Instituto Superior de Ciências do Trabalho e da Empresa

Recursos Discretos
uma Representação Baseada na Capacidade

Relatório Interno ISCTE-DCTI-1998-005
Departamento de Ciências e Tecnologias de Informação do I.S.C.T.E.

Joaquim Reis
Departamento de Ciências e Tecnologias da Informação
E-mail: Joaquim.Reis@iscte.pt

Janeiro de 1998

Resumo

Propõe-se uma forma de representação computacional da capacidade de recursos renováveis agregados (grupos de máquinas, veículos, trabalhadores) no tempo, para suporte ao escalonamento de recursos partilhados com capacidade finita, bem como algoritmos de afectação e desafectação de capacidade.

Índice de Matérias

1 Introdução

O presente texto propõe uma representação para a capacidade de recursos discretos através de perfis de capacidade, representando tempo e capacidade de uma forma explícita.

Um perfil de capacidade representa a evolução de uma variável de capacidade ao longo do tempo. Uma variável de capacidade pode representar a capacidade total do recurso ocupada, ou a capacidade do recurso utilizada (ocupada) num determinado contexto (por exemplo, utilizada num determinado grupo de tarefas, ou com um determinado produto, etc.), a capacidade disponível do recurso, a capacidade máxima do recurso, etc., ao longo do tempo. A representação computacional dos perfis de capacidade bem como operações computacionais de manipulação de perfis e de combinação de perfis é apropriada para servir como um componente num sistema de planeamento/programação de tarefas (*scheduling*). Basicamente, num sistema desta natureza, os perfis de capacidade permitirão a representação de restrições de capacidade dos recursos.

Um *recurso discreto* é aquele que é discretamente divisível, podendo ser afectado a tarefas numa das quantidades discretas de um conjunto finito de afectações possíveis (a afectação de uma única unidade de recurso por tarefa é um caso particular) [Blazewicz 1994]. Um *recurso renovável* é aquele que está restrito apenas no uso total em cada momento [Blazewicz 1994]. Quando um recurso desta categoria é afectado a uma tarefa, a sua capacidade disponível é reduzida numa quantidade igual à necessária à execução da tarefa durante o intervalo de tempo de execução desta; assume-se que essa quantidade é constante no intervalo. A partir do tempo de fim da tarefa (inclusive) aquela quantidade de capacidade fica novamente disponível. Exemplos de recursos discretos renováveis são tornos, fresadoras, robots, fornos, máquinas de soldar, gruas, veículos de transporte, operários, professores, etc.. Um *recurso não renovável*, ou *recurso consumível/produzível*, é aquele que está restrito apenas no consumo total até um certo momento. Quando um recurso não renovável é afectado a uma tarefa, a sua capacidade disponível fica reduzida numa quantidade igual à necessária à execução da tarefa a partir do tempo de início da tarefa e até ao fim do horizonte temporal de representação do recurso (mesmo que a sua capacidade seja alterada/aumentada num qualquer instante do horizonte). Portanto, ao contrário do que acontece com os recursos renováveis, mesmo a partir do tempo de fim da tarefa (inclusive) aquela quantidade de capacidade continua indisponível. Exemplos de recursos discretos não renováveis são peças (para montagem), ferramentas, tijolos, vigas, espaço disponível para armazenamento, etc..

Suponha-se que existem *eventos de capacidade* e que a cada evento está associado um *instante de tempo*, ou *instante*, e um *valor de capacidade* positivo ou negativo, significando incremento ou decremento de uma quantidade de capacidade (por exemplo, uma quantidade de componentes, peças, espaço disponível num armazém, etc.). Suponha-se também que existem *intervalos de capacidade* e que a cada intervalo está associado um par de instantes de tempo extremos do intervalo, o *instante inicial* e o *instante final*, e um *valor de capacidade*, positivo ou negativo, significando incremento ou decremento temporário de uma quantidade de capacidade (por exemplo, o número de máquinas, veículos, trabalhadores, etc., disponíveis ou não disponíveis). Neste caso, o incremento ou decremento da quantidade de capacidade é temporário no sentido em que

ocorre apenas durante o intervalo de tempo delimitado pelos instantes de tempo inicial (incluído) e final do intervalo (não incluído). Teremos assim duas formas de alteração do valor de uma variável de capacidade:

1- **Por meio de eventos de capacidade** - Com um evento de incremento (decremento) de uma determinada quantidade de capacidade, a capacidade é incrementada (decrementada) dessa quantidade para o instante associado ao evento e todos os instantes seguintes.

2- **Por meio de intervalos de capacidade** - Com um intervalo de incremento (decremento) de uma determinada quantidade de capacidade, a capacidade é temporàriamente incrementada (decrementada) dessa quantidade durante os instantes abrangidos pelo intervalo, isto é, no instante inicial do intervalo e todos os instantes seguintes até ao instante final exclusive.

No primeiro caso tem-se uma alteração, numa quantidade fixa, do valor de capacidade a partir de um instante. Tipicamente variáveis de capacidade de recursos não renováveis (consumíveis/produzíveis) são manipuladas desta forma. Por exemplo, se se está representando o número de unidades de determinado produto armazenadas em stock, a entrada, ou a saída, de determinada quantidade de produto serão representadas, respectivamente, por eventos de incremento, ou decremento, de capacidade nessa quantidade.

No segundo caso tem-se uma alteração, numa quantidade fixa, do valor de capacidade durante um intervalo de tempo. Tipicamente variáveis de capacidade de recursos renováveis são manipuladas desta forma. Por exemplo, se a capacidade é traduzida pelo número total de operários especializados num determinado tipo de tarefa numa fábrica, a afectação de um grupo de operários a uma tarefa desse tipo a decorrer entre dois instantes de tempo determinados é representada pelo correspondente intervalo de capacidade com um valor igual ao número de operários desse grupo, positivo (incremento) se se estiver representando a capacidade ocupada, ou negativo (decremento) se se estiver representando a capacidade disponível.

Embora estas duas formas alternativas pareçam mutuamente exclusivas, podem, no entanto, conceber-se situações em que elas possam coexistir. Por exemplo, com recursos renováveis um evento de incremento, ou de decremento, de capacidade pode representar respectivamente, uma ampliação, ou uma redução, da capacidade total a partir de um dado momento; com recursos não renováveis um intervalo de incremento de capacidade pode representar uma certa quantidade de unidades de um produto, de componentes, etc., que foram colocadas em stock e lá permaneceram, durante um certo intervalo de tempo, até serem consumidas, vendidas, etc.; se se está representando o espaço de armazenamento disponível a mesma situação corresponde a um intervalo de decremento. Por outro lado, para um recurso não renovável que é parte de uma rede de recursos - por exemplo, um armazém de um produto intermédio de uma cadeia de fornecimento - a primeira forma corresponde a uma gestão completamente local, ou descentralizada: eventos de decremento ocorrerão tipicamente até que a capacidade desça a um nível préfixado - o ponto de encomenda - após o que deverá ocorrer um evento de incremento para repor a capacidade; a segunda forma corresponde a uma gestão centralizada da rede de recursos em que a cada incremento de determinada quantidade de capacidade num dado instante corresponde um decremento da mesma quantidade num instante mais tarde. Sendo assim, assume-se sempre que um recurso pode ser manipulado das duas maneiras.

A idéia da representação do perfil de capacidade foi inspirada nas "histórias" introduzidas em [Williams 1990] para a representação do "comportamento" de variáveis de estado de um sistema ao longo do tempo. Também uma forma semelhante de representação é usada na representação de restrições de recurso e restrições de estado no sistema de scheduling GERRY [Zweben 1994].

2 Definições e Convenções de Representação

As definições e convenções aqui usadas são descritas a seguir.

Símbolos de operações ou relações - Os seguintes símbolos denotam operações ou relações entre objectos:

símbolo	descrição
+	exprime a operação de adição de valores numéricos.
-	exprime a operação de subtracção de valores numéricos.
=	exprime a relação de igualdade entre valores numéricos ou conjuntos.
≠	exprime a relação de desigualdade entre valores numéricos ou conjuntos.
<	exprime a relação menor que entre valores numéricos.
>	exprime a relação maior que entre valores numéricos.
≤	exprime a relação menor que, ou igual a, entre valores numéricos.
≥	exprime a relação maior que, ou igual a, entre valores numéricos.
∈	exprime a relação de pertença entre um elemento e um conjunto.
∉	exprime a negação da relação de pertença entre um elemento e um conjunto.
∪	exprime a operação de reunião de conjuntos.
\	exprime a operação de subtracção de conjuntos.
∧	exprime a conjunção lógica
∨	exprime a disjunção lógica
⇒	exprime a implicação lógica
∀	quantificador universal
∃	quantificador existencial

Objectos simples - Correspondem a variáveis de tempo ou de capacidade e têm identificadores típicos, ou correspondem a identificadores que fazem parte de objectos compostos.

Objectos compostos - Podem ser tuplos ou conjuntos:

- **Tuplos** - Tuplos são objectos compostos. O valor de um tuplo é o da sequência dos seus elementos, separados por "`,`", delimitada por "`<`" e "`>`". Podem ter ou não um componente específico (um elemento de tuplo) que é um identificador.
- **Conjuntos** - Conjuntos são objectos compostos. O valor de um conjunto é o da sequência dos seus elementos (tipicamente trata-se de conjuntos ordenados) separados por "`,`", delimitada por "`{`" e "`}`".

Componentes de objectos compostos podem ser expressos, para maior comodidade e simplicidade de expressão na descrição das propriedades desses objectos, por meio de operações de acesso aos componentes. Estas são funções que, aceitando como argumento o objecto composto, produzirão como valor o componente respectivo. As operações de acesso têm identificadores escritos em maiúsculas que sugerem o componente acedido. Por exemplo, para intervalos de tempo existem as operações `TI` e `TF`: se `h` é um intervalo de tempo, `TI(h)` e `TF(h)` são respectivamente, os instantes de tempo inicial e final de `h` (isto é, se o valor de `h` for `<ti,tf>`, são `ti` e `tf`). Podem existir operações diferentes com o mesmo identificador, sendo o tipo de operação deuzido através do tipo de argumento. Por exemplo, se `p` é um perfil de capacidade `TI(p)` e `TF(p)` são os instantes de tempo extremos do horizonte temporal de `p`.

Objectos simples:

Instante de tempo

Variável: `t`, `ti`, `tf`

Valor: inteiro

Valor, valor de capacidade ou capacidade

Variável: `c`

Valor: inteiro

Identificador

Variável: `id`

Valor: qualquer string de caracteres

Objectos compostos

Intervalo de tempo

Variável: `h`

Valor: `<ti,tf>`

Um intervalo de tempo `h=<ti,tf>` é um par de valores `ti` e `tf`, em que `ti` é o instante inicial e `tf` o instante final.

Acesso aos atributos de um intervalo de tempo:

`TI(h)=ti`

`TF(h)=tf`

`DUR(h)=tf-ti` redundante

Para qualquer intervalo de tempo h verifica-se que o instante final ocorre após o instante inicial, isto é:

`TF(h)>TI(h)`

Intervalo de capacidade

Variável: `i`

Valor: `<id,c,h>`

Um intervalo de capacidade `i=<id,c,h>` é um triplo de valores `id`, `c` e `h`, em que `id` é o identificador do intervalo de capacidade, `c` o valor de capacidade e `h` o intervalo de tempo associados ao intervalo de capacidade.

Acesso ao atributos de um intervalo de capacidade:

`ID(i)=id`

`VAL(i)=c`

`INT(i)=h`

`TI(i)=TI(INT(i))` redundante

`TF(i)=TF(INT(i))` redundante

Evento de capacidade

Variável: `e`

Valor: `<id,c,t>`

Formalmente, um evento de capacidade `e=<id,c,t>` é um triplo de valores `id`, `c` e `t`, em que `id` é o identificador do evento de capacidade, `c` o valor de capacidade e `t` o instante de tempo associados ao evento de capacidade.

Embora um evento de capacidade seja formalmente um triplo `<id,c,t>` eventos de capacidade serão internamente representados por intervalos de capacidade cujo instante final é igual ao instante final do horizonte temporal do perfil do recurso onde são carregados; o instante inicial é `t`. Isto faz sentido na medida em que o efeito do evento dura desde o instante `t` até ao fim deste horizonte, para além de que uniformiza o tratamento de intervalos e eventos de capacidade.

Portanto, um evento de capacidade `e=<id,c,t>` num perfil de capacidade `p`, é representado por um intervalo de capacidade `e=<id,c,h>`, em que `TI(e)=t` e `TF(e)=TF(p)`.

Conjunto de intervalos de capacidade

Variável: `I`

Valor: $\{i_1,...,i_n\}$

Um conjunto de intervalos de capacidade $I=\{i_1,...,i_n\}$ é um conjunto (eventualmente vazio) ordenado de intervalos de capacidade $i_1,...,i_n$. A ordem dos elementos de `I`, quando este não está vazio, é a da inclusão: i_1, foi incluído antes de i_2, i_2 foi incluído antes de i_3, etc., e i_{n-1} antes de i_n.

Segmento de capacidade

Variável: `s`

Valor: `<c,h,I>`

Um segmento de capacidade $s=<c,h,I>$ é um triplo de valores c, h e I, em que c é o valor de capacidade do segmento, h o intervalo de tempo do mesmo e I é o conjunto de intervalos de capacidade responsáveis pelo valor de capacidade c.

