CARACTERIZAÇÕES DE SOLUÇÕES ÓTIMAS ATRAVÉS DE ALGORITMOS POLINOMIAIS PARA PROBLEMAS EM SCHEDULING TIPO NO-WAIT FLOW-SHOP

Edgard Dias Batista Junior

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS (D.Sc.)

Aprovado por:

JAYME LUIZ SZWARCFITER (presidente)

ANTONIO ALBERTO F. OLIVEIRA

ANTONIO DE ALMEIDA PINHO

CELSO CARNEIRO RIBEIRO

HILTON MACHADO

RUY EDUARDO CAMPELLO

RIO DE JANEIRO - BRASIL
DEZEMBRO DE 1985

BATISTA JUNIOR, EDGARD DIAS

Caracterizações de Soluções Ótimas através de Algoritmos Polinomiais para Problemas em Scheduling Tipo No-Wait Flow-Shop (Rio de Janeiro) 1985.

ix, 138 p. 29.7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 1985)

Tese - Universidade Federal do Rio de Janeiro, COPPE.

1. Teoria de Scheduling. I. COPPE / UFRJ II. Título (série)

À Ronilda,
Edgard e
Dino

AGRADECIMENTOS

Desejo agradecer ao professor Jayme Luiz Szwarcfiter pela orientação recebida e, de uma maneira especial, pela amizade desprendida, aos professores da COPPE/UFRJ pelos ensinamentos, aos colegas do Departamento de Produção da FEG/UNESP pelo incentivo e à Margarida Corrêa Leite pela paciência e eficiência com que deu forma final a este trabalho. Desejo agradecer também o apoio financeiro recebido através do PICD/CAPES.

Resumo da Tese Apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

CARACTERIZAÇÕES DE SOLUÇÕES ÓTIMAS ATRAVÉS DE ALGORITMOS POLINOMIAIS PARA PROBLEMAS EM SCHEDULING TIPO NO-WAIT FLOW-SHOP

Edgard Dias Batista Junior

DEZEMBRO DE 1985

Orientador: Jayme Luiz Szwarcfiter
Programa: Engenharia de Sistemas e Computação/COPPE/UFRJ

Este trabalho apresenta caracterizações de soluções ótimas através de algoritmos polinomiais para os seguintes problemas em scheduling tipo no-wait flow-shop: (1) problemas com processadores uniformes, isto é, o tempo de processamento é uma função linear da quantidade de processamento; (2) problemas com processadores de velocidade controlável, de forma a resultar tarefas encaixadas e (3) problemas com colisões ocorrendo apenas no primeiro ou no último processador. Propõe-se ainda uma extensão do conceito no-wait flow-shop e, a partir desta, apresenta-se algoritmos polinomiais para a obtenção de soluções ótimas para o problema com dois processadores e também para o problema com um número arbitrário de processadores, mas com tarefas pertencentes a um número limitado de classes.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

CHARACTERIZATIONS OF OPTIMAL SOLUTIONS OF POLYNOMIAL ALGORITHMS FOR PROBLEMS IN NO-WAIT FLOW-SHOP SCHEDULING

Edgard Dias Batista Junior

DECEMBER, 1985

Chairman: Jayme Luiz Szwarcfiter
Department: Engenharia de Sistemas e Computação/COPPE/UFRJ

This work presents characterizations of optimal solutions through polynomial algorithms for the following problems in no-wait flow-shop scheduling: (1) problems with uniform processors, that is, the processing time is a linear function of the processing quantity; (2) problems with processors of controlable velocity, in such a way as to result in boxed jobs and (3) problems with collisions occurring only in the first or in the last processor. Further, an extension of the concept no-wait flow-shop is proposed, and, beginning with this, polinomial algorithms are presented for the obtaining of the optimal solutions for the problem with two processors, and also for the problem with an arbitrary number of processors, but with jobs relating to a limited number of classes.

ÍNDICE

CAPITULO I

INTRODUÇÃO

O presente trabalho relaciona-se com uma das áreas da Otimização Combinatória, usualmente denominada Scheduling, que trata de problemas envolvendo um conjunto de tarefas a serem executadas através de um conjunto de processadores. De uma forma geral, a questão consiste em se determinar o processador que executará cada tarefa e quando, de maneira a satisfazer um determinado critério de otimização.

Não obstante, a evidente importância dessa classe de problemas, os primeiros resultados somente começaram a surgir na década de cinquenta, ainda assim para problemas de formulação bastante particular. A partir daí, mesmo com o esforço de um grande número de pesquisadores, praticamente inexistem algoritmos eficientes que resolvam problemas gerais; ou melhor, mostra-se comumente que os problemas gerais pertencem à classe dos problemas NP-Difíceis. Assim sendo, aparecem duas linhas de pesquisa na área: uma procurando desenvolver algoritmos aproximativos para problemas gerais e outra na direção de encontrar algoritmos eficientes para problemas particulares. O presente trabalho pertence a essa linha.

No capítulo II os conceitos gerais de Teoria de Scheduling são apresentados, especialmente aqueles relacionados aos problemas estudados neste trabalho, compreendendo sistemas de processamento em série, usualmente denominados flow-shop. Nos problemas em scheduling do tipo flow-shop, as tarefas são subdivididas em operações que são executadas, de modo sucessivo, por todos os processadores, sendo a ordem de execução nos processadores igual para todas as tarefas. Assim

sendo, tem sentido referir-se a ordem dos processadores. Nos problemas tipo flow-shop pode-se, ou não, permitir a existência de um tempo de espera entre o término de uma operação e o início da seguinte. Os problemas tratatos neste trabalho se referem diretamente ao caso de não se permitir tal espera, usualmente denominado no-wait. Nos problemas em scheduling tipo no-wait flow-shop, comumente são feitas referências à ordenação das operações das tarefas e também à sequência de tarefas a serem executadas.

No capítulo III apresenta-se uma revisão da bibliografia, ficando evidenciado o crescente interesse de desenvolvimento de pesquisas na área. Esta revisão bibliográfica, além de permitir o conhecimento do estado-da-arte, possibilitou a definição dos problemas estudados no presente trabalho.

No capítulo IV são tratados problemas em Scheduling do tipo no-wait flow-shop com processadores uniformes. Nestes problemas considera-se o tempo de processamento das operações uma função linear da quantidade de processamento. Considerando-se problemas com essas características, mostra-se que a condição necessária e suficiente para uma sequência de tarefas minimizar a data média de término é tal que as quantidades de processamento estejam em ordem não-decrescente, ou seja, a solução ótima pode ser obtida através de um algoritmo de complexidade O(nlogn).

No capítulo V trata-se de problemas em scheduling do tipo no-wait flow-shop com processadores de velocidade controlável. Nos problemas estudados neste capítulo a velocidade dos processadores pode ser controlada de maneira que o tempo de processamento da i-ésima operação de uma dada tarefa seja igual ao tempo de processamento da (i-1)-ésima operação da tarefa seguinte. Considerando-se problemas com essas características: (1) mostra-se que a condição necessária e suficiente para uma sequência de tarefas minimizar a máxima data de término é tal que a quantidade de processamento da última tarefa da sequência seja mínima

(complexidade O(n)); (2) apresenta-se condições de otimalidade relacionadas com a minimização da data média de término e (3) mostra-se que a condição necessária para uma sequência de tarefas minimizar o tempo total de processamento é tal que as quantidades de processamento estejam em ordem não-crescente, logo a solução ótima pode ser obtida em tempo O(nlogn).

No capítulo VI estuda-se problemas em scheduling do tipo no-wait flow-shop com colisões ocorrendo apenas nos processadores extremos, isto é, o instante de término da primeira ou da última operação de uma determinada tarefa coincide, respectivamente, com o instante de início da primeira ou da última operação da tarefa seguinte; nas demais operações tal coincidência não se verifica. Neste caso, apresenta-se condições necessárias para uma permutação de tarefas minimizar a máxima data de término e que podem ser reconhecidas em tempo O(nlogn).

No capítulo VII apresenta-se uma extensão do conceito no-wait flow-shop, na qual o instante de início de uma dada operação não necessariamente coincide com o instante de término da operação anterior. Mostra-se, inicialmente, que uma solução ótima para o problema com dois processadores, considerando-se a extensão apresentada, objetivando minimizar a máxima data de término, pode ser encontrada através de algoritmo devido a Gilmore e Gomory [1964], em tempo O(nlogn). Complementando, mostra-se que o problema com um número arbitrário de processadores, mas com as tarefas pertencentes a um número limitado de classes, pode ser resolvido em tempo polinomial, utilizando-se algoritmos desenvolvidos por Szwarcfiter [1984A].

Finalizando, no capítulo VIII aponta-se algumas linhas para pesquisas posteriores, tendo em vista os resultados obtidos neste trabalho. Adicionalmente, sugere-se novas direções interessantes de pesquisa em scheduling relacionadas com a análise de problemas com multi-critérios de otimização, a análise de sensibilidade, a consideração de recursos escassos e a utilização do enfoque estocástico.

CAPITULO II

TEORIA DE SCHEDULING

Apresenta-se neste capítulo os conceitos gerais da Teoria de Scheduling e de uma maneira especial, aqueles diretamente relacionados com os problemas estudados neste trabalho. Noções básicas da Teoria de Grafos e Teoria de Complexidade Computacional são consideradas conhecidas e a notação utilizada segue basicamente a de Szwarcfiter [1984].

II.1 - CONSIDERAÇÕES GERAIS

De uma maneira geral, a Teoria de Scheduling pode ser considerada a área da Otimização Combinatória relacionada com o estudo da seguinte classe de problemas:

Problema Geral em Scheduling

dados:
- um conjunto de tarefas a serem executadas dentro de um intervalo de tempo,
- um conjunto de processadores capaz de executá-las

questão : qual o processador que executará cada tarefa e quando cada tarefa será executada, de maneira a satisfazer um critério de otimização.

A classe de problemas com as características gerais acima é denominada scheduling de máquinas ou simplesmente scheduling. Dependendo de particularidades tais problemas podem pertencer a classes especiais, tais como:

- scheduling de projetos,
- balanceamento de linha de produção,
- programação de mão de obra,
- programação de horário,
- programação de frota.

Nos problemas da classe scheduling de projetos, também conhecidos por problemas de coordenação, existe um conjunto de tarefas a serem processadas dentro de um intervalo de tempo. Tais tarefas, normalmente denominadas atividades, devem respeitar, entre si, uma determinada relação de precedência. Pode, ou não, haver limitação dos recursos, no entanto, não é relevante identificar o processador que executará cada tarefa. Os métodos PERT e CPM são clássicos nessa classe de problemas, podendo-se também utilizar programação inteira, programação 0-1 ou programação dinâmica.

Na classe balanceamento de linha de produção, dado um conjunto de tarefas e suas características (tempos de processamento, relação de precedência, etc.), procura-se minimizar o número de processadores, ou então minimizar a variância entre o total de processamento dos diversos processadores.

A classe programação de mão-de-obra compreende os problemas relacionados com a elaboração de escalas de serviço, por exemplo, em hospitais, quartéis, etc..

Um típico problema da classe programação de horário compreende a própria confecção de horário de aulas. Normalmente, neste caso o período é uma semana, as classes podem ser consideradas como sendo os "processadores" e as aulas das diferentes disciplinas, como sendo as "tarefas".

A classe programação de frota compreende os problemas relacionados com a programação dos serviços de uma frota de veículos. Os itinerários e os horários da frota devem ser estabelecidos de maneira a satisfazer uma determinada função objetivo (critério de otimização), além de considerar

as restrições relativas à frota e à demanda.

O presente trabalho se relaciona com problemas pertencentes à classe geral, scheduling de máquinas.

II.2 - DEFINIÇÕES BÁSICAS

Considere um conjunto de tarefas, $J = \{J_1, \ldots, J_n\}$, a serem executadas por um conjunto de processadores, $M = \{M_1, \ldots, M_m\}$, dentro de um intervalo de tempo $T = [t_0, t_f]$, onde t_0 é o instante inicial e t_f o instante final. Normalmente, considera-se $t_0 = 0$.

Um schedule S pode ser definido através da função:

$S : (T \times M) \rightarrow J$. Usa-se o neologismo schedular para indicar a formação de um schedule.

Seja J_0 a tarefa vazia, isto é, a ausência de tarefa ou ociosidade.

Assim, por exemplo:

$S([t_1, t_2], M_i) = J_j$, significa que o processador M_i está operando a tarefa J_j no intervalo de tempo $[t_1, t_2]$, $t_0 \leq t_1 < t_2 \leq t_f$;

$S([t_3, t_4], M_i) = J_0$, significa que o processador M_i está desocupado no intervalo de tempo $[t_3, t_4]$..

Um sistema constituído de processadores e que tenha como entrada as tarefas a serem processadas e como saída as próprias tarefas já executadas é denominado sistema de processamento.

Dependendo das particularidades do problema, um schedule S pode ser denotado de maneira mas simplificada, assim:

- caso o intervalo de tempo, necessário para processar cada tarefa seja uma constante Δt e a execução de toda tarefa, após iniciada, não puder ser interrompida até sua conclusão, tem-se:

  $S(t_1, M_i) = J_j$, significa que o processador M_i executa completamente a tarefa J_j, no intervalo de tempo $[t_1, t_1 + \Delta t]$;

- caso a própria permutação de tarefas σ identificar o schedule S, tem-se: $S = (J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)})$.

  Por exemplo: num sistema de processamento composto de um único processador e as tarefas possuem as características do caso acima.

De um modo geral, um problema π pode ser definido pelo par $\pi(\delta,\sigma)$, onde δ é o conjunto de todos os possíveis dados e σ é a questão que se deseja responder, ou seja, o objetivo do problema. O conjunto de todas as possíveis soluções viáveis de um problema $\pi(\delta,\sigma)$ é denominado conjunto viável e denotado por Ω. Uma instância $\imath$ de um problema $\pi(\delta,\sigma)$ são dados específicos pertencentes ao conjunto δ. Pode-se utilizar a notação $\pi(\imath)$, para se referir a uma determinada instância $\imath$ aplicada ao problema $\pi(\delta,\sigma)$, $\imath \in \delta$.

Diz-se que $\pi(\delta,\sigma)$ é um problema em scheduling, quando o conjunto de dados δ se relacionar com as características dos processadores $M_i \in M$ e das tarefas $J_j \in J$ e a questão σ se relacionar com a obtenção de um schedule S. Um schedule S é dito viável quando $S \in \Omega$.

Um problema em scheduling, de acordo com seu objetivo, pode ser identificado com um dos seguintes problemas

gerais:

Problema II.1: *Decisão*

dados: - um conjunto J de tarefas a serem executadas,
- num período de tempo T, através de
- um conjunto M de processadores

questão: existe um schedule S : (TxM) → J viável?

Problema II.2: *Localização*

dados: - um conjunto J de tarefas a serem executadas,
- num período de tempo T, através de
- um conjunto M de processadores

questão: localize, se existir, um schedule S ∈ Ω, isto é, construa um schedule S viável.

Problema II.3: *Otimização*

dados: - um conjunto J de tarefas a serem executadas,
- num período de tempo T, através de
- um conjunto M de processadores

questão: encontre, se existir, um schedule S ótimo, isto é, um schedule S ∈ Ω que satisfaça um critério de otimização C.

Os problemas de decisão, localização e otimização, podem ser associados entre si e neste caso é fácil observar que o problema de decisão apresenta dificuldade não maior do que o de localização, o qual, por sua vez, apresenta dificuldade não maior do que o de otimização. No presente trabalho estuda-se problemas de otimização da classe scheduling de máquinas, portanto os mais gerais. Dentro dessa classe é possível identificar um grande número de tipos de problemas em função da variação das características das tarefas e dos processadores, além da variação dos critérios de otimização. De acordo

com Graham et alii [1979], tais problemas podem ser identificados através do terno ; $\alpha/\beta/\gamma$. Os campos α e β identificam, respectivamente, as características dos processadores e das tarefas. O critério de otimização é identificado através do campo γ. Nas seções que seguem apresenta-se uma descrição dessas características.

II.3 - CAMPO β : CARACTERÍSTICAS DAS TAREFAS

Considere um conjunto de tarefas, $J = \{J_1, \ldots, J_n\}$.

Uma tarefa $J_j \in J$ pode ser particionada em operações: $J_j = (O_{1j}, \ldots, O_{kj}, \ldots, O_{m_j\, j})$, onde O_{kj} é a k-ésima operação da tarefa J_j, também denominada operação de ordem k; $m_j \geq 1$ é o número de operações da tarefa J_j.

Diz-se que uma tarefa é simples se $m_j = 1$; caso contrário é denominada composta. Utiliza-se simplesmente a denominação tarefa, no caso de se referir genericamente às tarefas simples e compostas e também, em casos específicos, quando não houver possibilidade de confusão.

A cada tarefa J_j associa-se:

- uma quantidade de processamento q_j, não negativa, que é o total de trabalho necessário para que a tarefa J_j seja completamente executada;
- um tempo de processamento, não negativo, que é o tempo necessário para que a tarefa J_j seja completamente executada.

Enquanto que a quantidade de processamento q_j é considerada única para cada tarefa, o tempo de processamento pode variar, por exemplo, em função do processador que executará a tarefa. Utiliza-se a notação p_{ij} para identificar o tempo necessário para o processador M_i executar completamente a

tarefa J_j. O caso do tempo de processamento de uma tarefa simples, ou das operações de uma tarefa composta, ser igual para todos os processadores, pode ser denotado simplesmente por p_j.

Diz-se que uma operação O_{kj} é predecessora da operação O_{ij} quando para o início da execução de O_{ij} é necessário que a operação O_{kj} tenha sido completamente executada. Neste caso a operação O_{ij} é dita sucessora da operação O_{kj}.

Diz-se que uma operação O_{ij} está liberada para o processamento quando sua execução ainda não teve início e todas as suas operações predecessoras tiverem sido executadas.

Uma operação é denominada independente quando não possuir predecessor, nem sucessor.

Os conceitos acima (predecessor, sucessor, liberada e independente) também se aplicam para tarefas.

Diz-se que uma tarefa J_j está disponível (para o processamento) a partir do instante de sua chegada no sistema de processamento.

A seguir são apresentadas as principais características das tarefas e a notação utilizada para identificá-las no campo β.

II.3.1. Interruptabilidade

Dependendo da aplicação, uma tarefa pode, ou não, ter sua execução interrompida e reassumida posteriormente no mesmo ou em outro processador. Denota-se por pmtn ("preemption"), o caso da interrupção ser permitida. O caso de não se permitir a existência de um tempo de espera entre operações de uma tarefa composta é denominado por no-wait. Neste caso uma tarefa após iniciada é processada ininterruptamente até a sua

conclusão. Finalmente, quando nada se especificar no campo β, subentende-se que a interrupção é permitida apenas entre operações de uma tarefa composta ("non preemption").

II.3.2. Relação de Precedência

As tarefas podem ser independentes entre si ou então existir uma relação de precedência entre as mesmas. Quando esta relação de precedência for qualquer, denota-se por prec; quando puder ser representada por uma árvore orientada qualquer, denota-se por tree; no caso de uma arborescência por out-tree e no caso de uma anti-arborescência por in-tree. Denota-se por chain o caso da relação de precedência ser do tipo cadeia. Duas cadeias de tarefas são ditas paralelas quando não existir relação de precedência entre as cadeias. Um bloco de cadeias paralelas é um conjunto de cadeias paralelas com a primeira (última) tarefa de cada cadeia tendo o mesmo predecessor (sucessor). Um bloco pode ser reduzido numa cadeia que inicia com o predecessor comum, seguido sequencialmente de cada cadeia do bloco e termina com o sucessor comum. Uma relação de precedência é do tipo série-paralelo quando puder ser reduzida a uma cadeia pela aplicação iterativa do seguinte procedimento: "reduzir um bloco numa cadeia".

II.3.3. Data de Chegada

Seja r_j a data de chegada da tarefa J_j no sistema de processamento. Denota-se por r_j o caso de existir tarefa com data de chegada diferente do instante inicial, t_0. Quando nada se especificar, subentende-se o caso de todas as tarefas estarem disponíveis a partir do instante inicial ($r_j = t_0$, $j = 1,\ldots, n$).

II.3.4. Recursos Limitados

Evidentemente, a execução de uma tarefa deve sempre satisfazer a restrição da disponibilidade dos processadores. No entanto, dependendo das aplicações, outros recursos escassos necessitam ser considerados. A notação a seguir é baseada em Blazewicz et alii [1983].

Considere a existência de s recursos adicionais: $R_1, \ldots, R_s$ e para cada recurso R_h, $1 \leq h \leq s$, seja q_h a quantidade total desse recurso disponível a qualquer tempo.

No caso de tarefas simples, seja r_{hj} a quantidade (não negativa) do recurso R_h requerida pela tarefa J_j, durante todo o tempo de sua execução. Neste caso, um schedule é viável se para cada instante t, o conjunto das tarefas executadas em t, S_t, satisfizer:

$$\sum_{J_j \in S_t} r_{hj} \leq q_h \quad , \quad h = 1, \ldots, s. \qquad (II.1)$$

No caso de tarefas compostas, seja r_{hij} a quantidade (não negativa) do recurso R_h requerida pela operação O_{ij} da tarefa J_j durante todo o tempo de execução de O_{ij}. Obviamente, a condição de viabilidade para esse caso é análoga a acima.

