assume-se, em geral, que existe *a priori* um problema bem definido, já decomposto em subproblemas, centrando-se a preocupação principal em questões como o método de ligação entre os nós das redes de computadores, a distribuição óptima das tarefas pelos nós, a afectação de recursos e evitar bloqueios.

[28] *Workshop* da conferência *European Conference of Artificial Intelligence* (ECAI).

Agentes e Modelos de Agentes

Existe uma multiplicidade de usos e definições da palavra agente, mesmo no contexto da IAD. Certas definições podem abranger entidades virtuais (programas de computador) e entidades físicas, biológicas ou não (seres vivos, robôs), outras limitam os agentes a programas de computador (ver definições de vários autores resumidas em [Franklin 1997], ou em [Coelho 1999]). Por exemplo, um agente pode ser definido como uma entidade autónoma, situada num ambiente, que percepciona o ambiente e actua nele de forma planeada ao longo do tempo de modo a atingir os seus objectivos [Russell 1995], [Franklin 1997], [Wooldridge 1999]; ou, um agente é um objecto pro-activo, *i.e.*, uma entidade que tem a encapsulação de dados e procedimental de um objecto de programa, controlo próprio e a capacidade de se executar autonomamente sem ser invocado do exterior [Parunak 1998b].

A maior parte das caracterizações de agentes coloca ênfase nos aspectos de autonomia, inteligência, pro-actividade e capacidades de interacção dos agentes individuais [Franklin 1997], [Ferber 1999], [Weiss 1999].

Por exemplo, em [Franklin 1997] coloca-se a tónica na autonomia e resume-se um conjunto de propriedades que podem ser atribuídas a agentes autónomos e que podem servir para os classificar de forma genérica, conforme exibem essas propriedades em maior ou menor grau. Estas propriedades são a reactividade (um agente é reactivo se percepciona e actua, respondendo a tempo a mudanças no ambiente), a autonomia (um agente é autónomo se tem controlo sobre o seu próprio comportamento e pode actuar sem a intervenção de outras entidades, seres humanos incluídos), a pro-actividade (um agente é pro-activo se actua guiado por objectivos, actuando não apenas em resposta ao ambiente[29]), a persistência (um agente é persistente se é um processo em execução de forma temporalmente contínua), a *c*omunicabilidade (um agente tem esta propriedade se é capaz de comunicar com outros agentes, humanos incluídos), a aprendizagem (um agente aprende se se adapta, mudando o seu comportamento com base em experiência prévia), a mobilidade (um agente é móvel se é capaz de se transportar de uma computador para outro), a flexibilidade (um agente é flexível se as suas acções não são pré-programadas), a capacidade de incarnar um personagem (ser credível e exibir estado emocional). Para descrever um agente autónomo é sugerida, em [Franklin 1997], uma caracterização em termos do tipo de ambiente em que o agente está inserido, das capacidades sensoriais (de percepção do ambiente) do agente, das acções possíveis do agente, das motivações e da arquitectura do agente para escolha de acções.

Em [Wooldridge 1999] distinguem-se os agentes inteligentes dos agentes não inteligentes, pelas capacidades flexíveis de actuação autónoma que um agente inteligente dispõe para atingir os objectivos para os quais foi projectado. A flexibilidade de actuação envolve a reactividade, a pro-actividade e a sociabilidade (capacidade de interagir com outros agentes, humanos incluídos, para atingir objectivos). Distinguem-se também agentes de objectos de programa, pela capacidade de decisão de um agente actuar ou não actuar quando para isso é solicitado por outro agente,[30] pela capacidade de comportamento flexível (reactivo, pro-activo e social) e

[29] Estes aspectos aparecem frequentemente associados a propriedades como inteligência [Weiss 1999] (um agente é inteligente se é guiado por objectivos individuais que tenta, de algum modo optimizar) ou racionalidade [Coelho 1999] (um agente é racional quando opta por realizar acções que sabe que o levarão a atingir os seus objectivos).

[30] No contexto da tecnologia actual da programação orientada por objectos, distinguem-se objectos activos de objectos passivos, pelo facto de os primeiros poderem exibir comportamento (*i.e.*, alteração do seu estado) sem

de os sistemas multi-agente poderem ter múltiplas sequências de processamento, em que cada agente pode ter um ou mais caminhos de controlo. Adicionalmente, faz-se a distinção entre sistemas periciais, como tradicionalmente entendidos na IA (*i.e.*, como sistemas capazes de resolver problemas ou dar conselhos no contexto de um determinado domínio específico de conhecimento) e agentes. Ao contrário de um agente, um sistema pericial não interage, em geral, directamente com um ambiente (normalmente interage com um utilizador); também, não é frequente haver cooperação entre vários sistemas periciais, ao contrário do que se preconiza para os agentes de um sistema multi-agente.

Em [Russell 1995] distinguem-se tipos de agentes diferentes por graus de sofisticação progressivamente maiores do programa que realiza o agente. Os agentes mais elementares são os agentes reflexos (ou reactivos) simples, que dispõem de um mecanismo interno simples para associar percepções a acções no ambiente, como a consulta de uma simples tabela, ou o recurso a regras do tipo condição-acção. Têm portanto, um comportamento do tipo estímulo-resposta. A seguir vêm os agentes reflexos que mantêm uma forma limitada de representação interna actual do ambiente, de modo a poderem percepcionar mudanças nele e escolher, de forma mais precisa, as suas acções futuras. O grau de sofisticação seguinte é o do agente baseado em objectivos, ou deliberativo, ou cognitivo. Este tipo de agente, para além de manter um modelo do ambiente, sobre o qual pode raciocinar, mantém informação interna acerca dos seus objectivos (que descrevem os estados, para o agente e o ambiente, pretendidos) e raciocina acerca dos efeitos das suas acções possíveis, para melhor escolher a acção ou sequência de acções que o levam a atingi-los. Estes agentes em que a escolha das acções é mais pensada, são, em geral, menos eficientes mas mais flexíveis do que os agentes reflexos. Um outro tipo de agente mais sofisticado é o agente baseado na utilidade. Este tipo agente, para além de ser baseado em objectivos, faz uso de uma função de utilidade que traduz preferências na escolha de soluções possíveis para atingir os objectivos (por exemplo, esta função de utilidade pode permitir um desempate no caso de haver vários objectivos em conflito, ou objectivos com diferentes graus de importância e diferentes graus de certeza quanto a poderem ser atingidos).

Em [Russell 1995] distinguem-se também vários tipos de ambientes de agentes como acessíveis/inacessíveis (dependendo do grau de acesso dos sensores do agente aos aspectos relevantes do estado do ambiente), determinísticos/não-determinísticos (conforme o estado seguinte do ambiente é ou não completamente determinado pelo estado actual e pelas acções do agente), episódicos/não-episódicos (conforme o ciclo de percepção-acção do agente se pode limitar ou não a episódios independentes), estáticos/dinâmicos (conforme o ambiente pode ou não mudar enquanto o agente está a deliberar) e discretos/contínuos (conforme há ou não um número discreto e limitado de percepções e acções possíveis).

Em [Parunak 1996a], [Parunak 1998b] e [Parunak 1998e] descrevem-se aplicações e discute-se a aplicabilidade da tecnologia de agentes à indústria. Em [Parunak 1996a] e [Parunak 1998b] elabora-se uma taxonomia de agentes, baseada numa síntese de outras taxonomias de vários autores, mas centrada em investigação e desenvolvimento de aplicações industriais da tecnologia dos SMA. Nesta taxonomia os SMA são classificados de acordo com:

- A função - Dizendo respeito ao tipo de tarefas que os agentes realizam. Por exemplo, funções típicas na indústria para os agentes são o escalonamento, o controlo, a

---

a intervenção de outros objectos [Booch 1994] exibindo, portanto, um certo grau de autonomia. Para os agentes coloca-se, comparativamente aos objectos activos, um ênfase maior no aspecto da autonomia.

monitorização, o diagnóstico, o projecto de produtos, a distribuição, a integração da informação, a logística.

- A arquitectura individual dos agentes - Dizendo respeito ao modelo do agente do ponto de vista da engenharia da programação. Aspectos como heterogeneidade, grau de sofisticação das capacidades de raciocínio dos agentes individuais e como modelam o mundo exterior e o tempo são aqui considerados. Com respeito à heterogeneidade, os agentes de um SMA poderão ter a mesma arquitectura ou arquitecturas diferentes. O grau de sofisticação das capacidades de raciocínio pode variar desde capacidades puramente reactivas, que originam um agente rápido mas menos flexível, a capacidades de planeamento prévio de acções, originando um agente com acções mais pensadas (mais lento, mas mais adaptável). Um agente pode ainda manter um modelo mais ou menos explícito e elaborado do seu mundo e raciocinar sobre o modelo; pode apenas representar o mundo no estado actual ou, adicionalmente, representá-lo também no estado ou estados em que ele pretende que o mundo esteja.
- A arquitectura do sistema - Dizendo respeito ao ambiente de apoio ao conjunto dos agentes do SMA. Este aspecto é subdividido na forma de comunicação entre os agentes do SMA, no protocolo de interacção (estrutura do diálogo, ou conversação) entre os agentes e em se o sistema é híbrido ou não (*i.e.*, se coexistem nele agentes humanos e agentes artificiais). Com respeito à forma de comunicação, são distinguidas duas formas: através de modificações físicas operadas, ou percepcionadas, no ambiente (por exemplo, presença ou ausência de um objecto num local), através da comunicação por sinais (por exemplo, a comunicação por sinais electrónicos convencionados, para controlo) e através da comunicação por mensagens (distinguindo-se se as mensagens são endereçadas a um único agente ou difundidas por todos os agentes do sistema, ou ainda se persistem ou não depois de enviadas). Relativamente aos protocolos de interacção referem-se vários. Nos protocolos mais simples, os agentes reagem, sem discussão, a mensagens ou comandos de outros agentes. Protocolos mais sofisticados são os de votação (os agentes enviam opiniões a um agente moderador, que as pondera de modo a decidir um curso de acção futura), de negociação (como o protocolo da rede de contratos, descrito mais adiante), propagação de restrições (ver o Apêndice C) ou protocolos mais elaborados baseados nos actos de discurso.

Conforme o que foi até aqui descrito, de acordo com as perspectiva de vários investigadores, e do ponto de vista do agente individual, o tipo de agente mais sofisticado (mais autónomo, mais flexível, mais inteligente) é, sem dúvida, o agente que, mantendo internamente uma representação simbólica e explícita do seu mundo e informação dos seus objectivos, pode raciocinar sobre os possíveis estados do mundo e sobre os efeitos das suas possíveis acções no mundo e planear essas acções de modo a melhor atingir os objectivos. Estes agentes são frequentemente apelidados de agentes cognitivos (por oposição a agentes reactivos).

Relativamente aos agentes cognitivos existem duas escolas de investigação importantes: a escola americana e a escola dos agentes racionais (ver, por exemplo, [Ferber 1999]). A escola americana baseia-se mais em sistemas multi-agente com um pequeno número de agentes fortemente baseados em conhecimento, com uma influência forte da IA (sistemas periciais, sistemas baseados em conhecimento). A escola dos agentes racionais, orienta-se para a formalização dos agentes autónomos, em especial no que respeita a modelos mentais do agente.

No contexto desta última escola, o desenvolvimento de modelos mentais de agentes racionais tem vindo a adquirir um grau de sofisticação elevado. Um tipo de arquitecturas de agente

racional apelidada de arquitectura BDI (de *Belief-Desire-Intention*, ou Crença-Desejo-Intenção), combina capacidades reactivas com capacidades cognitivas para obter um agente capaz de raciocínio prático. Estas arquitecturas são adequadas a agentes que devem actuar em ambientes dinâmicos e não-determinísticos (incluindo ambientes em que o agente pode ter de contar com as acções de outros agentes), em que o desempenho em tempo real é importante. Nas teorias que dão suporte a este tipo de agentes, manifesta-se um posicionamento importante, que é o de se assumir que é legítimo atribuir qualidades, ou atitudes mentais, a agentes artificiais à semelhança das que se atribuem agentes humanos. A questão de ser ou não legítimo atribuir qualidades mentais — como crenças, intenções, vontade, consciência, normalmente atribuídas a humanos — a agentes artificiais tem sido motivo de aceso debate entre filósofos e investigadores de IA (ver [McCarthy 1979], [Searle 1984], [Dennett 1987], [Shoham 1993]). Um ponto de vista pragmático é o que foi inicialmente expresso em [McCarthy 1979], que defende ser legítimo atribuir qualidades mentais a máquinas e que isso é útil e apropriado se nos ajuda a compreender a estrutura da máquina, o seu comportamento e como repará-la ou aperfeiçoá-la. Muita investigação teórica na área de IAD sobre os agentes racionais centra-se, então, em torno do grupo de qualidades mentais mais adequado e dos formalismos de suporte que permitem modelar estas qualidades mentais [Coelho 1999]. Em [Shoham 1993] propõe-se mesmo um paradigma, apelidado de programação orientada por agentes (*agent oriented programming*, AOP), como uma especialização do paradigma da programação orientada por objectos. Segundo este paradigma, os agentes são módulos de programa e são constituídos por componentes como crenças, capacidades e decisões; a computação consiste em os agentes competirem e cooperarem entre si trocando informação, pedidos, ofertas, aceitações, rejeições através de comunicação baseada nos actos de discurso (ver mais adiante).[31]

No modelo BDI (ver, por exemplo, [Haddadi 1996], [Georgeff 1998]) as qualidades mentais são as crenças, os desejos e as intenções (o agente mantém internamente uma representação actualizada das suas crenças, desejos e intenções). São três componentes — informacional, motivacional e deliberativo, respectivamente — que, em conjunto, constituem o estado interno, ou estado cognitivo, do agente e que lhe permitem escolha da acção seguinte adequada. Nas crenças é representado o estado actual do ambiente, devidamente actualizado pelas percepções. Os desejos e as intenções têm a ver com estados futuros possíveis do mundo. Os objectivos a atingir, bem como as prioridades ou utilidades associadas, são representados nos desejos. O terceiro componente, as intenções, representa o curso actual de acção escolhido no qual o agente está empenhado. Este componente permite, em situações em que o estado do ambiente muda rapidamente (requerendo-se, da parte do agente, uma resposta rápida à mudança), que haja um equilíbrio entre reconsiderar (e eventualmente abandonar), ou não reconsiderar (e prosseguir), o curso de acção escolhido.

As arquitecturas BDI foram aplicadas à resolução de problemas em domínios reais como o controlo de tráfego aéreo, gestão de processos de negócio e modelação de combate aéreo, com o desenvolvimento de produtos comerciais de eficácia comprovada [Georgeff 1998]. No campo teórico, os formalismos de suporte ao modelo BDI seguem vias que, ou recorrem ao uso de lógicas modais normais (*i.e.*, lógicas proposicionais ampliadas com operadores de necessidade e possibilidade), ou ao uso de representações que permitem uma descrição mais

[31] Por exemplo, em [Ventura 1999] descreve-se uma linguagem para programar equipas de agentes com uma aplicação ao jogo de futebol de robôs.

directa do estado cognitivo dos agentes (ver, por exemplo, [Cohen 1990b] e [Rao 1991] no primeiro caso, ou [Konolige 1993] no segundo).

Num extremo oposto ao das escolas dos agentes cognitivos posiciona-se, na IAD, a abordagem da escola dos agentes reactivos. Os agentes reactivos são agentes simples que não mantêm representação simbólica alguma do ambiente e actuam através de um comportamento reflexo, do tipo estímulo-resposta. Os investigadores que defendem esta abordagem, afirmam que não é necessário que os agentes de um sistema multi-agente tenham inteligência individual, para que o sistema exiba globalmente um comportamento inteligente (ver, por exemplo, [Agre 1987], [Maes 1989], [Schoppers 1989], [Brooks 1991]). Na arquitectura de agente reactivo apelidada de *arquitectura de subsunção* (proposta por Brooks e aplicada a robôs), por exemplo, a tomada de decisão do agente é realizada por um conjunto de módulos de comportamento, concretizados por máquinas de estado finitas, que mapeiam continuamente as percepções para as acções correspondentes. Estes módulos são organizados hierarquicamente e, em situações em que mais de um módulo de comportamento é aplicável, prioridade é dada aos módulos dos níveis mais baixos da hierarquia. A ideia subjacente é a de que os módulos nos níveis mais altos correspondem a comportamentos mais abstractos, que devem ser inibidos quando é necessária uma reacção prioritária, operada por um módulo de nível mais baixo.