Acesso ao atributos de um segmento de capacidade:

$VAL(s)=c$

$INT(s)=h$

$CI(s)=I$

$TI(s)=TI(INT(s))$ redundante

$TF(s)=TF(INT(s))$ redundante

Para qualquer segmento de capacidade s o que se segue é verdadeiro:

Os intervalos de tempo do segmento e do intervalo de capacidade sobrepõem-se, isto é:

$$\forall(I)\ i\in CI(s) \Rightarrow (TI(i)\geq TI(s) \wedge TI(i)<TF(s)) \vee (TF(i)\leq TF(s) \wedge TF(i)>TI(s))$$

Em qualquer instante t dentro do intervalo de tempo do segmento s, a soma dos valores de capacidade de todos os intervalos de capacidade do segmento que contêm t é igual ao valor de capacidade do segmento:

$$\forall(k,t)\ TI(i_k)\leq t \wedge TF(i_k)>t \wedge TI(s)\leq t \wedge TF(s)>t \Rightarrow VAL(s)=\Sigma_{k=1,n}VAL(i_k)$$

Troço de (perfil de) capacidade

Variável: S

Valor: $\{s_1,...,s_m\}$

Um troço de capacidade $S=\{s_1,...,s_m\}$ é um conjunto ordenado de segmentos de capacidade $s_1,...,s_m$, não vazio (contém um, ou mais, segmentos de capacidade), temporalmente contíguos e com valores de capacidade diferentes, isto é:

O segmentos de S são temporalmente contíguos:

$$\forall(j)\ j\geq 1 \wedge j<m \Rightarrow TF(s_j)=TI(s_{j+1})$$

Para qualquer conjunto S com mais de 1 segmento, cada segmento tem um valor de capacidade diferente do seguinte (excepto se é o último):

$$\forall(j)\ j\geq 1 \wedge j<m \Rightarrow VAL(s_j)\neq VAL(_{sj+1})$$

Perfil de capacidade

Variável: p

Valor: $<id,h,S,I>$

Um perfil de capacidade $p=<id,h,S,I>$ é um 4-tuplo de valores id, h, S e I, em que id é o identificador do perfil de capacidade, h é o intervalo de tempo que é o horizonte temporal do perfil de capacidade, S é um troço de capacidade do perfil que abrange temporalmente todo o horizonte temporal e I é o conjunto de todos os intervalos de capacidade responsáveis pelos valores de capacidade de todos os segmentos de capacidade do troço de capacidade S.

Acesso ao atributos de um perfil de capacidade:

$ID(p)=id$

$INT(p)=h$

$CS(p)=S$

$CI(p)=I$

$TI(p)=TI(INT(p))$ redundante

$TF(p)=TF(INT(p))$ redundante

Para qualquer perfil de capacidade p, sendo:

$CS(p)=\{s_1,...,s_m\}$

verifica-se o seguinte:

$CI(p)$ é o conjunto de todos os intervalos de capacidade responsáveis pelos valores de capacidade de todos os segmentos de capacidade do troço de capacidade S:

$CI(p)=\cup_{j=1,m}CI(s_j)$

Os intervalos de tempo de todos os intervalos de capacidade do perfil estão temporalmente contidos no horizonte temporal do perfil:

$$\forall(I)\ i\in CI(p) \quad \Rightarrow \quad TI(i)\geq TI(p) \quad \wedge \quad TF(i)\leq TF(p)$$

O primeiro intervalo de capacidade do perfil inicia-se no instante de tempo inicial do horizonte temporal do perfil:

$TI(s_1)=TI(p)$

O último intervalo de capacidade do perfil termina no instante de tempo final do horizonte temporal do perfil:

$TF(s_m)=TF(p)$

3 Operações sobre um Perfil de Capacidade

Descrevem-se a seguir operações sobre um perfil de capacidade. Incluem-se a operação de criação de um perfil de capacidade, operações de acesso aos componentes de um perfil de capacidade (operações selectoras) e as operações modificadoras de carga e de anulação de intervalos de capacidade num perfil de capacidade. As operações de carga e anulação serão descritas com maior detalhe, nomeadamente através de algoritmos propostos, dada a sua complexidade.

3.1 Operação de criação

`CriarPerfil(id,ti,tf)` - Operação que gera e produz um perfil de capacidade, identificado por `id`, com o horizonte temporal igual ao intervalo `<ti,tf>`. com um conjunto de segmentos contendo um único segmento e um conjunto de intervalos de capacidade vazio; o segmento único abrange todo o horizonte temporal do perfil de capacidade, tem um valor de capacidade 0 e um conjunto de intervalos de capacidade vazio. `ti` e `tf` devem ser tais que `tf>ti`, caso contrário nenhum perfil de capacidade é gerado.

3.2 Operações de acesso

`Ident(p)` - Produz o identificador do perfil `p` (idêntica à operação de acesso `ID(p)`).

`Tinicio(p)` - Produz o instante de tempo de inicial do horizonte temporal do perfil `p` (idêntica à operação de acesso `TI(p)`).

`Tfim(p)` - Produz o instante de tempo de final do horizonte temporal do perfil `p` (idêntica à operação de acesso `TF(p)`).

`Ivalor(p,id)` - Produz o intervalo de capacidade, identificado por `id` carregado no perfil `p`, um objecto da forma `<id,c,h>`.

`Segmentos(p,ti,tf)` - Operação que produz o troço mais curto do perfil de capacidade `p` que abrange o intervalo `<ti,tf>`, um troço de capacidade que é um subconjunto (subsequência) do troço de capacidade do perfil `p`, isto é, um objecto da forma $\{s_x, \ldots, s_z\}$, em que $x \geq 1$ e $z \leq m$ (sendo m o número de segmentos do troço de capacidade de `p`), tal que $TI(s_x)$ é o maior dos valores $TI(s_j)$ tal que $TI(s_j) \leq ti$ e $TF(s_z)$ é o menor dos valores $TF(s_j)$ tal que $TF(s_j) \geq tf$ (para $1 \leq j \leq m$).

3.3 Operações de modificação

`Carregar(p,id,c,ti,tf)` - Operação que constrói e carrega no perfil de capacidade `p` um intervalo de capacidade, identificado por `id`, com o valor de capacidade `c`, o instante inicial `ti` e o instante final `tf`. Se `id` identificar um intervalo de capacidade já carregado em `p`, ou `ti<TI(p)`, ou `tf>TF(p)`, o perfil permanecerá inalterado. O último argumento pode ser omitido (`Carregar(p,id,c,ti,)`); por omissão será assumido o valor do instante extremo do horizonte temporal de `p` (isto é `TF(p)`).

`Anular(p,id)` - Operação que retira do perfil de capacidade `p` o intervalo de capacidade `id`. Se `id` não identifica um intervalo de capacidade previamente carregado em `p` o perfil não é afectado.

A descrição que a seguir se faz destas operações tem como objectivo clarificar onde e como um perfil sofre alterações quando são usadas as operações Carregar ou Anular, fornecendo um maior detalhe apropriado à realização computacional dos respectivos algoritmos.

3.3.1 A operação de carga

A operação `Carregar(p,id,c,ti,tf)` constrói um intervalo de capacidade `i`, tal que:

`i=<id,c,<ti,tf>>`

e carrega, ou coloca, este intervalo de capacidade no perfil de capacidade `p`. Se `tf` é omitido, por exemplo `Carregar(p,id,c,t,)` o intervalo de capacidade construído é:

`i=<id,c,<t,TF(p)>>`

E é este intervalo de capacidade que é então carregado.

Pré-condições para a operação `Carregar` ter sucesso são:

1- Não existe nenhum intervalo identificado por `id` carregado em `p`, isto é:

$\forall(q,i_q)\ i_q \in CI(p) \Rightarrow id \neq ID(i_q)$

2- O intervalo de tempo de `i` está contido dentro do horizonte temporal de `p`, isto é:

$ti \geq TI(p) \wedge$

$tf \leq TF(p)$

Se estas pré-condições estão satisfeitas `i` será carregado em `p`.

3.3.2 A operação de anulação

A operação `Anular(p,id)` procura no perfil `p` um intervalo de capacidade `i`, identificado por `id` e, caso ele exista, retira-o do perfil `p`.

A pré-condição para a operação `Anular` ter sucesso é, portanto:

Existe um intervalo identificado por `id` carregado em `p`, isto é:

$\exists(q,i_q)\ \ i_q \in CI(p) \wedge$
$id=ID(i_q)$

Se esta pré-condição for satisfeita i_q será retirado de `p`.

3.3.3 Carga e Anulação de Intervalos de Capacidade: Casos Possíveis

A carga ou anulação de intervalos de capacidade num perfil de capacidade pode influir no número de segmentos de perfil, no valor de capacidade a eles associado e nos instantes de tempo iniciais e finais dos segmentos.

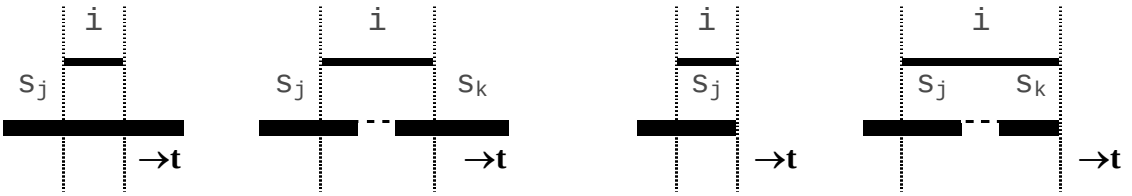
Os segmentos do perfil afectados com a carga ou anulação de um segmento de capacidade `i` dependem das relações, de igualdade e de ordem (`=`, `<` ou `>`), entre os valores dos instantes de tempo extremos do intervalo de capacidade `i` e os dos segmentos do perfil `p`. É particularmente importante a configuração do perfil nas zonas temporais na vizinhança dos extremos do intervalo `i`.

Após a carga ou anulação do intervalo de capacidade i no perfil de capacidade p o resultado é um perfil p′, que corresponde a um novo estado de p. Para uma maior eficiência das operações Carregar e Anular sugere-se um algoritmo que obtém um perfil novo, p′ por modificação do perfil original p. Assumir-se-á que isto é levado a cabo no sentido positivo do tempo, começando por modificar os segmentos na vizinhança temporal do instante inicial de i e concluindo pela modificação dos segmentos na vizinhança temporal do instante final de i. Há basicamente dois casos:

Caso 1 - O intervalo de tempo de i inicia-se num instante contido dentro do intervalo de tempo de um segmento do perfil, diferente dos instantes inicial e final deste, isto é:

$$\begin{aligned} \exists(j,s_j) \quad & j \geq 1 \quad \wedge \\ & j \leq m \quad \wedge \\ & s_j \in CS(p) \quad \wedge \\ & TI(i) > TI(s_j) \quad \wedge \\ & TI(i) < TF(s_j) \end{aligned}$$

Caem dentro deste caso as situações distintas representadas na Figura 1.



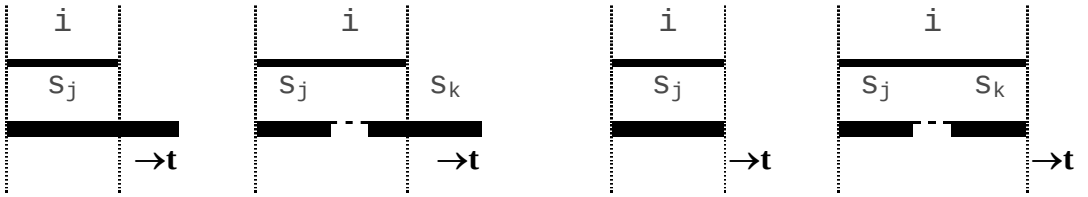
Caso 1-1 Caso 1-2 Caso 1-3 Caso 1-4

Figura 1- Casos de situações possíveis antes da carga ou anulação do intervalo de capacidade i, quando o intervalo de tempo de i começa dentro do intervalo de tempo de um segmento do perfil.

Caso 2 - O intervalo de tempo de i inicia-se no instante inicial de um segmento do perfil, isto é:

$$\begin{aligned} \exists(j,s_j) \quad & j \geq 1 \quad \wedge \\ & j \leq m \quad \wedge \\ & s_j \in CS(p) \quad \wedge \\ & TI(i) = TI(s_j) \end{aligned}$$

Caem dentro deste caso as situações distintas representadas na Figura 2.



Caso 2-1 Caso 2-2 Caso 2-3 Caso 2-4

Figura 2- Casos de situações possíveis antes da carga ou anulação do intervalo de capacidade i, quando o intervalo de tempo de i começa quando começa o intervalo de tempo de um segmento do perfil.

Conforme o posicionamento temporal do instante final de `i` assim estes dois se desdobram cada um em quatro casos específicos de tratamento diverso, denominados Caso 1-1 a Caso 1-4, representados na Figura 1, e Caso 2-1 a Caso 2-4, representados na Figura 2. Cada um destes casos, bem como o respectivo algoritmo apropriado, são descritos a seguir. A cada um destes casos é associado um algoritmo de carga e outro de anulação.

3.4 Algoritmos para Carga e Anulação

Na descrição que foi feita dos algoritmos que processam cada um dos casos específicos (do Caso 1-1 ao Caso 1-4 e do Caso 2-1 ao Caso 2-4) podem detectar-se padrões idênticos de processamento. Seria interessante, por razões de simplicidade, facilidade de compreensão e também de economia de linhas de código, dispor de um algoritmo sintético que fizesse uso de procedimentos, ou subalgoritmos, que realizam esses tratamentos idênticos. Nesta secção propõem-se tais procedimentos e um algoritmo geral, que trata todos os casos, construído com base nesses procedimentos.

A ideia base continua a ser a de o algoritmo processar a sequência de segmentos de capacidade do perfil de capacidade, `p`, no sentido crescente do tempo. Assim, os procedimentos a serem desenvolvidos operarão no perfil, modificando apenas a subsequência de segmentos que deve ser afectada, desde os segmentos de `p` na vizinhança temporal do instante inicial do intervalo de tempo do intervalo de capacidade a carregar/anular `TI(i)`, até aos segmentos de `p` na vizinhança temporal do instante final do mesmo intervalo de tempo, `TF(i)`. Em cada passo do algoritmo existirá uma variável que contém um segmento do perfil de capacidade dito *segmento actual*; cada procedimento receberá como parâmetro, ao ser invocado, o segmento actual (a variável) e é no segmento actual e a partir desse segmento que cada procedimento começa a operar; em geral os procedimentos modificam o valor da variável do segmento actual: o processamento de um procedimento conclui-se contendo esta variável o segmento onde o próximo procedimento a aplicar, se o houver, deve começar a operar no perfil.