A presença de recursos limitados é denotada no campo β por $\lambda\sigma\rho$, onde:

$$\lambda = \begin{cases} s, & \text{se o número de recursos s é fixo;} \\ \cdot, & \text{se o número de recursos adicionais é parte da entrada do problema, isto é, arbitrário.} \end{cases}$$

$$\sigma = \begin{cases} p, & \text{se as quantidades } q_h \text{ dos recursos } R_h,\ 1 \leq h \leq \ell \\ & \text{tem um limitante superior igual a p;} \\ \cdot, & \text{se } q_h \text{ é parte da entrada do problema.} \end{cases}$$

$$\rho = \begin{cases} r\ , & \text{se as quantidades } r_{hj}\ (r_{hij}) \text{ tem um limitante superior igual a } r; \\ \cdot\ , & \text{se as quantidades requeridas de cada recurso são arbitrárias.} \end{cases}$$

Pelo fato de $\sigma=p$ implicar em $\rho=r$, os casos possíveis se resumem nos seguintes: res s p r, res · p r, res s · r, res s · ·, res · · r e res ···.

II.4 - CAMPO α : CARACTERÍSTICAS DOS PROCESSADORES

Considere um sistema de processamento composto de um conjunto de processadores: $M = \{M_1, \ldots, M_m\}$.

A cada processador $M_i \in M$, associa-se uma velocidade de processamento, relacionada à quantidade de trabalho que o processador pode executar por unidade de tempo. Essa velocidade pode ser constante ou variável. No caso de cada processador possuir velocidade constante pode-se utilizar a notação v_i. A variação da velocidade de um processador M_i pode ser simplesmente função da tarefa J_j a ser processada, utilizando-se a notação v_{ij}, ou também função da tarefa J_k, imediatamente anterior à tarefa J_j, neste caso usa-se a notação v_{ikj}.

A seguir são apresentadas as principais características dos processadores e a respectiva notação a ser utilizada no campo α.

II.4.1. Processador Único

Neste caso o sistema de processamento é formado por um único processador, também denominado máquina simples e denotado pelo dígito 1.

II.4.2. Processadores em Paralelo

Diz-se que um sistema de processamento é formado de processadores em paralelo quando todas as tarefas são simples e qualquer processador pode executar qualquer tarefa.

Considere $J = \{J_1, \ldots, J_n\}$, um conjunto de tarefas e

$M = \{M_1, \ldots, M_m\}$, um conjunto de processadores.

Seja p_{ij}, o tempo de processamento da tarefa J_j no processador M_i.

Considere a velocidade de processamento, v_i de um processador M_i, $i = 1, \ldots, m$, igual a quantidade de processamento que M_i pode executar por unidade de tempo.

A seguir são apresentados alguns tipos especiais de processadores em paralelo.

II.4.2.1. Processadores Idênticos

É o caso de processadores em paralelo com velocidades iguais, isto é, os processadores são idênticos. Por outras palavras, o tempo de processamento de cada tarefa J_j, independe do processador que a executará: $p_{1j} = p_{2j} = \ldots = p_{mj}$. Neste caso, denotado por P, não é relevante a identificação do processador que executará cada tarefa. Quando o número de processadores é fixado, então este número deve aparecer no campo α. Assim, por exemplo, P3 significa que o sistema de processamento é composto de 3 processadores idênticos. Quando nada se especificar, subentende-se que o número de processadores é parte da entrada do problema. A consideração acima é válida para todos os tipos de processadores.

II.4.2.2. Processadores Uniformes

Diz-se que um sistema de processamento é formado de processadores uniformes, quando estes forem paralelos e a velocidade de cada processador é tal que o tempo de processamento, p_{ij}, para o processador M_i executar a tarefa J_j, é uma função linear da quantidade de processamento q_j:

$$p_{ij} = \alpha_i \cdot q_j + \beta_i \ , \quad \begin{matrix} i = 1, \ldots, m \\ j = 1, \ldots, n \end{matrix}$$

onde α_i e β_i são constantes de proporcionalidade.

Este caso, denotado por Q, é uma generalização do anterior, no qual as velocidades de todos os processadores são iguais, isto é, $\alpha_1 = \alpha_2 = \ldots = \alpha_m$ e

$$\beta_1 = \beta_2 = \ldots = \beta_m \ .$$

II.4.2.3. Processadores Ordenados

Neste caso, denotado por L, os processadores são paralelos e existe uma ordenação entre os tempos de processamento, satisfazendo:

(i) se $p_{ka} > p_{kb}$, para $J_a, J_b \in J$ e para algum $M_k \in M$

então $p_{ia} \geq p_{ib}$, para $\forall\ M_i \in M$;

(ii) se $p_{ra} > p_{sa}$, para algum $J_a \in J$ e para $M_r, M_s \in M$

então $p_{rj} \geq p_{sj}$, para $\forall\ J_j \in J$.

Evidentemente, processadores uniformes é um caso particular de processadores ordenados.

II.4.2.4. Processadores Não Relacionados

Diz-se que os processadores são não relacionados quando são paralelos mas não existe nenhuma ordenação entre as velocidades de processamento. Denota-se este caso por R.

II.4.3. Processadores em Série

Num sistema com processadores em série, também denominado flow-shop e denotado por F, as tarefas são compostas, existe uma relação de precedência entre as operações de cada tarefa, do tipo cadeia, e esta ordenação é a mesma para todas as tarefas. Por outras palavras, para todas as tarefas, as operações de mesma ordem devem ser executadas pelo mesmo processador. Assim, sem perda de generalidade, tem-se:

M_1 executa as operações O_{1j} , j = 1, ... , n

M_2 executa as operações O_{2j} , j = 1, ... , n

.
.
.

M_m executa as operações O_{mj} , j = 1, ... , n.

Os processadores M_1 e M_m são denominados processadores extremos.

Um estudo mais detalhado do tipo flow-shop é apresentado na seção II.8, por relacionar-se diretamente com os problemas estudados neste trabalho.

II.4.4. Processadores Tipo Job-Shop

Num sistema de processamento formado por processadores tipo job-shop, denotado por J, as operações de cada tarefa devem ser processadas numa dada ordem, mas esta ordenação não é a mesma para todas as tarefas.

II.4.5. Processadores Tipo Open-Shop

No caso de processadores tipo open-shop, denotado por O, todas as operações são independentes, isto é, não existe nunhema relação de precedência entre as operações de cada tarefa.

II.5 - CAMPO γ : OBJETIVO DOS PROBLEMAS EM SCHEDULING

O campo γ, que define o objetivo de um problema em scheduling, pode relacionar-se com as seguintes variáveis:

C_j - data de término da tarefa J_j ("completion time"), instante no qual a tarefa J_j é completada. Evidentemente, no caso flow-shop é o instante de término da m-ésima operação (última) da tarefa J_j.

L_j - tempo de atraso da tarefa J_j ("lateness"):

$L_j = C_j - d_j$,

onde d_j é a data prevista para o término da tarefa J_j ("due date"). É interessante observar que o valor de L_j pode ser negativo, significando neste caso que a tarefa J_j foi concluída antes da data prevista.

T_j - tempo de retardo ("tardiness") : $T_j = \max \{0, L_j\}$.

U_j - penalidade unitária ("unit penalty"): $U_j = \begin{cases} 0 \text{ se } C_j \leq d_j \\ 1 \text{ caso contrário} \end{cases}$,

F_j - tempo de permanência da tarefa J_j no sistema de processamento ("flow time") : $F_j = C_j - r_j$, onde r_j é a data de chegada.

P_j - tempo de processamento da tarefa J_j.

$C_{max}(S)$ - data de término do schedule S ("makespan"), isto é, instante de término da última tarefa a ser concluída em S: $C_{max} = \max \{C_j\}$, $j = 1, \ldots, n$.

A notação $C_{max}(S)$ pode ser simplificada para C_{max} quando não houver possibilidade de confusão. Utiliza-se a notação $C_{max}(\sigma)$, no caso do schedule S puder ser definido simplesmente através de uma permutação de tarefas : $S = (J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)})$. Denota-se por $C^k_{max}(\sigma)$, a data de término de um schedule considerando-se apenas k últimas tarefas da permutação σ.

O campo γ é formado por dois sub-campos $(\gamma_1 : \gamma_2)$. Sendo γ_1 utilizado para designar se o problema é de decisão, localização ou otimização, denotando-se por D, L e O, respectivamente. Utiliza-se o sub-campo γ_2 para especificar completamente o objetivo do problema. Quando não houver possibilidade de confusão o subcampo γ_1 pode ser omitido.

II.5.1. Problemas de Decisão e Localização

Um problema de decisão consiste em decidir se existe, ou não, um schedule que, além de viável, satisfaça o critério especificado por γ_2. Analogamente, um problema de localização consiste em construir, se possível, um schedule viável que satisfaça γ_2. Os seguintes critérios podem ser considerados:

- $C_{max} \leq \bar{D}$, significa que todas as tarefas do schedule devem possuir data de término menor ou igual a constante $\bar{D} > 0$.

- $C_j \leq d_j$, significa que cada tarefa do schedule deve terminar no máximo até a data prevista para seu término (d_j), não se admitindo atraso no término das tarefas.

II.5.2. Problemas de Otimização

Os problemas de otimização em scheduling utilizam, normalmente, critérios de minimização. Assim sendo, quando nada se especifica em γ_2 fica subentendido que o critério é de minimização. Apresenta-se a seguir os principais critérios:

- minimizar C_{max} , data máxima de término

- minimizar L_{max} , tempo de atraso máximo:

$$L_{max} = \max \{L_j\}$$

- minimizar T_{max} , tempo de retardo máximo:

$$T_{max} = \max \{T_j\}$$

- minimizar F_{max} , tempo de permanência máximo:

$$F_{max} = \max \{F_j\}$$

- minimizar ΣC_j , data média de término:

$$\Sigma C_j = \sum_{j=1}^{n} C_j / n$$

Utiliza-se também a notação: $\Sigma C(S)$ ou $\Sigma C(\sigma)$, ou simplesmente ΣC. Consideração análoga pode ser feita nos casos a seguir.

- minimizar ΣL_j , tempo médio de atraso:

$$\Sigma L_j = \sum_{j=1}^{n} L_j / n$$

• minimizar ΣT_j , tempo médio de retardo:

$$\Sigma \bar{T}_j = \sum_{j=1}^{n} \bar{T}_j / n$$

• minimizar ΣF_j , tempo médio de permanência:

$$\Sigma F_j = \sum_{j=1}^{n} F_j / n$$

• minimizar ΣU_j , número total de tarefas atrasadas:

$$\Sigma U_j = \sum_{j=1}^{n} U_j$$

• minimizar ΣP_j , tempo total de processamento:

$$\Sigma P_j = \sum_{j=1}^{n} P_j .$$

Associando-se um custo $w_j \geq 0$ à tarefa J_j, $j = 1, \ldots, n$, pode-se ainda, considerar os seguintes critérios de otimização:

• minimizar $\Sigma w_j C_j$, custo de término:

$$\Sigma w_j C_j = \sum_{j=1}^{n} w_j C_j$$

•• minimizar $\Sigma w_j L_j$, custo de atraso:

$$\Sigma w_j L_j = \sum_{j=1}^{n} w_j L_j$$

•• minimizar $\Sigma w_j T_j$, custo de retardo:

$$\Sigma w_j T_j = \sum_{j=1}^{n} w_j T_j$$

•• minimizar $\Sigma w_j F_j$, custo de permanência:

$$\Sigma w_j F_j = \sum_{j=1}^{n} w_j F_j$$

• minimizar $\Sigma w_j U_j$, custo por atrasos:

$$\Sigma w_j U_j = \sum_{j=1}^{n} w_j U_j$$

• minimizar $\Sigma w_j P_j$, custo do processamento:

$$\Sigma w_j P_j = \sum_{j=1}^{n} w_j P_j$$

Dado um problema de otimização em scheduling, A/B/C, um schedule viável, isto é, que considera todas as características A dos processadores e as características B das tarefas e também satisfaz o critério de otimização C, é denominado schedule ótimo e denotado por S*. O valor da função objetivo associada ao schedule S* é dito valor ótimo e denotado por s*. Utiliza-se também, a notação s* = min C.

Dois critérios de otimização, C' e C", são denominados equivalentes, denotando-se por C' ≃ C", quando diferirem apenas por uma constante.

Proposição II.1: As seguintes duplas de critérios de otimização, são equivalentes:

(a) $C' = \min \sum_{j=1}^{n} C_j$ e $C'' = \min \sum_{j=1}^{n} C_j / n$

(b) $C' = \min \sum_{j=1}^{n} L_j$ e $C'' = \min \sum_{j=1}^{n} L_j / n$

(c) $C' = \min \sum_{j=1}^{n} T_j$ e $C'' = \min \sum_{j=1}^{n} T_j / n$

(d) $C' = \min \sum_{j=1}^{n} F_j$ e $C'' = \min \sum_{j=1}^{n} F_j / n$

(e) $C' = \min \sum_{j=1}^{n} w_j C_j$ e $C'' = \min \sum_{j=1}^{n} w_j C_j / W$, $W = \sum_{j=1}^{n} w_j$

(f) $C' = \min \sum_{j=1}^{n} w_j L_j$ e $C'' = \min \sum_{j=1}^{n} w_j L_j / W$

(g) $C' = \min \sum_{j=1}^{n} w_j T_j$ e $C'' = \min \sum_{j=1}^{n} w_j T_j / W$

(h) $C' = \min \sum_{j=1}^{n} w_j F_j$ e $C'' = \min \Sigma w_j F_j / W$

(i) $C' = \min \sum_{j=1}^{n} w_j U_j$ e $C'' = \min \sum_{j=1}^{n} w_j U_j / W$

(j) $C' = \min \Sigma C_j$ e $C'' = \min \Sigma L_j$

(k) $C' = \min \Sigma C_j$ e $C'' = \min \Sigma F_j$

(l) $C' = \min \Sigma w_j C_j$ e $C'' = \min \Sigma w_j L_j$

(m) $C' = \min \Sigma w_j C_j$ e $C'' = \min \Sigma w_j F_j$

Prova:

(a) $C'' = \min \sum_{j=1}^{n} C_j / n = \frac{1}{n} \min \sum_{j=1}^{n} C_j = \frac{1}{n} C'$

($\frac{1}{n}$: constante)

Logo $C' \equiv C''$

(b) até (i) : análogas à prova (a)

(j) $C'' = \min \Sigma L_j = \min \sum_{j=1}^{n} (C_j - d_j) = \min \sum_{j=1}^{n} C_j - \min \sum_{j=1}^{n} d_j = C' - \sum_{j=1}^{n} d_j$

($\sum_{j=1}^{n} d_j$: constante)

Logo $C' \equiv C''$

(k) análoga à prova (j)

(l) $C'' = \min \sum_{j=1}^{n} w_j L_j = \min \sum_{j=1}^{n} w_j (C_j - d_j) =$

$= \min \sum_{j=1}^{n} w_j C_j - \min \sum_{j=1}^{n} w_j d_j$

$= C' - \sum_{j=1}^{n} w_j d_j$

($\sum_{j=1}^{n} w_j d_j$: constante)

(m) análoga à prova (l) Δ

Por uma questão de simplificação e considerando o resultado da Proposição II.1, as constantes (n e W) dos critérios de otimização podem ser desprezadas, mantendo-se no entanto as denominações.

II.6 - TEORIA DE COMPLEXIDADE COMPUTACIONAL EM SCHEDULING

Considere Π o conjunto de todos os problemas em scheduling. Dado um problema $\pi \in \Pi$, a preocupação natural que aparece é desenvolver um algoritmo eficiente (polinomial) que o resolva. Problemas em scheduling para os quais se conhece algoritmo eficiente, são denominados problemas polinomiais e o conjunto de tais problemas é denotado por Π^*. Dentro do conjunto dos problemas para os quais não se conhece algoritmo eficiente, existem problemas considerados intrinsicamente difíceis, denominados problemas NP-difíceis, abaixo definidos. O conjunto dos problemas NP-difíceis é denotado por $\Pi^!$. O conjunto $\Pi^? = \Pi - (\Pi^* \cup \Pi^!)$ é denominado conjunto dos problemas em aberto.

Um problema em scheduling $\pi_1(\delta_1, \sigma_1)$ é polinomialmente transformável ou redutível num problema $\pi_2(\delta_2, \sigma_2)$, denotando-se por $\pi_1 \to \pi_2$, quando existirem funções $f: \delta_1 \to \delta_2$ e $g: \Omega_2 \to \Omega_1$, tais que:

(i) f e g podem ser computadas em tempo polinomial e

(ii) para toda instância $\iota \in \delta_1$ tem-se:

$\pi_2(f(\iota))$ tem solução s_2 sse $\pi_1(\iota)$ tiver solução $s_1 = g(s_2)$.

Um problema π é NP-difícil, $\pi \in \Pi^!$, quando para todo $\pi' \in \Pi$ tem-se $\pi' \to \pi$.

Dois problemas, π_1 e π_2, são denominados equivalentes, denotando-se por $\pi_1 \approx \pi_2$, quando $\pi_1 \to \pi_2$ e $\pi_2 \to \pi_1$.

A relação $\rightarrow$ é uma ordenação parcial, com as seguintes propriedades:

(i) $\pi_i \rightarrow \pi_i$, para $\pi_i \in \Pi$ (propriedade reflexiva). Evidentemente, qualquer problema é redutível nele próprio;

(ii) $\pi_i \rightarrow \pi_j$ e $\pi_j \rightarrow \pi_i \Longrightarrow \pi_i \simeq \pi_j$, para Π_i, $\pi_j \in \Pi$ (propriedade anti-simétrica) e

(iii) $\pi_i \rightarrow \pi_j$ e $\pi_j \rightarrow \pi_k \Longrightarrow \pi_i \rightarrow \pi_k$, para π_i, π_j, $\pi_k \in \Pi$ (propriedade transitiva).

Considere dois problemas de otimização em scheduling: $\pi_1(\delta_1, \sigma_1)$ e $\pi_2(\delta_2, \sigma_2)$ e seja C_1 e C_2 os critérios de otimização de π_1 e π_2, respectivamente.

Proposição II.2: Se $\delta_1 = \delta_2$ e $C_1 \simeq C_2$ então $\pi_1 \simeq \pi_2$.

Prova: (i) $\pi_1 \rightarrow \pi_2$:

Como $\delta_1 = \delta_2$ então $f : \delta_1 \rightarrow \delta_2$, pode ser considerada a função identidade.
Sejam s_1 e s_2 soluções de π_1 e π_2, respectivamente: $s_1 = \min C_1$ e $s_2 = \min C_2$.
Como $C_1 \simeq C_2$ então, por definição, $C_2 = C_1 - k$, onde k é uma constante.
Logo: $s_2 = \min C_2 = \min(C_1 - k) = \min C_1 - k = s_1 - k$
Seja $g: \Omega_2 \rightarrow \Omega_1$ uma função tal que $g(s_2) = s_2 + k$, $s_2 \in \Omega_2$.
Logo $g(s_2) = s_1 - k + k = s_1$.
Então, por definição de transformação polinomial, tem-se $\pi_1 \rightarrow \pi_2$.

(ii) $\pi_2 \rightarrow \pi_1$: procedimento análogo a (i)

(iii) De (i) e (ii) tem-se $\pi_1 \simeq \pi_2$ Δ

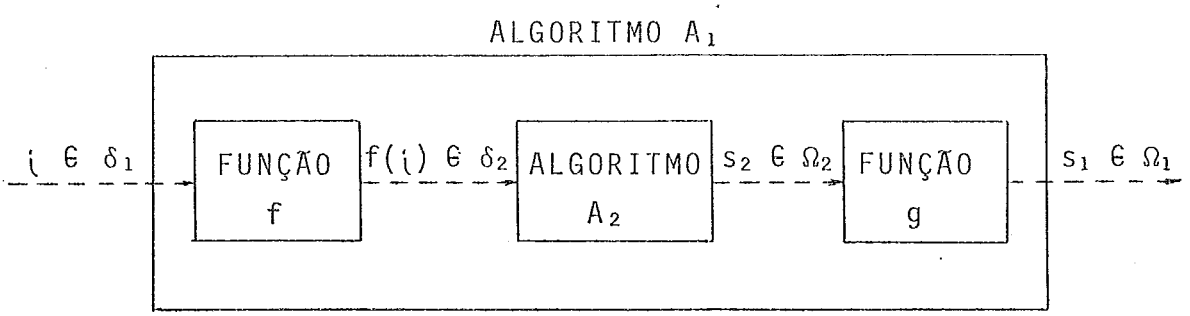
Proposição II.3: Se $\pi_1 \rightarrow \pi_2$ e $\pi_2 \in \Pi^*$ então $\pi_1 \in \Pi^*$.

Prova: Como $\pi_2 \in \Pi^*$ então existe um algoritmo eficiente, A_2, que resolve π_2.

Como $\pi_1 \rightarrow \pi_2$ então existem funções f e g que podem ser computadas em tempo polinomial de acordo com a definição de problema redutível.

