A · R · T · I · G · O · S

Programação da produção

Um algoritmo

*Victor Sequeira Roldão**

A flexibilidade do sistema de produção pode ser conseguida através da redução de tempo total de ciclo, que por sua vez decorre de uma boa programação. Nesse sentido é estruturado um algoritmo de sequenciamento que permite reduzir o tempo total do ciclo e cumprir mais eficientemente prazos de entrega. O algoritmo baseia-se na programação iterativa e sucessiva da máquina mais estrangulada e recorre à teoria dos grafos, e à combinação de regras heurísticas.

1. INTRODUÇÃO

Do Sistema Neurológico enquanto metáfora, resultam vários princípios orientadores para a empresa como um todo, e em particular para o sistema de produção: dos quais são de referir – a integração e a flexibilidade.

A flexibilidade pode ser conseguida através de um *mix* alargado de produtos, de equipamentos gerais (multipurpose), da possibilidade de encaminhamentos alternativos, ou ainda, de uma programação da produção eficiente. Neste último caso a flexibilidade vai decorrer da redução do tempo total de fluxo que vai permitir dispor de um intervalo de tempo maior para produzir outros produtos, e, do aumento de folgas em relação à data de entrega.

De acordo com SKINNER (1985) as actividades numa fábrica podem ser caracterizadas como: 75% em fluxo de informação e 25% em fluxo de materiais.

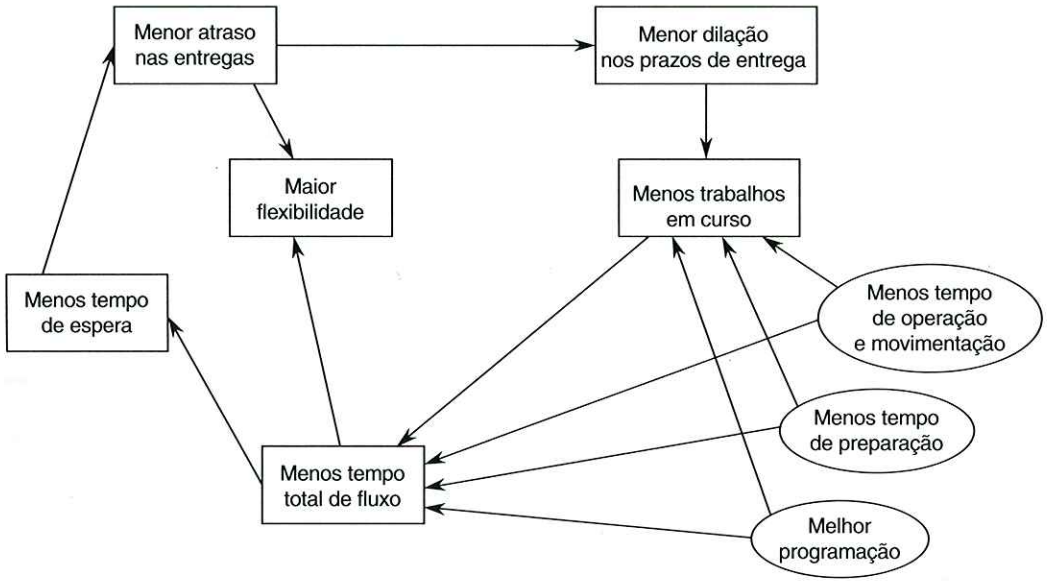
Assim é essencialmente actuando sobre o sistema de informação que se pode actuar na flexibilidade, mas pode considerar-se que várias outras variáveis são actuáveis nesse sentido:

A Programação de Produção, o Tempo de Preparação e a Dimensão do Lote, podem permitir reduzir o tempo total de ciclo e, na medida em que disponibiliza capacidade permite aumentar a flexibilidade (consultar Figura 1).

Se se actuar conjugadamente sobre: redução do tempo de preparação, redução de tempos de

* Engenheiro Mecânico, Director Geral de *Datinvest*, Docente no ISCTE.

Figura 1



operação e movimentação, redução do volume de trabalhos em curso, – o tempo total de fluxo sofre um decréscimo, que potencia ainda mais a redução do volume de trabalhos em curso, e vai reduzir por efeito dominó, o tempo total de ciclo, permitindo uma mudança mais rápida de tarefa (ou lote em fabrico).

Também o balanceamento dos trabalhos permite aumentar a flexibilidade, pois quando o fluxo de trabalhos não está balanceado, o tempo de fluxo é obviamente maior, pois os tempos de espera crescem quando a sequência não está sincronizada. O principal responsável pela falta de sincronização é a variabilidade dos tempos de processamento que origina «bloqueios» e «esperas» que diminuem a capacidade do sistema, o que, pode ser reduzido através de colocação de stocks de segurança nas sucessivas estações de trabalho.

A sincronização porque incide sobre o processo global, é da máxima relevância por permitir atingir um equilíbrio global de fluxos, o que passa por uma análise cuidada dos «pontos de estrangulamento» que determinam o débito do processo. Para gerir esses pontos de estrangulamento são acuadas ainda com maior intensidade as variáveis anteriormente descritas.

Do conjunto de variáveis que permite aumentar a flexibilidade, é de seguida abordada a programação da produção através do sequenciamento.

2. *SEQUENCIAMENTO*

Em sistemas intermitentes existem várias técnicas que permitem realizar o sequenciamento:

— Técnicas de Programação Matemática;

— Regras heurísticas simples;

— Regras heurísticas complexas (em que se incluem os algoritmos genéticos);

— Grafos;

— Algoritmos específicos.