Um esboço da sequência de passos do algoritmo geral é descrito a seguir:

1- São identificados os segmentos na vizinhança temporal do instante de tempo inicial do intervalo de tempo de `i`; o algoritmo processará esses segmentos de forma diferente de acordo com os dois casos diferentes mostrados na Figura 3 e designados por Caso-de-início 1 e Caso-de-início 2. Estes correspondem respectivamente ao facto de o intervalo de tempo de `i` se iniciar a meio (Caso-de-início 1) ou no início de um segmento do perfil (Caso-de-início 2) e terão um processamento diverso, através de dois procedimentos diferentes.

   Basicamente, estes procedimentos afectam a zona temporal do perfil correspondente ao primeiro segmento — s_j, na Figura 3 — da subsequência de segmentos a modificar, o que acontece sempre em ambos os casos, ou afectam o primeiro e também o segundo segmento — s_{j-1} (se existe, isto é, se `i>1`) e s_j na Figura 3 —, o que poderá acontecer no Caso-de-início 2. Esta afectação pode ser a subdivisão de um segmento em dois no Caso-de-início 1, a fusão de dois segmentos em um único no Caso-de-início 2 ou a simples alteração do valor de capacidade de um segmento do perfil no Caso-de-início 2. O conjunto `CI(s)` dos segmentos `s` afectados/gerados é também actualizado.

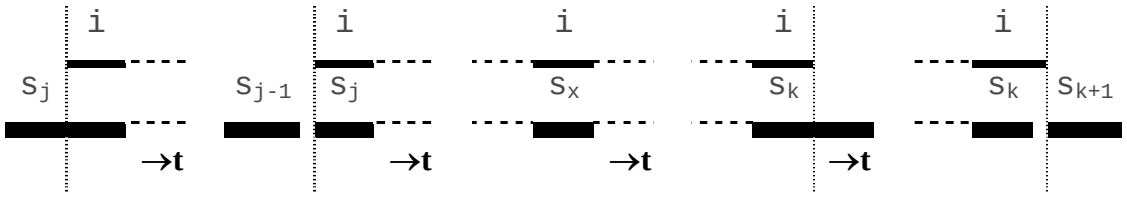
2- São identificados todos os segmentos cujo intervalo de tempo ocorre durante intervalo de tempo de `i` (não processados pelo passo 1-), se os houver, e ainda o segmento em cujo intervalo de tempo o instante final do intervalo de tempo de `i` está contido; o algoritmo processará estes segmentos de acordo com um outro procedimento. Este caso é na Figura 3 designado por Caso-do-meio e os segmentos nele afectados são genericamente designados por s_x.

   Este procedimento afecta a zona temporal do perfil correspondente ao troço intermédio da subsequência de segmentos a modificar (não processados pelo passo 1-) e também segmentos a serem processados posteriormente pelo passo 3-, com modificação do valor de capacidade dos segmentos s_x (incrementando-o, ou decrementando-o, de `VAL(i)`, conforme se trate de uma carga, ou de uma anulação) e actualização do conjunto $CI(s_x)$ (por inclusão, ou exclusão de `i`, conforme se trate de uma carga, ou de uma anulação).

3- É executado um processamento sobre os segmentos na vizinhança temporal do instante de tempo final do intervalo de capacidade a carregar/anular, `i`, de acordo com os dois casos diferentes mostrados na Figura 3 e designados por Caso-de-fim 1 e Caso-de-fim 2. Estes correspondem respectivamente ao facto de o intervalo de tempo de `i` terminar a meio (Caso-de-fim 1) ou no fim de um segmento do perfil (Caso-de-fim 2) e terão um padrão de processamento diverso, originando dois procedimentos diferentes. Estes dois procedimentos são, em parte, simétricos aos procedimentos associados aos dois casos no passo 1-.

   Basicamente, estes procedimentos afectam a zona temporal do perfil correspondente ao último segmento — s_k, na Figura 3 — da subsequência de segmentos a modificar, o que acontece no Caso-de-fim 1, ou podem ou não afectar o penúltimo e o último (ambos) segmento — s_k e s_{k+1} (se existe, isto é, se $k<m$) na Figura 3 —, no Caso-de-fim 2. Esta afectação pode ser a subdivisão de um segmento em dois seguida de exclusão de `i` do conjunto $CI(s_{k+1}')$ do segundo segmento (novo) apenas se se trata de um caso de carga, no Caso-de-fim 1, a fusão de dois segmentos em um único segmento no Caso-de-fim 2, ou simplesmente nenhuma modificação no Caso-de-fim 2. Note-se que, quando se inicia o passo 3-, o segmento s_k foi já alterado no passo anterior (com incrementação, ou decrementação, do valor de capacidade de s_k em `VAL(i)`, conforme se trate de uma carga, ou de uma anulação, respectivamente, e com inclusão no, ou exclusão do, conjunto $CI(s_k)$ de `i`, conforme se trate de uma carga, ou de uma anulação, respectivamente).

4- Por fim o intervalo de capacidade `i` é inlcuído no, ou excluído do, conjunto `CI(p)`, conforme se trate de uma carga, ou de uma anulação, respectivamente.



Caso-de-início 1 Caso-de-início 2 Caso-do-meio Caso-de-fim 1 Caso-de-fim 2

Figura 3- Casos de situações possíveis antes da carga ou anulação do intervalo de capacidade `i`.

A descrição dos algoritmos que se segue é feita no estilo da linguagem Pascal. Há um algoritmo para a operação `Carregar` e outro para a operação `Anular`. O algoritmo de `Carregar` utiliza procedimentos `Inicio-c1`, `Inicio-c2`, `Meio-c`, `Fim-c1` e `Fim-ca2`; estes são os que processam, respectivamente, os casos Caso-de-início 1, Caso-de-início 2, Caso-do-meio, Caso-de-fim 1 e Caso-de-fim 2, atrás referidos, para o caso da carga de intervalo de capacidade. O algoritmo de `Anular` utiliza procedimentos `Inicio-a1`, `Inicio-a2`, `Meio-a`, `Fim-a1` e `Fim-ca2`; estes são os que processam, respectivamente, os casos Caso-de-início 1, Caso-de-início 2, Caso-do-meio, Caso-de-fim 1 e Caso-de-fim 2, atrás referidos, para o caso da anulação de intervalo de capacidade. Ambos os algoritmos `Carregar` e `Anular` fazem uso do procedimento `Fim-ca2`.

As operações `SEGUINTE`, `PROCURA`, `INTERVALO`, `SEGMENTO` e `ORDEM` são usadas no algoritmo para maior simplicidade e operam como é descrito a seguir.

`SEGUINTE(s)` - Aceita um segmento `s`; produz o segmento de capacidade adjacente temporalmente seguinte a `s` no troço perfil de capacidade a que `s` pertence. Assume-se que, se `s` é o último segmento do perfil esta operação produz um valor que corresponde a *nenhum segmento*.

`PROCURAR(p,t)` - Aceita um perfil de capacidade `p` e um instante de tempo `t` tal que $t \geq TI(p)$ e $t \leq TF(p)$; procura e produz o primeiro segmento[1] `s` de `CS(p)` tal que $TF(s) \geq t$.

`INTERVALO(id,c,ti,tf)` - Aceita um identificador `id`, um valor de capacidade `c` e dois instantes de tempo `ti` e `tf`; constrói e produz um intervalo de capacidade novo cujo valor é `<id,c,<ti,tf>>`.

`SEGMENTO(c,ti,tf,I)` - Aceita um valor de capacidade `c`, dois instantes de tempo `ti` e `tf` e um conjunto de intervalos de capacidade `I`; constrói e produz um segmento de capacidade novo cujo valor é `<c,<ti,tf>,I>`.

`ORDEM(I,Io)` - Aceita dois conjuntos de intervalos de capacidade `I` e `Io` tais que $I \subseteq Io$; produz um conjunto de intervalos de capacidade exactamente com os mesmos elementos de `I` mas em que esses elementos estão ordenados pela ordem em que estão em `Io`.

O algoritmo para `Carregar` é:

[1] O primeiro segmento significa o segmento que ocorre temporalmente mais cedo.

```
PROCEDIMENTO Carregar(VAR     p          : Tipo-Perfil ;
                              id         : String ;
                              c, ti, tf : integer) ;
VAR  SegmentoActual : Tipo-Segmento ;
     i, iq          : Tipo-Intervalo ;
INÍCIO
1  SE   (∀(iq) iq ∈ CI(p) ⇒ id ≠ ID(iq)) ∧
       (ti ≥ TI(p)) ∧ (tf ≤ TF(p))              ENTÃO
2      i := INTERVALO(id, c, ti, tf) ;
3      SegmentoActual := PROCURAR(p, TI(i)) ;
4      SE   (TI(SegmentoActual) < TI(i)) ∧
            (TF(SegmentoActual) > TI(i))  ENTÃO
5           Inicio-c1(i, SegmentoActual)
6      SENÃO
7           Inicio-c2(p, i, SegmentoActual)
8      FIM ;
9      Meio-c(i, SegmentoActual) ;
10     SE   TF(SegmentoActual) > TF(i)    ENTÃO
11          Fim-c1(i, SegmentoActual)
12     SENÃO
13          SE   TF(i) < TF(p)  ENTÃO
14               SE   VAL(Sa) = VAL(SEGUINTE(Sa))   ENTÃO
15                    Fim-ca2(p, i, SegmentoActual)
16               FIM
17          FIM
18     FIM ;
19     CI(p) := CI(p) ∪ i
20 FIM
   FIM
```

O algoritmo para `Anular` é:

```
   PROCEDIMENTO Anular(VAR  p    : Tipo-Perfil ;
                       id        : String) ;
   VAR  SegmentoActual : Tipo-Segmento ;
        i, iq          : Tipo-Intervalo;
   INÍCIO
1  SE   (∃(iq) iq ∈ CI(p) ∧ id = ID(iq))   ENTÃO
2       i := iq ;
3       SegmentoActual := PROCURAR(p, TI(i)) ;
4       SE   (TI(SegmentoActual) < TI(i)) ∧
             (TF(SegmentoActual) > TI(i))  ENTÃO
5            Inicio-a1(i, SegmentoActual)
6       SENÃO
7            Inicio-a2(p, i, SegmentoActual)
8       FIM ;
9       Meio-a(i, SegmentoActual) ;
10      SE   TF(SegmentoActual) > TF(i)    ENTÃO
11           Fim-a1(i, SegmentoActual)
12      SENÃO
13           SE   TF(i) < TF(p)  ENTÃO
14                SE   VAL(Sa) = VAL(SEGUINTE(Sa))   ENTÃO
15                     Fim-ca2(p, i, SegmentoActual)
16                FIM
17           FIM
18      FIM ;
19      CI(p) := CI(p) \ { i }
20 FIM
   FIM
```

O algoritmo de `Inicio-c1` é:

```
   PROCEDIMENTO Inicio-c1(       i    : Tipo-Intervalo ;
                         VAR  Sa   : Tipo-Segmento) ;
   VAR  Snovo     : Tipo-Segmento ;
        iq        : Tipo-Intervalo;
   INÍCIO
1  Snovo := SEGMENTO(VAL(Sa)+VAL(i), TI(i), TF(Sa),
        (CI(Sa)\{iq : (iq ∈ CI(Sa)) ∧ (TF(iq)≤TI(i))})∪{i}) ;
2  SEGUINTE(Snovo) := SEGUINTE(Sa) ;
3  SEGUINTE(Sa) := Snovo ;
4  TF(Sa) := TI(i) ;
5  CI(Sa) := CI(Sa)\{ iq : (iq ∈ CI(Sa)) ∧ (TI(iq)≥TI(i))} ;
6  Sa := Snovo
   FIM
```

Nota:

1. No fim do procedimento `Inicio-c1`, a variável segmento actual, `Sa`, contém o segmento seguinte ao segmento novo (se `i` termina após o fim do segmento novo), ou o segmento novo (no caso contrário); note-se que, nessa altura, os valores `VAL(Sa)` e `CI(Sa)` foram modificados mas poderão não ser definitivos, pois este segmento poderá ser ainda processado pelos procedimentos a aplicar posteriormente.
2. Note-se que, no passo 2, `SEGUINTE(Sa)` poderá não existir, se `Sa` era o último segmento do perfil; nesse caso, o valor que `SEGUINTE(Snovo)` adquire será o de *nenhum segmento*.

Segue-se o algoritmo de `Inicio-a1`:

```
PROCEDIMENTO Inicio-a1(       i    : Tipo-Intervalo ;
                        VAR  Sa   : Tipo-Segmento) ;
VAR  Snovo     : Tipo-Segmento ;
     iq        : Tipo-Intervalo;
INÍCIO
1 Snovo := SEGMENTO(VAL(Sa)-VAL(i), TI(i), TF(Sa),
     (CI(Sa)\{iq : (iq ∈ CI(Sa)) ∧ (TF(iq)≤TI(i))})\{i}) ;
2 SEGUINTE(Snovo) := SEGUINTE(Sa) ;
3 SEGUINTE(Sa) := Snovo ;
4 TF(Sa) := TI(i) ;
5 CI(Sa) := (CI(Sa)\{ iq : (iq ∈ CI(Sa)) ∧ (TI(iq)≥TI(i))}) ;
6 Sa := Snovo
FIM
```

Nota:

1. Na linha 5, `i` é também excluído de `CI(Sa)`, porque a condição `TI(i)≥TI(i)` é satisfeita.
2. No fim do procedimento `Inicio-a1`, a variável segmento actual, `Sa`, contém o segmento seguinte ao segmento novo (se `i` termina após o fim do segmento novo), ou o segmento novo (no caso contrário); note-se que, nessa altura, os valores `VAL(Sa)` e `CI(Sa)` foram modificados mas poderão não ser definitivos, pois este segmento poderá ser ainda processado pelos procedimentos a aplicar posteriormente, onde também poderá vir a ser alterado o valor de `TF(Sa)`.
3. Note-se que, no passo 2, `SEGUINTE(Sa)` poderá não existir, se `Sa` era o último segmento do perfil; nesse caso, o valor que `SEGUINTE(Snovo)` adquire será o de *nenhum segmento*.