O aspecto mais interessante deste tipo de agentes não é a arquitectura, mas o facto de interacções simples entre agentes reactivos, ou entre componentes do mesmo agente, poderem levar à emergência de padrões complexos de actividades [Ferber 1996]. Por esta razão, para além da rejeição de representações simbólicas, a concepção reactiva dos agentes caracteriza-se por associar o comportamento inteligente e racional ao ambiente e à interacção com o ambiente do agente (outros agentes incluídos) e pela emergência daquele tipo de comportamento a partir da interacção entre elementos de comportamento simples.

Uma outra área de investigação, que se cruza com a área dos agentes reactivos, é a da vida artificial. Na área da vida artificial procura-se compreender e modelar sistemas com vida, *i.e.*, sistemas capazes de sobreviver, de se adaptar e de se reproduzir num ambiente possivelmente hostil. Esta área procura abstrair os princípios de organização dos seres vivos naturais, de modo a poder realizar seres artificiais, através de simulações com programas de computador, estudá-los e testá-los [Ferber 1999].

Coordenação Inter-Agente

Na resolução de problemas por um sistema multi-agente um aspecto muito importante é o da coordenação, *i.e.*, a sincronização das actividades dos agentes de modo a reduzir actividades desnecessárias, ou improdutivas, na resolução de problemas (por exemplo, reduzir a competição por recursos, evitar situações de bloqueio). Um outro aspecto, que se pode incluir na coordenação, é o do comportamento coerente, o comportamento do sistema como uma unidade para obter uma solução de boa qualidade eficientemente e de modo a acomodar situações de falha.

Formas gerais de coordenação entre agentes são a cooperação e a competição [Huhns 1999]. A cooperação é uma forma de coordenação entre agentes não antagónicos (com objectivos comuns); na competição os agentes são auto-interessados (têm objectivos possivelmente divergentes) e competem para atingir os seus objectivos. Exemplos de formas cooperativas e de formas competitivas de cooperação são, respectivamente, o planeamento multi-agente e a negociação.

A cooperação através de planeamento multi-agente baseia-se em alinhar o comportamento por meio de objectivos comuns e uma divisão de trabalho explícita. A geração de planos multi-agente (planos para execução por múltiplos agentes) pode ser feita por um único agente (planeamento multi-agente centralizado), responsável pela construção de um plano que especifica as acções dos agentes ou, quando não há um único agente com uma perspectiva global do problema, dividido por vários agentes (planeamento multi-agente distribuído) [Moulin 1996]. [Cammarata 1988] e [Martial 1992], por exemplo, descrevem formas de cooperação que se enquadram no planeamento multi-agente. Também o protocolo de negociação multi-estágio descrito em [Conry 1988], o planeamento global parcial (*partial global planning*, PGP) em [Durfee 1988] e em [Decker 1995], os mecanismos de controlo cooperativo em [Lesser 1988] e a procura heurística distribuída com restrições em [Sycara 1991b] podem ser incluídos como formas de planeamento multi-agente.

A negociação pode definir-se como o processo de melhoria de acordos entre agentes, por redução de inconsistência e incerteza, acerca de pontos de vista comuns, ou planos, através de um protocolo bem definido. Um protocolo de negociação muito conhecido é o protocolo da rede de contratos (*contract net*) [Smith 1988b], em que um conjunto de tarefas é distribuído por um grupo de agentes capazes de as realizar (ver mais adiante). Na negociação assume-se que os agentes são, pelo menos até certo grau, cooperativos. No entanto existem modelos da negociação entre agentes que assumem não cooperação e em que há necessidade de recorrer a um agente mediador (por exemplo, em [Sycara 1989]), que tenta minimizar os conflitos entre os objectivos dos agentes. Em [Zlotkin 1990] propõe-se um protocolo de negociação entre agentes racionais que pode abranger cooperação e não cooperação. Nesta abordagem é possível aos agentes inter-actuantes, mesmo em situações de conflito, cooperem até certo ponto.

Os sistemas apelidados de sistemas de mercado têm uma forma competitiva de coordenação diferente da negociação, que é a do ajustamento mútuo.[32] Nos sistemas de mercado os agentes estabelecem acordos, para partilha de recursos que controlam, para atingir objectivos mútuos. Os acordos baseiam-se na ideia da compra e venda e em preços atribuídos aos recursos. Exemplos de aplicação destes sistemas são descritos em [Wellman 1993], [Kis 1996], [Parunak 1998a] e [Walsh 1998].

Comunicação entre Agentes

Regra geral, a coordenação inter-agente num sistema multi-agente tem de envolver alguma forma de comunicação, mais ou menos sofisticada, entre os agentes, de modo a permitir a troca de informação necessária à coordenação. Formas não primitivas de comunicação (*i.e.*, mais do que simples sinais) podem ocorrer de duas maneiras: através de memória partilhada e através de troca de mensagens.

Nos sistemas com comunicação através de memória partilhada, também apelidados de sistemas de quadro preto (*blackboard*), existe uma área de memória acessível a todos os agentes, o

[32] Este mecanismo de coordenação está inerentemente ligado a uma estrutura organizacional descentralizada, a estrutura de mercado. O mecanismos de coordenação podem ser vistos com referência à estrutura organizacional. Dos três mecanismos de coordenação distinguidos em [Mintzberg 1979], que são a supervisão directa, o ajustamento mútuo e a padronização, os dois primeiros podem ser directamente ligados às estruturas organizacionais de hierarquia e de mercado, respectivamente (ver [Malone 1990]).

quadro preto, onde cada agente pode colocar mensagens e resultados parciais da resolução de um problema e aceder a essa informação lá colocada pelos outros agentes (ver por exemplo [Erman 1980], [Corkill 1988] e [Hayes-Roth 1988]). Esta área de memória é frequentemente dividida em níveis, que dizem respeito a níveis de abstracção diferentes do problema em questão; agentes que trabalham a um determinado nível de abstracção do problema têm acesso ao nível correspondente no quadro preto e aos níveis adjacentes. Deste modo, a informação gerada num nível pode ser passada para níveis mais acima, ou para níveis mais abaixo. Os sistemas de quadro preto são uma maneira simples de conjugar contribuições de diferentes agentes especialistas cooperantes na resolução de um problema. A actuação dos agentes envolvidos, frequentemente apelidados de fontes de conhecimento, pode ser desencadeada por eventos do quadro preto e eventos do exterior. Os eventos do quadro preto incluem adição, modificação ou remoção de informação no quadro preto. Um agente especial controla o curso de resolução por activação, em cada ciclo, de um agente que tenha sido escolhido, mediante uma informação do benefício geral da contribuição do agente para a resolução do problema. Um controlo deste tipo, permite ao sistema mudar rapidamente a sua estratégia de resolução do problema, adaptando-a ao curso da resolução e aos eventos que vão surgindo, *i.e.*, ter uma estratégia oportunista.

Nos sistemas com comunicação através de troca de mensagens, os agentes trocam informação entre si directamente por mensagens explícitas, num formato padronizado. Este método é particularmente adequado quando os agentes estão geograficamente descentralizados e ligados por uma rede electrónica. A quantidade de informação trocada e o número de trocas são, em geral, limitados, já que acarretam custos computacionais elevados. Vários sistemas multi-agente empregam esta forma de comunicação (por exemplo, [Lesser 1988], [Cammarata 1988], [Smith 1988b], já referidos atrás).

A comunicação entre agentes pode ser considerada, em parte percepção (receber mensagens) e em parte actuação (enviar mensagens). Num sistema multi-agente puramente baseado em computadores, aqueles podem mesmo ser os únicos meios de interacção de que os agentes dispõem. Na comunicação por mensagens, os agentes poderão ter capacidades de comunicação que podem ir desde a simples troca e compreensão de mensagens isoladas até ao envolvimento em diálogos, ou conversações, mais ou menos complexos. Estes diálogos são constituídos por várias mensagens trocadas de forma estruturada, em sequências de mensagens padronizadas [Huhns 1999]. Tipicamente, as mensagens (tipos de mensagens, formatos, conteúdos) e as sequências possíveis de mensagens são padronizados em convenções denominadas protocolos. Os protocolos de comunicação permitem aos agentes trocar e compreender mensagens. Os protocolos de interacção permitem ter trocas estruturadas de mensagens, *i.e.*, conversações.

Mesmo assumindo um papel passivo, um agente deve, no mínimo, ser capaz de receber informação e responder a perguntas. Se necessário, deve também poder comunicar informação e fazer perguntas, assumindo um papel activo na comunicação (deste modo o agente pode, potencialmente, controlar outro agente, levando-o a responder a perguntas ou a aceitar informação comunicada). Os tipos de mensagens mais simples que se podem ter são, então, a asserção (*assertion*) e a pergunta (*query*). Note-se que se está a falar aqui da comunicação de alto nível, não se envolvendo aspectos técnicos, de baixo nível da comunicação. Os protocolos de comunicação são normalmente definidos a vários níveis, no mais baixo especificando-se o método de inter-conexão, num nível intermédio o formato da informação nas mensagens e no nível de topo o significado, ou semântica, das mensagens (que se refere não apenas ao conteúdo mas também ao tipo de mensagem).

Para a comunicação por mensagens entre agentes toma-se frequentemente como base a teoria dos actos de discurso da comunicação humana. Esta teoria vê a comunicação de frases como actos, ou acções, de vários tipos, como pedidos, sugestões, compromissos, respostas [Searle 1984]. Um acto de discurso tem três aspectos: locução, ilocução e perlocução. A locução tem a ver com a produção física da frase; a ilocução tem a ver com o significado intencionado pelo emissor da frase; a perlocução tem a ver com a acção que resulta da locução.[33] Para comunicação entre agentes não deve existir qualquer ambiguidade quanto ao tipo de uma mensagem, *i.e.*, de qual a é "força ilocucionária" da "frase" (o que, na comunicação humana, nem sempre acontece). Na teoria dos actos de discurso usa-se o termo "performativa" para identificar a força ilocucionária de uma frase. Exemplos de verbos performativos são *tell* (dizer), *request* (pedir), *question* (perguntar), *inform* (informar). Os tipos de força ilocucionária podem ser classificados, de um modo muito geral, em assertivas (afirmação de um facto), directivas (ordens, ou comandos), comissivas (compromissos) e expressivas (expressão de emoção) [Huhns 1999]. O tipo de força ilocucionária restringe a semântica do acto de comunicação.

Um protocolo de comunicação proposto pela instituição americana DARPA (*Defense Advanced Research Projects Agency*), muito conhecido, é o KQML (de *Knowledge Query and Manipulation Language*). Este protocolo faz parte de uma iniciativa mais abrangente (apelidada de *Knowledge-Sharing Effort*) para desenvolver infra-estruturas técnicas de suporte à inter-operabilidade e à partilha de conhecimento entre sistemas multi-agente heterogéneos (o que inclui partilha de conhecimento declarativo, técnicas de resolução de problemas e raciocínio) [Neches 1991], [Finin 1992], [Patil 1998]. Desta iniciativa surgiram propostas de padrão de sintaxe para exprimir conhecimento, a linguagem KIF (de *Knowledge Interchange Format*) e para a comunicação inter-agente, o KQML. Outro componente da mesma iniciativa tem a ver com o desenvolvimento de ontologias (um vocabulário base, com especificações de objectos, conceitos, classes, funções e relações) de determinados domínios de conhecimento, expressas na linguagem KIF.

O protocolo KQML pode ser descrito pela seguinte estrutura básica de mensagem [Finin 1993]:

```
(<performativa-KQML>
    :sender     <palavra>
    :receiver   <palavra>
    :language   <palavra>
    :ontology   <palavra>
    :content    <expressão>
    ... )
```

Em que `<performativa-KQML>` identifica a performativa (o KQML tem cerca de `40` performativas; por exemplo, `ask-if`, `broadcast`, `forward`, `register`, `reply`, `subscribe`, `tell`, `unregister`, `untell`, são apenas algumas delas; ver [Finin 1993]), cuja semântica é independente do domínio da informação comunicada. Os campos, ou argumentos, da mensagem são assinalados (precedidos) por palavras chave (`:sender`,

[33] Por exemplo, se alguém diz a outrem "passa-me a chave de parafusos" este acto é composto pelos sons da frase (a locução), a intenção subjacente na frase, que é um pedido ou uma ordem (a ilocução) e a acção do interlocutor de dar (ao emissor da frase) a chave de parafusos.

`:receiver`, `:language`, `:ontology`, `:content`, ou ainda `:reply-with`, `:in-reply-to` e `:force`; ver [Finin 1993]). Os campos `:sender` e `:receiver` identificam o agente emissor da mensagem e o agente a que ela é endereçada, respectivamente. A semântica da mensagem é definida pelos campos `:content`, o conteúdo da mensagem, `:language`, a linguagem em que é expresso o conteúdo e `:ontology`, o vocabulário de base usado. Para compreender a mensagem, agente o receptor tem compreender a linguagem e ter acesso à ontologia.

Um outro protocolo de comunicação semelhante é o da linguagem ACL (de *Agent Communication Language*). Trata-se de um padrão mais recente que o KQML, desenvolvido pela organização FIPA (*Foundation for Intelligent Physical Agents*; ver [FIPA 1998a] e [FIPA 1998b]; a linguagem ACL é descrita em [FIPA 1998b]).

Interacção entre Agentes

Enquanto que os protocolos de comunicação especificam mecanismos para os agentes comunicarem simples mensagens, os protocolos de interacção especificam como é governada a troca de uma série de mensagens, isto é, quais são as sequências de mensagens possíveis numa conversação.[34] No caso em que os agentes do sistema multi-agente são agentes com objectivos que podem entrar em conflito, o objectivo do protocolo é normalmente o de maximizar valores de utilidade dos agentes; nos casos em que os agentes têm objectivos similares, ou têm problemas comuns (como na RDP), o objectivo dos protocolos é manter globalmente um desempenho coerente dos agentes sem violar a autonomia destes (sem um controlo global explícito) [Huhns 1999].

Um paradigma de protocolo de interacção para coordenação para sistemas multi-agente com múltiplos objectivos distribuídos pelos agentes é o dos Compromissos/Convenções [Jennings 1996]. Este paradigma preconiza definir um grafo de dependências de objectivos, atribuir regiões particulares do grafo aos agentes apropriados, controlar decisões sobre que regiões do grafo explorar, percorrer o grafo e assegurar que um percurso com sucesso seja relatado. Algumas destas actividades podem ser mais colaborativas, outras poderão ser executadas por agentes isolados. Os compromissos são garantias de tomar um certo curso de acção e fornecem um grau de previsibilidade, de modo que um agente possa tomar em consideração as acções futuras de outros agentes, ao tomar em conta dependências inter-agente, restrições globais, ou conflitos de utilização de recursos. As convenções são um meio para gerir os compromissos quando as circunstâncias mudam; elas restringem as condições sob as quais os compromissos devem ser reavaliados e especificam as acções que devem ser realizadas, *i.e.*, manter, alterar ou abandonar os compromissos [Huhns 1999].

Um protocolo de interacção para resolução cooperativa de problemas muito conhecido e aplicado é o *contract net*, ou rede de contratos [Davis 1988], [Smith 1988b], já anteriormente

[34] O aspecto da interacção entre agentes é muito importante pois o comportamento global de um sistema multi-agente pode ser quase completamente condicionado pela interacção. Existe mesmo investigação sobre o desenvolvimento de técnicas e ferramentas para a construção de sistemas multi-agente com base na coordenação e nas interacções pretendidas entre os agentes, o que é apelidado de paradigma da programação orientada por interacções (*interaction oriented programming*, IOP, ver [Singh 1998]). A ideia, aparentemente similar, de permitir integrar a especificação de protocolos com a especificação de aspectos funcionais do sistema multi-agente, é expressa em [d'Inverno 1998].

referido. Este protocolo, que se descreve a seguir, baseia-se nos mecanismos de contrato usados normalmente para negociar a troca de bens e serviços.

Existe um agente, apelidado de *manager* (gestor), que tem uma tarefa para ser realizada. Os outros agentes, que podem realizar a tarefa, são apelidados de potenciais *contractors* (empreiteiros). Do ponto de vista do *manager*, as actividades respeitantes ao protocolo consistem em:

1. Anunciar a tarefa que deve ser realizada;
2. Receber e avaliar propostas de potenciais *contractors*;
3. Dar a tarefa a realizar ao *contractor* apropriado;
4. Receber e sintetizar resultados.