— Programação Dinâmica (em que podem considerar-se incluídas as técnicas de Branch & Bound).

Numa programação intermitente em que existem n tarefas para m máquinas o número de programas possíveis é $(n\,!)^m$. Mesmo numa pequena oficina

$n = 10$

$m = 15$

O número de programas é $(10!)^{15}$ e, embora muitos dos programas não sejam viáveis, em virtude das restrições de encaminhamento, o número de programas possíveis é mesmo assim muito elevado.

Em virtude do grau de dificuldade exposto a utilização de técnicas de programação matemática e dinâmica torna-se morosa e muito complexa pelo que são neste caso utilizados com grande frequência regras e algoritmos heurísticos.

Os algoritmos heurísticos que funcionam em função de objectivos e têm um procedimento de verificação utilizam-se em resposta a situações que se repetem frequentemente e, embora considerando que o contexto pode originar uma enorme diversidade de estímulos, este tipo de regras se convenientemente simuladas e testadas podem originar bons comportamentos.

3. UM ALGORITMO PARA SEQUENCIAMENTO DE SISTEMAS INTERMITENTES

3.1 Descrição geral

As técnicas de sequenciamento vão assim das mais simples regras heurísticas, passando por algoritmos em vários estágios até técnicas de programação matemática complexas.

O algoritmo de sequenciamento que agora proponho é intermédio em termos de complexidade, mas de uma enorme eficiência. Iterativo em vários passos, recorre à teoria dos grafos, a regras heurísticas simples e à representação através de diagramas de GANT.

A metodologia é a seguinte:

1º Identificar a máquina mais estrangulada;

2º Sequenciar essa máquina segundo regras heurísticas «PP»;

3º Com a configuração obtida traçar um grafo e determinar as datas ao mais cedo e ao mais tarde para o conjunto;

4º Identificar a seguinte máquina mais estrangulada, e a partir da informação obtida em 3, sequenciar essa máquina segundo as regras heurísticas «PP»;

5º Desenhar o gráfico de GANT e proceder ao rearranjo das restantes máquinas já programadas;

6º Continuar a proceder identicamente enquanto houver máquinas para programar.

O algoritmo funciona com os seguintes pressupostos:

a) Cada máquina realiza apenas uma operação de cada vez.

b) O tempo de processamento de uma tarefa numa máquina é designado por operação.

c) Uma operação não pode ser processada sem que a anterior esteja concluída.

d) O único recurso limitado são as máquinas.

e) Não são consideradas interrupções por ruptura das máquinas.

f) As sequências dentro de cada tarefa são pré-definidas.

g) As tarefas são independentes umas das outras, isto é, não existe montagem.

h) Não existem máquinas em paralelo.

i) Não é considerado o tempo de movimentação e preparação quando se muda de máquina, o que é incluído nos tempos de processamento.

3.2 Identificação de estrangulamentos

O primeiro procedimento do algoritmo de sequenciamento que proponho consiste *em identificar a máquina mais estrangulada*, o que não é original, vários autores o têm feito, dos quais se destacam Goldratt e Adams.

GOLDRATT (1993) começa por identificar os estrangulamentos, a partir do que, «programa em desenvolvimento para diante» as tarefas críticas e «programa em desenvolvimento para trás» as tarefas não críticas.

ADAMS (1988) quando utiliza o SBP também baseia a resolução na identificação da máquina mais estrangulada na qual de seguida são sequenciadas as operações. A forma como identifica o estrangulamento baseia-se no maior atraso em relação à data de entrega.

RAMAN (1993) propõe um algoritmo GSP (*Global Scheduling Procedure*) em que a solução inicial é gerada por aplicação da regra heurística MOD, com os prazos de entrega das operações fixadas nos máximos valores que podem assumir sem atrasar as tarefas correspondentes. As tarefas processadas em todas as máquinas são então

agrupadas por ordem crescente do seu prazo de entrega.

CHOI (1994) desenvolve um algoritmo baseado em Branch & Bound em que o objectivo é no entanto maximizar as folgas dentro do cumprimento dos prazos de entrega – considera Choi que maximizar as folgas traz mais flexibilidade ao processo.

Outros autores como N. RAMAN (1993) identificam o estrangulamento através da carga relativa da cada máquina (Tempo de carga: tempo até à entrega).

No algoritmo de sequenciamento que utilizo, é identificada como máquina mais estrangulada a máquina com maior carga absoluta ou seja a máquina em que é máximo o somatório dos tempos de processamento.

$$\max \Sigma\ TP_{ij}$$

(Uma alternativa mais pesada, seria traçar o grafo, definir o caminho crítico, e, dentro do caminho crítico escolher a máquina mais solicitada).

3.3 Optimização do sequenciamento em cada máquina

Para resolver o problema de sequenciamento realizado em cada máquina (por ordem de estrangulamento) podem utilizar-se os algoritmos de sequenciamento de uma máquina só.

RAGHAVACHARI (1986) define que para qualquer data de entrega (mas com data de entrega comum), as tarefas são processadas por ordem decrescente dos tempos de processamento até que a tarefa com menor TPRC (Tempo de Processamento Mais Curto) seja processada, e de seguida por ordem crescente de tempos de processamento. Ou seja, ordenar todas as tarefas que acabem antes de d_i segundo a regra do tempo de processamento mais longo, e a partir daí segundo o tempo de processamento mais curto – É uma estrutura em V.