O algoritmo de `Inicio-c2` é o seguinte:

```
   PROCEDIMENTO Inicio-c2(  VAR  p    : Tipo-Perfil ;
                                 i    : Tipo-Intervalo ;
                            VAR  Sa   : Tipo-Segmento) ;
   INÍCIO
1  SE   TI(i) = TI(p)  ENTÃO
2       VAL(Sa) := VAL(Sa)+VAL(i) ;
3       CI(Sa) := CI(Sa)∪{i}
4  SENÃO
5       SE   VAL(Sa) ≠ VAL(SEGUINTE(Sa))+VAL(i) ENTÃO
6            Sa := SEGUINTE(Sa) ;
7            VAL(Sa) := VAL(Sa)+VAL(i) ;
8            CI(Sa) := CI(Sa)∪{i}
9       SENÃO
10           TF(Sa) := TF(SEGUINTE(Sa)) ;
11           CI(Sa) := ORDEM(CI(Sa)∪CI(SEGUINTE(Sa)),
                                                CI(p))∪{i} ;
12           SEGUINTE(Sa) := SEGUINTE(SEGUINTE(Sa))
13      FIM
14 FIM
   FIM
```

Nota:

1. Os passos 1, 2 e 3 tratam o caso em que o segmento actual inicial é o primeiro segmento do perfil.
2. Nos passos 5, 6, 7 e 8 trata-se o caso em que o segmento actual inicial não é afectado. Note-se que, na linha 6, a existência do segmento `SEGUINTE(Sa)` está previamente garantida, pois que Carrregar verificou previamente que `i` está contido no horizonte temporal do perfil e identificou o caso como um Caso-de-início 2 e se `i` não se inicia no instante inicial do primeiro segmento do perfil (caso tratado nos passos 1, 2, e 3) logo existirá forçosamente um segmento seguinte ao segmento `Sa` inicial. O mesmo se diz para o caso tratado nos passos 9, 10, 11, 12.
3. No passo 12 poderá não existir um segmento `SEGUINTE(SEGUINTE(Sa))`; nesse caso, o valor que `SEGUINTE(Sa)` adquire será o de *nenhum segmento*.

O algoritmo de `Inicio-a2` é o seguinte:

```
   PROCEDIMENTO Inicio-a2(  VAR  p    : Tipo-Perfil ;
                                 i    : Tipo-Intervalo ;
                            VAR  Sa   : Tipo-Segmento) ;
   INÍCIO
1  SE   TI(i) = TI(p)  ENTÃO
2       VAL(Sa) := VAL(Sa)-VAL(i) ;
3       CI(Sa) := CI(Sa)\{i}
4  SENÃO
5       SE   VAL(Sa) ≠ VAL(SEGUINTE(Sa))-VAL(i) ENTÃO
6            Sa := SEGUINTE(Sa) ;
7            VAL(Sa) := VAL(Sa)-VAL(i) ;
8            CI(Sa) := CI(Sa)\{i}
9       SENÃO
10           TF(Sa) := TF(SEGUINTE(Sa)) ;
11           CI(Sa) := ORDEM(CI(Sa)∪(CI(SEGUINTE(Sa))\{i}),
                                                    CI(p)) ;
12           SEGUINTE(Sa) := SEGUINTE(SEGUINTE(Sa))
13      FIM
14 FIM
   FIM
```

Nota:

1. Os passos 1, 2 e 3 tratam o caso em que o segmento actual inicial é o primeiro segmento do perfil.
2. Nos passos 5, 6, 7 e 8 trata-se o caso em que o segmento actual inicial não é afectado. Note-se que, na linha 6, a existência do segmento `SEGUINTE(Sa)` está previamente garantida, pois que Carrregar verificou previamente que `i` está contido no horizonte temporal do perfil e identificou o caso como um Caso-de-início 2 e se `i` não se inicia no instante inicial do primeiro segmento do perfil (caso tratado nos passos 1, 2, e 3) logo existirá forçosamente um segmento seguinte ao segmento `Sa` inicial. O mesmo se diz para o caso tratado nos passos 9, 10, 11, 12.
3. No passo 12 poderá não existir um segmento `SEGUINTE(SEGUINTE(Sa))`; nesse caso, o valor que `SEGUINTE(Sa)` adquire será o de *nenhum segmento*.

O algoritmo de `Meio-c` é o seguinte:

```
  PROCEDIMENTO Meio-c(     i    : Tipo-Intervalo ;
                      VAR  Sa   : Tipo-Segmento) ;
  INÍCIO
1 ENQUANTO  TF(Sa) < TF(i) FAZER
2      Sa := SEGUINTE(Sa) ;
3      VAL(Sa) := VAL(Sa)+VAL(i) ;
4      CI(Sa) := CI(Sa)∪{i}
5 FIM
  FIM
```

O algoritmo de `Meio-a` é o seguinte:

```
  PROCEDIMENTO Meio-a(     i    : Tipo-Intervalo ;
                      VAR  Sa   : Tipo-Segmento) ;
  INÍCIO
1 ENQUANTO  TF(Sa) < TF(i) FAZER
2      Sa := SEGUINTE(Sa) ;
3      VAL(Sa) := VAL(Sa)-VAL(i) ;
4      CI(Sa) := CI(Sa)\{i}
5 FIM
  FIM
```

O algoritmo de `Fim-c1` é o seguinte:

```
  PROCEDIMENTO Fim-c1(     i    : Tipo-Intervalo ;
                      VAR  Sa   : Tipo-Segmento) ;
  VAR  Snovo     : Tipo-Segmento ;
       iq        : Tipo-Intervalo ;
  INÍCIO
1 Snovo := SEGMENTO(VAL(Sa)-VAL(i), TF(i), TF(Sa),
       (CI(Sa)\{iq : (iq ∈ CI(Sa)) ∧ (TF(iq)≤TF(i))})) ;
2 SEGUINTE(Snovo) := SEGUINTE(Sa) ;
3 SEGUINTE(Sa) := Snovo ;
4 TF(Sa) := TF(i) ;
5 CI(Sa) := CI(Sa)\{ iq : (iq ∈ CI(Sa)) ∧ (TI(iq)≥TF(i))}
  FIM
```

Nota:

1. No passo 1, o valor `CI(Sa)\{iq:TF(iq)≤TF(i)}`, exclui automaticamanete `i` (que tinha sido previamente incluído com a aplicação dos procedimentos prévios), do segmento novo `Snovo`, pois `TF(i)≤TF(i)` verifica-se.
2. Note-se que, no passo 2, `SEGUINTE(Sa)` poderá não existir, se `Sa` era o último segmento do perfil; nesse caso, o valor que `SEGUINTE(Snovo)` adquire será o de *nenhum segmento*.

O algoritmo de `Fim-a1` é o seguinte:

```
PROCEDIMENTO Fim-a1(     i    : Tipo-Intervalo ;
                    VAR  Sa   : Tipo-Segmento) ;
VAR  Snovo     : Tipo-Segmento ;
     iq        : Tipo-Intervalo ;
INÍCIO
1 Snovo := SEGMENTO(VAL(Sa)+VAL(i), TF(i), TF(Sa),
     (CI(Sa)\{iq : (iq ∈ CI(Sa)) ∧ (TF(iq)≤TF(i))})) ;
2 SEGUINTE(Snovo) := SEGUINTE(Sa) ;
3 SEGUINTE(Sa) := Snovo ;
4 TF(Sa) := TF(i) ;
5 CI(Sa) := CI(Sa)\{ iq : (iq ∈ CI(Sa)) ∧ (TI(iq)≥TF(i))}
FIM
```

Nota:

No passo 2, `SEGUINTE(Sa)` poderá não existir, se `Sa` era o último segmento do perfil; nesse caso, o valor que `SEGUINTE(Snovo)` adquire será o de *nenhum segmento*.

O algoritmo de `Fim-ca2` é o seguinte:

```
PROCEDIMENTO Fim-ca2(    VAR  p    : Tipo-Perfil ;
                              i    : Tipo-Intervalo ;
                         VAR  Sa   : Tipo-Segmento) ;
INÍCIO
1 TF(Sa) := TF(SEGUINTE(Sa)) ;
2 CI(Sa) := ORDEM(CI(Sa)∪CI(SEGUINTE(Sa)),CI(p)) ;
3 SEGUINTE(Sa) := SEGUINTE(SEGUINTE(Sa))
FIM
```

Nota:

1. Os passos 3, 4 e 5 processam o caso em que o segmento actual inicial não é o último segmento do perfil e em que ele deve ser fundido com o segmento seguinte. Nos restantes casos possíveis todo o processamento foi já realizado por procedimentos previamente aplicados.
2. No passo 5 poderá não existir um segmento `SEGUINTE(SEGUINTE(Sa))`; nesse caso, o valor que `SEGUINTE(Sa)` adquire será o de *nenhum segmento*.
3. Note-se que, no passo 4, o valor que `CI(Sa)` adquire inclui `i` que foi previamente incluído em `Sa` por procedimentos previamente aplicados no caso de carga, ou não inclui `i`, que foi previamente excluído de `Sa` por procedimentos previamente aplicados, no caso de anulação, de `i`.
4. Note-se que, no passo 2, `VAL(Sa)` é um valor que já foi previamente modificado tendo em conta a carga, ou anulação, de `i`.

Segue-se um conjunto de figuras — da Figura 5 à Figura 22 —que ilustra o processamento da carga do intervalo de capacidade `i` pelo algoritmo de `Carregar`, para os casos Caso 1-1, Caso 1-2, Caso 1-3, Caso 1-4, Caso 2-1, Caso 2-2, Caso 2-3 e Caso 2-4, representados na Figura 1 e na Figura 2. As figuras ilustram a evolução do estado do perfil de capacidade nos quatro instantes, 1, 2, 3 e 4, de processamento a seguir descritos na Figura 4. Para o algoritmo de `Anular` as ilustrações (e respectivas legendas) são semelhantes mas com "`+`" substituído por "`-`".

Instante	Passo de `Carregar`	Passo de `Anular`
1	Antes de 4	Antes de 4
2	Antes de 9	Antes de 9
3	Antes de 10	Antes de 10
4	No fim do algoritmo	No fim do algoritmo

Figura 4- Instantes/estados no processamento da carga e anulação e passos dos algoritmos `Carregar` e `Anular`.

As figuras, Figura 5 a Figura 22, são mapas temporais onde se representam, dentro do horizonte temporal do perfil de capacidade, os segmentos `s` e número de segmentos presentes no perfil, visível por meio dos índices indicados nos segmentos `s`, os valores de capacidade `c` dos mesmos segmentos e aos instantes de início e fim dos respectivos intervalos de tempo. Um valor de capacidade c_k representa o valor de capacidade de um segmento s_k, isto é $c_k=VAL(s_k)$; `c` representa sempre o valor de capacidade do intervalo `i` a ser carregado, ou anulado, isto é, `c=VAL(i)`. Para cada um dos instantes/estados 2, 3 e 4 é assinalada a parte do segmento abrangida pela modificação, se houve alguma, feita entre o estado anterior e esse estado. Para os instantes/estados 1, 2 e 3, o segmento actual é também assinalado, com uma seta vertical.

Caso 1-1:

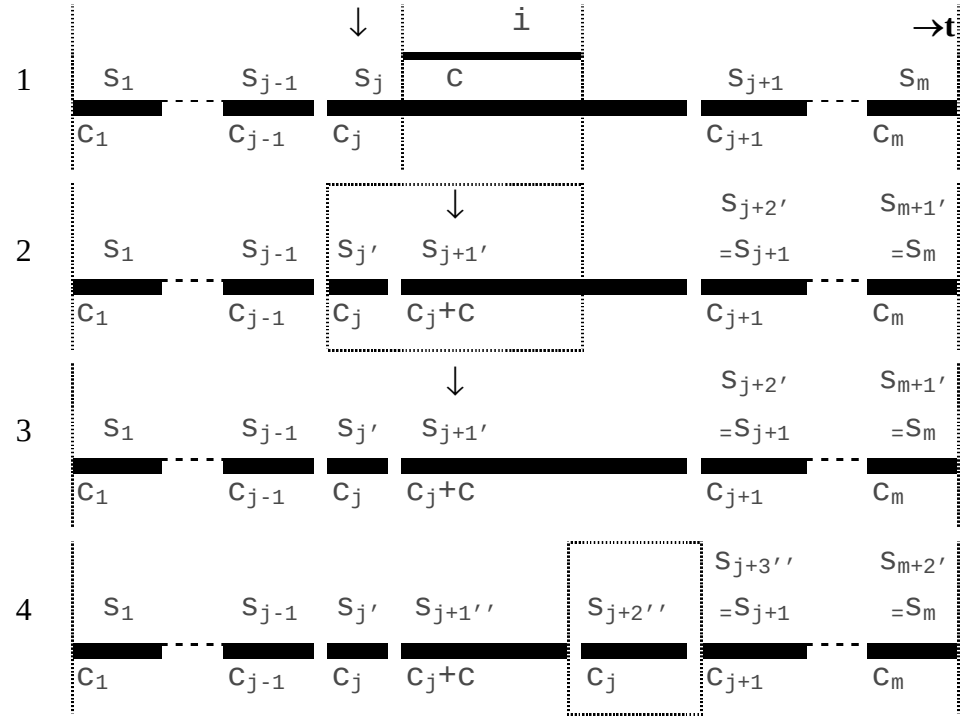


Figura 5- Caso 1-1: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade.