Do ponto de vista de um *contractor*, as actividades respeitantes ao protocolo consistem em:

1. Receber anúncios de tarefas;
2. Avaliar a sua capacidade para responder;
3. Responder (recusar, ou fazer uma oferta);
4. Se oferta feita foi aceite realizar a tarefa;
5. Relatar os resultados da realização da tarefa.

Os papéis dos agentes não são predeterminados e qualquer agente pode actuar como *manager* ou como *contractor*. Isto permite a decomposição de tarefas, com um *contractor* para uma dada tarefa a poder actuar também como *manager* e poder pedir ajuda a outros agentes para a realização de partes (sub-tarefas) da tarefa, com a consequente formação de uma hierarquia de controlo para a partilha de tarefas e síntese de resultados. Para além desta flexibilidade, o protocolo tem a vantagem de acomodar elegantemente situações de falha. Por exemplo, se um *contractor* não é capaz de fornecer uma solução satisfatória, o *manager* pode procurar outros potenciais *contractors* para a tarefa.

Extensões ao protocolo de interacção *contract net*, desenvolvidas pela FIPA, são o *FIPA-Contract-Net* (um *contract net* ampliado com dois actos de discurso adicionais) e o *FIPA-Iterated-Contract-Net* (semelhante ao *FIPA-Contract-Net*, mas permitindo vários ciclos das actividades de protocolo descritas acima). *FIPA-Auction-English* e *FIPA-Auction-Dutch* são outros exemplos de protocolos de interacção, também originados pela FIPA, baseados nos mecanismos de leilão (ver [FIPA 1998b]).

Referências

Referências

Aarts 1989 — Aarts, E.H.L.; Korst, J.H.M., *Simulated Annealing and Boltzman Machines*, Wiley, Chichester, 1989.

Abramson 1991 — Abramson, D., Constructing *School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms*, Management Science, 37 1991, 98-113.

Adams 1988 — Adams, Joseph; Balas, Egon; Zawack, Daniel, 1988, *The Shifting Bottleneck Procedure for Job Shop Scheduling*, Management Science, 34 3 1988.

Afentakis 1984 — Afentakis, Panayotis; Gavish, Bezalel; Karmarkar, Uday, 1984, *Computationally Efficient Optimal Solutions to the Lot-Sizing Problem in Multistage Assembly Systems*, Management Science, Vol. 30, No. 2, February 1984, 222-239.

Agre 1987 — Agre, Philip E.; Chapman, David, 1987, *Pengi: An Implementation of a Theory of Activity*, Proceedings of the AAAI-87, 268-272, 1987.

AITEAR 1997a — CIMI, 1997, *Final Report on Assessment of State-of-the-Art in Accurate Response*, Project AITEAR, CIMI/Cranfield University, Cranfield, Beds, MK43 0AL, UK, April 1997.

AITEAR 1997b — Benz, Harald, 1997, *Working Definition of EE and ARM for the AITEAR Project*, Discussion paper, AITEAR Project, FhG-IAO, Nobelstrasse 12, D 70569 Stuttgart, August 1997.

Allen 1983 — Allen, James F., 1983, *Mantaining Knowledge about Temporal Intervals*, Communications of the ACM 26 (11), 1983, 832-843.

Allen 1991 — Allen, James F., 1991, *Time and Time Again: The Many Ways to Represent Time*, International Journal of Intelligent Systems, 6 1991,

AMICE 1993 — *AMICE CIM-OSA: Open Systems Architecture for CIM, 2nd. Revised and Extended Version*, Springer-Verlag, Berlin, 1993.

Anderson 1994 — Anderson, Edward J., 1994, *The Management of Manufacturing*, Addison-Wesley Publishing Company, 1994.

Armstrong 1997 — Armstrong, Aaron; Durfee, Edmund, 1997, *Dynamic Prioritization of Complex Agents in Distributed Constraint Satisfaction Problems*, Proceedings of the IJCAI'97, 620-6625.

Arnold 1997 — Arnold, Jörg; Dudenhausen, Hans-Martin; Halmosi, Hans, 1997, *Production Planning and Control within Supply Chains*, Fraunhofer Institute Manufacturing Engineering and Automation, Stuttgart, Germany, ESPRIT project 20544, X-CITTIC, (do site da Web http://www.nimblesite.com/xcittic/default.htm).

Atabakhsh 1991 — Atabakhsh, H., 1991, *A Survey of Constraint Based Scheduling Systems Using an Artificial Intelligence Approach*, Artificial Intelligence in Engineering, 6 (2) 1991.

Axsäter 1988 — Axsäter, Sven, 1988, *A Sequential Lot Sizing Heuristic with Optimal Average Perfomance*, Management Science, Vol. 34, No. 11, November 1988, 1324-1332.

Axsäter 1993 — Axsater, Sven; Rosling, Kaj, 1993, *Installation vs. Echelon Stock Policies for Multilevel Inventory Control*, Management Science, Vol. 39, No. 10, October 1993, 1247-1280.

Bäckström 1994 — Bäckström, C.; Sandewall, E. (eds.), 1994, *Current Trends in AI Planning*, IOS Press, Amsterdam, Netherlands, 1994.

Badinelli 1992 — Badinelli, Ralph D., 1992, *A Model for Continuous-Review Pull Policies in Serial Inventory Systems*, Operations Research, Vol. 40, No. 1, January-February 1992, 142-156.

Bahl 1984 — Bahl, Harish C.; Ritzman, Larry P., 1984, *An Integrated Model for Master Scheduling, Lot Sizing and Capacity Requirements Planning*, Journal of the Operational Research Society, Vol. 35 No. 5 (4) 1986, 389-399.

Bahl 1987 — Bahl, Harish C.; Ritzman, Larry P.; Gupta, Jatinder N.D., 1987, *Determining Lot Sizes and Resource Requirements: A Review*, Operations Research, 35, No. 3, May-June, 1987, 329-345.

Baker 1974 — Baker, K.R., *Introduction to Sequencing and Scheduling*, New York, Wiley, 1974.

Barahona 1997 — Barahona, Pedro, 1997, *Constraint Programming*, Tutorial on Constraint Programming, Hand Notes, EPIA'97, 8th Portuguese Conference on Artificial Intelligence, University of Coimbra, Coimbra, Portugal, October 6-9, 1997.

Barbuceanu 1994 — Barbuceanu, Mihai; Fox, Mark S., 1994, *The Information Agent: An Infrastructure Agent Supporting Collaborative Enterprise Architectures*, Department of Industrial Engineering, University of Toronto, Canada (do site da Web http://www.ie.utoronto.ca/EIL/).

Barbuceanu 1995a — Barbuceanu, Mihai; Fox, Mark S., 1995, *COOL: A Language for Describing Coordination in Multi Agent Systems*, Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), 17-24, 1995.

Barbuceanu 1995b — Barbuceanu, Mihai; Fox, Mark S., 1995, *The Architecture of an Agent Based Infrastructure for Agile Manufacturing*, Department of Industrial Engineering, University of Toronto, Canada (do site da Web http://www.ie.utoronto.ca/EIL/).

Barnes 1995 — Barnes, J. Wesley; Laguna, Manuel; Glover, Fred, 1995, *An Overview of Tabu Search Approaches to Production Scheduling Problems*, In Intelligent Scheduling Systems, Donald E. Brown and William T. Scherer (eds.), Kluwer Academic Publishers, 1995, 101-127.

Beck 1994a — Beck, J. Christopher; Fox, Mark S., 1994, *Mediated Conflict Recovery by Constraint Relaxation*, Proceedings of the AAAI-94 Workshop on Models of Conflict Management in Cooperative Problem Solving, Seattle, WA, August 6, 1994.

Beck 1994b — Beck, J. Christopher; Fox, Mark S., 1994, *Supply Chain Coordination via Mediated Constraint Relaxation*, Proceedings of the First Canadian Workshop on Distributed Artificial Intelligence, Banff, AB, May 15, 1994

Beck 1995 — Beck, Howard; Tate, Austin, 1995, *Open Planning, Scheduling and Constraint Management Architectures for Virtual Manufacturing*, Artificial Intelligence Applications Institute, The University of Edinburgh (obtido na Web).

Beek 1991 — Beek, Peter van, 1991, *Approximation Algorithms for Temporal Reasoning*, IJCAI-91, 1291-1296, 1991.

Beek 1992 — Beek, Peter van, 1992, *Reasoning about Qualitative Temporal Information*, Artificial Intelligence 58 1992, 297-326.

Bellman 1957 — Bellman, R., *Dynamic Programming*, Princeton University Press, Priceton, New Jersey, 1957.

Bellman 1962 — Bellman, R.; Dreyfus, S.E., *Applied Dynamic Programming*, Princeton University Press, Priceton, New Jersey, 1962.

Bensana 1988 — Bensana, E.; Bel, G.; Dubois, D., *OPAL: A Multi-Knowledge-Based System for Industrial Job Shop Scheduling*, International Journal of Production Research, 1988, 26, (5), 795-819.

Bensana 1993 — Bensana, Eric, *Advanced Job-Shop Scheduling in Aeronautical Manufacturing*, In Scheduling of Production Processes, Dorn, J.; Froeschel, K. (eds.), Chapter 10, Elis Horwood, Ld., 1993.

Berry 1992 — Berry, Pauline M.; Choueiry, Berthe Y.; Friha, Lamia, 1992, *A Multi-Agent Architecture for a Distributed Approach to Resource Allocation Using Temporal Abstractions*, Technical report, EPFL No. TR-92/18, AI Laboratory, Swiss Federal Institute of Technology, EPFL-Ecublens, CH-1015 Lausanne, Switzerland (do site da Web http://liawww.epfl.ch/).

Bezirgan 1993 — Bezirgan, Atilla, 1993, *A Case-Based Approach to Scheduling Constraints*, In Scheduling of Production Processes, Dorn, J.; Froeschel, K. (eds.), Chapter 4, Elis Horwood, Ld., 1993.

Billington 1986 — Billington, Peter J.; McClain, John O.; Thomas, L. Joseph, 1986, *Heuristics for Multilevel Lot-Sizing with a Bottleneck*, Management Science, Vol. 32, No. 8, August 1986, 989-1006.

Bitran 1993 — Bitran, Gabriel R.; Tirupati, Devanath, 1993, *Hierarchical Production Planning*, In Handbooks in Operations Research and Management Science (Nemhauser, G.L.; Kan, A.H.G. Rinnooy, eds.), Volume 4: Logistics of Production and Inventory, Graves, S.C.; Kan, A.H.G. Rinnooy; Zipkin, P.H. (eds.), North-Holland, Amsterdam, 1993, Chapter 2, 523-568.

Blackburn 1982 — Blackburn, Joseph D.; Millen, Robert A., 1982, *Improved Heuristics for Multi-Stage Requirements Planning Systems,* Management Science, Vol. 28, No. 1, January 1982, 44-56.

Blazewicz 1994 — Blazewicz, J.; Ecker, K.H.; Schmidt, G.; Weglarz, J., *Scheduling in Computer and Manufacturing Systems*, Springer Verlag, 1994.

Boctor 1996 — Boctor, Fayez F., 1996, *A New and Efficient Heuristic for Scheduling Projects with Resource Restrictions and Multiple Execution Modes*, European Journal of Operational Research, 90, 1996, 349-361.

Bond 1988 — Bond, Alan H.; Les Gasser (eds.), 1988, *Readings in Distributed Artificial Intelligence*, Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

Booch 1994 — Booch, G., 1994, *Object-Oriented Analysis and Design*, Addison-Wesley, Reading, MA, 1994 (2nd. edition).

Botelho 2000 — Botelho, Luís M.; Ramos, Pedro N., 2000, *Emotionally Controlled Inter-Agent Communication*, Proceedings of the IEEE Intelligent Vehicle Symposium (IV'2000), 2000.

Bourland 1994 — Bourland, Karla E.; Yano, Candace A., 1994, *The Strategic Use of Capacity Slack in the Economic Lot Scheduling Problem with Random Demand*, Management Science, Vol. 40, No. 12, December 1994, 1690-1704.

Bowman 1959 — Bowman, E.H., *The Schedule-Sequence Problem*, Operations Research, vol. 7, pp. 621-624, 1959.

Brandimarte 1987 — Brandimarte, P.; Conterno, R.; Laface, P., *FMS Production Scheduling by Simulated Annealing*, Proceedings of the third International Conference on Simulation in Manufacturing, 235-245.

Brooks 1991 — Brooks, Rodney, 1991, *Intelligence without Reason*, Proceedings of the IJCAI-91, 569-595.

Brown 1995 — Brown, Donald E.; Marin, John A.; Scherer, William T., *A Survey of Intelligent Scheduling Systems*, In *Intelligent Scheduling Systems*, *Brown*, D.E.; Scherer, W.T. (eds.), Kluwer Academic Publications, pp. 1-40, 1995.

Browne 1996 — Browne, Jimmie; Harhen, John; Shivnan, James, 1996, *Production Management Systems*, Addison-Wesley, 1996 (2th edition).

Brucker 1998 — Brucker, Peter, 1998, *Scheduling Algorithms*, Springer-Verlag, Berlin, 1998.

Buchanan 1969 — Buchanan, B.G.; Sutherland, G.L.; Feigenbaum, E.A., 1969, *Heuristic DENDRAL: A Program for Generating Explanatory Hypothesis in Organic Chemistry*, In Meltzer, B., Michie, D., and Swann, M. (eds.), Machine Intelligence 4, 209-254, Edinburgh University Press, Edinburgh, Scotland.

Buffa 1997 — Buffa, Elwood S.; Sarin, Rakesh K., 1987, *Modern Production/Operations Management*, John Wiley & Sons, 1987.

Burke 1991 — Burke, P.; Prosser, P., 1991, *A Distributed Asynchronous System for Predictive and Reactive Scheduling*, Artificial Intelligence in Engineering, 6 (3) 1991, 106-124.

Burke 1994 — Burke, P.; Prosser, P., *The Distributed Asynchronous Scheduler*, In Intelligent Scheduling, Zweben, Monte; Fox, Mark S. (eds.), Cap.1, San Francisco, Morgan Kaufman, 1994.

Busby 1993 — Busby, J. S.; Fan, I. S., 1993, *The Extended Manufacturing Enterprise: Its Nature and its Needs*, International Journal of Technology Management, Special Issue on Manufacturing Technology Diffusion, Implementation and Management 1993, 8 (3, 4, 5), pp. 294-308.

Camarinha-Matos 1997a — Camarinha-Matos, L.M.; Afsarmanesh, H.; Garita, C.; Lima, C., 1997, *Towards an Architecture for Virtual Enterprises*, 2nd. World Congress on Intelligent Manufacturing Processes & Systems, Budapest, June 1997; ESPRIT project 22647, PRODNET II.

Camarinha-Matos 1997b — Camarinha-Matos, L.M., 1997, *A Platform to Support Production Planning and Management in a Virtual Enterprise*, Proceedings of CAPE'97- IFIP/SME Int. Conf. on Comp.Applications in Production and Engineering (Chapman & Hall), Detroit, USA, Nov 97; ESPRIT project 22647, PRODNET II.

Camarinha-Matos 1997c — Camarinha-Matos, L.M.; Lima, C.P.; Osorio, L., 1997, *The PRODNET Platform for Production Planning and Management in Virtual Enterprises*, Proceedings of ICE'97- Int. Conf. On Concurrent Enterprising, Nottingham, UK, Oct 97; ESPRIT project 22647, PRODNET II.

Camarinha-Matos 1999a — Camarinha-Matos, Luís M.; Afsarmanesh, Hamideh, 1999, *The Virtual Enterprise Concept*, In Infrastructures for Virtual Enterprises, Networking Industrial Enterprises, Luís M. Camarinha-Matos, Hamideh Afsarmanesh (eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, 3-14.

Camarinha-Matos 1999b — Camarinha-Matos, Luís M.; Afsarmanesh, Hamideh, 1999, *Tendencies and General Requirements for Virtual Enterprises*, In Infrastructures for Virtual Enterprises, Networking Industrial Enterprises, Luís M. Camarinha-Matos, Hamideh Afsarmanesh (eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, 15-30.

Camarinha-Matos 1999c — Camarinha-Matos, Luís M.; Silva, V. Santos; Rabelo, R.J., 1999, *Production Planning and Control in a Virtual Enterprise*, In Infrastructures for Virtual Enterprises, Networking Industrial Enterprises, Luís M. Camarinha-Matos, Hamideh Afsarmanesh (eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, 219-232.