Para datas de entrega variáveis, PANWALKAR (1993) apresenta um algoritmo heurístico $(P-S-K)$ que minimiza o atraso médio para o sequenciamento de uma máquina só, assim formulado: Considera-se um conjunto de tarefas ordenadas segundo o critério TPRC. Todas as tarefas não sequenciadas são de início ordenadas da esquerda para a direita começando do TPRC mais baixo. C é o somatório dos tempos de processamento de todas as tarefas já programadas, ou seja, o tempo de conclusão da última tarefa programada. TP_k e d_k são os tempos de processamento e a data de entrega da tarefa k.

As tarefas são inicialmente colocadas por ordem crescente de TPRC da esquerda para a direita. O algoritmo faz as passagens da esquerda para a direita, toma uma tarefa e programa-a na posição k, através de um conjunto de passos em que sucessivamente são comparados os tempos de processamento acumulados e as durações.

Verifica-se que a utilização das regras de sequenciamento de uma máquina só de Raghavachari e Panwalkar, quando aplicadas num conjunto de máquinas (uma de cada vez por ordem de criticidade de estrangulamento) originam resultados pobres (ensaiados no decurso desta investigação). Alternativamente, foi desenvolvido um algoritmo, que se apresenta de seguida:

3.3.1 Algoritmo PP

Pretende-se com este algoritmo sequenciar a máquina mais estrangulada através dos seguintes passos:

1. Agrupar as operações a processar na máquina k por ordem não decrescente de datas ao mais cedo em que a operação pode realizar-se, o que passa a constituir o conjunto U.
2. Escolher a operação com data ao mais cedo de U designada por operação ij e colocá-la no conjunto S. Acumular o tempo de processamento TP_{ij} ao tempo de processamento das operações já existentes em S, designando por C_{ij} o somatório do tempo de processamento total.
3. Escolher a próxima operação à direita no conjunto U, que é designada por $i+1$. Verificar se $C_{ij} + TP_{i+1,j} < d_{i+1,j}$. Se sim colocar a operação $i+1, j$ à direita de i,j no conjunto de S e ir ao Passo 5. Se não ir ao Passo 4.
4. Colocar $i+1, j$ à esquerda de i,j. Verificar se o somatório dos atrasos $L_{ij} = (C_{ij} - d_{ij})$ diminui considerando apenas as situações em que existe atraso $T_{ij} = \max(0, L_{ij})$. Se dimi-

nui, manter essa posição no conjunto S. Caso contrário, recolocar à direita de i no conjunto S.

5. Adicionar o tempo de processamento das tarefas já programadas a designar por C_{ij} e voltar ao Passo 3.

3.4 Interligação global a partir de um grafo

Após o sequenciamento da máquina mais estrangulada, é realizada a interligação global projectando a programação dessa máquina sobre um grafo contendo todas as tarefas.

Grafo G é um conjunto de pontos ou vértices $N_1, N_2, \ldots, N_n$ (designados pelo conjunto N) e um conjunto de linhas $a_1, a_2, \ldots, a_n$ (designados pelo conjunto A) ligando todos ou alguns desses pontos – O grafo é então completamente descrito através de $G(N, A)$ – CHRISTOFIDES (1975).

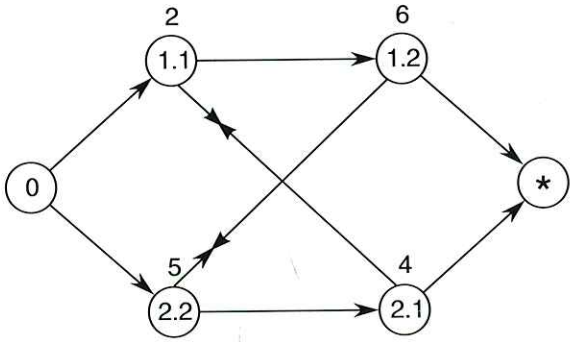
Se as linhas A têm uma direcção são designadas por arcos e o grafo é dirigido. Duas operações i e j executadas pela mesma máquina, não podem ser simultaneamente processadas, por isso são associadas com um par de arcos disjuntivos $[i,j] = \{(1,j),(j,i)\}$.

O problema passa então a ser representado por $G = (N, A, E)$, em que $G = (N, A)$ é um grafo conjuntivo e E um conjunto de disjunções. O conjunto N corresponde às operações representadas nos nodos, A representa as relações de precedência entre operações representadas nos arcos, e E representa o conjunto de operações a serem realizadas na mesma máquina.

Segundo ADAMS (1988), o conjunto de arcos disjuntivos E decompõe-se em cliques Ek.

A selecção de cada sequência Sk em cada máquina em Ek contém exactamente um membro de cada arco disjuntivo. Ou seja sequenciar uma máquina corresponde a fazer uma selecção acíclica em Ek. Uma selecção completa S consiste na união das selecções Sk em cada Ek. Ou seja, substituindo o conjunto de arcos disjuntivos E pelos arcos conjuntivos S, origina-se o grafo Ds. Cada selecção acíclica completa S, define uma família de programas.

No caso de duas tarefas cada uma das quais com duas operações e em duas máquinas, em que a operação (a, b) representa o processamento da tarefa a na máquina b, o grafo tem a seguinte configuração:



O tempo de processamento TP_{ij} da operação ij é posicionado sobre o nodo que representa a operação.