Caso 1-2:

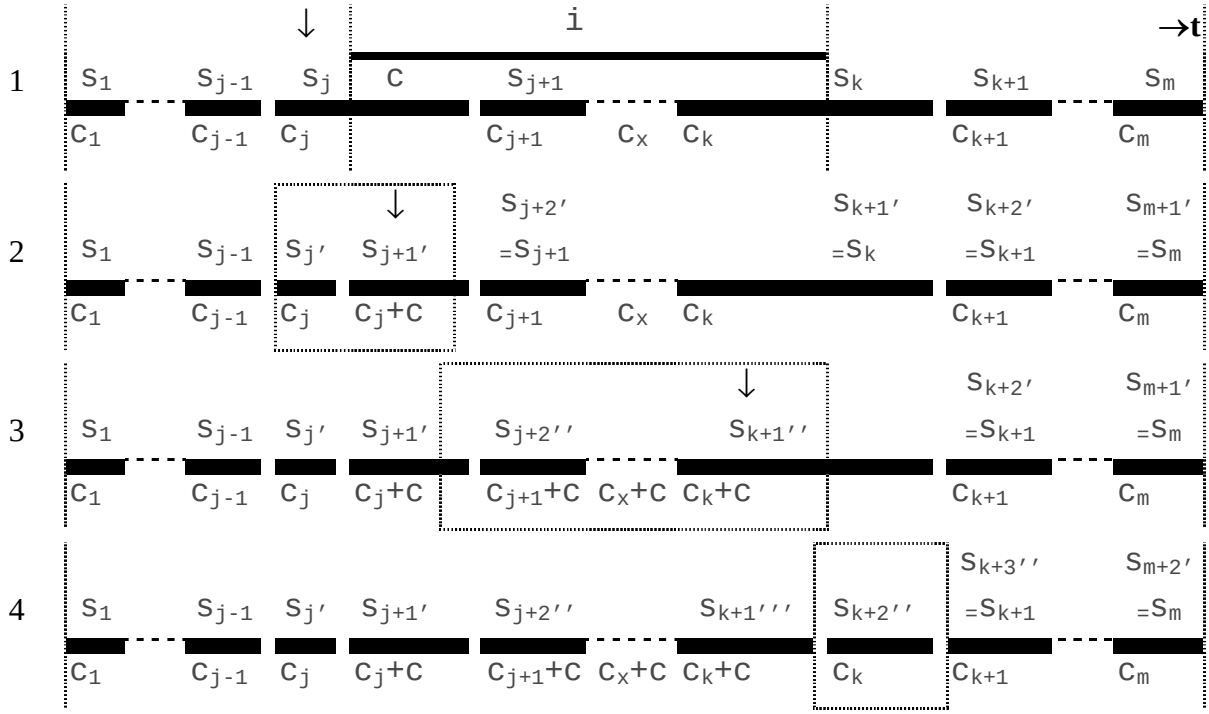


Figura 6- Caso 1-2: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade.

Caso 1-3-a):

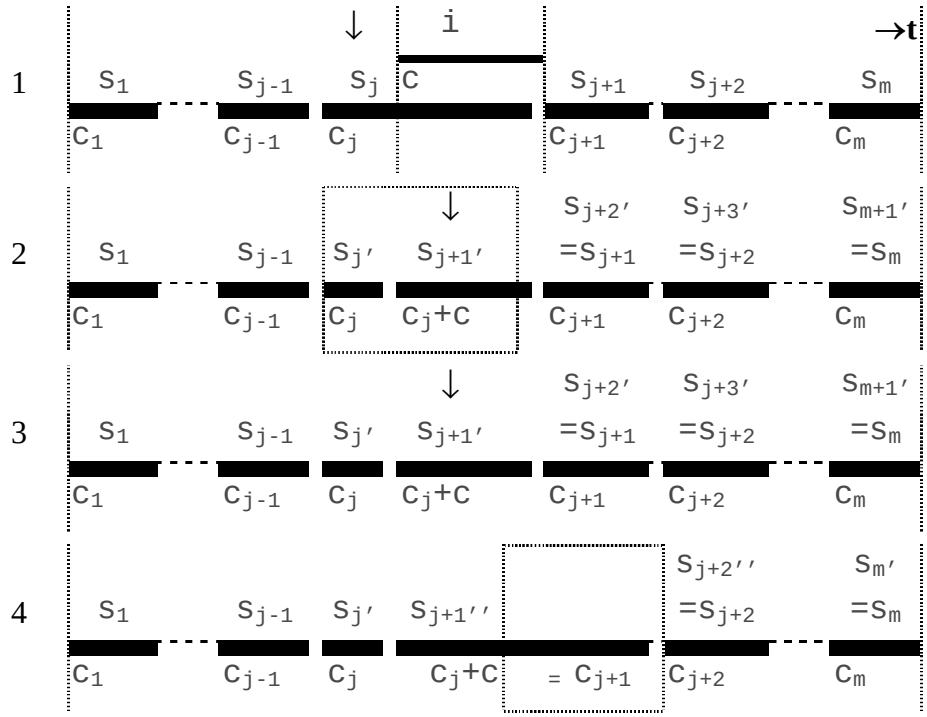


Figura 7- Caso 1-3: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_j)+VAL(i)=VAL(s_{j+1})$.

Caso 1-3-b):

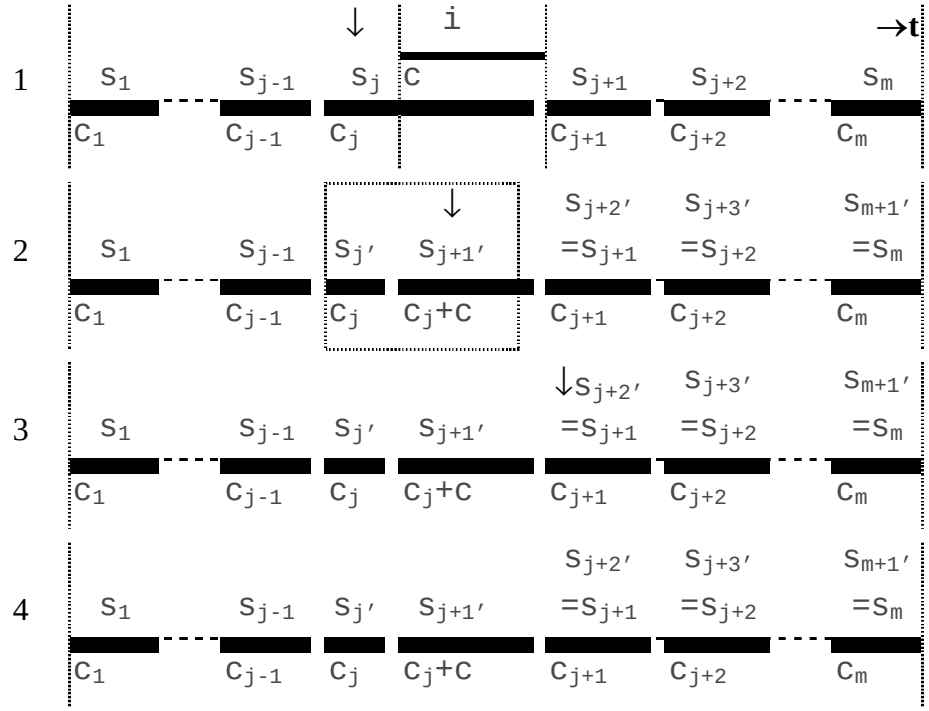


Figura 8- Caso 1-3: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_j)+VAL(i)\neq VAL(s_{j+1})$.

Caso 1-4-a):

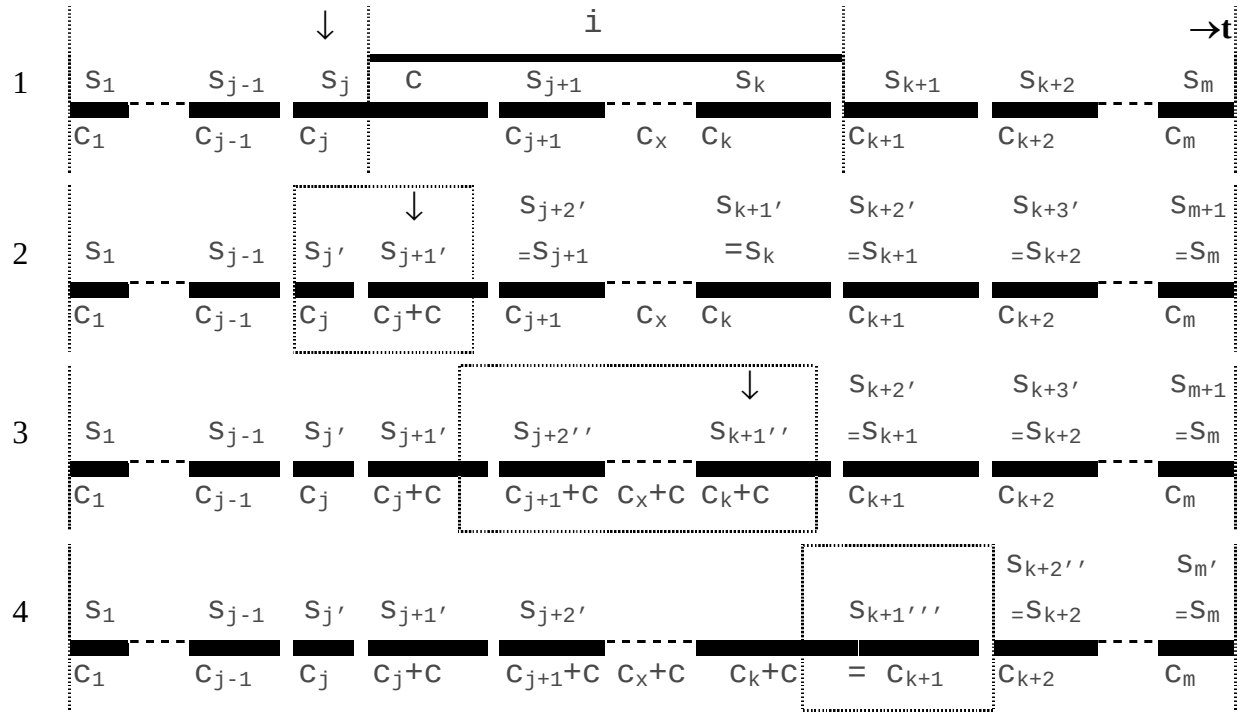


Figura 9- Caso 1-4: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_k)+VAL(i)=VAL(s_{k+1})$.

Caso 1-4-b):

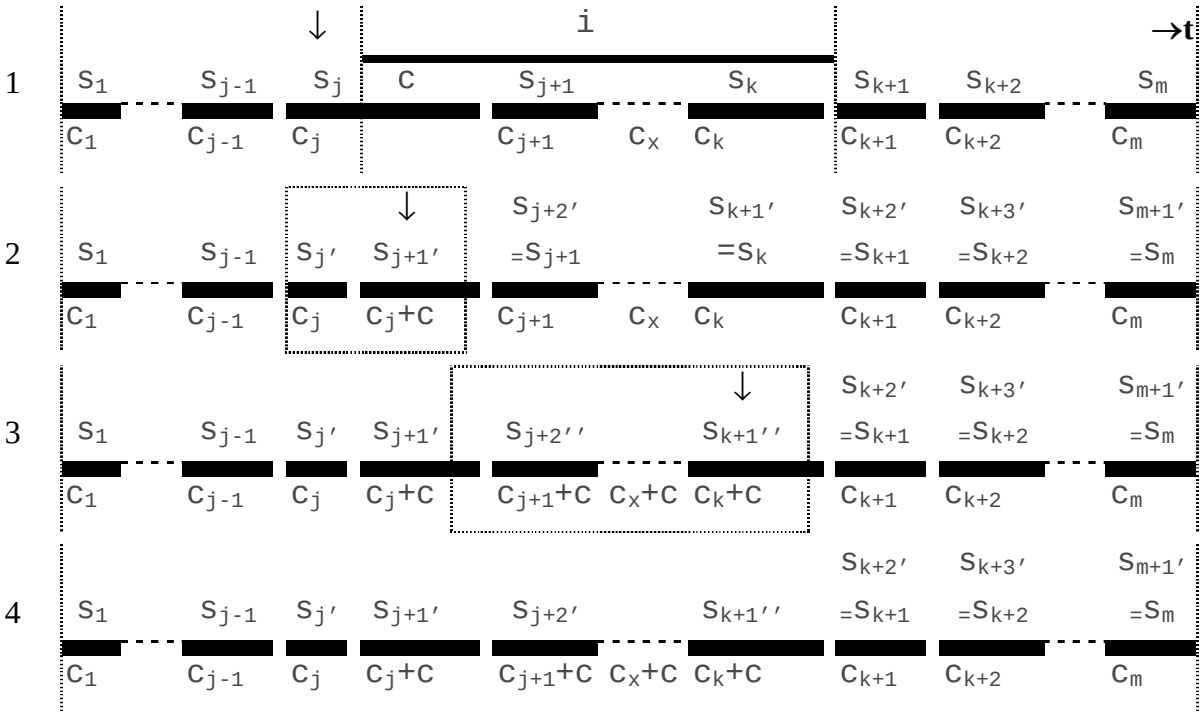


Figura 10- Caso 1-4: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_k)+VAL(i)\neq VAL(s_{k+1})$.