Cammarata 1988 — Cammarata, Stephanie; McArthur, David; Steeb, Randall, 1988, *Strategies of Cooperation in Distributed Problem Solving*, Readings in Distributed Artificial Intelligence, Bond, Alan H.; Les Gasser (eds.), Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

Cardelli 1985 — Cardelli, L.; Wegner, P., 1985, *On Understanding Types, Data Abstraction and Polymorphism*, ACM Computing Surveys, vol. 17, no. 4, pp. 471-522, 1985.

Carvalho 1996 — Carvalho, J.M. Crespo de, 1996, *Logística*, Edições Sílabo, Lda., Lisboa, Portugal, 1996.

Cesta 1998a — Cesta, Amedeo; Oddi, Angelo; Smith, Stephen F., 1998, *Scheduling Multi-Capacitated Resources under Complex Temporal Constraints*, Technical Report CMU-RI-98-17, School of Computer Science, Carnegie Mellon University, Pittsburg, Pennsylvania 15213, June 1998.

Cesta 1998b — Cesta, Amedeo; Oddi, Angelo; Smith, Stephen F., 1998, *Profile-Based Algorithms to Solve Multiple Capacitated Metric Scheduling Problems*, Proceedings of the 4th International Conference on Artificial Intelligence Planning Systems (AIPS-98), 1998.

Chase 1995 — Chase, Richard B.; Aquilano, Nicholas J., *Gestão da Produção e das Operações, Perspectiva do Ciclo de Vida*, Monitor, Lisboa, 1995.

Cheng 1995 — Cheng, C.; Smith, S., 1995, *A Constraint-Posting Framework for Scheduling Under Complex Constraints*, Proceedings Joint INRIA/IEEE Conference on Emerging Technologies and Factory Automation, October, 1995.

Choueiry 1993 — Choueiry, Berthe Y.; Faltings, Boi, 1993, *Interactive Resource Allocation by Problem Decomposition and Temporal Abstractions*, EWSP' 93 - 2nd. European Workshop on Planning, Vadstena, Sweder December 1993.

Christopher 1992 — Christopher, Martin, 1992, *Logistics and Supply Chain Management, Startegies for Reducing Costs and Improving Services*, Pitman Publishing, London, 1992.

Christopher 1993 — Christopher, Martin, 1993, *Logistics and Competitive Strategy*, in Cooper, James (ed.), *Strategy Planning in Logistics and Transportation*, The Cranfield Management Research Series/Koogan Page Limited, London, 1993, Chapter 2, 24-32.

Clark 1960 — Clark, A; Scarf, H., 1960, *Optimal Policies for a Multi-Echelon Inventory Problem*, Management Science 6(4), 475-490, 1960.

Clearwater 1991 — Clearwater, Scott H.; Huberman, Bernardo A.; Hogg, Tad, 1991, *Cooperative Solution of Constraint Satisfaction Problems*, Science, Vol. 254, 1181-1183

Coelho 1999 — Coelho, Helder; Paiva, Ana, 1999, *Inteligência Artificial Distribuída - Uma Introdução*, Texto de apoio à disciplina de Inteligência Artificial Distribuída, Mestrado de Informática, Faculdade de Ciências de Lisboa, 1999 (edição dos autores).

Cohen 1988 — Cohen, Morris A.; Lee, Hau L., 1988, *Strategic Analysis of Integrated Production-Distribution Systems: Models and Methods*, Operations Research, Vol. 36, No. 2, March-April 1988, 216-228.

Cohen 1990a — Cohen, Morris; Kamesam, Pasumarti V.; Lee, Hau; Tekerian, Armen, 1990, *Optimiser: IBM's Multi-Echelon Inventory System for Managing Service Logistics*, Interfaces 20: 1 January-February 1990, 64-82.

Cohen 1990b — Cohen, Philip R.; Levesque, Hector J., 1990, *Intention is Choice with Commitment*, Artificial Intelligence 42 (1990) 213-261.

Cohen 1995 — Cohen, Paul R., 1995, *Empirical Methods for Artificial Intelligence*, The MIT Press, Cambridge, Massachusetts, 1995.

Collinot 1988 — Collinot, Anne; Le Pape, Claude; Pinoteau, Gérard, 1988, *SONIA: A Knowledge-based Scheduling System*, Artificial Intelligence in Engineering Vol 3, Nº 2 (1988) 86-94.

Conry 1988 — Conry, Susan E.; Meyer, Robert A.; Lesser, Victor R., 1988, *Multistage Negotiation in Distributed Planning*, Readings in Distributed Artificial Intelligence, Bond, Alan H.; Les Gasser (eds.), Morgan Kaufman Publishers, Inc., San Mateo, California, 1988, 367-384.

Cooper 1993 — Cooper, James (ed.), 1993, *Strategy Planning in Logistics and Transportation*, Cooper, James (ed.), The Cranfield Management Research Series/Koogan Page Limited, London, 1993.

Corkill 1988 — Corkill, Daniel D.; Gallagher, Kevin Q.; Johnson, Philip M., 1988, *Achieving Flexibility, Efficiency, and Generality in Blackboard Architectures*, Readings in Distributed Artificial Intelligence, Bond, Alan H.; Les Gasser (eds.), Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

Costello 1995 — Costello, Damien; Jordan, Paddy; Browne, Jim, 1995, *A Knowledge-Based Tool for Reactive Scheduling*, In Artificial Intelligence in Reactive Scheduling, Kerr, Roger; Szelke, Elizabeth (eds.), Chapter 8, Chapman & Hall, 1995.

Courtois 1996 — Courtois, A.; Pillet, M.; Martin, C., 1996, Gestão da Produção, LIDEL, Edições Técnicas, Limitada, Lisboa, Portugal, 1997 (tradução do original *Gestion de Production*, Les Editions D' Organisatior 1996).

Custódio 1992 — Custódio, Luís; Bispo, Carlos; Sentieiro, João J.S., 1992, *A Hierarchical Fuzzy Control System for Short-Range Planning and Scheduling*, Proceedings of the Rensselaers' s Third Internationa Conference on Computer Integrated Manufacturing, Troy, NY, 1992.

Custódio 1993 — Custódio, Luís; Sentieiro, João; Bispo, Carlos, 1993, *Fuzzy Logic Applied to Production Planning and Scheduling*, relatório técnico AIMS Lab-TR 701-93, March 1993 (do site da Web http://www.isr.ist.utl.pt/~yoda/aims/publications.html).

Custódio 1998 — Custódio, Luis M.M.; Pinto-Ferreira, Carlos, 1998, *Control of Manufacturing Systems Using Societies of Agents*, Proceedings of Rensslear' s International Conference on Agile Intelligent an Computer Integrated Manufacturing, RPI, Troy, NY, USA, 1998.

Darby-Dowman 1995 — Darby-Dowman, Ken; Cormac, Lucas; Mitra, Gautam; Fink, Raymond; Kingsley, Leonard; Smith, J. Walter, 1995, *Intelligent Scheduling Support for the U.S. Coast Guard*, In Intelligent Scheduling Systems, Donald E. Brown and William T. Scherer (eds.), Kluwer Academic Publishers, 1995, 227-247.

Davis 1987 — Davis, Ernest, 1987, *Constraint Propagation with Interval Labels*, Artificial Intelligence 32 1987 281-331.

Davis 1988 — Davis, Randall; Smith, Reid G., 1988, *Negotiation as a Metaphor for Distributed Problem Solving*, Readings in Distributed Artificial Intelligence, Bond, Alan H.; Les Gasser (eds.), Morgan Kaufman Publishers, Inc., San Mateo, California, 1988, 333-356.

Davis 1993 — Davis, Tom, 1993, *Effective Supply Chain Management*, Sloan Management Review, 34(4), 1993, pp. 35-46.

Dean 1987 — Dean, Thomas L.; McDermott, Drew V., 1987, *Temporal Data Base Management*, Artificial Intelligence, 32 1987, 1-55.

Dean 1995 — Dean, Thomas; Allen, James; Aloimonos, Yiannis, 1995, *Artificial Intelligence, Theory and Practice*, The Benjamin/Cummings Publishing Company, Inc., USA, 1995.

Dechter 1988 — Dechter, Rina; Pearl, Judea, 1988, *Network-Based Heuristics for Constraint Satisfaction-Problems*, Artificial Intelligence, 34 1988.

Dechter 1990 — Dechter, Rina, 1990, *Enhancement Schemes for Constraint Processing: Backjumping, Learning, and Cutset Decomposition*, Artificial Intelligence, 41 1989/90, 273-312.

Dechter 1991 — Dechter, Rina; Meiri, Itay; Pearl, Judea, 1991, *Temporal Constraint Networks*, Artificial Intelligence, 49 1991.

Dechter 1994 — Dechter, Rina; Meiri, Itay, 1994, *Experimental Evaluation of Preprocessing Algorithms for Constraint Satisfaction Problems*, Artificial Intelligence, 68 1994 211-241.

Decker 1995 — Decker, Keith S.; Lesser, Victor R., 1995, *Designing a Family of Coordination Algorithms*, Multi-Agent Systems Laboratory, Department of Computer Science, University of Massachussetts at Amherst, USA, Technical Report 94-14.

Dennett 1987 — Dennett, D.C., 1987, *The Intentional Stance*, MIT Press, Cambridge, MA, 1987.

Diks 1996 — Diks, E.B.; Kok, A.G. de; Lagodimos, A.G., 1996, *Multi-Echelon Systems: A Service Measure Perspective*, European Journal of Operational Research, 95 (1996) 241-263.

d'Inverno 1998 — d' Inverno, Mark; Kinny, David; Luck Michael, 1998,*Interaction Protocols in Agentis*, Proceedings of the International Conference on Multi-Agent Systems, ICMAS-98, IEEE Computer Society, Los Alamitos, California, 1998, 112-119.

Dobson 1987 — Dobson, Gregory, 1987, *The Economic Lot-Scheduling Problem: Achieving Feasibility Using Time-Varying Lot Sizes*, Operations Research, Vol. 35, No. 5, September-October, 1987, 764-771.

Dorn 1994a — Dorn, Jürgen; Slany, Wolfgang, *A Flow Shop with Compatibility Constraints in a Steelmaking Plant*, In Intelligent Scheduling, Zweben, Monte; Fox, Mark S. (eds.), Cap.22, San Francisco, Morgan Kaufman, 1994.

Dorn 1994b — Dorn, Jürgen; Kerr, Roger; Thalhammer, Gabi, 1994, *Reactive Scheduling in A Fuzzy-Temporal Framework*, In Knowledge-based Reactive Scheduling, Proceedings of the IFIP TC5/WG5.7 International Workshop on Knowledge-Based Reactive Scheduling, Szelke, E., Kerr, R.M (eds.), Athens, Greece, October, 1993, Elsevier Science, 1994, 39-55.

Dorn 1995 — Dorn, Jürgen, 1995, *Case-Based Reactive Scheduling*, In Artificial Intelligence in Reactive Scheduling, Kerr, Roger; Szelke, Elizabeth (eds.), Chapter 4, Chapman & Hall, 1995.

Dowsland 1990 — Dowsland, K.A., *A Timetabling Problem in which Clashes Are Inevitable*, JORS, 41 1990, 907-918.

Dowsland 1993 — Dowsland, K.A., Chapter 2- *Simulated Annealing*, In Reeves, C.R. (ed.), Modern Heuristic Techniques for Combinatorial Problems, Reeves, C.R. (ed.), Blackwell Scientific Publications, 1993.

Dudenhausen 1997a — Dudenhausen, Hans-Martin; Halmosi, Hans; Löffler, Benno, 1997, *Real-Time Order Planning in Semiconductor Virtual Enterprises*, Fraunhofer Institute Manufacturing Engineering and Automation, Stuttgart, Germany, ESPRIT project 20544, X-CITTIC (do site da Web http://www.nimblesite.com/xcittic/default.htm).

Dudenhausen 1997b — Dudenhausen, Hans-Martin; Halmosi, Hans; Löffler, Benno, 1997, *Real-Time Order Planning in Semiconductor Virtual Enterprises*, Fraunhofer Institute Manufacturing Engineering and Automation, Stuttgart, Germany, ESPRIT project 20544, X-CITTIC (do site da Web http://www.nimblesite.com/xcittic/default.htm).

Durfee 1988 — Durfee, Edmund H.; Lesser, Victor R., 1988, *Using Partial Global Plans to Coordinate Distributed Problem Solvers*, Readings in Distributed Artificial Intelligence, Bond, Alan H.; Les Gasser (eds.), Morgan Kaufman Publishers, Inc., San Mateo, California, 1988, 150-158

Eastman 1959 — Eastman, W.L., *A Solution to the Travelling Salesman Problem*, Econometrica, vol. 27 1959, 282.

Eglese 1987 — Eglese, R.W.; Rand, G.K., *Conference Seminar Timetabling*, JORS, 38 1987, 591-598.

Elliott 1994 — Elliott, Clark, 1994, *Research Problems in the Use of a Shallow Artificial Intelligence Model of Personality and Emotion*, Proceedings of the AAAI'94, 9-15, 1994.

Ellram 1991 — Ellram, Lisa M., 1991, *Supply Chain Management: The Industrial Organisation Perspective*, International Journal of Phisical Distribution & Logistic Management 1991, 21(1), pp. 13-22

Elsayed 1994 — Elsayed, Elsayed A.; Boucher, Thomas, 1994, *Analysis and Control of Production Systems*, Prentice Hall, 1994.

Erman 1980 — Erman, L.D.; Hayes-Roth, F.; Lesser, V.R.; Reddy, R.D., 1980, *The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty*, ACM Computing Surveys, 12 (2), 1980, 213-253.

Faltings 1994 — Faltings, Boi, 1994, *Arc-Consistency for Continuous Variables*, Artificial Intelligence, 65 1994 363-376.

Famili 1992 — Famili, A. (Fazel); Nau, Dana S.; Kim, Steven H., 1992, *Artificial Intelligence Applications in Manufacturing*, Artificial Intelligence Applications in Manufacturing, Famili, A. (Fazel); Nau, Dana S.; Kim, Steven H. (eds.), The AAAI Press, Menlo Park, California, USA, 1992.

Farmer 1991 — Farmer, David; Amstel, Rien Ploos van, 1991, *Effective Pipeline Management, How to Manage Integrated Logistics*, Gower Publishing Company Limited, England, 1991.

Federgruen 1984 — Federgruen, Awi; Zipkin, Paul, 1984, *Approximations of Dynamic, Multilocation Production and Inventory Problems*, Management Science, Vol. 30, No. 1, January 1984, 69-84.

Federgruen 1989 — Federgruen, A., 1989, *Methodologies for the Evaluation and Control of Large Scale Production/Distribution Systems Under Uncertainty*, In Logistics: Where Ends Have to Meet: Proceedings of the Shell Conference on Logistics, Apeldoorn, The Netherlands, 2-3 November 1988, Rijn, C.F.H. van, 143-157.

Federgruen 1993a — Federgruen, A., 1993, *Centralized Planning Models for Multi-Echelon Inventory Systems under Uncertainty*, In Handbooks in Operations Research and Management Science (Nemhauser, G.L.; Kan, A.H.G. Rinnooy, eds.), Volume 4: Logistics of Production and Inventory, Graves, S.C.; Kan, A.H.G. Rinnooy; Zipkin, P.H. (eds.), North-Holland, Amsterdam, 1993, Chapter 3, 133-173.

Federgruen 1993b — Federgruen, A.; Zheng, Yu-Sheng, 1993, *Optimal Power-of-Two Replenishment Strategies in Capacitated General Production/Distribution Networks*, Management Science, Vol. 39, No. 6, June 1993, 710-727.

Ferber 1996 — Ferber, Jacques, 1996, *Reactive Distributed Artificial Intelligence: Principles and Applications*, In *Foundations of Distributed Artificial Intelligence*, O' Hare and Jennings (eds.), Wiley Inter-Science, 199 (Chapter 11).

Ferber 1999 — Ferber, Jacques, 1999, *Multi-Agent Systems, An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, 1999.

Fikes 1971 — Fikes, Richard E.; Nilsson, Nils J., 1971, *STRIPS: A New Approach of Theorem Proving to Problem Solving*, Artificial Intelligence, vol. 2, 189-208, 1971; Readings in Planning, Morgan Kaufman Publishers, Inc., San Mateo, California, 1990, 87-97.

Filipic 1993 — Filipic, Bogdan, 1993, *Enhancing Genetic Search to Schedule a Production Unit*, In Scheduling of Production Processes, Dorn, J.; Froeschel, K. (eds.), Chapter 5, Elis Horwood, Ld., 1993.