Considere o seguinte algoritmo, A_1, para resolver o problema π_1:

- dada uma instância $i \in \delta_1$, aplique a função f obtendo-se $f(i) \in \delta_2$, em tempo polinomial;
- entrando-se com $f(i)$, no algoritmo A_2, determina-se s_2, em tempo polinomial;
- aplicando-se a função g, obtem-se $s_1 = g(s_2)$, também em tempo polinomial, que é uma solução para o problema π_1.



O Algoritmo A_1, claramente, é eficiente e portanto o problema $\pi_1 \in \Pi^*$ Δ

Proposição II.4: Se $\pi_1 \rightarrow \pi_2$ e $\pi_1 \in \Pi^!$ então $\pi_2 \in \Pi^!$

Prova: Como $\pi_1 \in \Pi^!$ então, por definição, para todo $\pi' \in \Pi$ tem-se $\pi' \rightarrow \pi_1$.

Como $\pi_1 \rightarrow \pi_2$ então, por transitividade, tem-se $\pi' \rightarrow \pi_2$, para todo $\pi' \in \Pi$.

Portanto $\pi_2 \in \Pi^!$ Δ

Diz-se que um problema $\pi_1(\delta_1, \sigma_1)$ é uma δ-restrição de um problema $\pi_2(\delta_2, \sigma_2)$ quando $\delta_1 \subset \delta_2$ e $\sigma_1 = \sigma_2$. Neste caso, π_2 é denominado δ-extensão de π_1.

Proposição II.5: Se um problema $\pi_1 \in \Pi^!$ é uma δ-restrição de um problema π_2, então $\pi_2 \in \Pi^!$.

Prova: Sejam f e g funções identidade então, diretamente a partir da definição de transformação polinomial e de δ-restrição, tem-se $\pi_1 \propto \pi_2$.

Utilizando-se o resultado da Proposição II.4 tem-se $\pi_2 \in \Pi^!$ Δ

Um problema $\pi_1(\delta_1, \sigma_1)$ é uma σ-restrição de um problema $\pi_2(\delta_2, \sigma_2)$ quando $\delta_1 = \delta_2$ e a questão σ_1 estiver contida na questão σ_2. Isto é, os conjuntos de dados, δ_1 e δ_2, são iguais e a resposta da questão σ_1 está contida na resposta da questão σ_2, ou ainda, a questão σ_1 é um caso particular da questão σ_2. O problema π_2 é denominado σ-extensão do problema π_1. Como exemplo, verifica-se, facilmente, que o Problema II.1, de decisão, é uma σ-restrição do Problema II.2, de localização, o qual por sua vez é uma σ-restrição do Problema II.3, de otimização.

Quando não houver possibilidade de confusão utiliza-se simplesmente a denominação restrição, tanto para δ-restrição, como para σ-restrição.

Proposição II.6: Se s_1 é solução de π_1 e π_1 é uma δ-extensão de π_2 então s_1 é solução de π_2.

Prova: Seja $\pi_1(\delta_1, \sigma_1)$ e $\pi_2(\delta_2, \sigma_2)$.
Por hipótese s_1 é solução de π_1 e π_2 δ-extensão de π_1.
Como, por definição, $\delta_2 \subset \delta_1$ e $\sigma_1 = \sigma_2$, segue que:
s_1 é solução de $\pi_1(\delta_2, \sigma_2) = \pi_2$ Δ

Proposição II.7: Se um problema $\pi_1 \in \Pi^!$ é uma σ-restrição de um problema π_2, então $\pi_2 \in \Pi^!$.

Prova: Segue diretamente da definição de σ-restrição Δ

Proposição II.8: O problema $\pi_1 = \cdot/\cdot/F_{max}$ é uma σ-restrição do problema $\pi_2 = \cdot/\cdot/L_{max}$.

Prova: (i) $\min L_{max} = \min(\max\{L_j | j = 1,\ldots, n\})$

$= \min(\max\{C_j - d_j | j = 1,\ldots, n\})$

Considerando $v_j = d_j - r_j$, tem-se:

$\min L_{max} = \min(\max\{C_j - r_j - v_j | j = 1,\ldots, n\})$

(ii) $\min F_{max} = \min(\max\{C_j - r_j | j = 1,\ldots, n\})$

(iii) De (i) e (ii) conclue-se que $\min F_{max}$ é um caso particular de $\min L_{max}$, para $v_j = 0$, $j = 1,\ldots, n$.

Portanto: π_1 é uma σ-restrição de π_2 Δ

Evidentemente, a cardinalidade dos conjuntos Π^*, $\Pi^?$ e $\Pi^!$ é um indicador do progresso da pesquisa sobre problemas em scheduling. Com a finalidade de orientar o desenvolvimento de futuras pesquisas, define-se os seguintes conjuntos de problemas em scheduling, de acordo com Lageweg et alii [1982]:

Π^*_{max} - conjunto dos problemas fáceis maximais:

$\Pi^*_{max} = \{\pi \in \Pi^* \mid \nexists\, \pi' \in (\Pi^* - \{\pi\}) : \pi \to \pi'\}$;

$\Pi^?_{min}$ - conjunto dos problemas em aberto minimais:

$\Pi^?_{min} = \{\pi \in \Pi^? \mid \nexists\, \pi' \in (\Pi^? - \{\pi\}) : \pi' \to \pi\}$;

$\Pi^?_{max}$ - conjunto dos problemas em aberto maximais:

$\Pi^?_{max} = \{\pi \in \Pi^? \mid \nexists\, \pi' \in (\Pi^? - \{\pi\}) : \pi \to \pi'\}$ e

$\Pi^{!}_{min}$ - conjunto dos problemas difíceis minimais:

$$\Pi^{!}_{min} = \{\pi \in \Pi^{!} \mid \nexists\ \pi' \in (\Pi^{!} - \{\pi\}) : \pi' \rightarrow \pi\},$$

Os problemas em Π^{*}_{max} e $\Pi^{!}_{min}$ representam os resultados essenciais, no sentido de que outros resultados conhecidos podem ser obtidos a partir destes, evocando a relação $\rightarrow$. Os problemas em $\Pi^{?}_{min}$ e $\Pi^{?}_{max}$ são os mais indicados para futuras pesquisas. A figura abaixo representa a cardinalidade dos conjuntos Π^{*}, Π^{*}_{max}, $\Pi^{?}$, $\Pi^{?}_{min}$, $\Pi^{?}_{max}$, $\Pi^{!}$ e $\Pi^{!}_{min}$, considerando os problemas codificados em binário, de acordo com Lageweg et alii [1982].

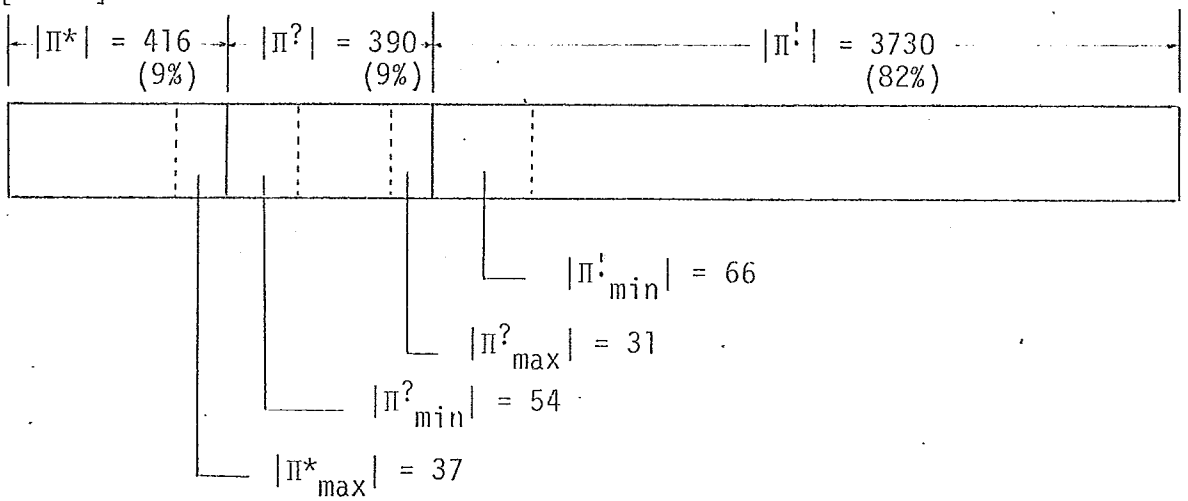


Fig. II.1: Cardinalidade dos conjuntos Π^{*}, Π^{*}_{max}, $\Pi^{?}$, $\Pi^{?}_{min}$, $\Pi^{?}_{max}$, $\Pi^{!}$ e $\Pi^{!}_{min}$

Considere um problema $\pi(\delta,\sigma)$. Seja A um algoritmo que resolve π. Considere todas as instâncias $\iota \in \delta$. Se a complexidade do algoritmo A é exponencial no caso da instância ι ser codificada em qualquer sistema de base maior ou igual a 2 e polinomial no caso codificada em unário, então o algoritmo A é denominado pseudo-polinomial.

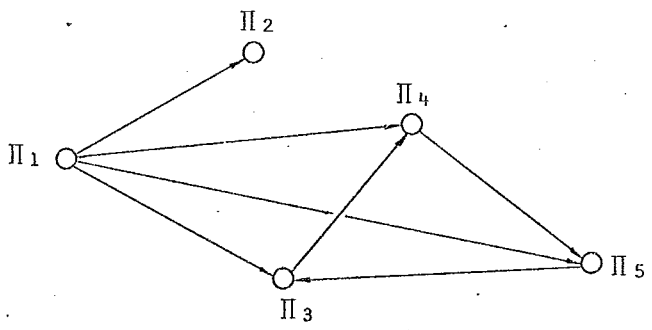
Um problema $\pi \in \Pi^{!}$ é dito fracamente NP-difícil quando existir um algoritmo pseudo-polinomial que resolve π.

Um problema $\pi \in \Pi'$ é denominado <u>fortemente NP-difícil</u> quando a existência de um algoritmo pseudo-polinomial que resolva π, implicar que qualquer problema em scheduling pertence à classe dos problemas polinomiais.

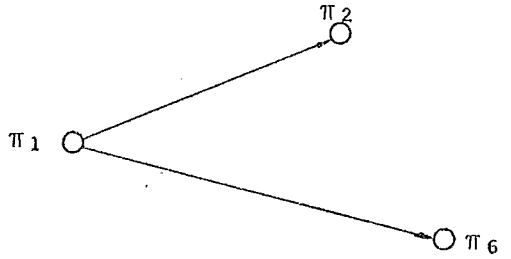
II.7 - REDUTIBILIDADE DE PROBLEMAS EM SCHEDULING

Considere um digrafo cujos vértices representam problemas $\pi \in \Pi$ e tal que para todo par de vértices π_i, π_j existe a aresta (π_i, π_j) sse $\pi_i \rightarrow \pi_j$. Este digrafo pode ser utilizado para mostrar a redutibilidade de problemas em scheduling. Os problemas pertencentes a uma mesma componente fortemente conexa de um digrafo, são equivalentes e neste caso, pode-se condensar o componente forte num único vértice. Para simplificar, a propriedade reflexiva da relação $\rightarrow$ pode não ser representada no digrafo, evitando-se assim, a formação de um laço em cada um de seus vértices.

<u>Exemplo II.1</u>: Dado: $\pi_1 \rightarrow \pi_2$, $\pi_1 \rightarrow \pi_3$, $\pi_1 \rightarrow \pi_4$, $\pi_3 \rightarrow \pi_4$, $\pi_4 \rightarrow \pi_5$ $\pi_5 \rightarrow \pi_3$, tem-se o seguinte digrafo:



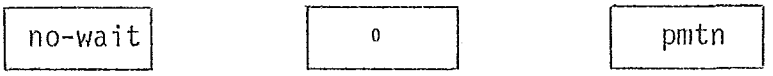
Como π_3, π_4 e π_5 são equivalentes, o digrafo pode ser condensado como segue:



$\pi_6 \approx \pi_3 \approx \pi_4 \approx \pi_5$ Δ

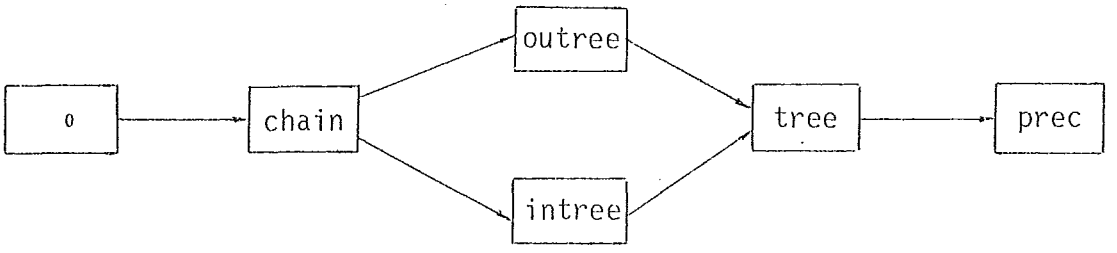
Os digrafos que seguem são utilizados para mostrar a redutibilidade entre problemas que, a menos da característica rotulada nos vértices, são idênticos (Graham et alii [1979]e Blazewicz et alii [1983]). O símbolo ° significa a negação da característica considerada.

G_1 : (interruptabilidade)



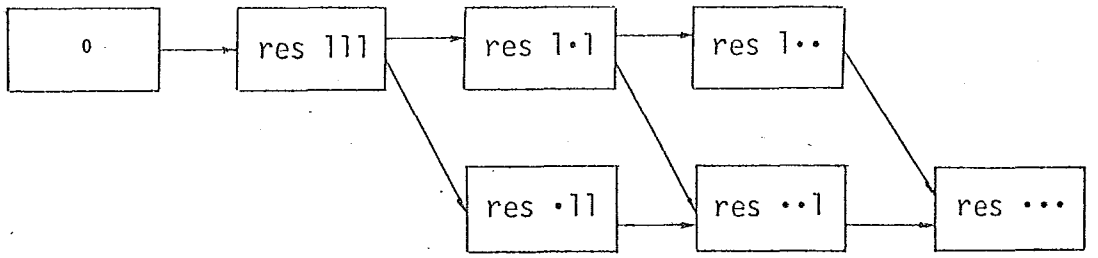
Observe que neste caso os problemas não são redutíveis.

G_2 : (relação de precedência)

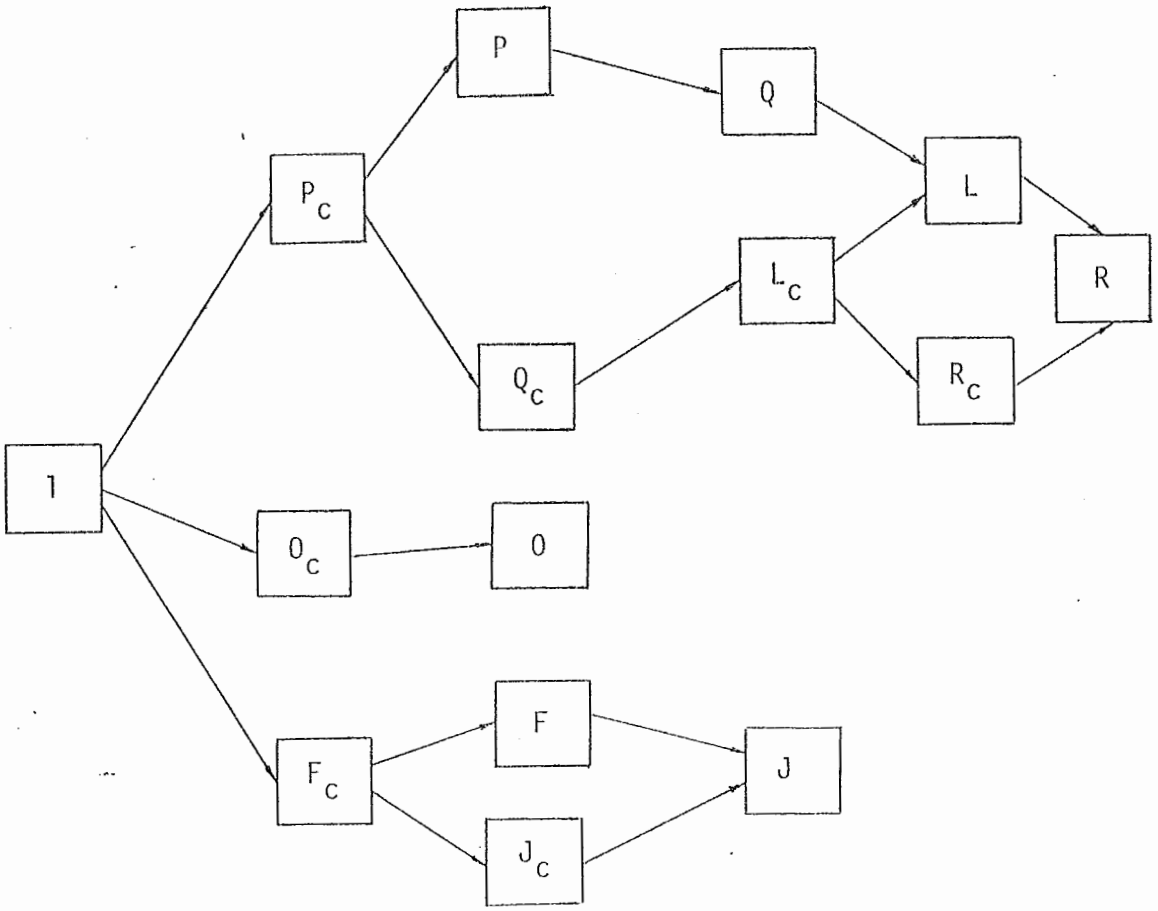


G_3 : (datas de chegada)

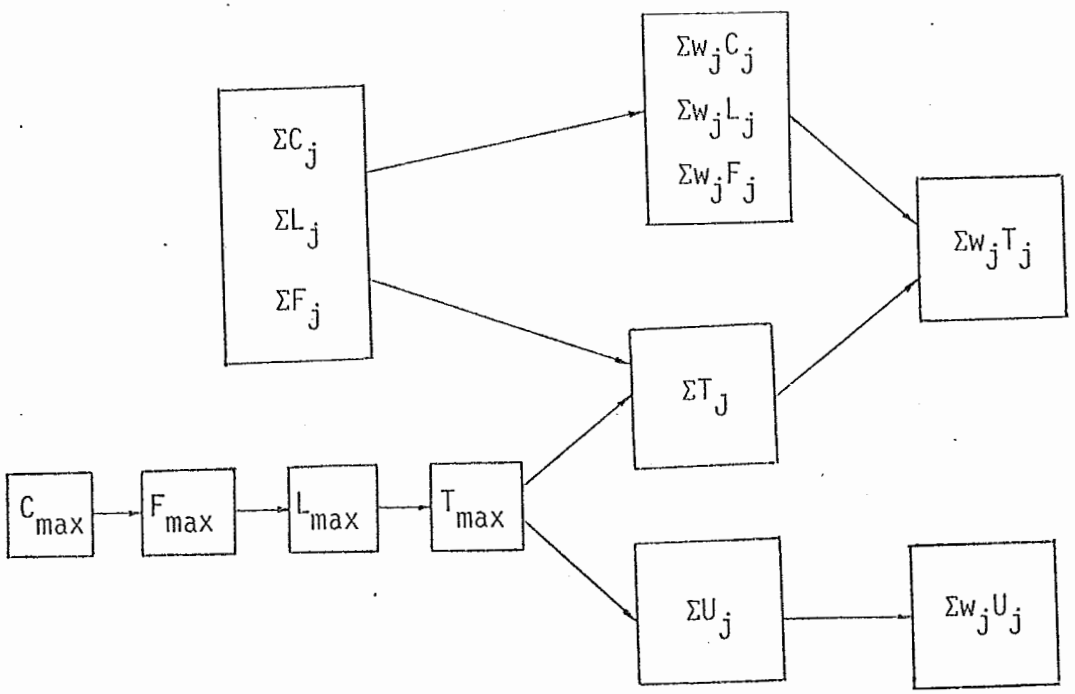
G_4 : (recursos limitados)



G_5 : (características dos processadores)



G_6 : (critério de otimização)



II.8 - FLOW SHOP SCHEDULING

II.8.1. Considerações Gerais

De acordo com o apresentado na seção II.4.3, nos problemas do tipo flow-shop:

- as tarefas são compostas: $J_j = (O_{1j}, \ldots, O_{ij}, \ldots, O_{mj})$, $j = 1, \ldots, n$
- e toda operação O_{ij}, $j = 1, \ldots, n$, é executada pelo processador M_i, $i = 1, \ldots, m$.

Logo existe uma associação biunívoca entre operações e processadores.