Caso 2-1-a):

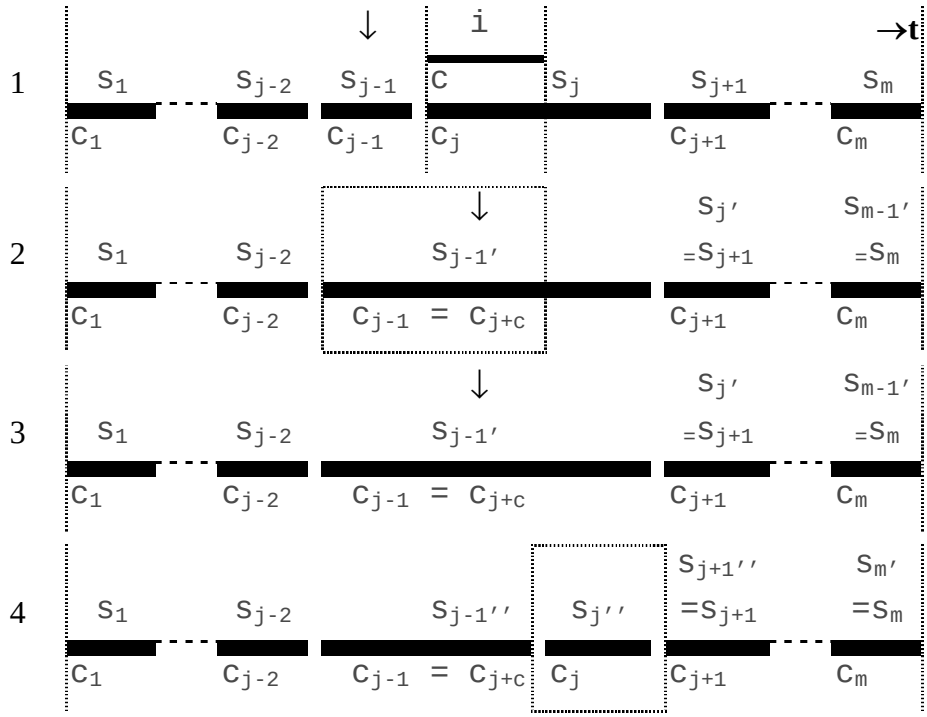


Figura 11- Caso 2-1: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_{j-1})=VAL(s_j)+VAL(i)$.

Caso 2-1-b):

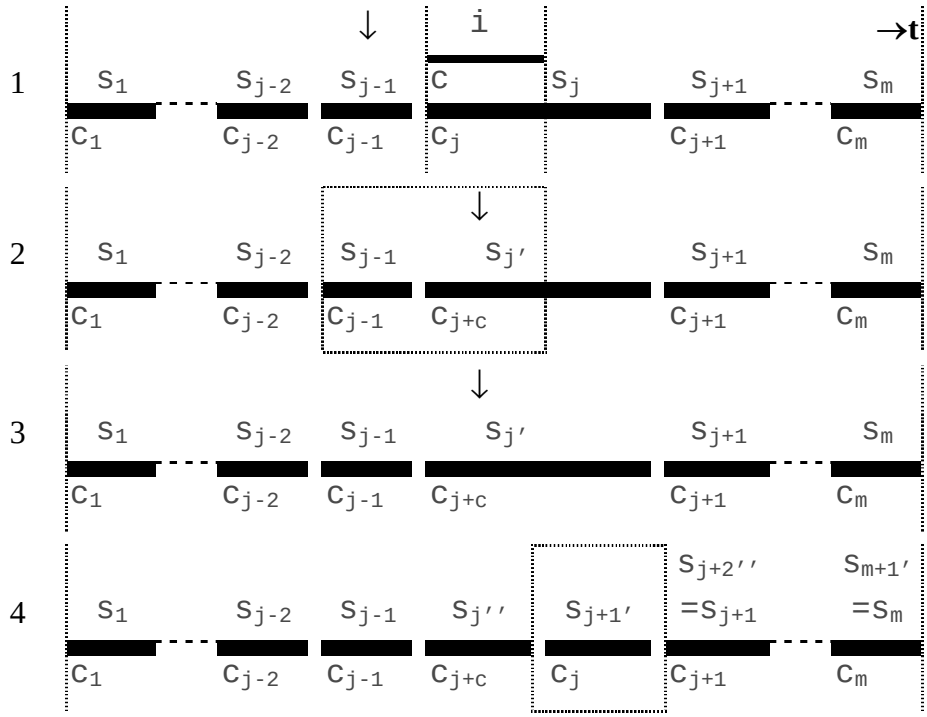


Figura 12- Caso 2-1: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_{j-1}) \neq VAL(s_j)+VAL(i)$.

Caso 2-2-a):

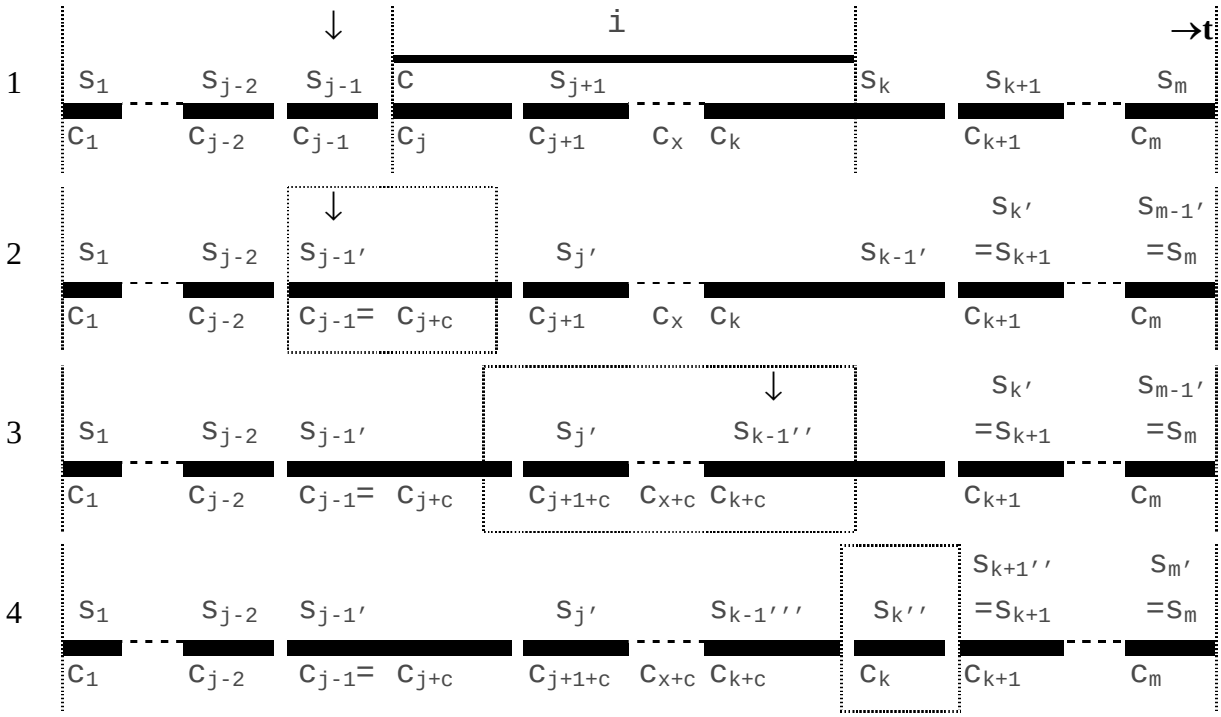


Figura 13- Caso 2-2: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_{j-1})=VAL(s_j)+VAL(i)$.

Caso 2-2-b):

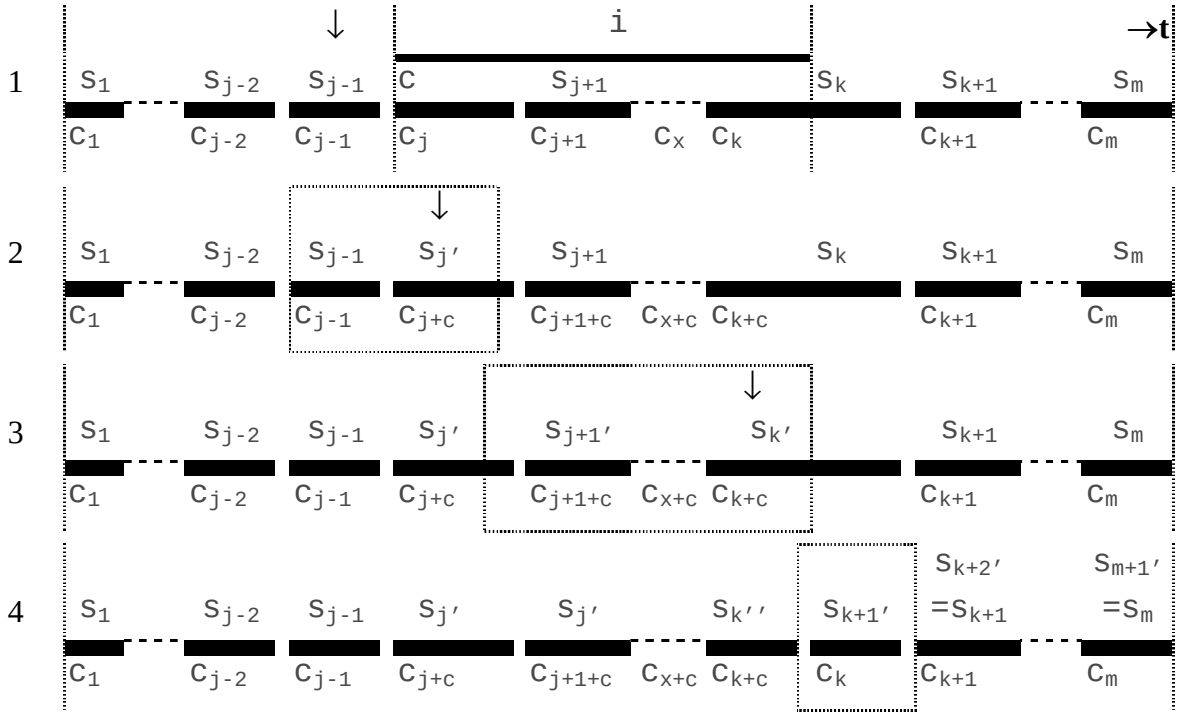


Figura 14- Caso 2-2: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_{j-1}) \neq VAL(s_j)+VAL(i)$.

Caso 2-3-a):

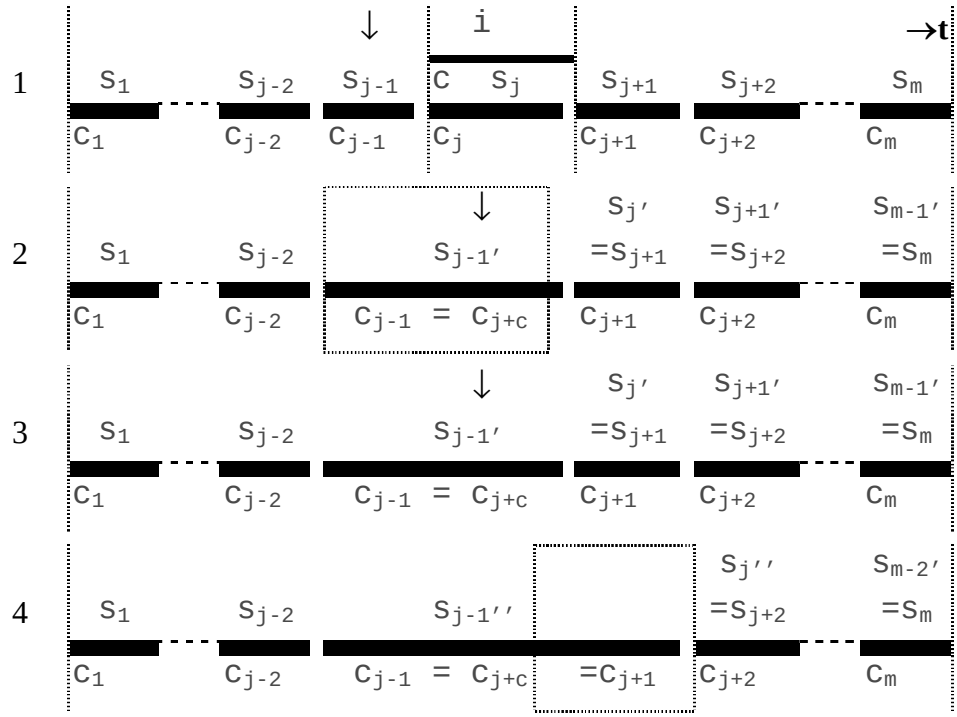


Figura 15- Caso 2-3: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_{j-1})=VAL(s_j)+VAL(i)$ e $VAL(s_{j+1})=VAL(s_j)+VAL(i)$.