Finin 1992 — Finin, Tim; Fritzson, Rich; McKay, Don, 1992, *A Language and Protocol to Support Intelligent Agent Interoperability*, Proceedings of the CE \& CALS Washington `92 Conference, June 1992.

Finin 1993 — Finin, Tim; Weber, Jay; Wiederhold, Gio; Genesereth, Michael; Fritzson, Richard; McKay, Donald; McGuire, James; Pelavin, Richard; Shapiro, Stuart; Beck, Chris, 1993, *DRAFT Specification of the Agent-Communication Language*, The DARPA Knowledge Sharing Initiative External Interfaces Working Group (do site da Web http://www.cs.umbc.edu/kqml/papers/).

FIPA 1998a — *Agent Management*, FIPA 97 Specification, Part 1, Version 2.0, Agent Management, Foundation for Intelligent Physical Agents, October 1998 (do site da Web http://www.fipa.org).

FIPA 1998b — *Agent Communication Language*, FIPA 97 Specification, Part 2, Version 2.0, Agent Communication Language, Foundation for Intelligent Physical Agents, October 1998 (do site da Web http://www.fipa.org).

Fischer 1994 — Fischer, Klaus, 1994, *Knowledge-Based Reactive Scheduling in a Flexible Manufacturing System*, In Knowledge-Based Reactive Scheduling, Proceedings of the IFIP TC5/WG5.7 International Workshop on Knowledge-Based Reactive Scheduling, Szelke, E., Kerr, R.M. (eds.), Athens, Greece, October, 1993, Elsevier Science, 1994, 1-18.

Fisher 1994 — Fisher, Marshall R.; Hammond, Janice H.; Obermeyer, Walter R.; Raman, Ananth, 1994, *Making Supply Meet Demand*, Harvard Business Review, May-June 1994.

Folie 1976 — Folie, M.; Tiffin, J., 1976, *Solution of a Multi-Product Manufacturing and Distribution Problem*, Management Science, Vol. 23 (1976) 286-296.

Forrester 1958 — Forrester, Jay W., 1958, *Industrial Dynamics a Major Breakthrough for Decision Makers*, Harvard Business Review, July-August 1958, 37-66.

Forrester 1961 — Forrester, Jay W., 1961, *Industrial Dynamics*, The MIT Press, Cambridge, Massachussetts, USA, 1961.

Fox 1982 — Fox, Mark S.; Allen, Brad P.; Strohm, Gary A., 1982, *Job-Schop Scheduling: An Investigation in Constraint-Directed Reasoning*, In Proceedings of the 2nd. National Conference on Artificial Intelligence, 155-158, Pittsburgh PA, 1982, AAAI.

Fox 1983 — Fox, Mark S., 1983, *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*, PhD. Thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh PA, 1983.

Fox 1988 — Fox, Mark S., 1988, *An Organizational View of Distributed Systems*, Readings in Distributed Artificial Intelligence, Bond, Alan H.; Les Gasser (eds.), Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

Fox 1989 — Fox, Mark S.; Sadeh, Norman; Baykan, Can, 1989, *Constrained Heuristic Search*, Proceedings of the 1989 International Joint Conference on Artificial Intelligence (IJCAI89) 309-315, 1989.

Fox 1990 — Fox, Mark S.; Sadeh, Norman, 1990, *Why Is Scheduling Difficult? A CSP Perspective*, Proceedings of the 9th European Conference on Artificial Intelligence (ECAI90) 754-767, 1990.

Fox 1993 — Fox, Mark S.; Chionglo, John F.; Barbuceanu, Mihai, 1993, *The Integrated Supply Chain Management System*, Department of Industrial Engineering, University of Toronto, Canada (do site da Web http://www.ie.utoronto.ca/EIL/iscm-descr.html).

Fox 1994 — Fox, Mark S., *ISIS: A Retrospective*, In Intelligent Scheduling, Cap.1, Zweben, Monte; Fox, Mark S., (eds.), San Francisco, Morgan Kaufman, 1994.

Franklin 1997 — Franklin, Stan; Graesser, Art, 1997, *Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents*, In Intelligent Agents III, Agent Theories, Architectures, and Languages, Proceedings of the ECAI' 96 Workshop (ATAL), Budapest, Hungary, August 12-13 1996, Müller, Jörg P.; Wooldridge, Michael J.; Jennings, Nicholas R. (eds.), Springer Verlag, 1997.

Freuder 1994 — Freuder, Eugene C.; Mackworth, Alan K. (eds.), 1994, *Constraint-Based Reasoning*, MIT Press, 1994.

Friedman 1997 — Friedman, Marc, 1997, *Job Shop Scheduling*, Technical Report, Department os Computer Science and Engineering, University of Washington, Seattle, WA, 1997 (do site da Web http://www.salford.ac.uk/planning).

Froeschl 1993 — Froeschl, Karl A., *Two Paradigms of Combinatorial Production Scheduling*, In Scheduling of Production Processes, Dorn, Jürgen; Froeschl, Karl, (eds.), Chapter 1, Elis Horwood Limited, 1993.

Garey 1979 — Garey, M.R.; Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, 1979.

Gasser 1989 — Gasser, L; Huhns, M. N. (eds.), 1989, *Distributed Artificial Intelligence*, L. Gasser e M. N. Huhns (eds.), Morgan Kaufmann, Los Altos, CA/Pitman, London, 1989.

Gattorna 1996 — Gattorna, J.L.; Walters, D.W., 1996, *Managing the Supply Chain*, Macmillan Business, 1996.

Georgeff 1998 — Georgeff, M.; Rao, A., 1998, *Rational Software Agents: From Theory to Practice*, In Agent Technology, Foundations, Applications, and Markets, Jennings, Nicholas R.; Wooldridge, Michael J. (eds.), Springer Verlag, 1998, 139-160.

Glover 1986 — Glover, F., *Future Paths for Integer Programming and Links to Artificial Intelligence*, Computers and Operations Research, 5 1986, 533-549.

Glover 1993 — Glover, F., Chapter 3- *Tabu Search*, In Modern Heuristic Techniques for Combinatorial Problems, Reeves, C.R. (ed.), Blackwell Scientific Publications, 1993.

Goldberg 1989 — Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Massachussets, 1989.

Graham 1994 — Graham, Paul, 1994, *On Lisp*, PrenticeHall, Inc., Englewood Cliffs, New Jersey, USA, 1994.

Graham 1996 — Graham, Paul, 1996, *ANSI Common Lisp*, PrenticeHall, Inc., Englewood Cliffs, New Jersey, USA, 1996.

Graves 1981 — Graves, Stephen C., *A Review of Production Scheduling*, Operations Research 28 1981, 646-675.

Graves 1982 — Graves, Stephen C., 1982, *A Multiple-Item Inventory Model with a Job Completion Criterion*, Management Science, Vol. 28, No. 11, November 1982, 1334-1336.

Graves 1993 — Graves, S.C.; Kan, A.H.G. Rinnooy; Zipkin, P.H., 1993, *Logistics of Production and Inventory*, Handbooks in Operations Research and Management Science (Nemhauser, G.L.; Kan, A.H.G. Rinnooy, eds.), Volume 4: Logistics of Production and Inventory, Graves, S.C.; Kan, A.H.G. Rinnooy; Zipkin, P.H. (eds.), North-Holland, Amsterdam, 1993.

Gruber 1993a — Gruber, Thomas R., 1993, *What is an Ontology*, (do site da Web http://ksl-web.stanford.edu/kst/what-is-an-ontology.html).

Gruber 1993b — Gruber, Thomas R., 1993, *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*, Technical Report KSL 93-04, Knowledge Systems Laboratory, Stanford University, USA (do site da Web http://ksl-web.stanford.edu/knowledge-sharing/papers/README.html#onto-design).

Haddadi 1996 — Haddadi, Afsaneh; Sundermeyer, Kurt, 1996, *Belief-Desire-Intention Agent Architectures*, In *Foundations of Distributed Artificial Intelligence*, G. M. P. O' Hare and N. R. Jennings (eds.), John Wiley & Sons Inc., 1996, (Chapter 5).

Hansen 1986 — Hansen, P., *The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming*, Congress on Numerical Methods, In Combinatorial Optimization, Capri, Italy, 1986.

Haralick 1980 — Haralick, Robert M.; Elliot, Gordon L., 1980, *Increasing Tree Search Efficiency for Constraint Satisfaction Problems*, Artificial Intelligence, 14 1980, 263-313.

Hayes-Roth 1988 — Hayes-Roth, Barbara, 1988, *A Blackboard Architecture for Control*, Readings in Distributed Artificial Intelligence, Bond, Alan H.; Les Gasser (eds.), Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

Henseler 1993 — Henseler, H., 1993, *REAKTION: A System for Event Independent Reactive Scheduling*, In Artificial Intelligence in Reactive Scheduling, Kerr, Roger; Szelke, Elizabeth (eds.), Chapter 3, Chapman & Hall, 1995.

Hentenryck 1992 — Hentenryck, Pascal; Simonis, Helmut; Dincbas, Mehmet, 1992, *Constraint Satisfaction Using Constraint Logic Programming*, Artificial Intelligence 58 1992113-159.

Hentenryck 1994 — Hentenryck, Pascal Van, 1994, *Scheduling and Packing in the Constraint Language cc(FD)*, In Intelligent Scheduling, Monte Zweben and Mark S. Fox (eds.), Morgan Kaufman, 1994, Cap. 5.

Hildum 1994 — Hildum, David W., 1994, *Flexibility in a Knowledge-Based System for Solving Dynamic Resource-Constrained Scheduling Problems*, PhD. Thesis/Technical Report 94-77, Multi-Agent Systems Laboratory, Department of Computer Science, University of Massachussetts at Amherst, USA.

Hildum 1997 — Hildum, David W.; Sadeh, Norman M.; Laliberty, Thomas J.; McA' Nulty, John; Smith, Stephen F.; Kjenstad, Dag, 199€ *Blackboard Agents for Mixed-Initiative Management of Integrated Process-Planning/Production-Scheduling Solutions across the Supply Chain*, Proceedings of the AAAI-97, AAAI Press/The MIT Press, Menlo Park, California, USA 1000-1005.

Hill 1996 — Hill, Roger M., 1996, *Order Splitting in Continuous Review (Q,r) Inventory Models*, European Journal of Operational Research 95 (1996) 53-61.

Hillier 1990 — Hillier, Frederick S.; Lieberman, Gerald J., 1990, *Introduction to Operations Research*, McGraw-Hill International Editions, 1990.

Holland 1975 — Holland, J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

Huhns 1998 — Huhns, Michael N.; Singh, Munindar P., 1998, *Readings in Agents*, Huhns, Michael N.; Singh, Munindar P. (eds.), Morgan Kaufmann Publishers, San Francisco, California, 1998.

Huhns 1999 — Huhns, Michael N.; Stephens, Larry M., 1999, *Multiagent Systems and Societies of Agents*, In *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence*, Gerhard Weiss (ed.), The MIT Press, 1999 (Chapter 2).

Hunt 1997 — Hunt, Ingrid; Pereira Klen, Alexandra Augusta P; Zhang, Jiangang, 1997, *Cross Border Enterprises: Virtual or Extended!*, Re-engineering for Sustainable Industrial Production, Camarinha-Matos, Luís M. (ed.), Chapman & Hall, London, 1997, 63-72.

ICMAS 1996 — ICMAS 1996, 1996, *Proceedings of the Second International Conference on Multi-Agent Systems (Contents)*, ICMAS-96, AAAI Press, Menlo Park, California, 1996.

Ivens 1996 — Ivens, Philip; Lambrecht, Marc, *Extending the Shifting bottlenect Procedure to Real-Life Applications*, European Journal of Operational Research, 90, 1996, 252-268.

Jackson 1990 — Jackson, Peter L.; Maxwell, William L.; Muckstadt, John A., 1988, *Determining Optimal Reorder Intervals in Capacitated Production-Distribution Systems*, Management Science, Vol. 34, No. 8, August 1988, 938-958.

Jacobs 1984 — Jacobs, F. Robert, 1984, *OPT Uncovered: Many Production Planning and Scheduling Concepts Can Be Applied with or without the Software*, Industrial Engineering, 1984, 16 (10), 32-41.

Jacobs 1989 — Jacobs, F. Robert; Bragg, Daniel J., 1989, *Repetitive Lots: Flow-Time Reductions through Sequencing and Dynamic Batch Sizing*, Decision Sciences, Vol. 19, No. 2, 1989, 281-294.

Jennings 1996 — Jennings, Nick R., 1996, *Coordination Techniques for Distributed Artificial Intelligence*, In *Foundations of Distributed Artificial Intelligence*, O' Hare and Jennings (eds.), Wiley Inter-Science, 199 (Chapter 6).

Jennings 1998 — Jennings, Nicholas R.; Wooldridge, Michael J., 1998, *Agent Technology, Foundations, Applications, and Markets*, Jennings, Nicholas R.; Wooldridge, Michael J. (eds.), Springer Verlag, 1998.

Johnson 1991 — Johnson, David S.; Aragon, Cecilia R.; Mcgeoch, Lyle A.; Schevon, Catherine, 1991, *Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning*, Operations Research, 39 (3) 1991.

Johnston 1994a — Johnston, Mark D.; Minton, Steven, 1994, *Analysing a Heuristic Strategy for Constraint-Satisfaction and Scheduling*, In Intelligent Scheduling, Monte Zweben and Mark S. Fox (eds.), Morgan Kaufman, 1994, Cap. 9.

Johnston 1994b — Johnston, Mark D.; Miller, Glenn E., 1994, *SPIKE: Intelligent Scheduling of Hubble Space Telescope Observations*, In Intelligent Scheduling, Monte Zweben and Mark S. Fox (eds.), Morgan Kaufman, 1994, Cap. 14.

Kan 1976 — Kan, A.H.G. Rinnooy, 1976, *Machine Scheduling Problems, Classification, Complexity and Computations*, Martinus Nijhoff, The Hague, 1976.

Karmarkar 1985 — Karmarkar, Uday S.; Schrage, Linus, 1985, *The Deterministic Dynamic Product Cycling Problem*, Operations Research, Vol. 33, No. 2, March-April, 1985, 326-345.

Karmarkar 1987a — Karmarkar, Uday S.; Kekre, Sham; Kekre, Sunder, 1987, *The Dynamic Lot-Sizing Problem with Startup and Reservation Costs*, Operations Research, 35, No. 3, May-June, 1987, 389-398.

Karmarkar 1987b — Karmarkar, Uday S., 1987, *Lot Sizes, Lead Times and In-Process Inventories*, Management Science, Vol. 33, No. 3, March 1987, 409-418.

Kaufmann 1977 — Kaufmann, A., 1977, *Introduction à la Théorie des Sous-Ensembles Flous à l'Usage des Ingénieurs*, Masson, Paris, 1977.

Kerr 1991 — Kerr, Roger, 1991, *Knowledge-Based Manufacturing Management, Applications of Artificial Intelligence to the Effective Management of Manufacturing Companies*, Addison-Wesley Publishing Company, 1991.

Kerr 1995 — Kerr, Roger; Szelke, Elizabeth (eds.), 1995, *Artificial Intelligence in Reactive Scheduling*, Chapman & Hall, 1995, ISBN 0412729008.

Kirkpatrick 1983 — Kirkpatrick, S.; Gellat, C.D.; Vecchi, M.P., *Optimization by Simulated Annealing*, Science, 220 1983, 671-680.

Kis 1996 — Kis, Tamás; Váncza, József; Márkus, András, 1996, *Controlling Distributed Manufacturing Systems by a Market Mechanism*, Proceedings of the ECAI-96, 12th European Conference on Artificial Intelligence, 1996, Budapest, Hungary, Wolfgang Wahlster (editor), p.534-538.

Kjenstad 1998 — Kjenstad, Dag, 1998, *Coordinated Supply Chain Scheduling*, PhD. Thesis, Norwegian University of Science and Technology - NTNU, Department of Production and Quality Engineering, Trondheim, Norway, 1998.

Klen 1998 — Klen, A.P.; Rabelo, R.J.; Spinosa, L.M.; Ferreira, A.C., 1998, *Integrated Logistics in the Virtual Enterprise: The PRODNET-II approach*, In proceedings of the 5th International Workshop on Intelligent Manufacturing Systems, IMS' 98, Gramado, Brasi November 1998.

Klen 1999 — Klen, A.A. Pereira; Rabelo, R.J.; Spinosa, L.M.; Ferreira, A.C., 1999, *Distributed Business Process Management*, In Infrastructures for Virtual Enterprises, Networking Industrial Enterprises, Luís M. Camarinha-Matos, Hamideh Afsarmanesh (eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, 241-258.