Normalmente, considera-se problemas do tipo flow-shop com as seguintes características:

- todas as tarefas J_j, $j = 1, \ldots, n$, são independentes e chegam simultaneamente no sistema de processamento, isto é, as tarefas estão liberadas a partir do instante inicial: $r_j = 0$, $j = 1, \ldots, n$;
- cada operação requer um processador diferente: se $O_{kj} \neq O_{hj}$ então $M_k \neq M_h$;
- diferentes operações de uma mesma tarefa, não podem ser executadas simultaneamente: se $S([t_1, t_2], M_k) = O_{kj}$ então $S([t_1, t_2], M_i) \neq O_{ij}$, para todo $i \neq k$;
- cada processador pode executar somente uma operação por vez: se $S([t_1, t_2], M_i) = O_{i\ell}$ então $S([t_1, t_2], M_i) \neq O_{ij}$, para todo $j \neq \ell$;
- o tempo de processamento, p_{ij}, da i-ésima operação da tarefa J_j (no processador M_i), é não negativo: $p_{ij} \geq 0$. Diz-se que a operação O_{ij} é vazia quando $p_{ij} = 0$;

- os tempos de preparação das operações são independentes da sequência de processamento das tarefas e por isso podem ser incluídos no tempo de processamento:

  p_{ij} = (tempo de preparação de O_{ij}) +

  (tempo de processamento, propriamente dito, de O_{ij} no processador M_i).

- não é permitida a interrupção da execução das operações, isto é, uma vez iniciada uma operação, ela é continuamente processada até o seu final:

  se $S([t_1, t_2], M_i) = O_{ij}$ então $p_{ij} = t_2 - t_1$;

- todos os m processadores estão continuamente em disponibilidade para o processamento. Na prática, essa característica representa uma simplificação da realidade, pois, evidentemente, em se tratando de máquinas, existe a necessidade de parada para manutenção e no caso de equipes formadas por pessoas, é evidente a necessidade de parada para descanso, a menos que se considere a existência de processadores de reserva. Um processador M_i é considerado ocioso quando existir operação liberada para o processamento em M_i e o processador M_i não estiver processando nenhuma operação.

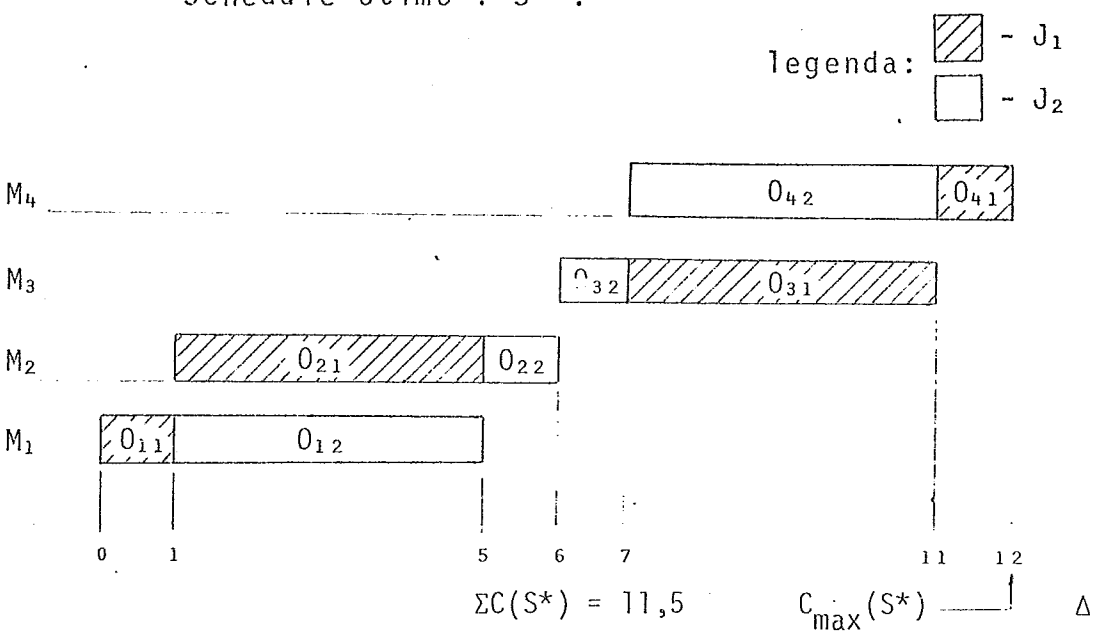
Considerando-se um sistema de processamento do tipo flow-shop com m processadores e n tarefas, é possível obter $(n!)^m$ schedules diferentes.

No exemplo II.2, devido a Baker [1974] é apresentado um schedule ótimo, com respeito à data máxima de término e também, à data média de término. É interessante observar que esse schedule ótimo mantém o processador M_3 ocioso no intervalo [5,6].

Exemplo II.2: tempos de processamento das operações

	J_1	J_2
M_4	1	4
M_3	4	1
M_2	4	1
M_1	1	4

Schedule ótimo : S* :



II.8.2. Permutation Flow-Shop

Pelo exposto, verifica-se a possibilidade da ocorrência de uma fila de operações liberadas $(O_{ig},\ldots,O_{ij},\ldots,O_{i\ell})$, aguardando o processamento em M_i, $i = 1,\ldots, m$. A disciplina do atendimento dessa fila de espera é uma característica importante da definição de um problema em flow-shop. Assim sendo, um problema é dito ser do tipo permutation flow-shop no caso da disciplina PEPS (primeira operação a entrar na fila em cada máquina, primeira a sair). Evidentemente, neste caso um schedule fica perfeitamente definido apenas com a identificação de uma permutação σ de tarefas:

$$S = (J_{\sigma(1)}, \ldots, J_{\sigma(n)}).$$

Um schedule com a característica acima é denominado de permutação. Obviamente, o schedule apresentado no Exemplo 2.2 não é um schedule de permutação.

II.8.3. Flow Shop com Processadores Uniformes e com Processadores de Velocidade Controlada

Considere um problema em scheduling do tipo flow-shop, definido como segue:

Seja: $J = \{J_1, \ldots, J_n\}$, conjunto de tarefas

q_j - quantidade de processamento constante para toda operação da tarefa J_j, $j = 1,\ldots, n$. Obviamente, a quantidade total de processamento da tarefa J_j é igual a $m \cdot q_j$, onde m é o número de processadores

p_{ij} - tempo de processamento da i-ésima operação da tarefa J_j

$M = \{M_1, \ldots, M_m\}$, conjunto de processadores uniformes, isto é:

$$p_{ij} = \alpha_i \cdot q_j + \beta_i \quad , \quad i = 1,\ldots, m \qquad j = 1,\ldots, n$$

α_i, β_i - constantes de proporcionalidade.

Um sistema de processamento com as características acima é denominado flow-shop com processadores uniformes e denotado por FQ.

Existem sistemas de processamento nos quais a velocidade dos processadores pode variar de maneira controlada. Um sistema de processamento do tipo flow-shop com essa característica é denotado por FC.

II.8.4. No-Wait Flow-Shop

Um caso particular de permutation flow-shop se relaciona com problemas nos quais não é permitida a formação de fila de espera de operações aguardando processamento. Neste caso, denominado no-wait flow-shop, uma vez iniciado o processamento da primeira operação de uma tarefa, as operações subsequentes são executadas sem interrupção, isto é, não se permite espera entre o término de uma operação e o início da operação seguinte.

Problemas do tipo no-wait flow-shop são encontrados nas siderúrgicas, metalúrgicas, indústrias químicas, hospitais, etc.. Nas siderúrgicas e metalúrgicas uma variável importante é a temperatura do metal. Este uma vez aquecido deve passar por todas as operações até o seu beneficiamento final, sem interrupção, pois caso contrário, se resfria, podendo dificultar ou mesmo inviabilizar as operações seguintes. Nas indústrias químicas, existe o problema do tempo associado às reações químicas. Assim sendo, uma espera entre uma operação e outra pode resultar num produto com características diferentes das desejadas. Com respeito aos hospitais, considere, por simplificação, um ato cirúrgico composto das seguintes atividades:

- providências pré-operatórias
- providências anestésicas
- operação propriamente dita
- providências pós-operatórias.

É razoável considerar, para a segurança do paciente, a inexistência de espera entre essas atividades. Neste caso, é interessante observar que a "tarefa" está relacionada com a operação cirúrgica em uma determinada pessoa e os "processadores" são as diferentes equipes profissionais, responsáveis pelas diversas atividades. Outra aplicação do conceito no-wait flow shop se refere a sistemas que processam produtos perecíveis.

Denomina-se intervalo entre duas tarefas consecutivas , $J_{\sigma(j-1)}$ e $J_{\sigma(j)}$, denotando-se por $d(\sigma(j-1), \sigma(j))$, o tempo decorrido entre o final do processamento da tarefa $J_{\sigma(j-1)}$ no processador M_m e o início do processamento da tarefa $J_{\sigma(j)}$, nesse mesmo processador.

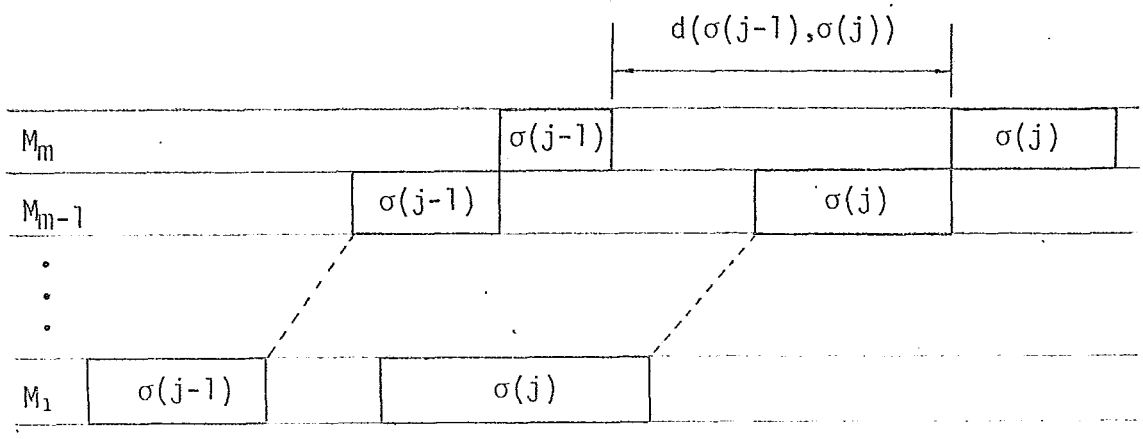


Fig. II.2 : $d(\sigma(j-1),\sigma(j))$

Evidentemente: $d(\sigma(j-1), \sigma(j)) \geq 0$ (II.3)

Seja: $\delta(\sigma(j-1), \sigma(j)) = d(\sigma(j-1), \sigma(j)) + p_{m,\sigma(j)}$ (II.4)

Por conveniência, considere:

$$d(\sigma(0), \sigma(1)) = \sum_{i=1}^{m-1} p_{i,\sigma(1)} \qquad (II.5)$$

onde $J_{\sigma(0)}$ pode ser considerada uma tarefa vazia.

Considere $\ell \in \{1, 2, \ldots, m-1\}$ tal que

$$\sum_{i=1}^{\ell} p_{m-i,\sigma(j)} - \sum_{i=1}^{\ell} p_{m-i+1,\sigma(j-1)} = \max_{k=1,\ldots,m-1} \{ \sum_{i=1}^{k} p_{m-i,\sigma(j)} - \sum_{i=1}^{k} p_{m-i+1,\sigma(j-1)} \} \qquad (II.6)$$

Assim sendo, tem-se:

$$d(\sigma(j-1),\sigma(j)) = \max_{1\leq k\leq m-1} \{0, \sum_{i=1}^{k} p_{m-i,\sigma(j)} - \sum_{i=1}^{k} p_{m-i+1,\sigma(j-1)}\} \qquad (II.7)$$

Se $d(\sigma(j-1),\sigma(j)) = \sum_{i=1}^{\ell} p_{m-i,\sigma(j)} - \sum_{i=1}^{\ell} p_{m-i+1,\sigma(j-1)} > 0$, diz-se que a colisão entre as tarefas $J_{\sigma(j-1)}$ e $J_{\sigma(j)}$ ocorre no processador $M_{m-\ell}$;

caso contrário ($d(\sigma(j-1),\sigma(j)) = 0$), a colisão ocorre no processador M_m.

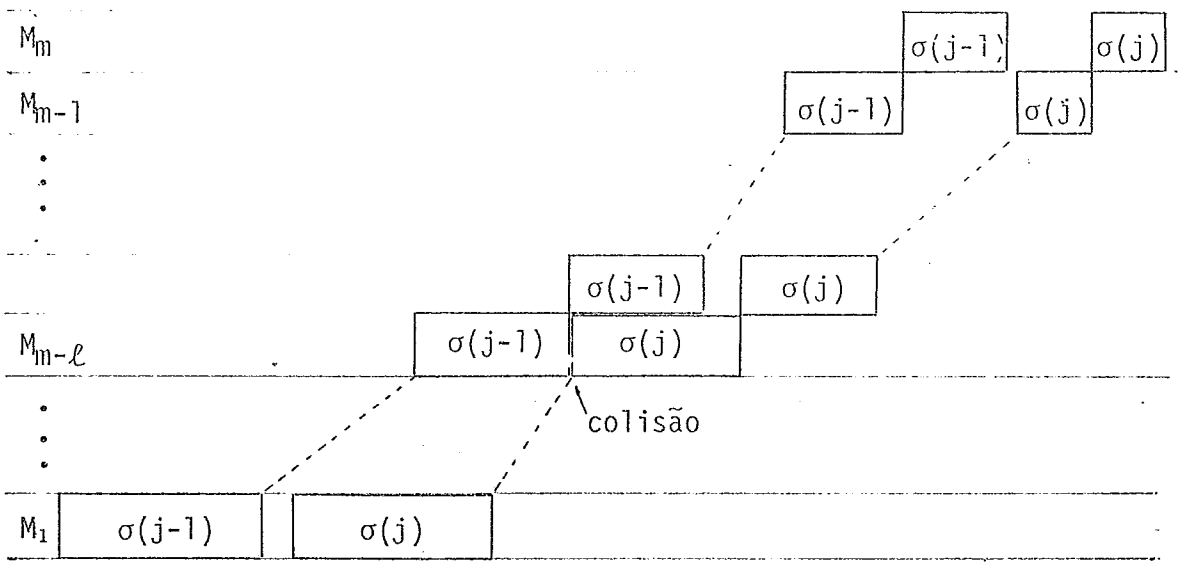


Fig. II.3 : Colisão entre as tarefas $J_{\sigma(j-1)}$ e $J_{\sigma(j)}$

Denomina-se distância entre duas tarefas, $J_{\sigma(j)}$ e $J_{\sigma(k)}$, denotando-se por $D(\sigma(j), \sigma(k))$, o tempo decorrido entre o início do processamento da tarefa $J_{\sigma(j)}$ no processador M_m e o término do processamento da tarefa $J_{\sigma(k)}$, nesse mesmo processador ($1 \leq j < k \leq n$).

Considere a seguinte notação:

$C^k_{max}(\sigma)$ - tempo gasto para processar exatamente as últimas k tarefas de uma permutação σ, isto é, processar a seguinte sequência de tarefas:

$$(J_{\sigma(n-k+1)} , J_{\sigma(n-k+2)} , \cdots , J_{\sigma(n)}) ;$$

$\text{Min } C^k_{max}$ - tempo mínimo para processar k tarefas, quaisquer.

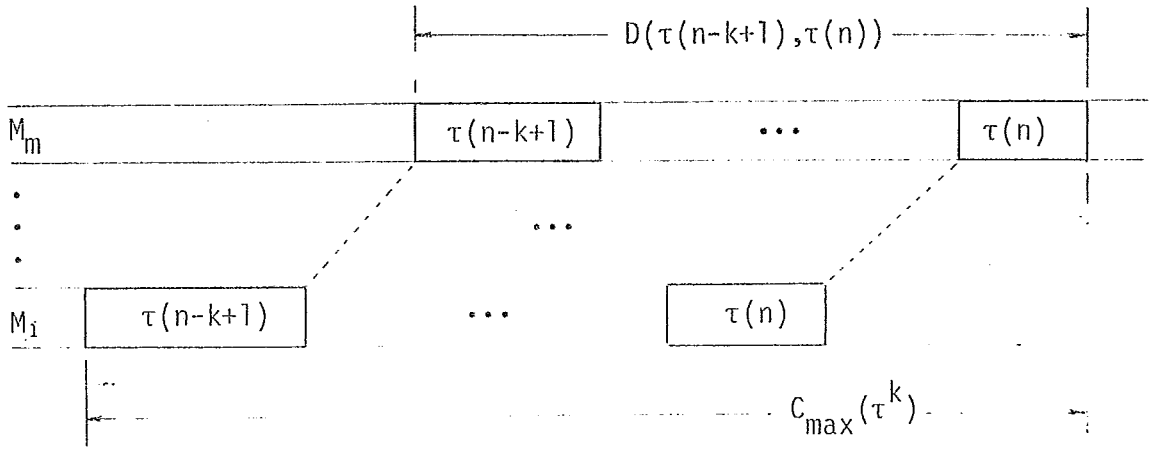


Fig. II.4 : $C^k_{max}(\tau)$, $D(\tau(n-k+1),\tau(n))$

Evidentemente:

$$C^n_{max}(\sigma) = C_{max}(\sigma) \qquad (II.8)$$

$$\text{Min } C^n_{max} = \text{Min } C_{max} \qquad (II.9)$$

No caso de processadores uniformes (seção II.8.3), tem-se:

$$C^k_{max}(\sigma) = q_{\sigma(n-k+1)} \cdot \sum_{i=1}^{m-1} \alpha_i + \sum_{i=1}^{m-1} \beta_i + D(\sigma(n-k+1), \sigma(n)) \qquad (II.10)$$

$$D(\sigma(n-k+1), \sigma(n)) = \alpha_m \cdot \sum_{j=n-k+1}^{n} q_{\sigma(j)} + \sum_{j=n-k+2}^{n} d(\sigma(j-1),\sigma(j)) + k \cdot \beta_m \qquad (II.11)$$

Considerando-se um sistema de processamento do tipo no-wait flow-shop, diz-se que duas tarefas, $J_{\sigma(j-1)}$ e $J_{\sigma(j)}$, são encaixadas quando:

$$p_{i,\sigma(j)} = p_{i+1,\sigma(j-1)} \quad , \quad \begin{matrix} i = 1,\ldots, m-1 \\ j = 2,\ldots, n \end{matrix} \qquad (II.12)$$

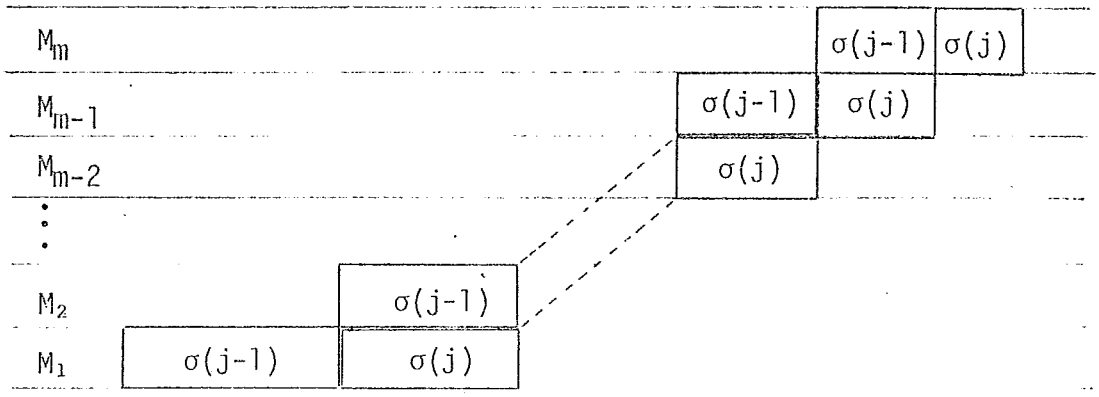


Fig. II.5 : Tarefas Encaixadas

Evidentemente, para tarefas encaixadas, tem-se:

$$d(\sigma(j-1),\sigma(j)) = 0 \qquad (II.13)$$

$$\sum_{i=1}^{k} p_{m-i,\sigma(j)} - \sum_{i=1}^{k} p_{m-i+1,\sigma(j-1)} = 0 \quad , \quad k = 1,\ldots, m-1 \qquad (II.14)$$

CAPITULO III

REVISÃO DA LITERATURA

III.1 - CONSIDERAÇÕES INICIAIS

Trabalhos pioneiros em Teoria de Scheduling, relacionados com máquinas simples, são devidos a Jackson [1955] e Smith [1956]. O primeiro mostra que um schedule com as datas previstas para o término das tarefas em ordem não decrescente é uma solução ótima para o problema $1//L_{max}$. Smith mostra que o problema $1//\Sigma w_j C_j$ também pode ser resolvido em tempo $O(n\log n)$, sequenciando as tarefas em ordem não decrescente da razão w_j/p_j. Lawler [1977] desenvolveu um algoritmo pseudo-polinomial, $O(n^4 \cdot \Sigma p_j)$, que resolve o problema $1//\Sigma T_j$. No entanto permanece em aberto se esse problema é ou não polinomial, sendo esse um dos poucos problemas em aberto com máquinas simples (Lenstra e Rinnooy-Kan [1984]).