ADAMS (1988) estrutura a metodologia de utilização de grafos aplicados à programação, na mesma linha CARLIER (1982) definindo que, o tempo total de fluxo que é óptimo para S é igual o comprimento do caminho mais longo em Ds – ou seja, a minimização do caminho mais longo obtém-se optimizando para cada máquina a sequência das outras máquinas, o que, em termos globais corresponde a encontrar dentro do caminho mais longo a sua mínima duração. Utiliza para isso o algoritmo de CARLIER (1982), que é do tipo *Branch & Bound*, o qual, em cada nodo da árvore de *Branch & Bound* (associada ao grafo) utiliza uma heurística – maior volume de trabalho remanescente.

Na implementação que desenvolvo, o grafo só é construído explicitamente sempre que é escolhida a sequência de operações para cada máquina, sendo determinadas as datas de início ao mais cedo e ao mais tarde, com a particularidade de, as datas de conclusão ao mais tarde do último acontecimento serem diferentes tarefa a tarefa.

3.5 Síntese descritiva do algoritmo

A. Identificar a máquina mais estrangulada.
A.1.

No algoritmo de sequenciamento que utilizo, é identificada como máquina mais estrangulada a máquina com maior carga absoluta ou seja a máquina em que é máximo o somatório dos tempos de processamento.

$$\max \sum TP_i$$

A.2. Optimização do sequenciamento em cada máquina.

1. Agrupar as operações a processar na máquina k por ordem não decrescente de datas ao mais cedo em que a operação pode realizar-se, o que passa a constituir o conjunto U.
2. Escolher a operação com data ao mais cedo de U designada por operação ij e colocá-la no conjunto S. Acumular o tempo de processamento TP_{ij} ao tempo de processamento das operações já existentes em S, designando por C_{ij} o somatório do tempo de processamento total.
3. Escolher a próxima operação à direita no conjunto U, que é designada por $i+1, j$. Verificar se $C_{ij} + TP_{i+1,j} < d_{i+1,j}$. Se sim colocar a operação $i+1, j$ à direita de i, j no conjunto S e ir ao Passo 5. Se não ir ao Passo 4.
4. Colocar $i+1, j$ à esquerda de i, j. Verificar se o somatório dos atrasos $L_{ij} = (C_{ij} - d_{ij})$ diminui considerando apenas as situações em que existe atraso $T_{ij} = \max(0, L_{ij})$. Se diminui, manter essa posição no conjunto S. Caso contrário recolocar à direita de i no conjunto S.
5. Adicionar o tempo de processamento das tarefas já programadas a designar por C_{ij} e voltar ao Passo 3.

A.3. Determinar datas de início ao mais cedo e ao mais tarde.

O grafo é construído explicitamente sempre que é escolhida a sequência de operações para cada máquina, sendo determinadas as datas de início ao mais cedo e ao mais tarde de cada operação.

B.

B.1. Identificar o próximo estrangulamento.

B.2. Sequenciar segundo o algoritmo de sequenciamento máquina a máquina.

B.3. Estabelecer o grafo considerando a máquina mais estrangulada.

Calcular datas de início ao mais cedo e ao mais tarde tomando nas datas de início ao mais cedo o valor mais elevado entre o valor calculado e o valor que já se possui da iteração anterior.

B.4. Desenhar o gráfico de GANT respectivo.

B.5. Readaptar o posicionamento das restantes máquinas já programadas caso exista incompatibilidade, deslocando operações ao mínimo para diante ou para trás.

C. Repetir o procedimento B até à última máquina.

4. UM EXEMPLO

Pretende-se sequenciar em três máquinas um conjunto de quatro tarefas constituídas por diferentes operações, e que têm diferentes prazos de entrega. Os dados são de seguida apresentados:

Tarefas	Sequência por máquina e duração	Prazo de entrega d_i
1	A – 2 H B – 3 H C – 4 H	12
2	C – 4 H B – 1 H	7
3	C – 5 H A – 3 H B – 1 H	10
4	B – 1 H A – 2 H C – 3 H	10

A.1. Identificar a máquina mais estrangulada:

A = 2 H + 3 H + 2 H = 7 H

B = 3 H + 1 H + 1 H + 1 H = 6H

C = 4 H + 4 H + 5 H + 3 H = 16H

↑ Máquina mais estrangulada

Designam-se as operações que decorrem na máquina A por 1 na máquina B por 2 e na máquina C por 3.

A.2. Optimização do sequenciamento na máquina 3.

1. Estruturação da Informação

Tarefa	Operação	Duração	Data início mais cedo	Prazo de entrega	Operação	Duração	Data início mais cedo	Prazo de entrega	Operação	Duração	Data início mais cedo	Prazo de entrega
1	11	2	0	5	12	3	2	8	13	4	5	12
2	23	4	0	6	22	1	4	7				
3	33	5	0	6	31	3	5	9	32	1	8	10
4	42	1	0	5	41	2	1	7	43	3	3	10

2. Agrupar as operações que decorrem na máquina 3, por datas de início ao mais cedo.

	T_{23}	T_{33}	T_{43}	T_{13}
Data início mais cedo R_{ij}	0	0	3	5
Duração TP_{ij}	4	5	3	4
Prazo entrega d_{ij}	6	6	10	12