Konolige 1993 — Konolige, Kurt; Pollack, Martha E., 1993, *A Representationalist Theory of Intention*, Proceedings of the IJCAI'93, 390-395.

Koomen 1989 — Koomen, Johannes A.G.M., 1989, *Localizing Temporal Constraint Propagation*, KR-89, 198-202, 1989.

Kuik 1990 — Kuik, R.; Salomon, M., *Multi-Level Lot-Sizing Problem: Evaluation of a Simulated Annealing Approach*, EJOR, 45 1990, 25-37.

Kuik 1994 — Kuik, Roelof; Salomon, Marc; Wassenhove, Luk N. van, 1994, *Batching Decisions: Structure and Models*, European Journal of Operational Research 75 (1994) 243-263.

Kuipers 1994 — Kuipers, Benjamin, 1994, *Qualitative Reasoning, Modelling and Simulation with Incomplete Knowledge*, The MIT Press, 1994.

Kumar 1992 — Kumar, Vipin, 1992, *Algorithms for Constraint-Satisfaction Problems: A Survey*, AI Magazine, 13 (1) 1992.

Land 1960 — Land, A.H.; Doig, A.G., *An Automatic Method for Solving Discrete Programming Problems*, Econometrica, vol. 28 1960, 497-520.

Lawler 1989 — Lawler, E.L.; Lenstra, J.K.; Rinnooy Kan, A.H.G.; Shmoys, D.B., *Sequencing and Scheduling: Algorithms and Complexity*, Report BS-R8909, Centre for Mathematics and Computer Science - Dep. of Oper. Research, Statisctics and System Theory, 1989.

Lawler 1993 — Lawler, Eugene L.; Lenstra, Jan Karel; Kan, Alexander H.G.Rinnooy; Shmoys, David B., 1993, *Sequencing and Scheduling: Algorithms and Complexity*, In Handbooks in Operations Research and Management Science (Nemhauser, G.L.; Kan, A.H.G. Rinnooy, eds.), Volume 4: Logistics of Production and Inventory, Graves, S.C.; Kan, A.H.G. Rinnooy; Zipkin, P.H. (eds.), North-Holland, Amsterdam, 1993, Chapter 2, 445-522.

Le Pape 1994 — Le Pape, Claude, *Scheduling as Intelligent Control of Decision-Making and Constraint Propagation*, In Intelligent Scheduling, Cap.3, Zweben, Monte; Fox, Mark S. (eds.), San Francisco, Morgan Kaufman, 1994.

Leach 1997a — Leach, N.P.; Makatsoris, H.; Richards, H.D., 1997, *Supply Chain Control: Trade-Offs and System Requirements*, Imperial College, UK, ESPRIT project 20544, X-CITTIC (do site da Web http://www.nimblesite.com/xcittic/default.htm).

Leach 1997b — Leach, Nick; Ristic, Mike, 1997, *Capacity Models for Scheduling the Virtual Enterprise*, Department of Mechanical Engineering, Imperial College, London, UK, ESPRIT project 20544, X-CITTIC (do site da Web http://www.nimblesite.com/xcittic/default.htm).

Lee 1992 — Lee, Hau L.; Billington, Corey, 1992, *Managing Supply Chain Inventory: Pitfalls and Opportunities*, Sloan Management Review, Spring 1992, pp. 65-73.

Lee 1993a — Lee, Hau L.; Billington, Corey, 1993, *Material Mangement in Decentralized Supply Chains*, Operations Research, Vol. 41, No. 5, September- October 1993, 835-847.

Lee 1993b — Lee, Hau L.; Billington, Corey; Carter, Brent, 1993, *Hewlett-Packard Gains Control of Inventory and Service Through Design for Localization*, Interfaces 23: 4 July-August 1993, 1-11.

Lee 1995 — Lee, Hau L.; Billington, Corey, 1995, *The Evolution of Supply-Chain-Management Models and Practice at Hewlett-Packard*, Interfaces 25: 5 September-October 1995, 42-63.

Lee 1996 — Lee, Hau L., 1996, *Effective Inventory and Service Management through Product and Process Redesign*, Operations Research, Vol. 44, No. 1, January-February 1996, 151-159.

Lenstra 1975 — Lenstra, J.K.; Rinnooy Kan, A.H.G., *A Recursive Approach to the Generation of Conbinatorial Configurations*, Working Paper WP/75/25, Graduate School of Management, Delft, 1975.

Lesser 1988 — Lesser, Victor R.; Corkill, Daniel D., 1988, *Functionally Accurate, Cooperative Distributed Systems*, Readings in Distributed Artificial Intelligence, Bond, Alan H.; Les Gasser (eds.), Morgan Kaufman Publishers, Inc., San Mateo, California, 1988, 295-310.

Liskov 1986 — Liskov, B.; Guttag, J., 1986, *Abstraction and Specification in Program Development*, Cambridge, Mass./New York: The MIT Press/MCgGraw-Hill, 1986.

Liu 1993 — Liu, JyiShane; Sycara, Katia P., 1993, *Distributed Constraint Satisfaction through Constraint Partition and Coordinated Reaction*, Proceedings of the 12th International Workshop on Distributed Artificial Intelligence, Hidden Valley, Pennsylvania, May 19-21, 1993, 263-279.

Liu 1994 — Liu, Jyi-Shane, 1994, *Collective Problem Solving Through Coordination in a Society of Reactive Agents*, Technical Report CMU-RI-TR-94-23, The Robotics Institute, Carnegie Mellon University, Pittsburg, Pennsylvania 15213, June 1994.

Luger 1993 — Luger, George F.; Stubblefield, William A., 1993, *Artificial Intelligence, Structures and Strategies for Complex Problem Solving*, The Benjamin/Cummings Publishing Company, Inc., Redwood City, California, USA, 1993.

Luo 1993a — Luo, Q.Y.; Hendry, P.G.; Buchanan, J.T., 1993, *Heuristic Search for Distributed Constraint Satisfaction Problems*, Technical Report KEG-6-93, Dep. of Computer Science, University of Strathclyde, Glasgow, Scotland, UK.

Luo 1993b — Luo, Q.Y.; Hendry, P.G.; Buchanan, J.T., 1993, *Comparison of Different Approaches for Solving Distributed Constraint Satisfaction Problems*, Technical Report KEG-7-93, Dep. of Computer Science, University of Strathclyde, Glasgow, Scotland, UK.

Luo 1993c — Luo, Q.Y.; Hendry, P.G.; Buchanan, J.T., 1993, *An Integrated Distributed Scheduler*, Technical Report KEG-8-93, Dep. of Computer Science, University of Strathclyde, Glasgow, Scotland, UK.

Mackworth 1977 — Mackworth, Alan K., 1977, *Consistency in Networks of Relations*, Artificial Intelligence, 8 1977, 99-118.

Mackworth 1985 — Mackworth, Alan K.; Freuder, Eugene C., 1985, *The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems*, Artificial Intelligence, 25 1985, 65-74.

Maes 1989 — Maes, Pattie, 1989, *The Dynamics of Action Selection*, Proceedings of the, IJCAI-89, 991-997.

Malone 1988 — Malone, Thomas W., 1988, *Modeling Coordination in Organizations and Markets*, Readings in Distributed Artificial Intelligence, Bond, Alan H.; Les Gasser (eds.), Morgan Kaufman Publishers, Inc., San Mateo, California, 1988, 285-293.

Malone 1990 — Malone, T. W., 1990, *Organizing Information Processing Systems: Parallels between Human Organizations and Computer Systems*, In *Cognition, Computation and Cooperation*, W. W. Zachary and S. P. Robertson (eds.), Ablex, Norwood, NJ, 1990, 56-83.

Mandrioli 1987 — Mandrioli, Dino; Ghezzi, Carlom, 1987, *Theoretical Foundations of Computer Science*, John Wiley & Sons, Inc., 1987, New York, USA.

Martial 1992 — Martial, Frank von, 1992, *Coordinating Plans of Autonomous Agents*, Springer-Verlag, Berlin, 1992.

Martin 1996 — Martin, Charles F., 1996, *The Next Generation of Analytic Techniques for Manufacturing Planning*, APICS - The Educational Society for Resource Management, 1996 (do site da Web http://www.apics.org/SIGs/Articles/next.htm).

Maxwell 1985 — Maxwell, William L.; Muckstadt, John A., 1985, *Establishing Consistent and Realistic Reorder Intervals in Production-Distribution Systems*, Operations Research, Vol. 33, No. 6, November-December, 1985, 1316-1341.

McCarthy 1979 — McCarthy, John, 1979, *Ascribing Mental Qualities to Machines*, Technical Report No. STAN-CS-79-725, Stanford University.

McDermott 1995 — McDermott, Drew; Hendler, James, *Planning: What it is, What it could be, An Introduction to the Special Issue on Planning and Scheduling*, Artificial Intelligence, 76 (1995), 1-16.

Minton 1992 — Minton, Steven; Johnston, Mark D.; Philips, Andrew B.; Laird, Philip, 1992, *Minimizing Conflicts: a Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems*, Artificial Intelligence 58, 1992; In Constraint-Based Reasoning, Freuder, Eugene C.; Mackworth, Alan K. (eds.), MIT Press, 1994, 161-205.

Mintzberg 1979 — Mintzberg, H., 1979, *The Structuring of Organizations*, Prentice-Hall, Englewoods Cliffs, NJ., 1979.

Miyashita 1994 — Miyashita, Kazuo; Sycara, Katia, 1994, *Adaptive Case-Based Control of Schedule Revision*, In Intelligent Scheduling, Monte Zweben and Mark S. Fox (eds.), Morgan Kaufman, 1994, Cap. 10.

Moily 1986 — Moily, Jaya P., 1986, *Optimal and Heuristic Procedures for Component Lot-Splitting in Multi-Stage Manufacturing Systems*, Management Science, Vol. 32, No. 1, January 1986, 113-125.

Morgado 1989 — Morgado, Ernesto M., 1989, *Tipos Abstractos de Informação*, 1ª Edição preliminar (notas de apoio à disciplina de Estruturas de Informação, Mestrado de Engenharia Mecânica, IST, Lisboa, 1989).

Morton 1988 — Morton, T.E.; Lawrence, S.R.; Rajagopolan, S.; Kekre, S., 1988, *SCHED-STAR: A Price-Based Shop Scheduling Module*, Journal of Manufacturing and Operations Management, 1988, 1, 131-181.

Morton 1993 — Morton, Thomas E.; Pentico, David W., 1993, *Heuristic Scheduling Systems, with Applications to Production Systems and Project Management*, John Wiley & Sons, Inc., USA.

Moulin 1996 — Moulin, Bernard; Chaib-Draa, Brahim, 1996, *An Overview of Distributed Artificial Intelligence*, In *Foundations of Distributed Artificial Intelligence*, G. M. P. O' Hare, N. R. Jennings (eds.),John Wiley & Sons, Inc., New York, USA., 1996 (Chapter 1).

Muckstadt 1987 — Muckstadt, John A.; Roundy, Robin, 1987, *Multi-Item, One-Warehouse, Multi-Retailer Distribution Systems*, Management Science, Vol. 33, No. 12, December 1987, 1613-1621.

Muckstadt 1993 — Muckstadt, John A.; Roundy, Robin O., 1993, *Analysis of Multistage Production Systems*, In Handbooks in Operations Research and Management Science (Nemhauser, G.L.; Kan, A.H.G. Rinnooy, eds.), Volume 4: Logistics of Production and Inventory, Graves, S.C.; Kan, A.H.G. Rinnooy; Zipkin, P.H. (eds.), North-Holland, Amsterdam, 1993, Chapter 2, 59-131.

Mulkens 1994 — Mulkens, Hubert, 1994, *Revisiting the Johnson Algorithm for Flow-Shop Scheduling with Genetic Algorithms*, In Knowledge-based Reactive Scheduling, Proceedings of the IFIP TC5/WG5.7 International Workshop on Knowledge-Based Reactive Scheduling, Szelke, E., Kerr, R.M (eds.), Athens, Greece, October, 1993, Elsevier Science, 1994, 69-80.

Muller 1994 — Muller, Claude; Magill, Evan H.; Prosser, Patrick; Smith, D. Geoffrey, 1994, *Distributed Genetic Algorithms for Resource Allocation*, In Scheduling of Production Processes, Dorn, J.; Froeschel, K. (eds.), Chapter 6, Elis Horwood, Ld., 1993.

Müller 1997 — Müller, Jörg P.; Wooldridge, Michael J.; Jennings, Nicholas R. (eds.), 1997, *Intelligent Agents III, Agent Theories, Architectures, and Languages*, Proceedings of the ECAI' 96 Workshop (ATAL) Budapest, Hungary, August 12-13, 1996, Springer Verlag, 1997.

Muscettola 1994a — Muscettola, Nicola, *HSTS: Integrating Planning and Scheduling*, In Intelligent Scheduling, Cap.6, Zweben, Monte; Fox, Mark S. (eds.), San Francisco, Morgan Kaufman, 1994.

Muscettola 1994b — Muscettola, Nicola, 1994, *An Experimental Analysis of Bottleneck-Centered Opportunistic Scheduling*, In Current Trends in AI Planning, Bäckström, C.; Sandewall, E. (eds.), IOS Press, Amsterdam, Netherlands, 1994, 240-253.

Neches 1991 — Neches, Robert; Fikes, Richard; Finin, Tim; Gruber, Thomas; Patil, Ramesh; Senator, Ted; Swartout, William R., 1991, *Enabling Technology for Knowledge Sharing*, AI Magazine, Volume 12, No. 3, Fall 1991.

Nilsson 1998 — Nilsson, Nils J., 1998, *Artificial Intelligence, A New Synthesis*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 1998.

Norvig 1992 — Norvig, Peter, 1992, *Paradigms of Artificial Intelligence Programming*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, USA, 1992.

Nudel 1983 — Nudel, Bernard, 1983, *Consisten-Labeling Problems and their Algorithms: Expected-Complexities and Theory-Based Heuristics*, Artificial Intelligence 21 (1983) 135-178.

Nuijten 1996 — Nuijten, W.P.M.; Aarts, E.H.L., 1996, *A Computational Study of Constraint Satisfaction for Multiple Capacitated Job Shop Scheduling*, European Journal of Operational Research, 90, 1996, 269-284.

Numao 1994 — Numao, Masyuki, 1994, *Development of a Cooperative Scheduling System for the Steel-Making Process*, In Intelligent Scheduling, Morgan Kaufman, 1994, Cap. 21.

Odell 2000 — Odell, James; Parunak, H. Van Dyke; Bauer, Bernard, 2000, *Representing Agent Interaction Protocols in UML*, artigo submetido à conferência AAAI Agents 2000, Barcelona, 3-7 Junho 2000 (do site da Web http://www.erim.org/~vparunak/papers.htm).

Ogbu 1990 — Ogbu, F.A.; Smith, D.K., *The Application of the Simulated Annealing Algorithm to the Solution of the n/m/Cmax Flowshop Problem*, Computers & Operations Research, 17 1990, 243-253.

Ogbu 1991 — Ogbu, F.A.; Smith, D.K., *Simulated Annealing for the Permutation Flow-Shop Scheduling*, OMEGA, 19 1991, 65-67.

O'Hare 1996 — G. M. P. O' Hare, N. R. Jennings (eds.),*Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, Inc., 1996, New York, USA.

O'Neill 1994 — O' Neill, Henrique; Sackett, Peter, 1994,*The Extended Manufacturing Enterprise Paradigm*, Management Decision, 1994, 32(8), pp. 42-49.

O'Neill 1996 — O' Neill, H.; Sackett, P., 1996,*The Extended Enterprise Reference Framework*, Balanced Automation Systems II, Luís M. Camarinha-Matos and Hamideh Afsarmanesh (eds.), 1996, Chapman & Hall, London, UK, pp 401-412.

Osman 1989 — Osman, L.H.; Potts, C.N., *Simulated Annealing for the Permutation Flow-Shop Scheduling*, OMEGA, 17 1989, 551-557.

Panwalkar 1977 — Panwalkar, S.S.; Iskander, Wafik, 1977, *A Survey of Scheduling Rules*, Operations Research, Vol. 25, No. 1, 45-61, 1977.

Parnell 1996 — Parnell, Clay, 1996, *Supply Chain Management*, APICS - The Educational Society for Resource Management, 1996 (do site da Web http://www.apics.org/SIGs/Articles/supply.htm).