Um dos primeiros resultados relacionados com processadores em paralelo deve-se a McNaughton [1959] que apresenta um algoritmo $O(n)$ para resolver o problema $P/pmtn//C_{max}$. A maioria dos problemas com processadores em paralelo pertencem à classe NP-Difícil, mesmo aqueles de formulação bastante simples, tais como: $P2//C_{max}$ e $P2//\Sigma w_j C_j$ (Bruno et alii [1974] e Lenstra et alii [1977]). O problema $P3/prec, p_j = 1/C_{max}$ tem permanecido na lista dos problemas em aberto por muitos anos (Lenstra e Rinnooy-Kan [1984]).

Existem dezenas de outros importantes trabalhos em Scheduling, como pode ser visto em Graham et alii [1979] e Lawler et alii [1982]. No entanto, de acordo com o escopo do presente trabalho, são referenciados a seguir apenas os prin

cipais trabalhos relacionados com problemas do tipo flow-shop e de uma maneira especial aqueles com a restrição no-wait.

III.2 - FLOW-SHOP SCHEDULING

Um resultado pioneiro e fundamental em flow-shop scheduling é devido a Johnson [1954], que apresenta um algoritmo O(nlogn) para resolver o problema F2// C_{max}. A existência, nesse problema, de um intervalo mínimo entre o início (fim) da primeira operação e o início (fim) da segunda operação é considerada nos trabalhos de Mitten [1958 e 1959], Johnson [1958], Nabeshima [1963], Szwarc [1968], Rinnooy-Kan [1976] e Kurisu [1976]. Nos trabalhos de Johnson [1954], Conway et alii [1967], Monma [1977 e 1979], Gonzales e Sahni [1978], Sidney [1979] e Szwarc [1981 e 1983] encontram-se generalizações adicionais do resultado de Johnson. Outro trabalho pioneiro em flow-shop scheduling deve-se a Heller [1960].

Sabe-se que os problemas gerais em flow-shop pertencem à classe NP-Difícil, e segundo Lawler [1983], mesmo casos particulares são extremamente difíceis de resolver. Garey e Johnson [1976], Garey et alii [1976], Lenstra et alii [1977] e Cho e Sahni [1981] mostram que os problemas F2/tree/C_{max}, F2/ /r_j/C_{max}, F2/pmtn, r_j/C_{max}, F2// ΣC_j, F2// L_{max}, F2/pmtn/L_{max}, F2// ΣT_j, F2// ΣU_j e F3// C_{max} pertencem à classe NP-Difícil. Em Blazewicz et alii [1983], a partir do problema 3-Partição, mostra-se que o problema F2/res 111/C_{max} é fortemente NP-Difícil. A complexidade dos problemas F2/pmtn, res 111/C_{max} e F2/ /pmtn/ΣC_j permanece em aberto. Estudos adicionais sobre complexidade de problemas em flow-shop são encontrados nos trabalhos de Garey e Johnson [1976 e 1979], Lenstra et alii [1977], Gonzales e Sahni [1978] e Achugbue e Chin [1982].

O problema F2/tree/C_{max}, com relação de precedência do tipo série paralelo pode ser resolvido por um algoritmo O(nlogn) devido a Sidney [1979]. Extensões desse resultado são encontradas nos trabalhos de Monma e Sidney [1977] e Monma [1979]. Chin e Tsai [1981] e Pinho [1983] mostram, de forma independente, que o problema F/p_j/C_{max} pode ser resolvido em tempo linear. Pinho [1983] apresenta também um algoritmo O(nlogn) para o problema F/p_j/ΣC_j.

A aplicação da técnica "branch-and-bound" na busca de solução para problemas em flow-shop encontra-se nos trabalhos de Ignall e Schrage [1965], Lomnicki [1965], Brown e Lomnicki [1966], McMahon e Burton [1967], McMahon [1969 e 1971], Ashour [1970], Szwarc [1971 e 1973], Krone e Steiglitz [1974], Dannenbring [1977], Lageweg et alii [1978], Kang e Kook[1981], Grabowski [1982] e Grabowski et alii [1983]. Nos trabalhos de Palmer [1965], Campbell et alii [1970], Gupta [1971 e 1972], Silver et alii [1980], Stinson e Smith [1982] e Enscore et alii [1983] utiliza-se o enfoque heurístico. Algoritmos aproximativos são encontrados nos trabalhos de Bárány [1981], Potts [1981] e Röck e Schmith [1983]. Revisões bibliográficas e estudos comparativos podem ser encontrados em Ashour [1970A], Baker [1975], Palma [1977], Graham et alii [1979] e Lenstra e Rinnooy-Kan [1984A]. Uma linha de pesquisa em recente desenvolvimento utiliza a abordagem estocástica, como nos trabalhos de Regendra-Prasad [1981], Pinedo [1982 e 1983], Brumelle et alii [1983], Forst [1983], Wermus [1983], Dempster et alii [1983], Frenk e Rinnooy-Kan [1984], Lenstra et alii [1984] e Möhring et alii [1984].

III.3 - NO-WAIT FLOW-SHOP SCHEDULING

O problema geral F/no-wait/C_{max} pertence à classe NP-Difícil podendo ser formulado em termos do bem conhe

cido problema do Caixeiro Viajante, como nos trabalhos de Piehler [1960], Reddi e Ramamoorthy [1972], Wismer [1972], Lenstra e Rinnooy-Kan [1975] e Lenstra et alii [1977]. Gilmore e Gomory [1964] apresenta um algoritmo polinomial para resolver o problema F2/no-wait/C_{max}. Algumas generalizações desse resultado são encontrados nos trabalhos de Reddi e Ramamoorthy [1972], Gupta [1976], Panwalkar e Woollan [1979] e Szwarc [1983A]. Papadimitriou e Kanellakis [1980] mostra que o problema F4/no-wait/C_{max} é NP-Difícil. A complexidade do problema F3/no-wait/C_{max} permaneceu na lista dos problemas em aberto durante vários anos, inclusive em Lenstra et alii [1977] estimula-se o desenvolvimento de pesquisa na área. Finalmente, Röck [1982 e 1984] mostra que o problema F3/no-wait/C_{max} é NP-Difícil.

Outro problema geral, F/no-wait/ΣC_j, também pertence à classe NP-Difícil, de acordo com Lenstra et alii [1977]. No trabalho de Van Deman e Baker [1974] esse problema é abordado utilizando-se a técnica "branch-and-bound". No trabalho de Röck [1984A] mostra-se que o problema F2/no-wait/ΣC_j é NP-Difícil, da mesma forma que os problemas F2/no-wait/L_{max}, F2/no-wait, r_j/C_{max}, F2/no-wait, $p_{ij} = 1$, res $2\cdot\cdot/C_{max}$, F2/no-wait, $p_{ij} = 1$, res $1\cdot\cdot/L_{max}$, F2/no-wait, $p_{ij} = 1$, res $1\cdot\cdot$, r_j/C_{max}, F2/no-wait, $p_{ij} = 1$, res $1\cdot\cdot/\Sigma C_j$. No trabalho de Blazewicz et alii [1985] prova-se que os problemas F2/no-wait, $p_{ij} = 1$, res $\cdot 11/C_{max}$ e F/no-wait, $p_{ij} = 1$, res $111/C_{max}$ pertencem à classe NP-Difícil. Röck [1984A] mostra que o problema F2/no-wait, $p_{ij} = 1$, res $1\cdot\cdot/C_{max}$ pode ser resolvido em tempo $O(n\log n)$, utilizando-se o algoritmo de Gilmore e Gomory [1964]. A complexidade do problema F2/no-wait, res $111/C_{max}$ permanece em aberto.

Resultados adicionais em no-wait flow-shop e extensões para job-shop e open-shop são encontrados nos trabalhos de Goyal [1973], Dutta e Cunningham [1975], Arora e Rana [1980], Panwalkar e Woollam [1980], Szwarc [1981A], Kalra e Bagga [1981], Bellman et alii [1982], Axsäter [1982], Pinho [1983] e Sahni e Cho [1983].

CAPITULO IV

PROBLEMAS EM SCHEDULING DO TIPO NO-WAIT FLOW-SHOP COM PROCESSADORES UNIFORMES E CONSTANTES DE PROPORCIONALIDADE NÃO-CRESCENTES

IV.1 - CONSIDERAÇÕES INICIAIS

Neste capítulo apresenta-se um estudo de problemas em scheduling do tipo no-wait flow-shop com processadores uniformes e constantes de proporcionalidade não-crescentes.

Considere um conjunto de processadores, $M = \{M_1, \ldots, M_m\}$ e um conjunto de tarefas, $J = \{J_1, \ldots, J_n\}$. O fato dos processadores serem uniformes significa que o tempo de processamento, p_{ij}, é uma função linear da quantidade de processamento, q_j, ou seja:

$$p_{ij} = \alpha_i \cdot q_j + \beta_i, \text{ para todo } M_i \in M \text{ e } J_j \in J. \qquad (IV.1)$$

Considere as constantes de proporcionalidade, α_i e β_i, satisfazendo as seguintes condições:

$$\alpha_i \geq \alpha_{i+1} > 0, \; i = 1, \ldots, m-1 \qquad (IV.2)$$

$$\beta_i \geq \beta_{i+1}, \quad i = 1, \ldots, m-1 \qquad (IV.3)$$

Na última seção deste capítulo, mostra-se que a condição necessária e suficiente para uma permutação de tarefas σ minimizar a data média de término de um problema com as características acima descritas, denotado por FQ/no-wait/ΣC, é tal que

$$q_{\sigma(j)} \leq q_{\sigma(j+1)}, \quad j = 1, \ldots, n-1 \qquad (IV.4)$$

A seção IV.2 relaciona-se com a determinação de $C_{max}(\sigma)$, $\Sigma C(\sigma)$ e $d(\sigma(j-1), \sigma(j))$ e a seguinte apresenta resultados relacionados com intervalos entre tarefas.

IV.2 - DETERMINAÇÃO DE $C_{max}(\sigma)$, $\Sigma C(\sigma)$ e $d(\sigma(j-1), \sigma(j))$

De acordo com o apresentado na seção II.8, todo schedule S relacionado com os problemas tratados neste capítulo fica perfeitamente definido pela simples apresentação de uma permutação σ de tarefas:

$$S(\sigma) = (J_{\sigma(1)}, \ldots, J_{\sigma(j)}, \ldots, J_{\sigma(n)}),$$

como mostra a figura a seguir.

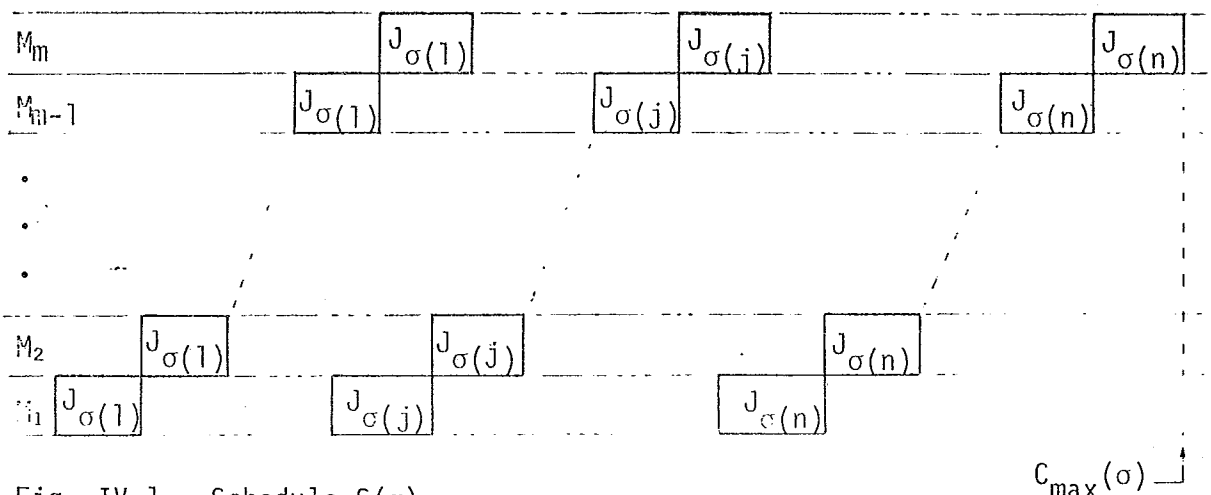


Fig. IV.1 - Schedule $S(\sigma)$

A partir da figura acima é imediato verificar que C_{max} é igual ao tempo total que um processador qualquer M_i esteve operando (OP_i), acrescido do tempo total que M_i esteve ocioso (OC_i) até a conclusão de todas as tarefas. Particularmente, para o processador M_m, tem-se:

$$C_{max}(\sigma) = OP_m + OC_m \qquad (IV.5)$$

Claramente:

$$OP_m = \sum_{j=1}^{n} p_{m,\sigma(j)} = \sum_{j=1}^{n} (\alpha_m \cdot q_{\sigma(j)} + \beta_m) = \alpha_m \cdot \sum_{j=1}^{n} q_{\sigma(j)} + n \cdot \beta_m =$$

$$= \alpha_m \sum_{j=1}^{n} q_j + n \cdot \beta_m \qquad (IV.6)$$

$$OC_m = \sum_{i=1}^{m-1} p_{i,\sigma(1)} + \sum_{j=2}^{n} d(\sigma(j-1), \sigma(j)) =$$

$$= q_{\sigma(1)} \cdot \sum_{i=1}^{m-1} \alpha_i + \sum_{i=1}^{m-1} \beta_i + \sum_{j=2}^{n} d(\sigma(j-1), \sigma(j)) \qquad (IV.7)$$

Substituindo-se os resultados (IV.6) e (IV.7) na equação (IV.5), tem-se:

$$C_{max}(\sigma) = \alpha_m \cdot \sum_{j=1}^{n} q_j + n \cdot \beta_m + q_{\sigma(1)} \cdot \sum_{i=1}^{m-1} \alpha_i + \sum_{i=1}^{m-1} \beta_i +$$

$$+ \sum_{j=2}^{n} d(\sigma(j-1), \sigma(j)) \qquad (IV.8)$$

Considere uma tarefa vazia, J_0, tal que $\sum_{i=1}^{m} p_{i,\sigma(0)} = 0$.

Logo: $q_{\sigma(0)} = - \sum_{i=1}^{m} \beta_i \Big/ \sum_{i=1}^{m} \alpha_i \qquad (IV.9)$

O intervalo entre $J_{\sigma(0)}$ e $J_{\sigma(1)}$ é dado por:

$$d(\sigma(0), \sigma(1)) = q_{\sigma(1)} \cdot \sum_{i=1}^{m-1} \alpha_i + \sum_{i=1}^{m-1} \beta_i \qquad (IV.10)$$

Substituindo-se (IV.10) em (IV.8), tem-se:

$$C_{max}(\sigma) = \alpha_m \cdot \sum_{j=1}^{n} q_j + n \cdot \beta_m + \sum_{j=1}^{n} d(\sigma(j-1), \sigma(j)) \qquad (IV.11)$$

Evidentemente, $C_{\sigma(n)} = C_{max}(\sigma) \qquad (IV.12)$

De uma forma geral, a data de término $C_{\sigma(k)}$, pode ser determinada como segue:

$$C_{\sigma(k)} = \alpha_m \cdot \sum_{j=1}^{k} q_{\sigma(j)} + k \cdot \beta_m + \sum_{j=1}^{k} d(\sigma(j-1), \sigma(j)),$$

$$k = 1, \ldots, n \qquad (IV.13)$$

A determinação da data média de término, $\Sigma C(\sigma)$, pode ser feita diretamente a partir de (IV.13):

$$\Sigma C(\sigma) = \sum_{k=1}^{n} (\alpha_m \cdot \sum_{j=1}^{k} q_{\sigma(j)} + k \cdot \beta_m + \sum_{j=1}^{k} d(\sigma(j-1), \sigma(j))) \qquad (IV.14)$$

$$\therefore \Sigma C(\sigma) = \alpha_m \cdot \sum_{k=1}^{n} \sum_{j=1}^{k} q_{\sigma(j)} + \beta_m \cdot \sum_{k=1}^{n} k +$$

$$+ \sum_{k=1}^{n} \sum_{j=1}^{k} d(\sigma(j-1), \sigma(j)) \qquad (IV.15)$$

Analisando-se as equações (IV.13) e (IV.15) verifica-se que a determinação de $C_{max}(\sigma)$ e $\Sigma C(\sigma)$ depende do cálculo de $d(\sigma(j-1), \sigma(j))$, $j = 1, \ldots, n$, o qual pode ser feito considerando-se os seguintes casos:

Caso IV.1: $q_{\sigma(j-1)} \leq q_{\sigma(j)}$

O Lema IV.1 a seguir, garante que o presente caso pode ser representado como segue:

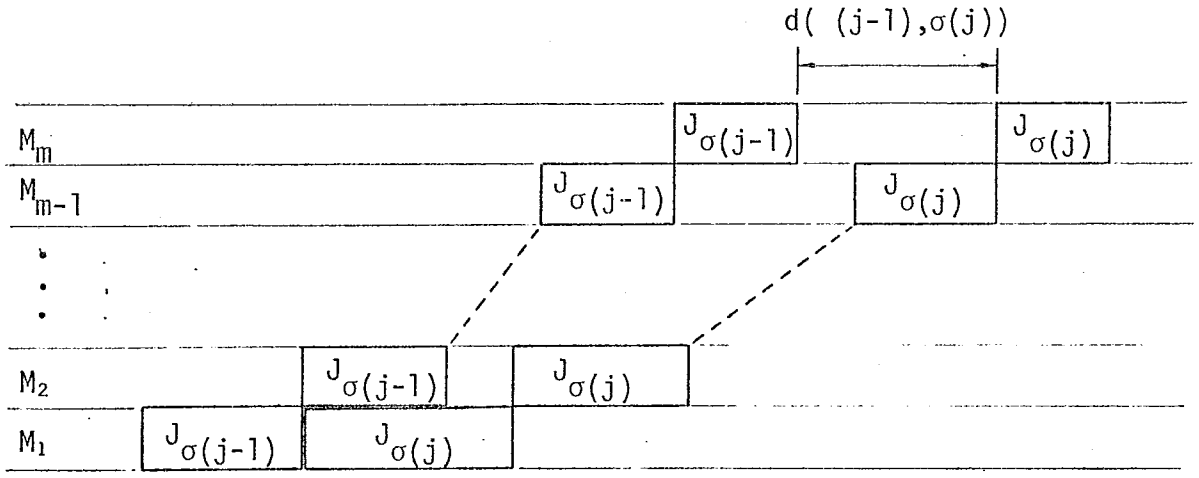


Fig. IV.3 : Caso IV.1

Lema IV.1: Se $q_{\sigma(j-1)} \leq q_{\sigma(j)}$

então $p_{i+1,\,\sigma(j-1)} \leq p_{i,\,\sigma(j)}$, $i = 1, \ldots, m-1$

prova:

De (IV.2) : $\alpha_i \geq \alpha_{i+1} > 0$, $i = 1, \ldots, m-1$ (i)

Por hipótese: $q_{\sigma(j)} \geq q_{\sigma(j-1)}$ (ii)

De (i) e (ii):

$$\alpha_i \cdot q_{\sigma(j)} - \alpha_{i+1} \cdot q_{\sigma(j-1)} \geq 0, \quad i = 1,\ldots, m-1 \qquad \text{(iii)}$$

De (IV.3)

$$\beta_i - \beta_{i+1} \geq 0, \quad i = 1, \ldots, m-1 \qquad \text{(iv)}$$

Somando-se (iii) e (iv):

$$\alpha_{i+1} \cdot q_{\sigma(j-1)} + \beta_{i+1} \leq \alpha_i \cdot q_{\sigma(j)} + \beta_i, \quad i = 1,\ldots, m-1$$

$$\therefore\ p_{i+1,\sigma(j-1)} \leq p_{i,\sigma(j)}\ , \quad i = 1, \ldots, m-1 \quad \Delta$$

A Fig. IV.3 permite verificar que a determinação de $d(\sigma(j-1), \sigma(j))$ para o caso $q_{\sigma(j-1)} \leq q_{\sigma(j)}$, pode ser feita como segue:

$$d(\sigma(j-1), \sigma(j)) = \sum_{i=1}^{m-1} p_{i,\sigma(j)} - \sum_{i=2}^{m} p_{i,\sigma(j-1)} \qquad \text{(IV.16)}$$

A seguir são apresentadas algumas expressões alternativas:

$$d(\sigma(j-1),\sigma(j)) = \sum_{i=1}^{m-1} p_{m-i,\sigma(j)} - \sum_{i=1}^{m-1} p_{m-i+1,\sigma(j-1)} \qquad \text{(IV.17)}$$

$$= q_{\sigma(j)} \cdot \sum_{i=1}^{m-1} \alpha_i - q_{\sigma(j-1)} \cdot \sum_{i=2}^{m} \alpha_i + \beta_1 - \beta_m \qquad \text{(IV.18)}$$

$$d(\sigma(j-1), \sigma(j)) = (q_{\sigma(j)} - q_{\sigma(j-1)}) \cdot \sum_{i=1}^{m} \alpha_i + \alpha_1 \cdot q_{\sigma(j-1)} +$$

$$+ \beta_1 - \alpha_m \cdot q_{\sigma(j)} - \beta_m \qquad \text{(IV.19)}$$

É imediato verificar que neste caso a colisão ocorre no primeiro processador, isto é:

$$\sum_{i=1}^{m-1} p_{m-i,\sigma(j)} - \sum_{i=1}^{m-1} p_{m-i+1,\sigma(j-1)} =$$

$$= \max_{1 \le k \le m-1} \{ \sum_{i=1}^{k} p_{m-i,\sigma(j)} - \sum_{i=1}^{k} p_{m-i+1,\sigma(j-1)} \} \qquad \text{(IV.20)}$$