3. Reordenar tarefas segundo a Folga

$Tp_{23} + Tp_{33} > d_{33}$

$(4 + 5) > 6$

Experimentar colocar T_{33} à esquerda de T_{23} e comparar

	T_{23}	T_{33}
$c_{ij} = \sum TP_{ij}$	4	4+5
d_{ij}	6	6
$T_{ij} = \max L_{ij} = (c_{ij} - d_{ij})$	0	−3

$\Sigma -3$

$\rightarrow$

T_{33}	T_{23}
5	5+4
6	6
0	−3

$\Sigma -3$

Não houve melhoria. Manter posições T_{23} . T_{33}

$Tp_{23} + Tp_{33} + Tp_{43} > d_{43}$

$(4 + 5 + 3) > 10$

	T_{23}	T_{33}	T_{43}
c_{ij}	4	4+5	4+5+3
d_{ij}	6	6	10
T_{ij}	0	−3	−2

$\Sigma -5$

Experimentar colocar T_{43} à esquerda de T_{33}

$\rightarrow$

T_{23}	T_{43}	T_{33}
4	4+3	4+3+5
6	10	6
0	0	−6

$\Sigma -6$

Não houve melhoria. Manter posições T_{23} . T_{33} . T_{43}

$Tp_{23} + Tp_{33} + Tp_{43} + Tp_{13} > d_{13}$

$(4 + 5 + 3 + 4) > 12$

	T_{23}	T_{33}	T_{43}	T_{13}
c_{ij}	4	4+5	4+5+3	4+5+3+4
d_{ij}	6	6	10	12
T_{ij}	0	−3	−2	−4

Σ −9

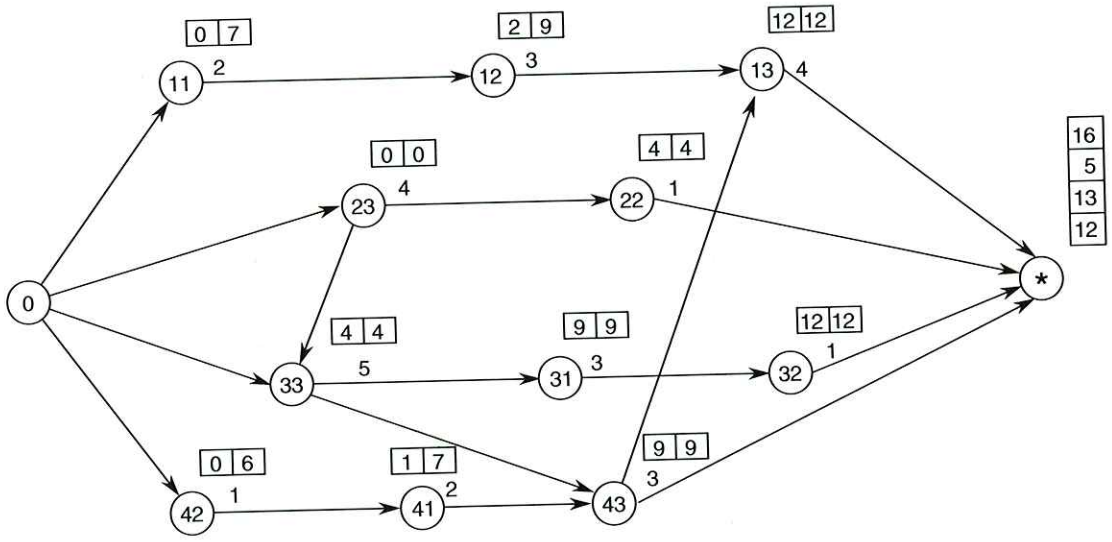
Experimentar colocar T_{13} à esquerda de T_{43} →

T_{23}	T_{33}	T_{13}	T_{43}
4	4+5	4+5+4	4+5+4+3
6	6	12	10
0	−3	−1	−6

Σ −10

Manter posições T_{23} . T_{33} . T_{43} . T_{13}

A.3 Determinar datas de início ao mais cedo e de conclusão ao mais tarde.



B.1 Identificar a próxima máquina mais estrangulada.

Máquina 1 = Σ 7 H

B.2. Optimização do sequenciamento da máquina 1.

1. Agrupar as operações que ocorrem da máquina 1, por datas de início ao mais cedo.

	T_{11}	T_{41}	T_{31}
Data início mais cedo R_{ij}	0	1	5
Duração TP_{ij}	2	2	3
Prazo entrega d_{ij}	9	9	12