Parunak 1995 — Parunak, H. Van Dyke, 1995, *The Heartbeat of the Factory:Understanding the Dynamics of Agile Manufacturing Enterprises*, Technical Report, Industrial Technology Institute (do site da Web http://www.erim.org/cec/sched/schedpubs.htm).

Parunak 1996a — Parunak, V., 1996, *Applications of Distributed Artificial Intelligence in Industry*, In *Foundations of Distributed Artificial Intelligence*, G. M. P. O' Hare, N. R. Jennings (eds.),Wiley Inter-Science, 1996 (Chapter 4).

Parunak 1996b — Parunak, H. Van Dyke, 1996, *Visualizing Agent Conversations: Using Enhanced Dooley Graphs for Agent Design and Analysis*, Proceedings of the 2nd. International Conference on Multi-Agent Systems, 275-282, AAAI Press, 1996.

Parunak 1996c — Parunak, H. Van Dyke; Savit, Robert; Riolo, Rick L.; Clark, Steven J., 1996, *DASCh: Dynamic Analysis of Supply Chains* (do site da Web http://www.erim.org/~vparunak/papers.htm).

Parunak 1998a — Parunak, H. Van Dyke; Ward, Allen; Sauter, John, 1998, *The MarCon Algorithm: A Systematic Market Approach To Distributed Constraint Problems*, Environmental Research Institute of Michigan/Center for Electronic Commerce (do site da Web http://www.erim.org/cec/sched/schedpubs.htm).

Parunak 1998b — Parunak, H. Van Dyke, 1998, *Practical and Industrial Applications of Agent-Based Systems*, Industrial Technology Institute (do site da Web http://www.erim.org/cec/sched/schedpubs.htm).

Parunak 1998c — Parunak, H. Van Dyke; VanderBok, Ray, 1998, *Modeling the Extended Supply Network*, Industrial Technology Institute (do site da Web http://www.erim.org/cec/sched/schedpubs.htm).

Parunak 1998d — Parunak, H. Van Dyke, 1998, *The DASCh Experience: How to Model a Supply Chain*, Environmental Research Institute of Michigan/Center for Electronic Commerce (apresentado na conferência ICCS' 98; d site da Web http://www.erim.org/cec/sched/schedpubs.htm).

Parunak 1998e — Parunak, H. Van Dyke, 1998, *What can Agents do in Industry, and Why? An Overview of Industrially-Oriented R&D at CEC*, In *Cooperative Information Agents II*, Klusch and Weiss (eds.), LNCS 1435, pp. 1-18 (também: Technical Report, Industrial Technology Institute, do site da Web http://www.erim.org/cec/sched/schedpubs.htm).

Patil 1998 — Patil, Ramesh S.; Fikes, Richard E.; Patel-Schneider, Peter F.; Mckay, Don; Finin, Tim, Gruber, Thomas, Neches, Robert, 1998, *The DARPA Knowledge Sharing Effort: Progress Report*, Readings in Agents, Huhns, Michael N.; Singh, Munindar P. (eds.), Morgan Kaufmann Publishers, San Francisco, California, 1998, 243-254.

Porter 1985 — Porter, Michael E.; Millar, Victor E., 1985, *How Information Gives you Competitive Advantage*, Harvard Business Review, July-August 1985, 149-160.

Pritsker 1969 — Pritsker, A.A.B.; Watters, L.J.; Wolfe, P.M., *Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach*, Management Science, vol. 16, pp. 93-108, 1969.

Prosser 1988 — Prosser, Patrick, 1988, *Reactive Factory Scheduling as a Dynamic Constraint Satisfaction Problem*, Technical Report AISL-31-88, University of Strathclyde, Glasgow, Scotland, UK, August 1988.

Prosser 1991 — Prosser, Patrick, 1991, *Hybrid Algorithms for the Constraint Satisfaction Problem*, Technical Report AISL-46-91, University of Strathclyde, Glasgow, Scotland, UK, September 1991.

Purdom 1983 — Purdom, Paul Walton, Jr., 1983, *Search Rearrangement Backtracking and Polynomial Average Time*, Artificial Intelligence 21 (1983) 117-133.

Rabelo 1994a — Rabelo, Ricardo J.; Camarinha-Matos, L.M., 1994, *Negotiation in Multi-Agent Based Dynamic Scheduling*, International Journal on Robotics & Computer-Integrated Manufacturing, Vol. 11, No. 4, December 1994.

Rabelo 1994b — Rabelo, Ricardo J.; Camarinha-Matos, L.M., 1994, *Control and Dynamic Scheduling in Virtual Organization of Production Resources*, IFIP WG5.7 Working Conference on Evaluation of Production Management Methods, Gramado-RS, Brasil, 1994 (do site da Web http://www.gsigma-grucon.ufsc.br/).

Rabelo 1996a — Rabelo, R.J.; Camarinha-Matos, L.M., 1996, *Towards Agile Scheduling in Extended Enterprise*, Balanced Automation Systems II, Luís M. Camarinha-Matos and Hamideh Afsarmanesh (eds.), 1996, Chapman & Hall, London, UK, pp 413-422.

Rabelo 1996b — Rabelo, Ricardo J.; Camarinha-Matos, Luís Manuel, 1996, *Deriving Particular Agile Scheduling Systems Using the HOLOS Methodology*, Studies in Informatics and Control, Vol. 5, Nº. 2, June 1996.

Rabelo 1997 — Rabelo, Ricardo José, 1997, *Um Enquadramento para o Desenvolvimento de Sistemas de Escalonamento Ágil da Produção*, Tese de Doutoramento, Engenharia Electrotécnica, Especialidade de Robótica e Manufactura Integrada, Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia, Lisboa, 1997.

Rabelo 1998a — Rabelo, R.J.; Camarinha-Matos, M., 1998, *Generic Framework for Conflict Resolution in Negotiation-Based Agile Scheduling Systems*, In proceedings of the 5th International Workshop on Intelligent Manufacturing Systems, IMS'98, Gramado, Brasil, November 1998.

Rabelo 1998b — Rabelo, Ricardo J.; Camarinha-Matos, Luis M.; Afsarmanesh, Hamideh, 1998, *Multiagent Perspectives to Agile Scheduling*, Basys' 98 – IEEE / IFIP International Conference on Balanced Automation Systems, Prague, Czech Republic, 26-28/08/98.

Rabelo 1998c — Rabelo, Ricardo J., 1998, *Escalonamento Ágil Inter-Organizacional: Uma Nova Dimensão à Coordenação da Produção*, GSIGMA, Federal University of Santa Catarina, Florianópolis, SC, Brazil (do site da Web http://www.gsigma-grucon.ufsc.br/).

Rabelo 1999 — Rabelo, R.J.; Camarinha-Matos, L.M.; Afsarmanesh, H., 1999, *Multiagent-based Agile Scheduling*, (do site da Web http://centaurus.dee.fct.unl.pt/~massyve/).

Rao 1991 — Rao, Anand S.; Georgeff, Michael P, 1991, *Modeling Rational Agents within a BDI-Architecture*, KR' 91, Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning, 1991, 473-484.

Reeves 1993a — Reeves, C.R.; Beasley, J.E., Chapter 1- *Introduction*, In Modern Heuristic Techniques for Combinatorial Problems, Reeves, C.R. (ed.), Blackwell Scientific Publications, 1993.

Reeves 1993b — Reeves, C.R., Chapter 4- *Genetic Algorithms*, In Modern Heuristic Techniques for Combinatorial Problems, Reeves, C.R. (ed.), Blackwell Scientific Publications, 1993.

Reis 1998 — Reis, J., *Recursos Discretos, uma Representação Baseada na Capacidade*, Relatório Interno ISCTE-DCTI-1998-005 do Departamento de Ciências e Tecnologias de Informação do I.S.C.T.E., Janeiro de 1998.

Ribeiro 1996 — Ribeiro, C., *Meta-Heuristics and Applications*, Chapter 4.3, Notas do EAIA-96, Escola Avançada de Inteligência Artificial 1996, Estoril, Portugal, 1996.

Rich 1991 — Rich, Elaine; Knight, Kevin, 1991, *Artificial Intelligence*, McGraw-Hill, 1991.

Roldão 1995 — Roldão, Victor Sequeira, *Planeamento e Programação da Produção*, Monitor, Lisboa, 1995.

Ross 1996 — Ross, David F., 1996, *Designing an Effective Quick Response System*, APICS - The Educational Society for Resource Management, 1996 (do site da Web http://www.apics.org/SIGs/Articles/designin.htm).

Roundy 1990 — Roundy, Robin O., 1990, *Computing Nested Reorder Intervals for Multi-Item Distribution Systems*, Operations Research, Vol. 38, No. 1, January-February, 1990, 37-52.

Rumbaugh 1999 — Rumbaugh, James; Jacobson, Ivar; Booch, Grady, 1999, *The Unified Modeling Language Reference Manual*, Addison Wesley Longman, Inc., 1999.

Russell 1995 — Russell, Stuart; Norvig, Peter, 1995, *Artificial Intelligence, A Modern Approach*, Prentice Hall, 1995.

Sackett 1994 — Sackett, P.; Wortmann, H.; Brown, J., 1994, *Manufacturing Business Challenges in the late 1990's*, Proc. 1st. SCMA Conference on Outstanding Business Success in Manufacturing, London.

Sadeh 1991 — Sadeh, Norman, 1991, *Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling*, PhD. Thesis/Technical Report CMU-CS-91-102, School of Computer Science, Carnegie Mellon University.

Sadeh 1994 — Sadeh, Norman, *Micro-Oportunistic Scheduling: The Micro-Boss Factory Scheduler*, In Intelligent Scheduling, Cap.4, Zweben, Monte; Fox, Mark S. (eds.), San Francisco, Morgan Kaufman, 1994.

Sadeh 1995a — Sadeh, Norman; Sycara, Katia; Xiong, Yalin, 1995, *Backtracking Techniques for the Job Shop Scheduling Constraint Satisfaction Problem*, Artificial Intelligence 76 (1995) 455-480.

Sadeh 1995b — Sadeh, Norman M.; Fox, Mark S., 1995, *Variable and Value Ordering Heuristics for the Job Shop Scheduling Constraint Satisfaction Problem*, Technical Report CMU-RI-TR-95-39, School of Computer Science, Carnegie Mellon University, Pittsburg, PA, USA. (obtido na Web).

Sadeh 1996 — Sadeh, Norman M.; Hildum, David W.; Laliberty, Thomas J.; Smith, Stephen F.; McA' Nulty, John; Kjenstad, Dag, 1996,*An Integrated Process-Planning/Production-Scheduling Shell for Agile Manufacturing*, Technical Report CMU-RI-TR-96-10, The Robotics Institute, Carnegie Mellon University, Pittsburg, Pennsylvania 15213, May, 1996.

Sauer 1993 — Sauer, Jürgen, 1993, *Meta-Scheduling Using Dynamic Scheduling Knowledge*, In Scheduling of Production Processes, Dorn, J.; Froeschel, K. (eds.), Chapter 14, Elis Horwood, Ld., 1993.

Schoppers 1989 — Schoppers, Marcel J., 1989, *In Defense of Reaction Plans as Caches*, AI Magazine, 1989

Schrag 1992 — Schrag, Robert; Boddy, Mark; Carciofini, Jim, 1992, *Managing Disjunction for Practical Temporal Reasoning*, Principles of Knowledge Representation and Reasoning, Proceedings of the Third International Conference (KR' 92), 36-46, 1992, Morgan Kaufmar California.

Schroeder 1993 — Schroeder, Roger G., 1993, *Operations Management, Decision Making in the Operations Function*, McGraw-Hill International Editions, 1993.

Schwarz 1975 — Schwarz, L.B.; Schrage, L., 1975, *Optimal and System Myopic Policies for Multi-Echelon Production/Inventory Assembly Systems*, Management Science 21, 1285-1294, 1975.

Schwarz 1978 — Schwarz, L.B.; Schrage, L., 1978, *On Echelon Holding Costs*, Management Science 24, 865-866, 1978.

Scott 1991 — Scott, Charles; Westbrook, Roy, 1991, *New Strategic Tools for Supply Chain Management*, International Journal of Physical Distribution & Logistics Management, Vol. 21, No. 1, 1991, 23-33.

Searle 1969 — Searle, J., 1969, *Speech Acts*, Cambridge University Press, Cambridge, UK, 1969.

Searle 1984 — Searle, John, 1984, *Minds, Brains and Science*, Mente, Cérebro e Ciência (edição em português), Biblioteca de Filosofia Contemporânea, Edições 70, 1997.

Shapiro 1993 — Shapiro, Jeremy F., *Mathematical Programming Models and Methods for Production Planning and Scheduling*, In Handbooks in OR & MS, Graves, S.C. et al. (eds.), Vol. 4, Cap. 8, Elsevier Publishers B.V., 1993.

Shoham 1993 — Shoham, Yoav, 1993, *Agent-Oriented Programming*, Artificial Intelligence 60 (1993), 51-92.

Sichman 1998 — Sichman, Jaime S.; Conte, Rosaria; Gilbert, Nigel, 1998, *Mulit-Agent Systems and Agent-Based Simulation*, Proceedings of the First International Workshop, MABS' 98, Paris France, July 199ε Springer-Verlag, Berlin, 1998.

Singh 1998 — Singh, Munindar P., 1998, *Developing Formal Specifications to Coordinate Heterogeneous Autonomous Agents*, Proceedings of the International Conference on Multi-Agent Systems, ICMAS-98, IEEE Computer Society, Los Alamitos, California, 1998, 261-268.

Smith 1988a — Smith, Reid G.; Davis, Randall, 1988, *Frameworks for Cooperation in Distributed Problem Solving*, Readings in Distributed Artificial Intelligence, Bond, Alan H.; Les Gasser (eds.), Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

Smith 1988b — Smith, Reid G., 1988, *The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver*, Readings in Distributed Artificial Intelligence, Bond, Alan H.; Les Gasser (eds.), Morgan Kaufman Publishers, Inc., San Mateo, California, 1988, 357-366.

Smith 1990 — Smith, Stephen F.; Ow, Peng Si; Potvin, Jean-Yves; Muscettola, Nicola; Matthys, Dirk C., 1990, *An Integrated Framework for Generating and Revising Factory Schedules*, Journal of the Operational Research Society, 41 (6) 539-552, 1990.

Smith 1994 — Smith, Stephen, *OPIS: A Methodology and Architecture for Reactive Scheduling*, In Intelligent Scheduling, Cap.2, Zweben, Monte; Fox, Mark S. (eds.), San Francisco, Morgan Kaufman 1994.

Smith 1995 — Smith, Barbara M., 1995, *A Tutorial on Constarint Programming*, Research Report 95.15, Apr-1995, Division of Artificial Intelligence, University of Leeds.

Soares 1998 — Soares, Pedro Miguel Gouveia Monteiro, 1998, *Construção Automática de Horários*, Tese de Mestrado, Departamento de Engenharia Electrotécnica e de Computadores, Instituto Superior Técnico, Universidade Técnica de Lisboa, Junho de 1998.

Spearman 1990 — Spearman, Mark L.; Woodruff, David L.; Hopp, Wallace J., 1990, *CONWIP: A Pull Alternative to Kanban*, International Journal of Production Research 1990, Vol. 28, No. 5, 879-894.

Spearman 1992 — Spearman, Mark L.; Zazanis, Michael A., 1992, *Push and Pull Production Systems: Issues and Comparisons*, Operations Research, Vol.40, No. 3, May-June 1992.

Spinosa 1998a — Spinosa, L.M.; Rabelo, R. J.; Klen, A.P.; Ferreira, A.C., 1998, *An Oriented Decision Support System Model for Virtual Enterprise Coordination*, In proceedings of Prolamat' 98 / The Tenth Internationa IFIP TC5 WG-5.2 WG-5.3 Conference, September, 1998, Trento, Italy.

Spinosa 1998b — Spinosa, Luiz Marcio; Hofmann, Ana C.M.; Rabelo, Ricardo J.; Klen, Alexandra A. Pereira, 1998, *Basic Services for the management of Virtual Enterprises: A Case Study*, Basys' 98, IEEE IFIP Internationa Conference on Balanced Automation System, Praga, Rep. Tcheca.

Stallman 1977 — Stallman, Richard M.; Sussman, Gerald J., 1977, *Forward Reasoning and Dependency-Directed Backtracking in a System for a Computer-Aided Circuit Analysis*, Artificial Intelligence 9 (1977) 135-196.

Steele 1990 — Steele, Guy L., 1990, *Common Lisp, the Language*, Digital Press/Butterworth-Heinemann, 1990.