CASO IV.2: $q_{\sigma(j-1)} > q_{\sigma(j)}$ e

$$\sum_{i=1}^{k} p_{m-i,\sigma(j)} < \sum_{i=1}^{k} p_{m-i+1,\sigma(j-1)} , \quad k=1, \ldots , m-1$$

Este caso pode ser visualizado através da seguinte figura:

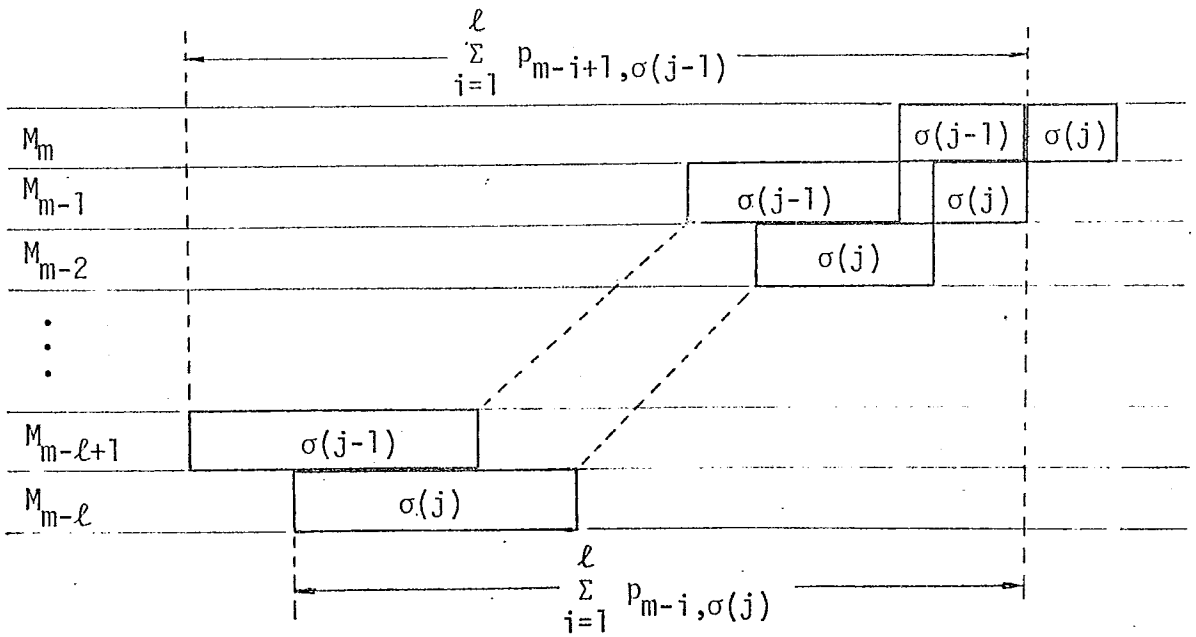


Fig. IV.4 : Caso IV.2

Segue diretamente da figura IV.4 que:

$$(i)\quad d(\sigma(j-1),\ \sigma(j)) = 0 \qquad \text{(IV.21)}$$

$$(ii)\quad \sum_{i=1}^{k} p_{m-i,\sigma(j)} - \sum_{i=1}^{k} p_{m-i+1,\sigma(j-1)} \leq 0,\quad k = 1,\ldots,\ m-1 \qquad \text{(IV.22)}$$

CASO IV.3: $q_{\sigma(j-1)} > q_{\sigma(j)}$ e

$$\sum_{i=1}^{k} p_{m-i,\sigma(j)} = \sum_{i=1}^{k} p_{m-i+1,\sigma(j-1)}\ ,\quad k = 1,\ldots,\ m-1$$

Evidentemente, tem-se:

$$p_{i,\sigma(j)} = p_{i+1,\sigma(j-1)}\ ,\quad i = 1,\ldots,\ m-1\ ;$$

(tarefas encaixadas)

$$d(\sigma(j-1),\ \sigma(j)) = 0 \qquad \text{(IV.23)}$$

No presente caso os coeficientes de proporcionalidade devem satisfazer o seguinte lema:

Lema IV.2: Duas tarefas, $J_{\sigma(j-1)}$ e $J_{\sigma(j)}$, satisfazem as condições do Caso IV.3 sse

$$\alpha_{i+1} = \alpha_1 \cdot (q_{\sigma(j)} / q_{\sigma(j-1)})^{i} +$$

$$+ \sum_{k=1}^{i} (\beta_k - \beta_{k+1}) \cdot (q_{\sigma(j)})^{i-k} \cdot (q_{\sigma(j-1)})^{-i+k-1}\ ,$$

$$i = 1,\ \ldots\ ,\ m-1$$

prova:

(=>) condição necessária, por indução:

(a) Para i=1:

Por hipótese: $p_{1,\sigma(j)} = p_{2,\sigma(j-1)}$

$\therefore \quad \alpha_1 \cdot q_{\sigma(j)} + \beta_1 = \alpha_2 \cdot q_{\sigma(j-1)} + \beta_2$

$\therefore \quad \alpha_2 = \alpha_1 \cdot (q_{\sigma(j)} / q_{\sigma(j-1)}) + (\beta_1 - \beta_2) / q_{\sigma(j-1)}$

(b) Hipótese de indução: supor válido para i-1, $i < m - 1$:

$$\alpha_i = \alpha_1 \cdot (q_{\sigma(j)} / q_{\sigma(j-1)})^{i-1} + \sum_{k=1}^{i-1} (\beta_k - \beta_{k+1}) \cdot (q_{\sigma(j)})^{i-k-1} \cdot (q_{\sigma(j-1)})^{-i+k}$$

(c) Para i, $i \leq m - 1$:

Por hipótese: $p_{i,\sigma(j)} = p_{i+1,\sigma(j-1)}$

$\therefore \quad \alpha_i \cdot q_{\sigma(j)} + \beta_i = \alpha_{i+1} \cdot q_{\sigma(j-1)} + \beta_{i+1}$

$\therefore \quad \alpha_{i+1} = \alpha_i \cdot (q_{\sigma(j)} / q_{\sigma(j-1)}) + (\beta_i - \beta_{i+1}) / q_{\sigma(j-1)}$

Substituindo-se α_i (hipótese de indução), tem-se:

$$\alpha_{i+1} = [\alpha_1 \cdot (q_{\sigma(j)} / q_{\sigma(j-1)})^{i-1} + \sum_{k=1}^{i-1} (\beta_k - \beta_{k+1}) \cdot (q_{\sigma(j)})^{i-k-1} \cdot (q_{\sigma(j-1)})^{-i+k}] \cdot (q_{\sigma(j)} / q_{\sigma(j-1)}) + (\beta_i - \beta_{i+1}) / q_{\sigma(j-1)}$$

$$\therefore \quad \alpha_{i+1} = \alpha_1 \cdot (q_{\sigma(j)} / q_{\sigma(j-1)})^{i} + \sum_{k=1}^{i} (\beta_k - \beta_{k+1}) \cdot (q_{\sigma(j)})^{i-k} \cdot (q_{\sigma(j-1)})^{-i+k-1}$$

(<=) condição suficiente:

Por hipótese, para $1 \leq i \leq m-1$, tem-se:

$$\alpha_i = \alpha_1 \cdot (q_{\sigma(j)} / q_{\sigma(j-1)})^{i-1} + \sum_{k=1}^{i-1} (\beta_k - \beta_{k+1}) \cdot (q_{\sigma(j)})^{i-1-k} \cdot (q_{\sigma(j-1)})^{-i+k}$$

$$\therefore \alpha_i \cdot q_{\sigma(j)} = (q_{\sigma(j)} / q_{\sigma(j-1)})^{i-1} \cdot (\alpha_1 \cdot q_{\sigma(j)} + \sum_{k=1}^{i-1} (\beta_k - \beta_{k+1}) \cdot (q_{\sigma(j)})^{1-k} \cdot (q_{\sigma(j-1)})^{-1+k}) \quad (i)$$

$$\alpha_{i+1} = \alpha_1 \cdot (q_{\sigma(j)} / q_{\sigma(j-1)})^{i} + \sum_{k=1}^{i} (\beta_k - \beta_{k+1}) \cdot (q_{\sigma(j)})^{i-k} \cdot (q_{\sigma(j-1)})^{-i+k-1}$$

$$\therefore \alpha_{i+1} \cdot q_{\sigma(j-1)} = (q_{\sigma(j)} / q_{\sigma(j-1)})^{i-1} \cdot (\alpha_1 \cdot q_{\sigma(j)} + \sum_{k=1}^{i-1} (\beta_k - \beta_{k+1}) \cdot (q_{\sigma(j)})^{1-k} \cdot (q_{\sigma(j-1)})^{-1+k}) + \beta_i - \beta_{i+1} \quad (ii)$$

Substituindo-se (i) em (ii):

$$\alpha_{i+1} \cdot q_{\sigma(j-1)} = \alpha_i \cdot q_{\sigma(j)} + \beta_i - \beta_{i+1}$$

Portanto:

$$p_{i,\sigma(j)} = p_{i+1,\sigma(j-1)}, \quad i = 1, \ldots, m-1 \qquad \Delta$$

CASO IV.4: $q_{\sigma(j-1)} > q_{\sigma(j)}$ e

existe $\ell \in \{1,\ldots, m-1\}$ tal que

$$\sum_{i=1}^{\ell} p_{m-i+1,\sigma(j-1)} < \sum_{i=1}^{\ell} p_{m-i,\sigma(j)}$$

Seja $r \in \{1,\ldots, m-1\}$, índice tal que:

$$\sum_{i=1}^{r} p_{m-i,\sigma(j)} - \sum_{i=1}^{r} p_{m-i+1,\sigma(j-1)} = \max_{1\leq k\leq m-1} \{ \sum_{i=1}^{k} p_{m-i,\sigma(j)} - \sum_{i=1}^{k} p_{m-i+1,\sigma(j-1)}\} \qquad (IV.24)$$

Evidentemente, a colisão ocorre no processador M_{m-r} e o intervalo entre as tarefas $J_{\sigma(j-1)}$ e $J_{\sigma(j)}$, neste caso, é dado por:

$$d(\sigma(j-1), \sigma(j)) = \sum_{i=1}^{r} p_{m-i,\sigma(j)} - \sum_{i=1}^{r} p_{m-i+1,\sigma(j-1)} \qquad (IV.25)$$

De acordo com os resultados obtidos nos casos IV.1 a IV.4, apresenta-se as seguintes expressões gerais para a determinação do intervalo entre tarefas:

$$d(\sigma(j-1),\sigma(j)) = \max_{1\leq \ell\leq m-1} \{0, \sum_{i=1}^{\ell} p_{m-i,\sigma(j)} - \sum_{i=1}^{\ell} p_{m-i+1,\sigma(j-1)}\} \qquad (IV.26)$$

$$\therefore d(\sigma(j-1),\sigma(j)) = \max_{1\leq \ell\leq m-1} \{0, q_{\sigma(j)} \cdot \sum_{i=1}^{\ell} \alpha_{m-i} + \sum_{i=1}^{\ell} \beta_{m-i} - q_{\sigma(j-1)} \cdot \sum_{i=1}^{\ell} \alpha_{m-i+1} - \sum_{i=1}^{\ell} \beta_{m-i+1}\} \qquad (IV.27)$$

$$\therefore d(\sigma(j-1),\sigma(j)) = \max_{1 \leq \ell \leq m-1} \{0,\ q_{\sigma(j)} \cdot \sum_{i=m-\ell}^{m-1} \alpha_i - q_{\sigma(j-1)} \cdot \sum_{i=m-\ell+1}^{m} \alpha_i + \beta_{m-\ell} - \beta_m\} \qquad (IV.28)$$

$$\therefore d(\sigma(j-1),\sigma(j)) = \max_{1 \leq \ell \leq m-1} \{0,\ (q_{\sigma(j)} - q_{\sigma(j-1)}) \cdot \sum_{i=m-\ell}^{m} \alpha_i + q_{\sigma(j-1)} \cdot \alpha_{m-\ell} - q_{\sigma(j)} \cdot \alpha_m + \beta_{m-\ell} - \beta_m\} \qquad (IV.29)$$

A determinação de $C_{max}(\sigma)$ e $\Sigma C(\sigma)$ pode ser efetuada através do seguinte algoritmo de complexidade $O(n \cdot m)$:

Algoritmo IV.1: Determinação de $C_{max}(\sigma)$ e $\Sigma C(\sigma)$

1 entrada: α_i e β_i , $i = 1,\ldots, m$

q_j , $j = 1,\ldots, n$

permutação σ de tarefas: $(J_{\sigma(1)},\ldots, J_{\sigma(n)})$

2 $C_{\sigma(1)} := q_{\sigma(1)} \cdot \sum_{i=1}^{m} \alpha_i + \sum_{i=1}^{m} \beta_i$

3 $\Sigma C(\sigma) := C_{\sigma(1)}$

4 para $j = 2,\ldots, n$ efetuar

$d := 0$

para $\ell = 1,\ldots, m-1$ efetuar

$$d := \max \{d,\ q_{\sigma(j)} \cdot \sum_{i=m-\ell}^{m-1} \alpha_i - q_{\sigma(j-1)} \cdot \sum_{i=m-\ell+1}^{m} \alpha_i + \beta_{m-\ell} - \beta_m\}$$

$$C_{\sigma(j)} := C_{\sigma(j-1)} + d + \alpha_m \cdot q_{\sigma(j)} + \beta_m$$

$$\Sigma C(\sigma) := \Sigma C(\sigma) + C_{\sigma(j)}$$

$C_{max}(\sigma) := C_{\sigma(n)}$

5 saída : $C_{max}(\sigma)$ e $\Sigma C(\sigma)$ Δ

IV.3. RESULTADOS RELACIONADOS COM INTERVALO ENTRE TAREFAS

Apresenta-se a seguir uma série de lemas relacionados com intervalo entre tarefas, os quais simplesmente constituem resultados adicionais, não utilizados na prova dos teoremas deste capítulo.

Lema IV.3: Se $d(g, j) = 0$ e $q_j \geq q_k$
então $d(g, k) = 0$

prova:

Por hipótese: $d(g, j) = 0$
logo, de (IV.28), segue que:

$$q_j \cdot \sum_{i=m-\ell}^{m-1} \alpha_i - q_g \cdot \sum_{i=m-\ell+1}^{m} \alpha_i + \beta_{m-\ell} - \beta_m \leq 0, \quad \ell = 1, \ldots, m-1$$

Por hipótese, $q_j \geq q_k$, logo:

$$q_k \cdot \sum_{i=m-\ell}^{m-1} \alpha_i - q_g \cdot \sum_{i=m-\ell+1}^{m} \alpha_i + \beta_{m-\ell} - \beta_m \leq 0, \quad \ell = 1, \ldots, m-1$$

Portanto, de (IV.28), tem-se: $d(g, k) = 0$ Δ

Lema IV.4: Se $d(j, k) = 0$ e $q_j \leq q_g$
então $d(g, k) = 0$

prova: análoga a anterior Δ

Lema IV.5: Se $q_j \geq q_k$
então, para q_g qualquer, tem-se:

$d(g, j) \geq d(g, k)$

prova: Existem 3 casos a considerar:

(a) $d(g, j) = 0$

Do Lema IV.3, segue que $d(g, k) = 0$

Logo, neste caso: $d(g, j) = d(g, k)$

(b) $d(g, j) > 0$ e $d(g, k) = 0$: trivial

(c) $d(g, j) > 0$ e $d(g, k) > 0$

Considere a colisão das tarefas J_g e J_k ocorrendo no processador M_{m-v} :

$$d(g, k) = q_k \cdot \sum_{i=m-v}^{m-1} \alpha_i - q_g \cdot \sum_{i=m-v+1}^{m} \alpha_i + \beta_{m-v} - \beta_m \qquad (i)$$

De (IV.28), segue que:

$$d(g, j) \geq q_j \cdot \sum_{i=m-v}^{m-1} \alpha_i - q_g \cdot \sum_{i=m-v+1}^{m} \alpha_i + \beta_{m-v} - \beta_m$$

Por hipótese: $q_j \geq q_k$, logo:

$$d(g, j) \geq q_k \cdot \sum_{i=m-v}^{m-1} \alpha_i - q_g \cdot \sum_{i=m-v+1}^{m} \alpha_i - \beta_{m-v} - \beta_m$$

Portanto, de (i): $d(g, j) \geq d(g, k)$ Δ

Lema IV.6: Se $q_j \geq q_k$

então, para q_g qualquer, tem-se:

$d(j, g) \leq d(k, g)$

prova: análoga a anterior Δ

IV.4 - O PROBLEMA FQ/no-wait/ΣC

Nesta seção mostra-se que a condição necessária e suficiente para uma permutação σ de tarefas ser solução ótima do problema FQ/no-wait/ΣC é tal que

$$q_{\sigma(1)} \leq q_{\sigma(2)} \leq \cdots \leq q_{\sigma(n)} .$$

Para tanto demonstra-se, inicialmente, os seguintes lemas:

Lema IV.7: Seja σ uma permutação de tarefas tal que $q_{\sigma(j)} \leq q_{\sigma(j+1)}$, $j = 1,\ldots, n-1$ e τ uma permutação qualquer de tarefas.

Então, $\sum_{j=1}^{k} q_{\sigma(j)} \leq \sum_{j=1}^{k} q_{\tau(j)}$, $k = 1,\ldots, n$

prova: Por indução

(a) Para k=1

Trivial, pois $q_{\sigma(1)} \leq q_j$, $j = 1,\ldots, n$

(b) Supor válido para $k-1 \leq n-1$

$$\sum_{j=1}^{k-1} q_{\sigma(j)} \leq \sum_{j=1}^{k-1} q_{\tau(j)} , \quad k = 2,\ldots, n$$

(c) Para k :

Existem dois casos a considerar:

1º caso: $q_{\sigma(k)} \leq q_{\tau(k)}$

$$\sum_{j=1}^{k} q_{\sigma(j)} = q_{\sigma(k)} + \sum_{j=1}^{k-1} q_{\sigma(j)} \leq q_{\tau(k)} + \sum_{j=1}^{k-1} q_{\tau(j)} = \sum_{j=1}^{k} q_{\tau(j)}$$

2º caso: $q_{\sigma(k)} > q_{\tau(k)}$

Por hipótese: $q_{\sigma(1)} \leq \cdots \leq q_{\sigma(k-1)} \leq q_{\sigma(k)} \leq \cdots \leq q_{\sigma(n)}$.

Então, neste caso, necessariamente existe $q_{\tau(s)}$, $s \leq k - 1$

tal que: $q_{\tau(s)} \geq q_{\sigma(k)}$ (i)

Considere π uma permutação de tarefas tal que:

$$q_{\pi(j)} = q_{\tau(j)} \ , \quad j \neq s \ , \quad j=1,\ldots,k-1 \qquad \text{(ii)}$$

$$q_{\pi(s)} = q_{\tau(k)} \qquad \text{(iii)}$$

$$q_{\pi(k)} = q_{\tau(s)} \qquad \text{(iv)}$$

Por hipótese de indução:

$$\sum_{j=1}^{k-1} q_{\sigma(j)} \leq \sum_{j=1}^{k-1} q_{\pi(j)} \qquad \text{(v)}$$

De (i) e (iv): $q_{\sigma(k)} \leq q_{\pi(k)}$ (vi)

Como, por construção:

$$\sum_{j=1}^{k} q_{\tau(j)} = \sum_{j=1}^{k} q_{\pi(j)} = q_{\pi(k)} + \sum_{j=1}^{k-1} q_{\pi(j)} \qquad \text{(vii)}$$

Considerando-se (v) e (vi) em (vii), tem-se:

$$\sum_{j=1}^{k} q_{\tau(j)} \geq q_{\sigma(k)} + \sum_{j=1}^{k-1} q_{\sigma(j)} = \sum_{j=1}^{k} q_{\sigma(j)} \quad \Delta$$

Lema IV.8: Seja σ uma permutação de tarefas tal que

$$q_{\sigma(j)} \leq q_{\sigma(j+1)} \ , \quad j = 1,\ldots, n-1$$

e τ uma permutação qualquer de tarefas.