$Tp_{11} + Tp_{41} < d_{41}$

$(2 + 2)$

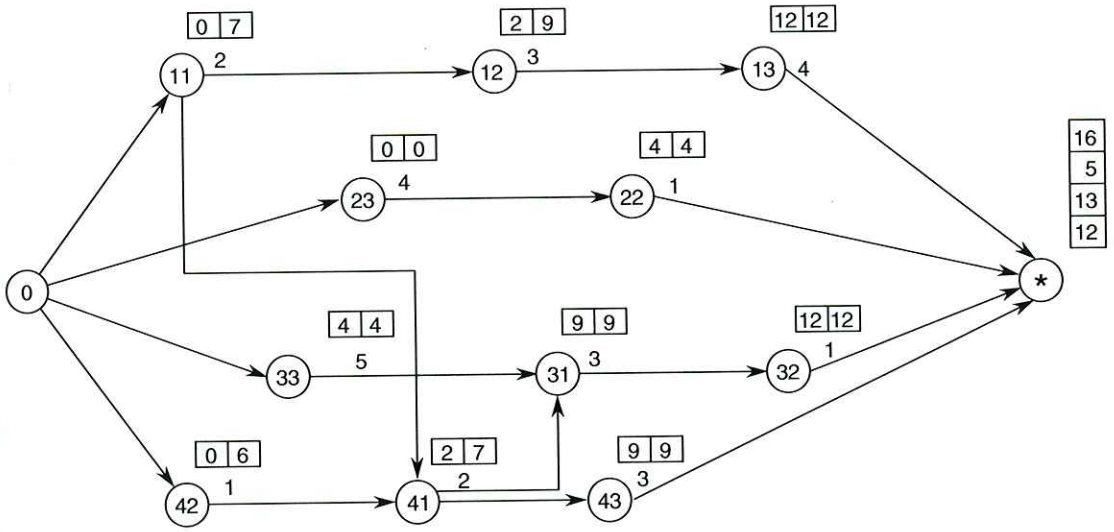
Manter ordem T_{11} . T_{41}

$Tp_{11} + Tp_{41} + Tp_{31} < d_{31}$

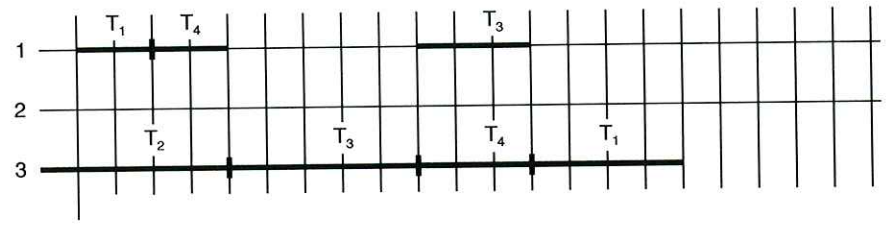
$2 + 2 + 2 < 12$

Manter ordem T_{11} . T_{41} . T_{31}

B.3 Traçar um novo grafo



B.4 Traçado do gráfico de Gantt e eventual readaptação de actividades anteriormente programadas.



Não é necessário reordenar actividades anteriormente programadas.

C.1 Identificar a próxima máquina mais estrangulada.

Máquina 2 = $\sum$ 6 H

C.2 Optimização do sequenciamento da máquina 2.

1. Agrupar as operações que decorrem na máquina 2, por datas de início ao mais cedo.

	T_{42}	T_{12}	T_{22}	T_{32}
Data início mais cedo R_{ij}	0	2	4	12
Duração TP_{ij}	1	3	1	1
Prazo entrega d_{ij}	7	12	5	13

$Tp_{42} + Tp_{12} < d_{12}$ Manter ordem T_{42} . T_{12}

$1 + 3 < 12$

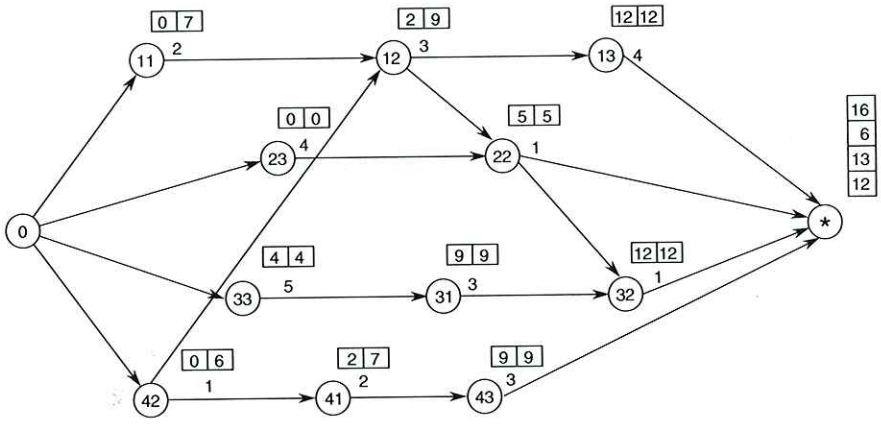
$Tp_{42} + Tp_{12} + Tp_{22} < d_{22}$ Manter ordem T_{42} . T_{12} . T_{22}

$1 + 3 + 1 < 5$

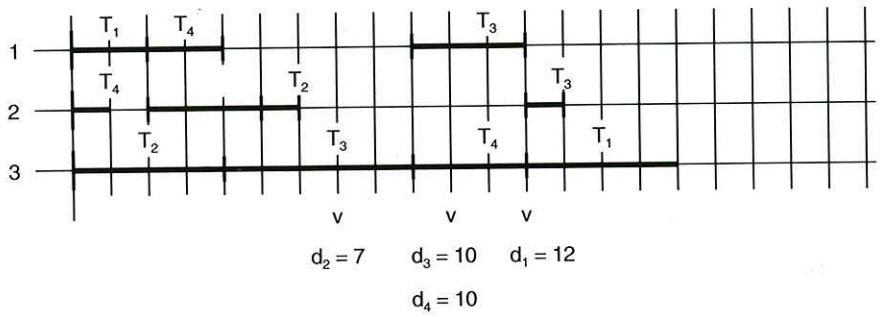
$Tp_{42} + Tp_{12} + Tp_{22} + Tp_{32} < d_{32}$ Manter ordem T_{42} . T_{12} . T_{22} . T_{32}

$1 + 3 + 1 + 1 < 13$

C.3 Traçar novo grafo.