Stefik 1995 — Stefik, Mark, 1995, *Introduction to Knowledge Systems*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 1995.

Sullivan 1990 — Sullivan, Gerald; Fordyce, Kenneth, 1990, *IBM Burlington' s Logistic Management System*, Interfaces 20: 1 January-February 1990, 43-64.

Sycara 1989 — Sycara, K., 1989, *Multiagent Compromise via Negotiation*, In Distributed Artificial Intelligence, L. Gasser and M. N. Huhns (eds.), Vol. 2, 119-137, Morgan Kaufmann, Los Altos, CA/Pitman, London, 1989.

Sycara 1991a — Sycara, Katia P.; Roth, Steven F.; Sadeh, Norman; Fox, Mark S., 1991, *Resource Allocation in Distributed Factory Scheduling*, IEEE Expert, February, 1991, 29-40.

Sycara 1991b — Sycara, Katia; Roth, Steve; Sadeh, Norman; Fox, Mark, 1991, *Distributed Constrained Heuristic Search*, IEEE Transactions on System, Man, and Cybernetics, Vol. 21, No.6, November/December 1991, 1446-1461.

Sycara 1994 — Sycara, P. Katia; Miyashita, Kazuo, 1994, *Adaptive Schedule Repair*, In Knowledge-based Reactive Scheduling, Proceedings of the IFIP TC5/WG5.7 International Workshop on Knowledge-Based Reactive Scheduling, Szelke, E., Kerr, R.M (eds.), Athens, Greece, October, 1993, Elsevier Science, 1994, 39-55.

Szelke 1994 — Szelke, Elizabeth; Kerr, Roger (eds.), 1994, *Knowledge Based Reactive Scheduling*, Proceedings of the IFIP TC5/WG5.7 International Workshop on Knowledge-Based Reactive Scheduling, Athens, Greece, October, 1993, Elsevier Science, 1994, ISBN 0444818146.

Szelke 1995 — Szelke, E.; Markus, G., 1995, *A Blackboard Based Perspective of Reactive Scheduling*, In Artificial Intelligence in Reactive Scheduling, Kerr, Roger; Szelke, Elizabeth (eds.), Chapter 6, Chapman & Hall, 1995.

Szendrovits 1975 — Szendrovits, A.Z., 1975, *Manufacturing Cycle Time Determination for a Multi-Stage Economic Production Quantity Model*, Management Science 22(3), 298-308, 1975.

Szendrovits 1978 — Szendrovits, A.Z., 1978, *A Comment on 'Optimal and System Myopic Policies for Multi-Echelon Production/Inventory Assembly Systems'*, Management Science 24, 863-864, 1978.

Tate 1995 — Tate, Austin; Drabble, Brian; Dalton, Jeff, 1995, *An Engineer' Approach to the Application of Knowledge Based Planning and Scheduling Techniques to Logistics - The O-Plan Project*, Artificial Intelligence Applications Institute, The University of Edinburgh, 14 July 1995.

Tate 1996 — Tate, Austin; Drabble, Brian; Dalton, Jeff, 1996, *O-Plan: a Knowledge-Based Planner and its Application to Logistics*, Technical Report, Artificial Intelligence Applications Institute, The University of Edinburgh, 1996.

Tatsiopoulos 1983 — Tatsiopoulos, I.P.; Kingsman, B.G., 1983, *Lead Time Management*, European Journal of Operational Research, 14 1983, 351-358.

Teixeira 1997 — Teixeira, E.M.A.; Makatsoris, C.; Besant, C.B., 1997, *Distributed Capacity Analysis for Proactive Planning in Semiconductor Virtual Enterprises*, ESPRIT project 20544, X-CITTIC (do site da Web http://www.nimblesite.com/xcittic/default.htm).

Tel 1999 — Tel, Gerard, 1999, *Distributed Control Algorithms for AI*, In *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence*, Gerhard Weiss (ed.), The MIT Press, 1999 (Chapter 13).

Tempelmeier 1994 — Tempelmeier, Horst; Helber, Stefan, 1994, *A Heuristic for Dynamic Multi-Item Multi-Level Capacitated Lotsizing for General Product Structures*, European Journal of Operational Research 75 (1994) 296-311.

Terpstra 1996 — Terpstra, Victor, 1997, *Batch Scheduling within the Context of Intelligent Supervisory Process Control*, PhD thesis, Department of Electrical Engineering, Delft University of Technology, The Netherlands, 1996, edicao do autor, ISBN 90-9009848-8.

Tersine 1988 — Tersine, Richard J., 1988, *Principles of Inventory and Materials Management*, North-Holland/Elsevier SciencePublishing Co., 1988.

Thomas 1996 — Thomas, Douglas J.; Griffin, Paul M., 1996, *Coordinated Supply Chain Management*, European Journal of Operational Research, 94 (1996) 1-15.

Thorelli 1986 — Thorelli, Hans B., 1986, *Networks: Between Markets and Hierarchies*, Strategic Management Journal, 7, 1986, 37-51.

Towill 1982 — Towill, D.R., 1982, *Dynamic Analysis of an Inventory and Order Based Production Control System*, International Journal of Production Research, 1982, Vol. 20, No. 6, 671-687.

Towill 1992 — Towill, D. R.; Naim, M. M.; Wilkner, J., 1992, *Industrial Dynamics Simulation Models in the Design of Supply Chains*, International Journal of Phisical Distribution & Logistic Management 1992, 22(5), pp. 3-13.

Van Laarhoven 1988 — Van Laarhoven, P.J.M.; Aarts, E.H.L., *Simulated Annealing: Theory and Applications*, Kluwer, Dordrecht, 1988.

Veinott 1965 — Veinott, A. F., Jr., 1965, *Optimal Policy for a Multi-Product, Dynamic Non-Stationary Inventory Problem*, Management Science 12(3), 206-222, 1965.

Veinott 1966 — Veinott, A. F., Jr., 1966, *The Status of Mathematical Inventory Theory*, Management Science 12(11), 745-777, 1966.

Ventura 1999 — Ventura, Rodrigo; Aparício, Pedro; Lima, Pedro, 1999, *Agent-Based Programming Language for Multi-Agent Teams*, ISR Internal Report RT-701-99, RT-401-99 (16 pages), 1999 (do site da Web http://www.isr.ist.utl.pt/~yoda/aims/publications.html).

Vere 1983 — Vere, Steven A., 1990, *Planning in Time: Windows and Durations for Activities and Goals*, IEEE, 1983, 5 (3).

Vilain 1990 — Vilain, Marc; Kautz, Henry; Beek, Peter van, 1990, *Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report*, Qualitative Reasoning about Physical Systems, Morgan Kaufman Publishers, Inc., San Mateo, California, 1990, 373-381.

Vollmann 1997 — Vollmann, Thomas E.; Berry, William L.; Whybark, D. Clay, 1997, *Manufacturing Planning and Control Systems*, Irwin/McGRaw-Hill, 1997.

Walsh 1998 — Walsh, William E.; Wellman, Michael P., 1998, *A Market Protocol for Decentralized Task Allocation*, Proceedings of the International Conference on Multi-Agent Systems, ICMAS-98, IEEE Computer Society, Los Alamitos, California, 1998, 325-332.

Watt 1990 — Watt, David A., 1990, *Programming Language Concepts and Paradigms*, Prentice Hall International (UK) Ltd., 1990.

Weiss 1999 — Weiss, Gerhard (ed.), 1999, *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, 1999.

Weld 1990 — Weld, Daniel S.; Kleer, Johan de (eds.), 1990, *Qualitative Reasoning about Physical Systems*, Qualitative Reasoning about Physical Systems, Morgan Kaufman Publishers, Inc., San Mateo, California, 1990.

Wellman 1993 — Wellman, Michael P., 1993, *A Market-Oriented Programming Environment and its Application to Distributed Multicommodity Flow Problems*, Journal of Artificial Intelligence Research 1 (1993) 1-23 (do site da Web http://ai.eecs.umich.edu/people/wellman/).

Wikner 1991 — Wikner, J.; Towill, D.R.; Naim, M., 1991, *Smoothing Supply Chain Dynamics*, International Journal of Production Economics, 22 (1991) 231-248.

Williams 1981 — Williams, Jack F., 1981, *Heuristic Techniques for Simultaneous Scheduling of Production and Distribution in Multi-Echelon Structures: Theory and Empirical Comparisons*, Management Science, Vol. 27, No. 3, March 1981, 336-352.

Williams 1983 — Williams, Jack F., 1983, *A Hybrid Algorithm for Simultaneous Scheduling of Production and Distribution in Multi-Echelon Structures*, Management Science, Vol. 29, No. 1, January 1983, 77-92.

Williams 1986 — Williams, Brian C., 1986, *Doing Time: Putting Qualitative Reasoning on Firmer Ground*, Proceedings of AAAI-86, 105-112.

Williamson 1975 — Williamson, O. E., 1975, *Markets and Hierarchies: Analysis and Antitrust Implications*, Free Press-Macmillan, New York, 1975.

Wooldridge 1995 — Wooldridge, Michael; Jennings, Nicholas R., 1995, *Inteligent Agents: Theory and Practice*, artigo submetido a *Knowledge Engineering Review*, em Outubro de 1994 (revisão com data de Janeiro de 1995).

Wooldridge 1999 — Wooldrige, Michael, 1999, *Intelligent Agents*, In *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence*, Weiss, Gerhard (ed.), The MIT Press, 1999 (Chapter 1).

Wright 1989 — Wright, M.B., *Applying Stochastic Algorithms to a Locomotive Scheduling Problem*, JORS, 38 1989, 187-192.

Yang 1988 — Yang, Y.C.; Liu, M.C.; Cochran, J.K., 1988, *Knowledge Based Scheduling for Flexible Manufacturing Systems*, ACM, 0-89791-271-3-/88/0006/1075, 1988.

Yokoo 1990 — Yokoo, Makoto; Ishida, Toru; Kuwabara, K., 1990, *Distributed Constraint Satisfaction for DAI Problems*, Proceedings of the 10th International Workshop on DAI, 1990.

Yokoo 1991 — Yokoo, Makoto; Durfee, Edmund H., 1991, *Distributed Constraint Optimization as a Formal Model of Partially Adversarial Cooperation*, Technical Report CSE-TR-101-91, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, USA.

Yokoo 1993 — Yokoo, Makoto, 1993, *Dynamic Variable/Value Ordering Heuristics for Solving Large-Scale Distributed Constraint Satisfaction Problems*, Proceedings of the 12th International Workshop on Distributed Artificial Intelligence, Hidden Valley, Pennsylvania, May 19-21, 1993, 407-422.

Yokoo 1999 — Yokoo, Makoto; Ishida, Toru, 1999, *Search Algorithms for Agents*, In *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence*, Gerhard Weiss (ed.), The MIT Press, 1999, (Chapter 4).

Zangwill 1966 — Zangwill, W.I., 1966, *A Deterministic Multi-Period Production Scheduling Model with Backlogging*, Management Science 13 (1966) 105-119.

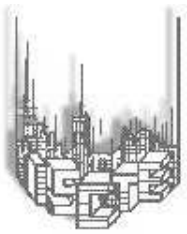
Zangwill 1969 — Zangwill, W. I., 1969, *A Backlogging Model and a Multi-Echelon Model of a Dynamic Economic Lot Size Production System*, Management Science 15, 506-527, 1969.

Zlotkin 1990 — Zlotkin, Gilad; Rosenschein, Jeffrey S., 1990, *Negotiation and Conflict Resolution in Non-Cooperative Domains*, Proceedings Eigth National Conference on Artificial Intelligence, Volume One, AAAI Press/The MIT Press, 1991, 100-105.

Zweben 1992 — Zweben, Monte; Davis, Eugene; Drascher, Ellen; Deale, Michael; Eskey, Megan, 1992, *Learning to Improve Constraint-Based Scheduling*, Artificial Intelligence 58, 1992, 271-296.

Zweben 1994a — Zweben, Monte; Fox, Mark, 1994, *Intelligent Scheduling*, Morgan Kaufman, 1994, ISBN 1-55860-260-7.

Zweben 1994b — Zweben, Monte; Daun, Brian; Davis, Eugene; Deale, Michael, 1994, *Scheduling and Rescheduling with Iterative Repair*, In Intelligent Scheduling, Morgan Kaufman, 1994, Cap. 8.



Instituto Superior
de Ciências do Trabalho e da Empresa

Um Modelo de Escalonamento Multi-Agente na Empresa Estendida

—— ERRATA ——

Joaquim António Marques dos Reis
(Mestre em Engenharia Mecânica)

ERRATA
de:
Dissertação para obtenção do Grau de Doutor
em Ciências e Tecnologias de Informação
(especialidade de Inteligência Artificial)

Lisboa, Abril de 2002

(versão revista e corrigida em Abril de 2002)

Errata

Após a impressão dos exemplares da dissertação "Um Modelo de Escalonamento Multi-Agente na Empresa Estendida", foi verificada uma incorrecção que aqui é indicada e para a qual se sugere correcção.

página 174 (Capítulo 4), na nota de pé nº 11 há duas pequenas incorrecções, quando se afirma serem as tarefas de retalho e de matéria-prima idênticas aos pedidos que as originam. Embora esta incorrecção não tenha importância maior para efeitos da actividade de escalonamento na perspectiva temporal, existe uma incoerência com a definição dada no Capítulo 3 para aquelas classes de tarefas, em que aí se definem como sendo idênticas a eventos no nível físico do modelo. Ora eventos e pedidos diferem (apenas) no significado do conteúdo do componente local, que no evento deve ser um local físico e no pedido deve ser um local virtual. Em rigor, as identidades sugeridas na nota de pé nº11, que são:

$\mathbb{O}^{r}_{i_r,r}$=TAREFA($\mathbb{d}^{0,r}_{i_r,r}$) e $\mathbb{O}^{m}_{i_r,r}$=TAREFA($\mathbb{d}^{i,m}_{i_r,r}$)

deveriam ser, respectivamente:

$\mathbb{O}^{r}_{i_r,r}$=TAREFA(EVENTO(PQ-CONJ($\mathbb{d}^{0,r}_{i_r,r}$),TEMPO($\mathbb{d}^{0,r}_{i_r,r}$),v_r)) e

$\mathbb{O}^{m}_{i_r,r}$=TAREFA(EVENTO(PQ-CONJ($\mathbb{d}^{i,m}_{i_r,r}$),TEMPO($\mathbb{d}^{i,m}_{i_r,r}$),v_m))

em que v_r e v_m são os identificadores dos nós de retalho e de matéria-prima (locais físicos) associados aos agentes de retalho e de matéria-prima g_r e g_m, respectivamente. As incorrecções aqui referidas repetem-se mais à frente na página 182 (Capítulo 4), linhas de texto 13 e 20, na página 249 (Capítulo 4), linha de texto nº 25, e na página 252 (Capítulo 4), linha de texto nº 2, onde são aplicáveis correcções similares à que foi indicada, substituindo $\mathbb{d}^{i,m}_{i_r,r}$ por $\mathbb{d}^{k,m}_{i_r,r}$. As mesmas incorrecções voltam a repetir-se na página 250, linha de texto nº 9, e na página 252, linha de texto nº 15. A correcção destas últimas envolveria a redefinição das duas operações onde ocorrem, isto é:

TAREFA($\mathbb{PE}^{r}_{i_r,r}$) ::= TAREFA(PEDIDO-DE-JUSANTE($\mathbb{PE}^{r}_{i_r,r}$)).

e

TAREFA($\mathbb{PE}^{m}_{i_r,r}$) ::= TAREFA(PEDIDO-DE-JUSANTE($\mathbb{PE}^{m}_{i_r,r}$)).

que deveriam ser, respectivamente:

TAREFA($\mathbb{PE}^{r}_{i_r,r}$,v_r) ::=
TAREFA(EVENTO(PQ-CONJ(PEDIDO-DE-JUSANTE($\mathbb{PE}^{r}_{i_r,r}$)),
TEMPO(PEDIDO-DE-JUSANTE($\mathbb{PE}^{r}_{i_r,r}$)),
v_r)).

e

TAREFA($\mathbb{PE}^{m}_{i_r,r}$,v_m) ::=
TAREFA(EVENTO(PQ-CONJ(PEDIDO-DE-JUSANTE($\mathbb{PE}^{m}_{i_r,r}$)),
TEMPO(PEDIDO-DE-JUSANTE($\mathbb{PE}^{m}_{i_r,r}$)),
v_m)).

em que v_r e v_m têm o significado já acima referido. A incoerência subjacente às incorrecções descritas não mais voltará a ser referida nesta errata.