Então, $\sum_{k=1}^{\ell} \sum_{j=1}^{k} q_{\sigma(j)} \leq \sum_{k=1}^{\ell} \sum_{j=1}^{k} q_{\tau(j)} \ , \quad \ell = 1,\ldots, n$

prova:

$$\sum_{k=1}^{\ell} \sum_{j=1}^{k} q_{\sigma(j)} = \sum_{j=1}^{1} q_{\sigma(j)} + \sum_{j=1}^{2} q_{\sigma(j)} + \ldots + \sum_{j=1}^{\ell} q_{\sigma(j)} \ , \quad \ell = 1,\ldots, n \qquad \text{(i)}$$

Do Lema IV.7: $\sum_{j=1}^{\ell} q_{\sigma(j)} \leq \sum_{j=1}^{\ell} q_{\tau(j)}$, $\ell = 1,\ldots, n$ (ii)

Considerando-se (ii) em (i), tem-se:

$$\sum_{k=1}^{\ell} \sum_{j=1}^{k} q_{\sigma(j)} \leq \sum_{j=1}^{1} q_{\tau(j)} + \sum_{j=1}^{2} q_{\tau(j)} + \ldots + \sum_{j=1}^{\ell} q_{\tau(j)} , \quad \ell = 1,\ldots, n$$

$$\therefore \quad \sum_{k=1}^{\ell} \sum_{j=1}^{k} q_{\sigma(j)} \leq \sum_{k=1}^{\ell} \sum_{j=1}^{k} q_{\sigma(j)} , \quad \ell = 1,\ldots, n \quad \Delta$$

Teorema IV.1: Se σ é uma permutação de tarefas tal que

$$q_{\sigma(1)} \leq q_{\sigma(2)} \leq \cdots \leq q_{\sigma(n)}$$

então $\Sigma C(\sigma) = \text{Min } \Sigma C_j$

prova:

Basta mostrar que:

$$\sum_{j=1}^{n} C_{\sigma(j)} \leq \sum_{j=1}^{n} C_{\tau(j)} , \text{ para qualquer permutação } \tau.$$

(a) De (IV.15):

$$\sum_{j=1}^{n} C_{\sigma(j)} = \sum_{k=1}^{n} \sum_{j=1}^{k} d(\sigma(j-1),\sigma(j)) + \alpha_m \cdot \sum_{k=1}^{n} \sum_{j=1}^{k} q_{\sigma(j)} + \beta_m \cdot \sum_{k=1}^{n} k \quad (i)$$

Por hipótese $q_{\sigma(j-1)} \leq q_{\sigma(j)}$, $j = 2,\ldots, n$, logo de (IV.18):

$$d(\sigma(j-1),\sigma(j)) = q_{\sigma(j)} \cdot \sum_{i=1}^{m-1} \alpha_i - q_{\sigma(j-1)} \cdot \sum_{i=2}^{m} \alpha_i + \beta_1 - \beta_m , \quad j = 1,\ldots, n$$

Substituindo-se esse resultado em (i):

$$\sum_{j=1}^{n} C_{\sigma(j)} = \sum_{k=1}^{n} \sum_{j=1}^{k} \left(q_{\sigma(j)} \cdot \sum_{i=1}^{m-1} \alpha_i - q_{\sigma(j-1)} \cdot \sum_{i=2}^{m} \alpha_i + \beta_1 - \beta_m\right) +$$

$$+ \alpha_m \cdot \sum_{k=1}^{n} \sum_{j=1}^{k} q_{\sigma(j)} + \beta_m \cdot \sum_{k=1}^{n} k$$

$$\therefore \quad \sum_{j=1}^{n} C_{\sigma(j)} = \left(\sum_{i=1}^{m-1}\alpha_i\right) \cdot \left(\sum_{k=1}^{n}\sum_{j=1}^{k} q_{\sigma(j)}\right) - \left(\sum_{i=2}^{m}\alpha_i\right) \cdot \left(\sum_{k=1}^{n}\sum_{j=1}^{k} q_{\sigma(j-1)}\right) +$$

$$+ \beta_1 \cdot \sum_{k=1}^{n} k - \beta_m \cdot \sum_{k=1}^{n} k + \alpha_m \cdot \sum_{k=1}^{n}\sum_{j=1}^{k} q_{\sigma(j)} + \beta_m \cdot \sum_{k=1}^{n} k \qquad (ii)$$

Como:

$$\sum_{k=1}^{n}\sum_{j=1}^{k} q_{\sigma(j-1)} = \sum_{j=1}^{1} q_{\sigma(j-1)} + \sum_{j=1}^{2} q_{\sigma(j-1)} + \cdots + \sum_{j=1}^{n} q_{\sigma(j-1)}$$

$$\therefore \quad \sum_{k=1}^{n}\sum_{j=1}^{k} q_{\sigma(j-1)} = q_{\sigma(0)} + (q_{\sigma(0)} + q_{\sigma(1)}) + \cdots + (q_{\sigma(0)} + q_{\sigma(1)} + \cdots +$$

$$+ q_{\sigma(n-1)})$$

$$\therefore \quad \sum_{k=1}^{n}\sum_{j=1}^{k} q_{\sigma(j-1)} = n \cdot q_{\sigma(0)} + (q_{\sigma(1)}) + \cdots + (q_{\sigma(1)} + \cdots + q_{\sigma(n-1)})$$

$$\therefore \quad \sum_{k=1}^{n}\sum_{j=1}^{k} q_{\sigma(j-1)} = n \cdot q_{\sigma(0)} + \sum_{k=1}^{n-1}\sum_{j=1}^{k} q_{\sigma(j)}$$

De (IV.9): $q_{\sigma(0)} = -\sum_{i=1}^{m}\beta_i \Big/ \sum_{i=1}^{m}\alpha_i$, logo:

$$\sum_{k=1}^{n}\sum_{j=1}^{k} q_{\sigma(j-1)} = -n \cdot \sum_{i=1}^{m}\beta_i \Big/ \sum_{i=1}^{m}\alpha_i + \sum_{k=1}^{n-1}\sum_{j=1}^{k} q_{\sigma(j)} \qquad (iii)$$

Substituindo-se (iii) em (ii):

$$\sum_{j=1}^{n} C_{\sigma(j)} = \left(\sum_{i=1}^{m}\alpha_i\right) \cdot \left(\sum_{k=1}^{n}\sum_{j=1}^{k} q_{\sigma(j)}\right) + \beta_1 \cdot \sum_{k=1}^{n} k -$$

$$- \left(\sum_{i=2}^{m}\alpha_i\right) \cdot \left(-n \cdot \sum_{i=1}^{m}\beta_i \Big/ \sum_{i=1}^{m}\alpha_i + \sum_{k=1}^{n-1}\sum_{j=1}^{k} q_{\sigma(j)}\right)$$

$$\therefore \sum_{j=1}^{n} C_{\sigma(j)} = \left(\sum_{i=1}^{m}\alpha_i\right)\cdot\left(\sum_{k=1}^{n-1}\sum_{j=1}^{k} q_{\sigma(j)} + \sum_{j=1}^{n} q_{\sigma(j)}\right) + \beta_1 \cdot \sum_{k=1}^{n} k +$$

$$+ n \cdot \left(\sum_{i=2}^{m}\alpha_i\right)\cdot\left(\sum_{i=1}^{m}\beta_i \Big/ \sum_{i=1}^{m}\alpha_i\right) - \left(\sum_{i=2}^{m}\alpha_i\right)\left(\sum_{k=1}^{n-1}\sum_{j=1}^{k} q_{\sigma(j)}\right)$$

$$\therefore \sum_{j=1}^{n} C_{\sigma(j)} = \left(\sum_{k=1}^{n-1}\sum_{j=1}^{k} q_{\sigma(j)}\right)\cdot\left(\sum_{i=1}^{m}\alpha_i - \sum_{i=2}^{m}\alpha_i\right) + \left(\sum_{i=1}^{m}\alpha_i\right)\left(\sum_{j=1}^{n} q_{\sigma(j)}\right) +$$

$$+ \beta_1 \cdot \sum_{k=1}^{n} k + n \cdot \left(\sum_{i=2}^{m}\alpha_i\right)\cdot\left(\sum_{i=1}^{m}\beta_i\right) \Big/ \left(\sum_{i=1}^{m}\alpha_i\right)$$

$$\therefore \sum_{j=1}^{n} C_{\sigma(j)} = \alpha_1 \cdot \sum_{k=1}^{n-1}\sum_{j=1}^{k} q_{\sigma(j)} + \left(\sum_{i=1}^{m}\alpha_i\right)\cdot\left(\sum_{j=1}^{n} q_{\sigma(j)}\right) + \beta_1 \cdot \sum_{k=1}^{n} k +$$

$$+ n \cdot \left(\sum_{i=2}^{m}\alpha_i\right)\cdot\left(\sum_{i=1}^{m}\beta_i \Big/ \sum_{i=1}^{m}\alpha_i\right) \qquad \text{(iv)}$$

De (IV.2): $\alpha_1 > 0$ e $\sum_{i=1}^{m}\alpha_i > 0$ (v)

Do Lema IV.7: $\sum_{j=1}^{n} q_{\sigma(j)} \leq \sum_{j=1}^{n} q_{\tau(j)}$ (vi)

Do Lema IV.8: $\sum_{k=1}^{n-1}\sum_{j=1}^{k} q_{\sigma(j)} \leq \sum_{k=1}^{n-1}\sum_{j=1}^{k} q_{\tau(j)}$ (vii)

Considerando-se (v) - (vii) em (iv), tem-se:

$$\sum_{j=1}^{n} C_{\sigma(j)} \leq \alpha_1 \cdot \sum_{k=1}^{n-1}\sum_{j=1}^{k} q_{\tau(j)} + \left(\sum_{i=1}^{m}\alpha_i\right)\cdot\left(\sum_{j=1}^{n} q_{\tau(j)}\right) + \beta_1 \cdot \sum_{k=1}^{n} k +$$

$$+ n \cdot \left(\sum_{i=2}^{m}\alpha_i\right)\cdot\left(\sum_{i=1}^{m}\beta_i \Big/ \sum_{i=1}^{m}\alpha_i\right) \qquad \text{(viii)}$$

(b) Por outro lado, de (IV.15):

$$\sum_{j=1}^{n} C_{\tau(j)} = \sum_{k=1}^{n} \sum_{j=1}^{k} d(\tau(j-1),\tau(j)) + \alpha_m \cdot \sum_{k=1}^{n} \sum_{j=1}^{k} q_{\sigma(j)} + \beta_m \cdot \sum_{k=1}^{n} k \qquad (ix)$$

De (IV.28):

$$d(\tau(j-1),\tau(j)) \geq q_{\tau(j)} \cdot \sum_{i=1}^{m-1} \alpha_i - q_{\tau(j-1)} \cdot \sum_{i=2}^{m} \alpha_i + \beta_1 - \beta_m$$

Substituindo-se esse resultado em (ix):

$$\sum_{j=1}^{n} C_{\tau(j)} \geq \sum_{k=1}^{n} \sum_{j=1}^{k} \left(q_{\tau(j)} \cdot \sum_{i=1}^{m-1} \alpha_i - q_{\tau(j-1)} \cdot \sum_{i=2}^{m} \alpha_i + \beta_1 - \beta_m\right) +$$

$$+ \alpha_m \cdot \sum_{k=1}^{n} \sum_{j=1}^{k} q_{\sigma(j)} + \beta_m \cdot \sum_{k=1}^{n} k$$

Seguindo-se procedimento análogo ao do item (a) acima, tem-se:

$$\sum_{j=1}^{n} C_{\tau(j)} \geq \alpha_1 \cdot \sum_{k=1}^{n-1} \sum_{j=1}^{k} q_{\tau(j)} + \left(\sum_{i=1}^{m} \alpha_i\right) \cdot \left(\sum_{j=1}^{n} q_{\tau(j)}\right) + \beta_1 \cdot \sum_{k=1}^{n} k +$$

$$+ n \cdot \left(\sum_{i=2}^{m} \alpha_i\right) \cdot \left(\sum_{i=1}^{m} \beta_i \,/\, \sum_{i=1}^{m} \alpha_i\right) \qquad (x)$$

(c) De (viii) e (x), segue que:

$$\sum_{j=1}^{n} C_{\sigma(j)} \leq \sum_{j=1}^{n} C_{\tau(j)}$$

Logo: $\Sigma C(\sigma) = \text{Min } \Sigma C_j \quad \Delta$

A seguir mostra-se que toda permutação σ que minimiza a data média de término (min ΣC_j), satisfaz a condição:

$$q_{\sigma(1)} \leq q_{\sigma(2)} \leq \cdots \leq q_{\sigma(n)}$$

ou seja, demonstra-se a suficiência dessa condição.

Teorema IV.2: Se σ e τ são duas permutações de tarefas, tais que

$$q_{\sigma(1)} \leq q_{\sigma(2)} \leq \cdots \leq q_{\sigma(n)} \quad \text{e}$$

$$q_{\tau(\ell)} > q_{\tau(\ell+1)} \text{ , para algum } \ell \in \{1,\ldots, n-1\}$$

$$\text{então} \quad \sum_{j=1}^{n} C_{\sigma(j)} < \sum_{j=1}^{n} C_{\tau(j)}$$

prova:

Da equação (iv) do Teorema IV.1, tem-se:

$$\sum_{j=1}^{n} C_{\sigma(j)} = \alpha_1 \left(\sum_{j=1}^{1} q_{\sigma(j)} + \ldots + \sum_{j=1}^{\ell} q_{\sigma(j)} + \ldots + \sum_{j=1}^{n-1} q_{\sigma(j)} \right) +$$

$$+ \left(\sum_{i=1}^{m} \alpha_i\right) \cdot \left(\sum_{j=1}^{n} q_{\sigma(j)} \right) + \beta_1 \cdot \sum_{k=1}^{n} k + n \cdot \left(\sum_{i=2}^{m} \alpha_i\right) \cdot \left(\sum_{i=1}^{m} \beta_i \,/\, \sum_{i=1}^{m} \alpha_i\right) \quad \text{(i)}$$

Do Lema IV.7:

$$\sum_{j=1}^{k} q_{\sigma(j)} \leq \sum_{j=1}^{k} q_{\tau(j)} \quad , \quad k = 1,\ldots, n \qquad \text{(ii)}$$

Por hipótese, $q_{\tau(\ell)} > q_{\tau(\ell+1)}$ e $q_{\sigma(1)} \leq q_{\sigma(2)} \leq \cdots \leq q_{\sigma(\ell)}$

Logo: $$\sum_{j=1}^{\ell} q_{\sigma(j)} < \sum_{j=1}^{\ell} q_{\tau(j)} \qquad \text{(iii)}$$

De (i), (ii) e (iii), segue que:

$$\sum_{j=1}^{n} C_{\sigma(j)} < \alpha_1 \left(\sum_{j=1}^{1} q_{\tau(j)} + \ldots + \sum_{j=1}^{\ell} q_{\tau(j)} + \ldots + \sum_{j=1}^{n-1} q_{\tau(j)} \right) +$$

$$+ \left(\sum_{i=1}^{m} \alpha_i\right) \cdot \left(\sum_{j=1}^{n} q_{\tau(j)} \right) + \beta_1 \cdot \sum_{k=1}^{n} k + n \cdot \left(\sum_{i=2}^{m} \alpha_i\right) \cdot \left(\sum_{i=1}^{m} \beta_i \,/\, \sum_{i=1}^{m} \alpha_i\right)$$

$$\therefore \sum_{j=1}^{n} C_{\sigma(j)} < \alpha_1 \cdot \sum_{k=1}^{n-1} \sum_{j=1}^{k} q_{\tau(j)} + \left(\sum_{i=1}^{m} \alpha_i\right) \cdot \left(\sum_{j=1}^{n} q_{\tau(j)} \right) +$$

$$+ \beta_1 \cdot \sum_{k=1}^{n} k + n \cdot \left(\sum_{i=2}^{m} \alpha_i\right) \cdot \left(\sum_{i=1}^{m} \beta_i \,/\, \sum_{i=1}^{m} \alpha_i\right) \qquad \text{(iv)}$$

Da equação (x) do Teorema IV.1, tem-se:

$$\sum_{j=1}^{n} C_{\tau(j)} \geq \alpha_i \sum_{k=1}^{n-1} \sum_{j=1}^{k} q_{\tau(j)} + \left(\sum_{i=1}^{m} \alpha_i\right) \cdot \left(\sum_{j=1}^{n} q_{\tau(j)} \right) +$$

$$+ \beta_1 \cdot \sum_{k=1}^{n} k + n \cdot \left(\sum_{i=2}^{m} \alpha_i\right) \cdot \left(\sum_{i=1}^{m} \beta_i \,/\, \sum_{i=1}^{m} \alpha_i\right) \qquad \text{(v)}$$

Finalmente, de (iv) e (v):

$$\sum_{j=1}^{n} C_{\sigma(j)} < \sum_{j=1}^{n} C_{\tau(j)} \qquad \Delta$$

De acordo com o resultado do Teorema IV.1 e utilizando-se o Algoritmo IV.1, pode-se determinar o valor de uma solução ótima para o problema FQ/no-wait/ ΣC em tempo $O(n(m + \log n))$.

CAPITULO V

PROBLEMAS EM SCHEDULING DO TIPO NO-WAIT FLOW-SHOP COM PROCESSADORES DE VELOCIDADE CONTROLÁVEL

V.1 - CONSIDERAÇÕES INICIAIS

Neste capítulo estuda-se problemas em scheduling do tipo no-wait flow-shop, nos quais as velocidades dos processadores podem ser controladas.

Considere um conjunto de processadores $M = \{M_1, \ldots, M_i, \ldots M_m\}$, e um conjunto de tarefas, $J = \{J_1, \ldots, J_n\}$. Nestes problemas, denotados por FC/no-wait/· , as velocidades dos processadores podem ser controladas de tal forma que a velocidade que o processador M_i executa a tarefa J_j é uma função da velocidade que o processador M_{i-1} executa a tarefa J_{j+1}.

Na seção V.2 define-se os problemas tratados neste capítulo e na seguinte apresenta-se procedimentos para a determinação do tempo de processamento (p_{ij}) e da data de término (C_j). Finalizando, na seção V.4 apresenta-se resultados para os problemas FC/no-wait/C_{max}, FC/no-wait/ΣC e FC/no-wait/ΣP.

V.2 - DEFINIÇÃO DOS PROBLEMAS

Considere problemas em scheduling do tipo no-wait flow-shop com processadores de velocidade controlável.

Seja: $(J_{\sigma(1)},\ldots, J_{\sigma(n)})$, uma permutação qualquer de tarefas

$(O_{1,\sigma(j)},\ldots, O_{m,\sigma(j)})$, operações da tarefa $J_{\sigma(j)} \in J$

$p_{i,\sigma(j)}$, tempo de processamento da i-ésima operação da tarefa $J_{\sigma(j)} \in J$, no processador $M_i \in M$, considerando-se a permutação de tarefas σ.

Considere que as velocidades dos processadores possam ser controladas de modo que o tempo que o processador M_i executa a operação $O_{i,\sigma(j)}$ seja igual ao tempo que M_{i-1} executa a operação $O_{i-1,\sigma(j+1)}$, ou seja:

$$p_{i,\sigma(j)} = p_{i-1,\sigma(j+1)} \qquad \begin{matrix} i = 2,\ldots, m \\ j = 1,\ldots, n-1 \end{matrix} \qquad (V.1)$$

De acordo com (II.12), as tarefas $J_{\sigma(j)}$ e $J_{\sigma(j+1)}$, $j = 1,\ldots, n-1$, são encaixadas.

Seja: v_i, velocidade nominal do processador $M_i \in M$ ($v_i > 0$);

q_j, quantidade de processamento de cada operação da tarefa $J_j \in J$ ($q_j > 0$);

$v_{i,\sigma(j)}$, velocidade real do processador $M_i \in M$, executando a operação $O_{i,\sigma(j)}$ da tarefa $J_{\sigma(j)} \in J$, considerando a permutação de tarefas σ.

Evidentemente:

$$v_{i,\sigma(j)} = q_{\sigma(j)} / p_{i,\sigma(j)}, \qquad \begin{matrix} i = 1,\ldots, m \\ j = 1,\ldots, n \end{matrix} \qquad (V.2)$$

Por conveniência, considere:

$$\alpha_i = 1 / v_i, \quad i = 1,\ldots, m \qquad (V.3)$$

Considere as seguintes condições de contorno:

(i) o primeiro processador, M_1, opera sempre na sua velocidade nominal:

$$p_{1,\sigma(j)} = q_{\sigma(j)} / v_1 = \alpha_1 \cdot q_{\sigma(j)} , \quad j = 1,\ldots, n \qquad (V.4)$$

(ii) para qualquer permutação σ de tarefas, as operações da última tarefa, $J_{\sigma(n)}$, são executadas na velocidade nominal:

$$p_{i,\sigma(n)} = q_{\sigma(n)} / v_i = \alpha_i \cdot q_{\sigma(n)} , \quad i = 2,\ldots, m \qquad (V.5)$$

Na seção V.4 apresenta-se resultados relacionados com os seguintes problemas:

FC/no-wait/C_{max} : mostra-se (Teorema V.1) que a condição necessária e suficiente para uma permutação de tarefas σ minimizar a máxima data de término é tal que $q_{\sigma(n)} = \min_{1 \leq j \leq n} \{q_j\}$;

FC/no-wait/ΣC : os Teoremas V.2 e V.3 apresentam condições de otimalidade relacionadas com a minimização da data média de término;

FC/no-wait/ΣP : o Teorema V.4 mostra que a condição necessária para uma permutação de tarefas σ minimizar o tempo total de processamento é tal que $q_{\sigma(j)} \geq q_{\sigma(j+1)}$, $j = 1,\ldots, n-1$.