Caso 2-3-b):

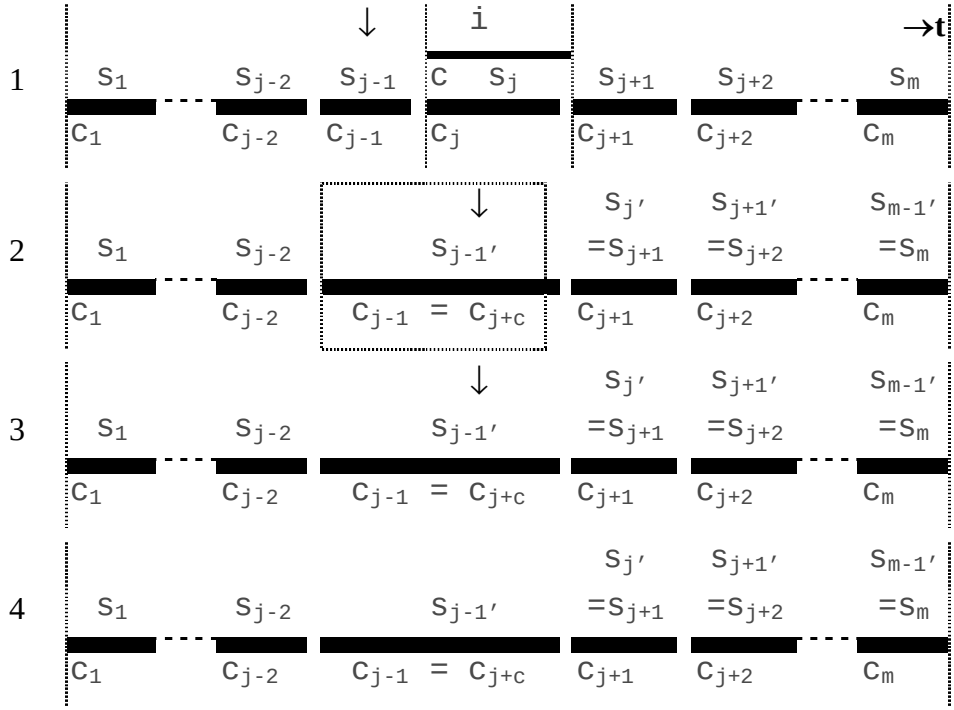


Figura 16- Caso 2-3: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_{j-1})=VAL(s_j)+VAL(i)$ e $VAL(s_{j+1})\neq VAL(s_j)+VAL(i)$.

Caso 2-3-c):

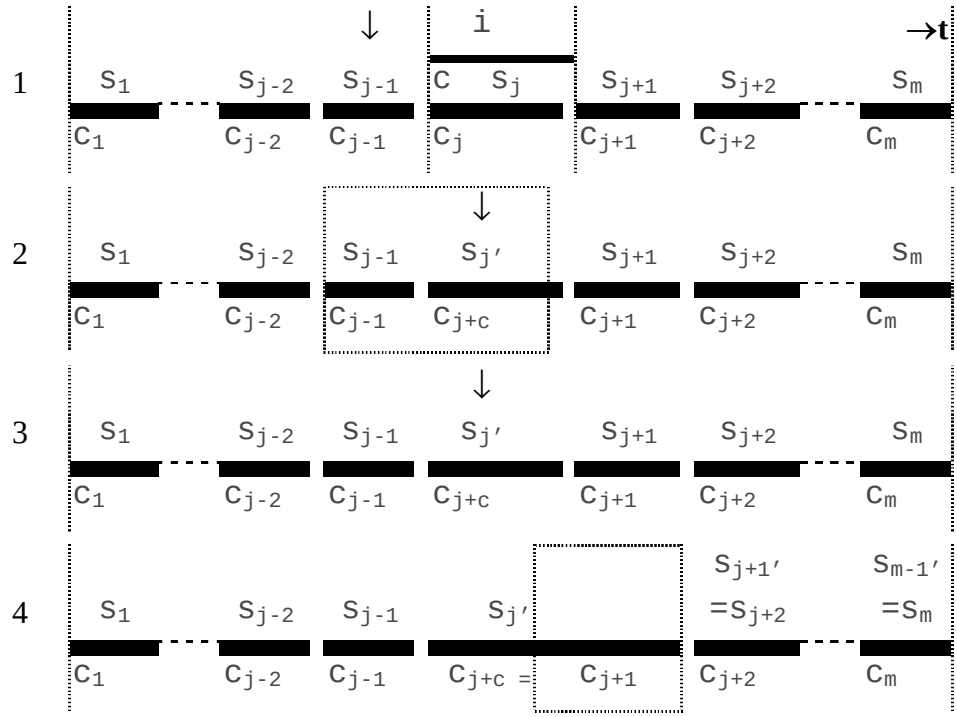


Figura 17- Caso 2-3: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_{j-1}) \neq VAL(s_j)+VAL(i)$ e $VAL(s_{j+1})=VAL(s_j)+VAL(i)$.

Caso 2-3-d):

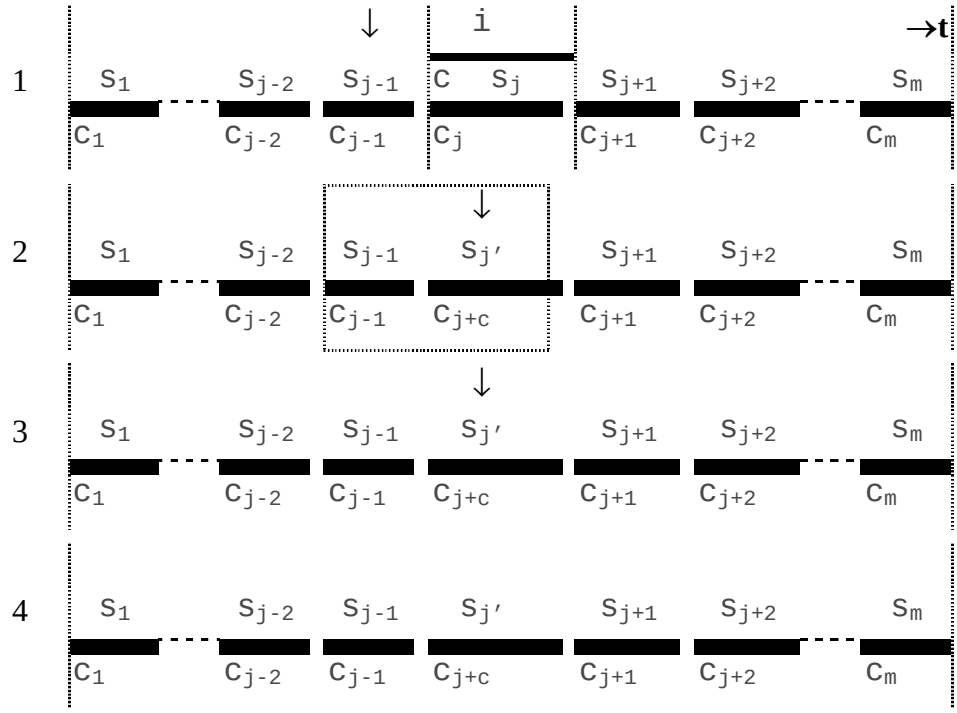


Figura 18- Caso 2-3: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_{j-1}) \neq VAL(s_j)+VAL(i)$ e $VAL(s_{j+1}) \neq VAL(s_j)+VAL(i)$.

Caso 2-4-a):

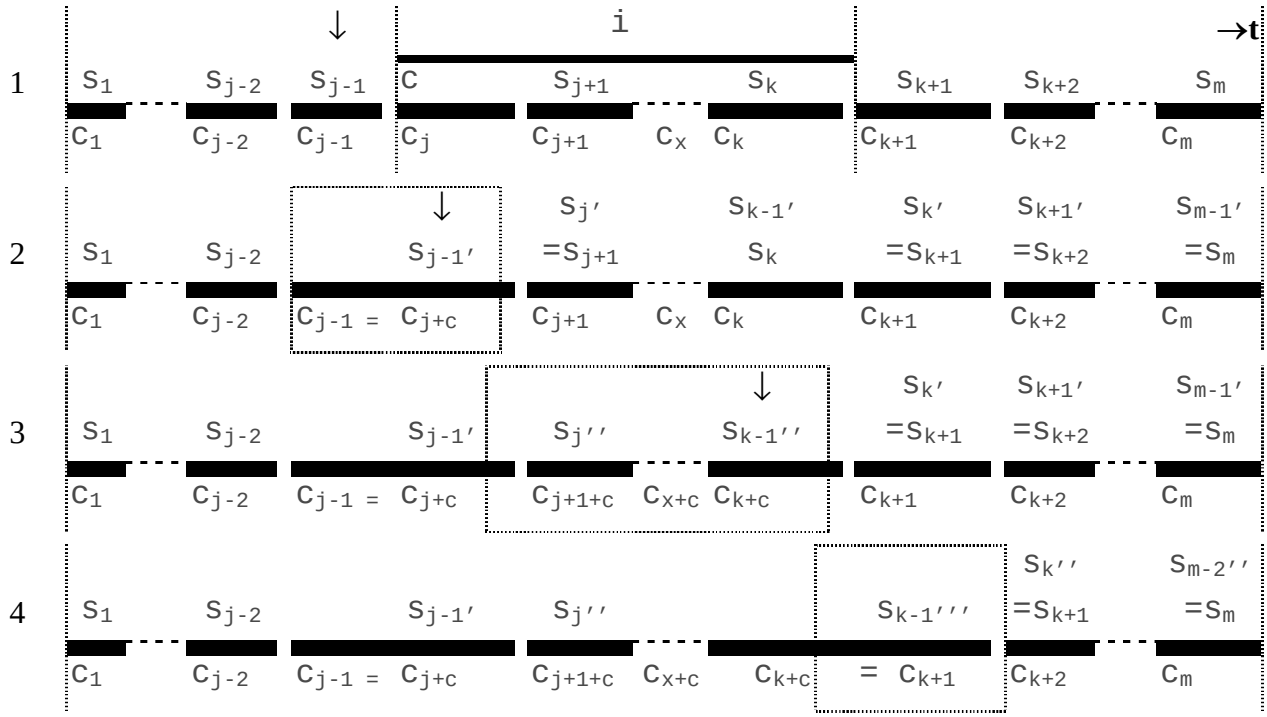


Figura 19- Caso 2-4: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_{j-1})=VAL(s_j)+VAL(i)$ e $VAL(s_{k+1})=VAL(s_k)+VAL(i)$.

Caso 2-4-b):

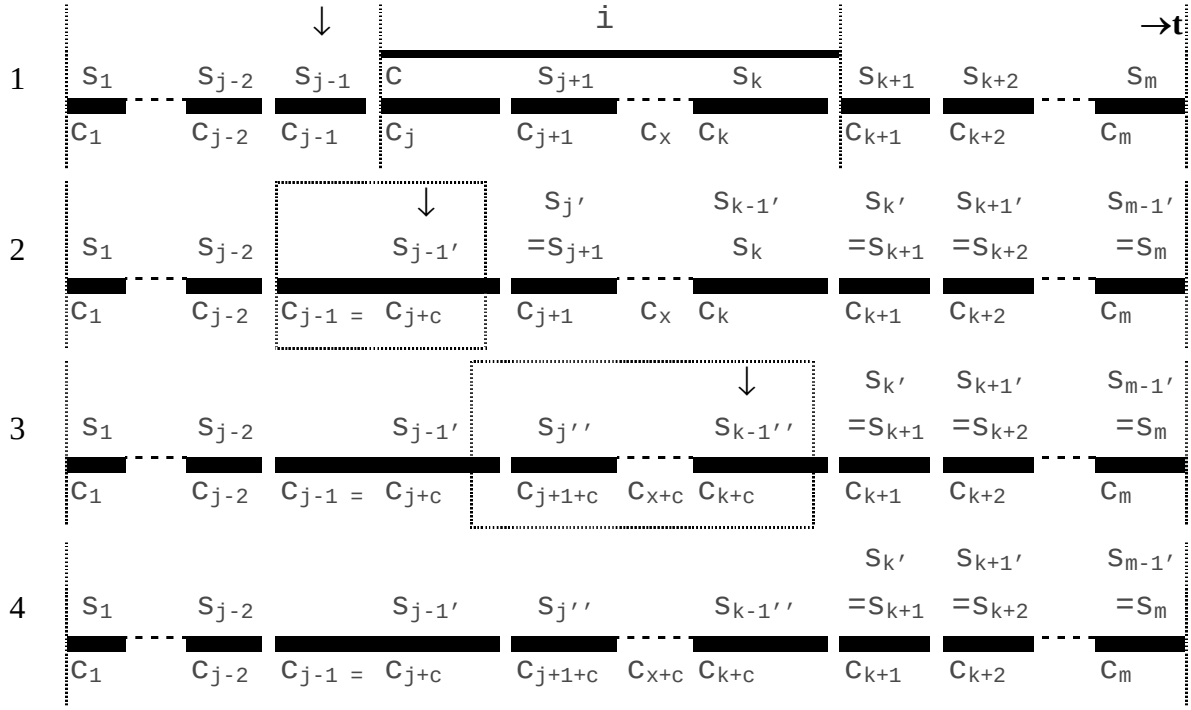


Figura 20- Caso 2-4: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_{j-1})=VAL(s_j)+VAL(i)$ e $VAL(s_{k+1})\neq VAL(s_k)+VAL(i)$.

Caso 2-4-c):

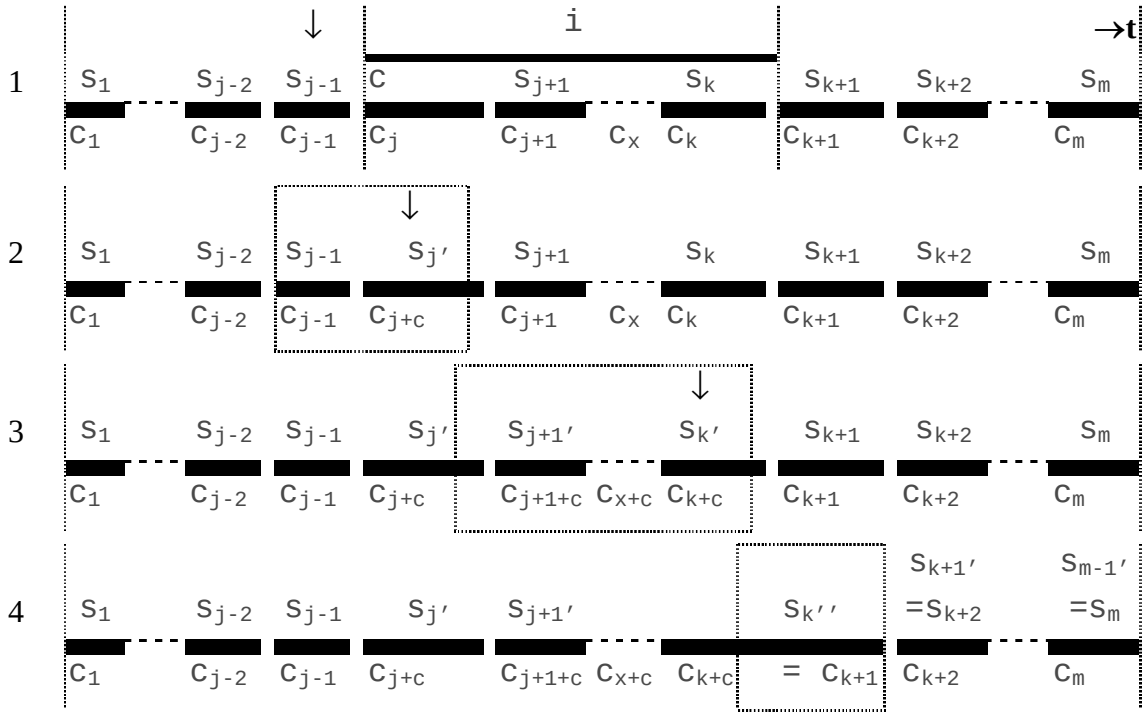


Figura 21- Caso 2-4: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_{j-1}) \neq VAL(s_j)+VAL(i)$ e $VAL(s_{k+1})=VAL(s_k)+VAL(i)$.