C.4 Traçado do gráfico de Gantt e eventual readaptação de actividades anteriormente programadas.



Não é necessário reordenar actividades anteriormente programadas.

5. AVALIAÇÃO DE DESEMPENHO

O desempenho da programação avalia-se normalmente em função dos objectivos criados, no entanto há um conjunto de indicadores que são mais comummente utilizados:

1. O Atraso Total
2. O Atraso Médio por tarefa
3. O Número de Tarefas atrasadas
4. O Tempo Total de Fluxo de Todas as Tarefas
5. O Tempo de Fluxo da Tarefa Mais Longa (Data de Entrega Total)
6. O Tempo Total de Não Utilização das Máquinas
7. O Tempo Médio de Não Utilização das Máquinas
8. A Percentagem de Utilização de cada Máquina

1. Atraso Total

$T_1 = -4H$

$T_2 = 0$

$T_3 = -3\,H$

$T_4 = -2\,H$

$\Sigma = -9\,H$

2. Atraso Médio por tarefa

$$\frac{-9\,H}{4} = -2{,}25\,H$$

3. Número de Tarefas atrasadas

3

4. Tempo Total de Fluxo de Todas as Tarefas

$T_1 = 16\,H$

$T_2 = 6\,H$

$T_3 = 13\,H$

$T_4 = 12\,H$

$\Sigma = 47\,H$

5. Tempo de Fluxo de Tarefa Mais Longa

16 H

6. Tempo Total de Não Utilização das Máquinas

Máquina 1 = 5 H

Máquina 2 = 7 H

Máquina 3 = 0 H

$\Sigma = 12\,H$

7. Tempo Médio de Não Utilização Das Máquinas

$$\frac{12\,H}{3} = 4\,H / \text{Máquina}$$

8. Percentagem de Utilização das Máquinas

Máquina 1 = 7 / 12 = 58%

Máquina 2 = 6 /12 = 46%

Máquina 3 = 16 / 16 = 100%

6. COMPARAÇÃO COM OUTROS CRITÉRIOS

6.1. Critérios a utilizar para efeitos de comparação

Compare-se de seguida o algoritmo PP com as regras heurísticas mais correntes, que podem ser utilizadas numa base estática ou, numa base dinâmica:

A) DEM (Data de Entrega Mínima)
$[\min (d_i - t)]$

B) FOLGA
$[\min (d_i - t - TP_i (t))]$

Estes critérios cumprem bem quando o objectivo principal é cumprir datas de entrega, sem que tenham acontecido deslizamentos para além do tempo previsto e quando a carga não é elevada. Quando os tempos de espera entre operações são muito curtos esta regra já não cumpre bem; Igualmente em ambientes congestionados não cumpre bem.

A DEM é considerada ainda uma regra míope que podendo resultar num curto prazo, gera problemas no médio-longo prazo por aumentar o nível de congestionamento.

C) TPRC (Tempo de Processamento Mais Curto)
$[\min TP_i (t)]$

Cumpre bem quando o objectivo é mazimizar a eficiência, e quando há congestionamento, uma vez que a sua utilização reduz o tempo médio de fluxo, o tempo de espera, e portanto o volume de trabalhos em curso.

Apresenta desvios standard muito elevados e funciona menos bem para tarefas com um tempo de processamento muito longo que podem ficar indefinidamente na fila de espera. Minimiza normalmente o tempo médio de fluxo e cumpre bem prazos de entrega em ambientes congestionados.

D) RC (Rácio Crítico) =

$$= [\,\text{Min}\,\frac{d_i - t - TP_i (t)}{[TP_i (t)]}\,]$$

Indicador muito ligado à data de Entrega apresenta a vantagem de fornecer em simultâneo várias indicações.

Quando
RC > 1 Existe avanço
RC < 1 Existe atraso

Segundo MONTANEZO et al (1990) parece funcionar bem quando as datas de entrega estão ultrapassadas.

6.2 Avaliação de desempenho de 4 regras heurísticas aplicadas ao exemplo anterior

DEM

$T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_1$

1. Atraso Total = – 10 H
2. Atraso Médio por Tarefa = – 2,5 H
3. Número Tarefas Atrasadas = 3
4. Tempo Total de Fluxo = 46 H
5. Tempo Fluxo da Tarefa Mais Longa = 16 H
6. Tempo Total de Não Utilização = 12 H
7. Tempo Médio Não Utilização = 4H / Máquina
8. % de Utilização Máquinas

   Máquina 1 = 7 / 12 = 58%
   Máquina 2 = 6 / 13 = 43%
   Máquina 3 = 16 / 16 = 100%

Cumpre Pior que o algoritmo PP segundo 3 dos Critérios

Cumpre Igual segundo 4 dos Critérios

Cumpre Melhor segundo 1 dos Critérios

FOLGA

$T_3 \rightarrow T_2 \rightarrow T_1 \rightarrow T_4$

1. Atraso Total = – 10 H
2. Atraso Médio por Tarefa = – 2,5 H
3. Número Tarefas Atrasadas = 3
4. Tempo Total de Fluxo = 49 H
5. Tempo Fluxo da Tarefa Mais Longa = 16 H
6. Tempo Total de Não Utilização = 5 H
7. Tempo Médio Não Utilização = 1,7 H
8. % de Utilização Máquinas