Caso 2-4-d):

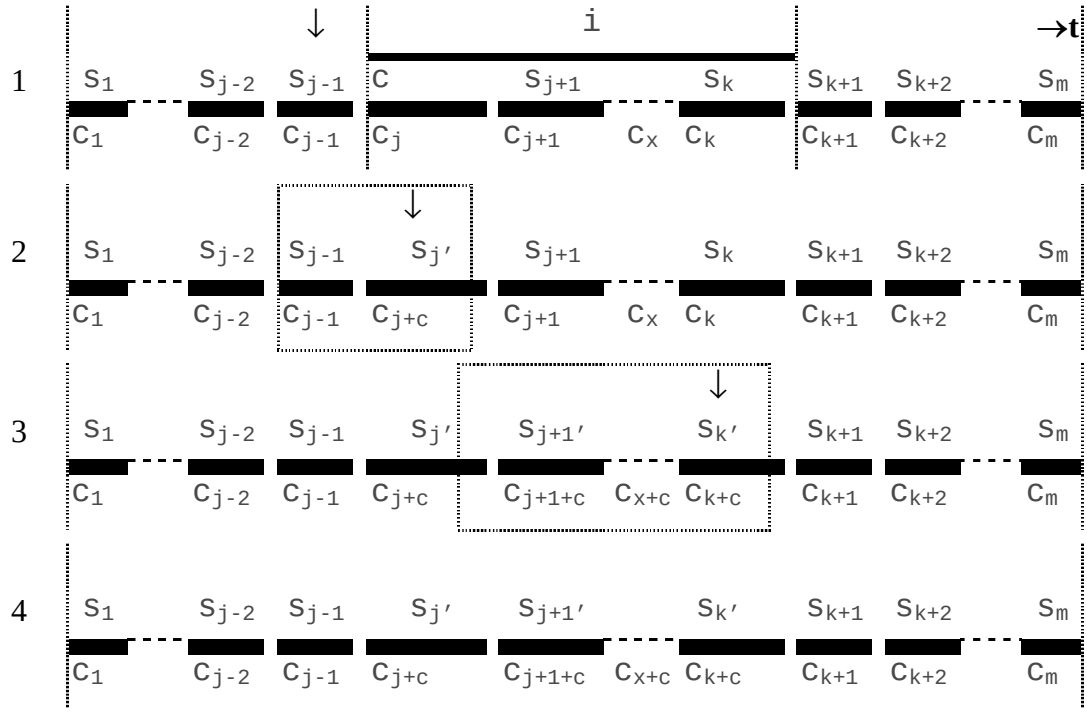


Figura 22- Caso 2-4: estado do perfil de capacidade antes (1), durante (2 e 3) e depois (4) da carga/anulação do intervalo de capacidade, quando $VAL(s_{j-1}) \neq VAL(s_j)+VAL(i)$ e $VAL(s_{k+1}) \neq VAL(s_k)+VAL(i)$.

4 Operações sobre Perfis de Capacidade

Descreve-se a seguir uma operação que combina dois perfis de capacidade para obter um terceiro, a operação `Combinar`.

`Combinar(op,p1,p2,id,p)` - Aceita uma operação aritmética, `op`, dois perfis de capacidade, `p1` e `p2`, e um identificador para o perfil novo `p`. Devolve em `p` um perfil de capacidade com segmentos de capacidade cujos valores de capacidade resultam dos dos segmentos de `p1` e `p2`, operados por `op`.

A operação `Combinar` é descrita adiante através de um algoritmo. Exemplos de casos em que esta operação tem interesse são os de quando há que obter um perfil de capacidade a partir de outros que representam capacidades parcelares, ou quando há que obter um perfil de capacidade disponível, com base nos perfis de capacidade máxima e de capacidade ocupada.

As operações `MIN` e `APLICAR` são usadas no algoritmo de `Combinar` para maior simplicidade e operam como é descrito a seguir.

`MIN(t1,t2)` - Aceita dois instantes de tempo, `t1` e `t2`, e produz o menor deles, ou um deles, se forem iguais.

`APLICAR(op,c1,c2)` - Aceita uma operação aritmética, `op` (tipicamente a operação `+`, ou a operação `-`) e dois valores de capacidade, `c1` e `c2`, aplica a operação aos valores e produz o resultado (*i.e.*, o resultado de `c1 op c2`).

O algoritmo da operação `Combinar` é:

```
   PROCEDIMENTO Combinar(        op        : Tipo-Operacao ;
                                 p1, p2    : Tipo-Perfil ;
                                 id        : string ;
                           VAR   p         : Tipo-Perfil) ;
   VAR  Sa1, Sa2, Sa, Snovo : Tipo-Segmento ;
        tia, tfa            : integer ;
   INÍCIO
1  SE   TI(p1) = TI(p2) ∧ TF(p1) = TF(p2) ENTÃO
        CriarPerfil(id, TI(p1),TF(p1)) ;
4       CI(p) := CI(p1) ∪ CI(p2) ;
5       tia := TI(p) ;
6       tfa := MIN(TF(Sa1), TF(Sa2)) ;
7       Sa1 := PROCURAR(p1, TI(p1)) ;
8       Sa2 := PROCURAR(p2, TI(p2)) ;
9       Sa := SEGMENTO(APLICAR(op, VAL(Sa1), VAL(Sa2)),
                       tia, tfa, CI(Sa1) ∪ CI(Sa2)) ;
10      CS(p) := Sa ;
11      ENQUANTO tfa < TF(p) FAZER
12           SE   tfa = TF(Sa1)  ENTÃO
13                Sa1 := SEGUINTE(Sa1)
14           FIM ;
15           SE   tfa = TF(Sa2)  ENTÃO
16                Sa2 := SEGUINTE(Sa2)
17           FIM ;
18           SE   APLICAR(op,  VAL(Sa1),  VAL(Sa2))  =  VAL(Sa)
        ENTÃO
19                tfa := MIN(TF(Sa1), TF(Sa2)) ;
20                TF(Sa) := tfa ;
21                CI(Sa) := CI(Sa) ∪ CI(Sa1) ∪ CI(Sa2)
22           SENÃO
23                tia := tfa ;
24                tfa := MIN(TF(Sa1), TF(Sa2)) ;
25                Snovo := SEGMENTO(APLICAR(op, VAL(Sa1),
                  VAL(Sa2)), tia, tfa, CI(Sa1) ∪ CI(Sa2)) ;
26                SEGUINTE(Sa) := Snovo ;
27                Sa := Snovo
28           FIM
29      FIM
30 FIM
   FIM
```

5 Aspectos a Refinar

Enumeram-se aqui alguns aspectos que podem constituir trabalho futuro sobre a matéria tratada neste documento.

5.1 Reservas

Uma reserva é um pré-aviso de afectação, em que apenas se indica que o recurso será afectado a uma certa tarefa, sem se saber ainda o tempo de início preciso desta, mas sabendo-se qual o intervalo de tempo, que abrange o intervalo de execução, em que a tarefa deverá ser executada.

Tome-se por exemplo uma situação em que a tarefa Ox (um transporte de uma mercadoria, uma palestra, etc.) com duração de 2 horas, necessitando de uma unidade de um recurso determinado Ry (um veículo, um conferencista, etc.) deve ocorrer/ser executada forçosamente numa dada semana de um dado mês. Trata-se de uma situação que não é rara e que envolveria a reserva/pré-aviso de afectação, para aquela semana daquele recurso, relegando-se para mais tarde a definição do tempo de início preciso da tarefa (isto é, a afectação do recurso à tarefa - a efectiva programação da tarefa).

Uma representação e operações que permitam manipular e raciocinar sobre reservas poderá ser necessária. Estas detectarão situações de conflito não através da capacidade disponível do recurso em cada instante, mas através da área contida entre a linha do perfil de capacidade disponível e o eixo dos tempos dentro do intervalo da reserva, no gráfico do perfil de capacidade disponível ao longo do tempo.

5.2 O Horizonte Temporal de Recurso

Poder ser conveniente, ou não, ter operações de manipulação de recursos que modifiquem os extremos e/ou a extensão do intervalo do horizonte temporal de recurso. A definirem-se estas operações deverá ter-se em conta que, se o novo horizonte contém o antigo, nenhuma questão crítica se põe (a não ser a reoptimização possível da programação das tarefas anteriormente afectadas ao recurso que, no entanto, é tarefa do sistema cliente). Mas se isso não for o caso poderá haver tarefas programadas no recurso cujo intervalo de tempo de execução passa a cair fora do horizonte.

Poderá ser conveniente uma representação de horizontes que sejam não determinados nos extremos. Isto contemplaria casos em que o recurso sempre esteve (ou esteve, desde um tempo indeterminado) disponível até um certo instante tf (horizonte (-∞ tf)), ou o recurso estará sempre disponível (ou vai estar, por um tempo indeterminado) a partir de um certo instante ti (horizonte (ti +∞)), ou o recurso está sempre disponível (horizonte (-∞ +∞)).

5.3 Operações de Input/Output

Operações de entrada e saída de informação com recursos discretos renováveis poderão ser realizadas, basicamente apoiadas nas operações de acesso propostas. Uma interface gráfica interactiva, baseada em gráficos de Gantt, é tipicamente útil em sistemas de

scheduling. Nestas o próprio utilizador pode alterar interactivamente a programação actual das tarefas (aqui estariam em causa não só as operações de acesso mas também as de modificação). De qualquer modo, este tipo de interfaces estará sempre a um nível mais elevado do que o da representação dos recursos, pois tem mais a ver com a representação de programas (schedules, ou calendários). E um programa poderá ser o programa de um recurso apenas, ou de um conjunto limitado de tarefas relacionadas (um processo) do problema, ou de todas as tarefas do problema.

5.4 Realização

Uma realização da representação e das operações proprostas deverá ser levada a cabo. Deverá prever-se a sua integração num sistema mais amplo. Sistemas clientes desta representação são tipicamente sistemas para Scheduling/Programação das Tarefas.

As estruturas de dados que forem usadas na realização poderão não se limitar aos componentes sugeridos (horizonte, capacidade máxima, perfil de capacidade, conflitos, tarefas), ou à sua simplicidade. Assim, por exemplo, o uso de alguma forma de indexação (no tempo) das tarefas, do prefil de capacidade, ou dos conflitos, ou de hash-tables (por exemplo para as tarefas poderá conveniente, em particular porque o os horizonte temporal poderá ser extenso e a quantidade de informação acumulada elevada (muitas tarefas, muitos conflitos, etc.).

Também, em relação à forma de representação sugerida para objectos diferentes do recurso, como por exemplo, as tarefas, os requerimentos de recurso e até mesmo os intervalos de tempo, note-se que os componentes aqui incluídos são aqueles que se julgam estritamente necessários para a representação de recursos aqui investigada. Quer isto dizer que, de um modo geral, a representação poderá ou deverá incluir outros componentes que, para o caso, não parecem ter interesse. Por exemplo, num sistema para scheduling baseado em restrições [Kumar 1992], o domínio temporal actual de uma tarefa em cada passo do processo de programação das tarefas (a ser refinado ao longo deste processo) terá de ser representado na tarefa, ou associado à tarefa [Haralick 1980].

Uma linguagem orientada para objectos e que permita a criação de uma hierarquia de estruturas de dados, de classes, subclasses, instâncias e permita um mecanismo de herança de propriedades e ligação procedimental seria conveniente.

Sugestões podem ser dadas para C++, CLOS ou um sistema baseado em “frames”, ou Prolog orientado para objectos.

6 REFERÊNCIAS

Blazewicz 1994 Blazewicz, J.; Ecker, K.H.; Schmidt, G.; Weglarz, J., "Scheduling in Computer and Manufacturing Systems", Springer Verlag, 1994.

Haralick 1980 Haralick, Robert M.; Elliot, Gordon L., "Increasing Tree Search Efficiency for Constraint Satisfaction Problems", Artificial Intelligence, 14 1980, 263-313.

Kumar 1992 Kumar, Vipin, "Algorithms for Constraint Satisfaction: A Survey", AI Magazine, 13 (1) 1992: 32-44.

Williams 1990 Williams, Brian C., 1990, *Doing Time: Putting Qualitative Reasoning on Firmer Ground*, Morgan Kaufman Publishers, Inc., San Mateo, California, 1990, 353-360.

Zweben 1994 Zweben, Monte; Daun, Brian; Davis, Eugene; Deale, Michael, "Scheduling and Rescheduling with Iterative Repair", in: Zweben, Monte; Fox, Mark S., "Intelligent Scheduling", Cap.8, San Francisco, Morgan Kaufman, 1994.