   Máquina 1 = 7 / 8 = 87%
   Máquina 2 = 6 / 10 = 60%
   Máquina 3 = 16 / 16 = 100%

Cumpre Pior que o algoritmo PP segundo 3 Critérios

Cumpre Igual segundo 2 Critérios

Cumpre Melhor segundo 3 Critérios

TPRC

$T_2 \rightarrow T_4 \rightarrow T_1 \rightarrow T_3$

1. Atraso Total = [0 + 0 - 11 - 0] = -11 H
2. Atraso Médio por Tarefa = -2,75 H
3. Número Tarefas Atrasadas = 1
4. Tempo Total de Fluxo = 45 H
5. Tempo Fluxo da Tarefa Mais Longa = 21 H
6. Tempo Total de Não Utilização = 29 H
7. Tempo Médio Não Utilização = 9,7 H.
8. % de Utilização Máquinas

   Máq. 1 = 7 / 20 = 35%
   Máq. 2 = 6 / 21 = 28%
   Máq. 3 = 16 / 17 = 94,7%

Cumpre Pior que o algoritmo PP segundo 6 Critérios

Cumpre Igual segundo – Critérios

Cumpre Melhor segundo 2 Critérios

RC

$T_3 \rightarrow T_1 \rightarrow T_2 \rightarrow T_4$

1. Atraso Total = – 13 H
2. Atraso Médio por Tarefa = – 3,25 H
3. Número Tarefas Atrasadas = 2
4. Tempo Total de Fluxo =
   = [13 + 10 + 9 + 16] = 48 H
5. Tempo Fluxo da Tarefa Mais Longa = 16 H
6. Tempo Total de Não Utilização = 9 H
7. Tempo Médio Não Utilização = 2,25 H

8. % de Utilização Máquinas

Máquina 1 = 7/8 = 87%

Máquina 2 = 6/10 = 60%

Máquina 3 = 16/16 = 100%

Cumpre Pior que o algoritmo PP segundo 4 dos Critérios

Cumpre Igual segundo 1 dos Critérios

Cumpre Melhor segundo 3 dos Critérios

NOTAÇÃO

M – Conjunto de máquinas

A – Conjunto de pares de operações ligadas por uma relação de precedência

Ek – Conjunto de pares de operações a serem realizadas na máquina k

t – Momento de decisão

n – Número de tarefas na oficina

i – Número de ordem da tarefa

j – Número de ordem da máquina

ij – Operação da tarefa i na máquina j

P_{ij} – Tempo de processamento da tarefa i na máquina j vector de processamento $[\, p_{i1}, p_{i2}, ..., p_{in} \,]$

TO_i – Número total de operações da tarefa i

TP_i – Tempo total de processamento da tarefa i

$RO_i(t)$ – Número de operações remanescentes da tarefa i

$RP_i(t)$ – Tempo de processamento remanescente da tarefa i

R_{ij} – Momento em que a tarefa i fica disponível para a operação j

R_{il} – Momento em que a tarefa i chega à oficina

C_i – Momento em que a tarefa i é concluída

d_{ij} – Data de entrega da tarefa i na máquina j

L_i – Estado de avanço da tarefa $L_i = C_i - d_i$

T_i – Atraso da tarefa T_i = máx $(0, L_i)$

F_i – Tempo de fluxo = $C_i - R_i$

$S_i(t)$ – Folga = $d_i - t - RP_i(t)$

S_i – Folga estática = $d_i - R_i1 - TP_i$

Conjunto de tarefas na fila de espera no momento da operação j da tarefa i

W_{iq} – Tempo de espera da tarefa i para a operação q

$Z_i(t)$ – Prioridade da tarefa i no momento t

BIBLIOGRAFIA

ADAMS, J.; BALAS, E.; ZAWACK, D., «The shifting bottleneck procedure for job shop scheduling», *Management Science*, 34 (3):391-401, 1988.

ANDERSON, E.J.; NYIRENDA, J.C., «Two new rules to minimize tardiness in a job shop», *Int. J. Prod. Research*, 28 (12):2277-2292, 1990.

BAKER, K.R.; SCUDDER, G.D., «Sequencing with earliness and tardiness penalties: a review», *Operations Research*, 38(1):22-36, 1990.

CARLIER, J.; PINSON, E., «An algoritm for solving the job-shop problem», *Management Science*, 35(2):164-176, 1989.

CHENG, T.C.E.; GUPTA, M.C., «Survey of scheduling research involving due date determination decisions», *European Journal of Operational Research*, 38:156-166, 1989.

CHRISTOFIDES, N., *Graph Theory – Algoritmic Approach*, Academic Press, London, 1975.

DE, P.; GHOSH, J.B.; WELLS, C.E., «Optimal due-date assignment and sequencing» *European Journal of Operational Research*, 57:323-331, 1992.

GORDON, V.S., «A note on optimal assignment of slack due-dates in single machine scheduling», *European Journal of Operational Research*, 70:311-315, 1993.

PANWALKAR, S.S.; SMITH, M.L., KOULAMAS, C.P., «A heuristic for the single machine tardiness problem», *European Journal of Operational Research*, 70:304-310, 1993.

RAGHAVACHARI, M., «A V-shape property of optimal schedule of jobs about a common due date» *European Journal of Operational Research*, 23:401-402, 1986.

RAMAN, N.; TALBOT, F.B., «The job shop tardiness problem: A decomposition approach», *European Journal of Operational Research*, 69:187-199.