V.3 - DETERMINAÇÃO DE $p_{i,\sigma(j)}$, $C_{\sigma(j)}$, $\Sigma C_{\sigma(j)}$ e $\Sigma P_{\sigma(j)}$

A determinação de $p_{i,\sigma(j)}$ pode ser feita diretamente a partir do lema a seguir:

Lema V.1: Se $i + j \leq n + 1$ então $p_{i,\sigma(j)} = \alpha_1 \cdot q_{\sigma(i+j-1)}$

caso contrário $p_{i,\sigma(j)} = \alpha_k \cdot q_{\sigma(n)}$, para $k = i+j-n$

prova: (por indução)

(a) Caso: $i+j \leq n+1$

- Para i=1 ($j \leq n$), o resultado segue diretamente de (V.4):

$$p_{1,\sigma(j)} = \alpha_1 \cdot q_{\sigma(j)} \quad , \quad j = 1,\ldots, n$$

- Supor válido para i-1

$$p_{i-1,\sigma(j)} = \alpha_1 \cdot q_{\sigma(i-1+j-1)} = \alpha_1 \cdot q_{\sigma(i+j-2)} \qquad (i)$$

- Para $i \leq n-j+1$, tem-se:

de (V.1): $p_{i,\sigma(j)} = p_{i-1,\sigma(j+1)}$

de (i) : $p_{i-1,\sigma(j+1)} = \alpha_1 \cdot q_{\sigma(i+j+1-2)} = \alpha_1 \cdot q_{\sigma(i+j-1)}$

logo: $p_{i,\sigma(j)} = \alpha_1 \cdot q_{\sigma(i+j-1)}$

(b) Caso: $i+j > n+1$

- Para j=n ($i > 1$), o resultado segue diretamente de (V.5):

$$p_{i,\sigma(n)} = \alpha_i \cdot q_{\sigma(n)} \quad , \quad i = 2,\ldots, m$$

- Supor válido para j+1

$$p_{i,\sigma(j+1)} = \alpha_k \cdot q_{\sigma(n)} \text{ , para } k=i + (j+1) - n = i+j-n+1 \qquad (ii)$$

- Para $j > n-i+1$, tem-se:

de (V.1): $p_{i,\sigma(j)} = p_{i-1,\sigma(j+1)}$

de (ii): $p_{i-1,\sigma(j+1)} = \alpha_k \cdot q_{\sigma(n)}$, para $k = (i-1) + j-n+1$

logo: $p_{i,\sigma(j)} = \alpha_k \cdot q_{\sigma(n)}$, para $k = i+j-n$ Δ

A partir de (V.4) e (V.5) é imediato verificar que:

$$C_{\sigma(n)} = \alpha_1 \cdot \sum_{j=1}^{n} q_{\sigma(j)} + q_{\sigma(n)} \cdot \sum_{i=2}^{m} \alpha_i \qquad (V.6)$$

Como, $\sum_{j=1}^{n} q_{\sigma(j)} = \sum_{j=1}^{n} q_j$ e $C_{\sigma(n)} = C_{max}(\sigma)$, para qualquer permutação σ de tarefas, tem-se:

$$C_{max}(\sigma) = \alpha_1 \cdot \sum_{j=1}^{n} q_j + q_{\sigma(n)} \cdot \sum_{i=2}^{m} \alpha_i \qquad (V.7)$$

A partir do resultado do Lema V.1 e da equação (V.6), pode-se determinar as datas de término, $C_{\sigma(j)}$, $j = 1,\ldots, n$, através do seguinte algoritmo linear:

Algoritmo V.1: Determinação de $C_{\sigma(j)}$

entrada: - permutação σ de tarefas; m, número de processadores.
- $q_{\sigma(j)}$, $j = 1,\ldots, n$

para $j = 1,\ldots, n$ efetuar

se $j \leq n - m + 1$ então $p_{m,\sigma(j)} := \alpha_1 \cdot q_{\sigma(j+m-1)}$

caso contrário $p_{m,\sigma(j)} := \alpha_{j+m-n} \cdot q_{\sigma(n)}$

$$C_{\sigma(n)} := \alpha_1 \cdot \sum_{j=1}^{n} q_{\sigma(j)} + q_{\sigma(n)} \cdot \sum_{i=2}^{m} \alpha_i$$

para $j = n-1, n-2,\ldots, 1$ efetuar

$$C_{\sigma(j)} := C_{\sigma(j+1)} - p_{m,\sigma(j+1)}$$

saída : $C_{\sigma(j)}$, $j = 1,\ldots, n$ Δ

Alternativamente, a determinação de $C_{\sigma(j)}$ pode ser feita analisando-se os seguintes casos:

(i) Caso $n < m$

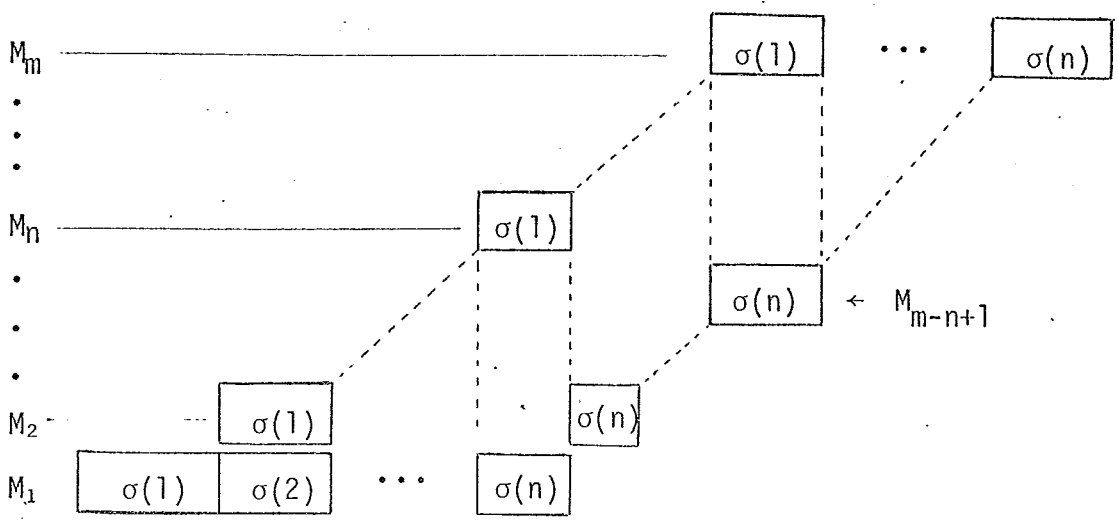


Fig. V.1 : Schedule, caso $n < m$

Segue diretamente da Fig. V.1 que:

$$C_{\sigma(j)} = \alpha_1 \cdot \sum_{k=1}^{n} q_k + q_{\sigma(n)} \cdot \sum_{i=2}^{m-n+j} \alpha_i \quad , \quad j = 1, \ldots, n \qquad (V.8)$$

(ii) Caso $n \geq m$

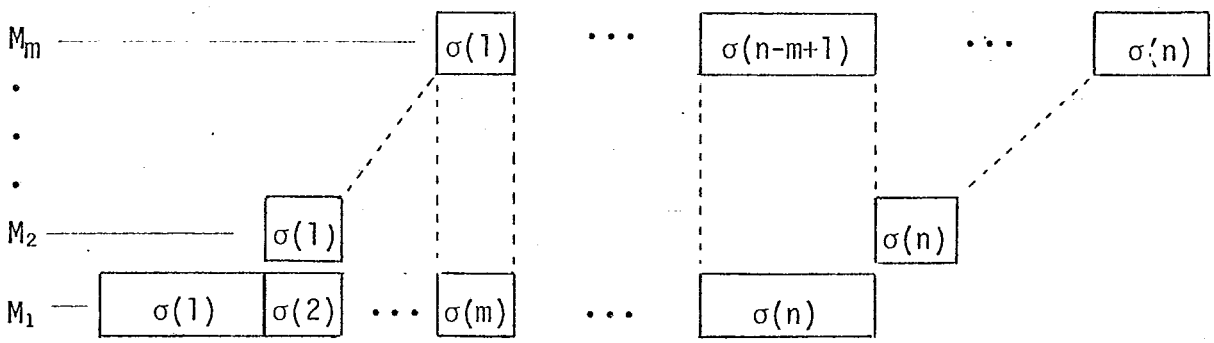


Fig. V.2 : Schedule, caso $n \geq m$

Neste caso tem-se:

$$C_{\sigma(j)} = \alpha_1 \cdot \sum_{k=1}^{m+j-1} q_{\sigma(k)} \quad , \text{ para } j = 1,\ldots, n-m+1 \qquad (V.9)$$

$$C_{\sigma(j)} = \alpha_1 \cdot \sum_{k=1}^{n} q_{\sigma(k)} + q_{\sigma(n)} \cdot \sum_{i=2}^{m-n+j} \alpha_i \quad , \text{ para } j = n-m+2,\ldots, n$$

A data média de término, $\sum_{j=1}^{n} C_{\sigma(j)} = \Sigma C_{\sigma(j)}$, pode ser determinada analisando-se os seguintes casos:

(i) Caso $n < m$

De (V.8) segue que:

$$\Sigma C_{\sigma(j)} = \sum_{j=1}^{n} (\alpha_1 \cdot \sum_{k=1}^{n} q_{\sigma(k)} + q_{\sigma(n)} \cdot \sum_{i=2}^{m-n+j} \alpha_i) \qquad (V.10)$$

Como $\sum_{k=1}^{n} q_{\sigma(k)} = \sum_{k=1}^{n} q_k$ = constante, tem-se:

$$\Sigma C_{\sigma(j)} = n \cdot \alpha_1 \cdot \sum_{k=1}^{n} q_k + q_{\sigma(n)} \cdot \sum_{j=1}^{n} \sum_{i=2}^{m-n+j} \alpha_i \qquad (V.11)$$

(ii) Caso $n = m$

De (V.9) e (V.10) tem-se:

$$\Sigma C_{\sigma(j)} = \alpha_1 \cdot \sum_{k=1}^{n} q_k + \sum_{j=2}^{n} (\alpha_1 \cdot \sum_{k=1}^{n} q_k + q_{\sigma(n)} \cdot \sum_{i=2}^{j} \alpha_i) \qquad (V.12)$$

$$\therefore \quad \Sigma C_{\sigma(j)} = n \cdot \alpha_1 \cdot \sum_{k=1}^{n} q_k + q_{\sigma(n)} \cdot \sum_{j=2}^{n} \sum_{i=2}^{j} \alpha_i \qquad (V.13)$$

$$\therefore \quad \Sigma C_{\sigma(j)} = n \cdot \alpha_1 \cdot \sum_{k=1}^{n} q_k + q_{\sigma(n)} \cdot \sum_{i=2}^{m} (m+1-i) \cdot \alpha_i \qquad (V.14)$$

(iii) Caso $n > m$

Diretamente, a partir da Fig. V.2, tem-se:

$$\Sigma C_{\sigma(j)} = n \cdot \alpha_1 \cdot \sum_{k=1}^{m} q_{\sigma(k)} + (n-1) \cdot \alpha_1 \cdot q_{\sigma(m+1)} + \ldots$$

$$\ldots + (m+1) \cdot \alpha_1 \cdot q_{\sigma(n-1)} + m \cdot \alpha_1 \cdot q_{\sigma(n)} +$$

$$+ (m-1) \cdot \alpha_2 \cdot q_{\sigma(n)} + \ldots + 2 \cdot \alpha_{m-1} \cdot q_{\sigma(n)} + \alpha_m \cdot q_{\sigma(n)} \quad (V.15)$$

$$\therefore \Sigma C_{\sigma(j)} = n \cdot \alpha_1 \cdot \sum_{k=1}^{m} q_{\sigma(k)} + \alpha_1 \cdot \sum_{k=m+1}^{n} (n+m-k) \cdot q_{\sigma(k)} +$$

$$+ q_{\sigma(n)} \cdot \sum_{i=2}^{m} (m+1-i) \cdot \alpha_i \quad (V.16)$$

Finalmente, o tempo total de processamento, $\sum_{j=1}^{n} P_{\sigma(j)} = \Sigma P_{\sigma(j)}$, pode ser determinado de modo análogo:

(i) Caso $n \leq m$:

$$\Sigma P_{\sigma(j)} = \alpha_1 \cdot q_{\sigma(1)} + 2 \cdot \alpha_1 \cdot q_{\sigma(2)} + \ldots + (n-1) \cdot \alpha_1 \cdot q_{\sigma(n-1)} +$$

$$+ n \cdot \alpha_1 \cdot q_{\sigma(n)} + n \cdot \alpha_2 \cdot q_{\sigma(n)} + \ldots + n \cdot \alpha_{m-n+1} \quad q_{\sigma(r)} +$$

$$+ (n-1) \cdot \alpha_{m-n+2} \cdot q_{\sigma(n)} + \ldots + 2 \cdot \alpha_{m-1} \cdot q_{\sigma(n)} +$$

$$+ \alpha_m \cdot q_{\sigma(n)} \quad (V.17)$$

$$\therefore \Sigma P_{\sigma(j)} = \alpha_1 \cdot \sum_{k=1}^{n-1} k \cdot q_{\sigma(k)} + q_{\sigma(n)} \cdot (n \cdot \sum_{i=1}^{m-n+1} \alpha_i + \sum_{i=m-n+2}^{m} (m-i+1) \cdot \alpha_i)$$

(V.18)

(ii) Caso $n > m$

$$\Sigma P_{\sigma(j)} = \alpha_1 \cdot \left(\sum_{k=1}^{m-1} k \cdot q_{\sigma(k)} + m \cdot \sum_{k=m}^{n-1} q_{\sigma(k)} \right) +$$

$$+ q_{\sigma(n)} \cdot \sum_{i=1}^{m} (m-i+1) \cdot \alpha_i \qquad (V.19)$$

V.4 - OS PROBLEMAS FC/no-wait/C_{max}, FC/no-wait/ΣC e FC/no-wait/ΣP

O Teorema V.1 a seguir mostra que a condição necessária e suficiente para uma permutação de tarefas σ ser solução ótima do problema FC/no-wait/C_{max} é tal que a quantidade de processamento da última tarefa, $q_{\sigma(n)}$, seja mínima.

Teorema V.1: σ é uma permutação de tarefas tal que

$$q_{\sigma(n)} = \min \{q_j\}, \quad j = 1, \ldots, n$$

sse $C_{max}(\sigma) = \text{Min } C_{max}$

prova:

(==>) De (V.7), para qualquer permutação τ de tarefas, tem-se:

$$C_{max}(\tau) = \alpha_1 \cdot \sum_{j=1}^{n} q_j + q_{\tau(n)} \cdot \sum_{i=2}^{m} \alpha_i \qquad (i)$$

Por hipótese: $q_{\sigma(n)} = \min \{q_j\}, \quad j = 1, \ldots, n \qquad (ii)$

De (i) e (ii) segue que:

$C_{max}(\sigma) = \min \{C_{max}(\tau)\}$, para qualquer permutação τ de tarefas

Logo: $C_{max}(\sigma) = \text{Min } C_{max}$

($\Leftarrow$) Por hipótese $C_{max}(\sigma) = \text{Min } C_{max}$ (iii)

De (V.7): $C_{max}(\sigma) = \alpha_1 \cdot \sum_{j=1}^{n} q_j + q_{\sigma(n)} \cdot \sum_{i=2}^{m} \alpha_i$

Suponha, por absurdo, que existe uma permutação τ de tarefas, tal que $q_{\tau(n)} < q_{\sigma(n)}$

Logo: $C_{max}(\tau) = \alpha_1 \cdot \sum_{j=1}^{n} q_j + q_{\tau(n)} \cdot \sum_{i=2}^{m} \alpha_i < C_{max}(\sigma)$,

contradizendo (iii) Δ

Uma solução ótima para o problema FC/no-wait/ /C_{max} também é ótima para o problema FC/no-wait/ΣC, no caso $n \leq m$, como mostra o teorema a seguir.

Teorema V.2: Se σ é uma permutação de tarefas tal que

$q_{\sigma(n)} = \min \{q_j\}$, $j = 1, \ldots, n$ e $n \leq m$

então $\Sigma C_\sigma = \text{Min } \Sigma C_j$

prova:

De (V.11) e (V.13), para qualquer permutação τ de tarefas, tem-se:

$$\Sigma C_\tau = n \cdot \alpha_1 \cdot \sum_{k=1}^{n} q_K + q_{\tau(n)} \cdot \sum_{j=1}^{n} \sum_{i=2}^{m-n+j} \alpha_i \quad , \quad n \leq m$$

Por hipotese: $q_{\sigma(n)} \leq q_{\tau(n)}$, para qualquer permutação τ de tarefas

Logo: $\Sigma C_\sigma \leq \Sigma C_\tau$, para qualquer permutação τ de tarefas

$\therefore \quad \Sigma C_\sigma = \text{Min } \Sigma C_j \quad \Delta$

A seguir mostra-se alguns resultados utilizados na prova dos Teoremas V.3 e V.4.

Lema V.2: Se σ é uma permutação de tarefas tal que

$$q_{\sigma(j)} \geq q_{\sigma(j+1)} \quad , \quad j = 1, \ldots, n-1$$

e a_j são números positivos tal que

$$a_j \leq a_{j+1} \quad , \quad j = 1, \ldots, n-1$$

então, para qualquer permutação de tarefas τ, tem-se:

$$\sum_{j=1}^{n} a_j \cdot q_{\sigma(j)} \leq \sum_{j=1}^{n} a_j \cdot q_{\tau(j)}$$

prova: (por indução)

(a) Para 1 tarefa : trivial

(b) Para 2 tarefas:
Deve-se mostrar que:

$$a_1 \cdot q_{\sigma(1)} + a_2 \cdot q_{\sigma(2)} \leq a_1 \cdot q_{\tau(1)} + a_2 \cdot q_{\tau(2)}$$

Existem dois casos possíveis:

(b.1) $q_{\sigma(1)} = q_{\tau(1)}$ e $q_{\sigma(2)} = q_{\tau(2)}$: trivial

(b.2) $q_{\sigma(1)} = q_{\tau(2)}$ e $q_{\sigma(2)} = q_{\tau(1)}$ (i)

Por hipótese: $q_{\sigma(1)} - q_{\tau(2)} \geq 0$ (ii)

$$a_2 - a_1 \geq 0 \qquad \text{(iii)}$$

Multiplicando-se (ii) por (iii), segue que:

$$a_1 \cdot q_{\sigma(1)} + a_2 \cdot q_{\sigma(2)} \leq a_1 \cdot q_{\sigma(2)} + a_2 \cdot q_{\sigma(1)} \qquad \text{(iv)}$$

O resultado segue diretamente de (i) e (iv)

(c) Supor válido para n-1 tarefas:

$$\sum_{j=1}^{n-1} a_j \cdot q_{\sigma(j)} \leq \sum_{j=1}^{n-1} a_j \cdot q_{\tau(j)} \qquad \text{(v)}$$

(d) Para n tarefas:

Por hipótese: $q_{\sigma(j)} \geq q_{\sigma(j+1)}$, $j = 1,\ldots, n-1$

$\therefore \quad q_{\sigma(n)} \leq q_j$, $j = 1,\ldots, n$

$\therefore \quad q_{\sigma(n)} \leq q_{\tau(n)}$, para qualquer permutação τ (vi)

Multiplicando-se (vi) por $a_n > 0$, tem-se:

$$a_n \cdot q_{\sigma(n)} \leq a_n \cdot q_{\tau(n)} \qquad \text{(vii)}$$

Somando-se (v) e (vii), segue o resultado esperado:

$$\sum_{j=1}^{n} a_j \cdot q_{\sigma(j)} \leq \sum_{j=1}^{n} a_j \cdot q_{\tau(j)} \text{ , para qualquer permutação } \tau \quad \Delta$$

Lema V.3: Se σ é uma permutação de tarefas tal que

$$q_{\sigma(j)} \leq q_{\sigma(j+1)} \text{ , } j = 1,\ldots, n-1$$

e a_j são números positivos tal que

$$a_j \geq a_{j+1} \text{ , } j = 1,\ldots, n-1$$

então, para qualquer permutação de tarefas τ, tem-se:

$$\sum_{j=1}^{n} a_j \cdot q_{\sigma(j)} \leq \sum_{j=1}^{n} a_j \cdot q_{\tau(j)}$$

prova: diretamente a partir do Lema V.2, utilizando-se a propriedade comutativa da soma Δ

O teorema a seguir apresenta um resultado para o problema FC/no-wait/ΣC, no caso $n > m$.

Teorema V.3: Se σ é uma permutação de tarefas tal que

$$q_{\sigma(j)} \leq q_{\sigma(m+1)} \leq q_{\sigma(m+2)} \leq \cdots \leq q_{\sigma(n)} \ , \quad j = 1,\ldots, m \ ,$$

$$n > m \quad e \quad \sum_{i=1}^{m} \alpha_j \cdot (m+1-i) \leq (m+1) \cdot \alpha_1$$

então $\Sigma C_\sigma = \text{Min} \Sigma C_j$

prova:

(a) Da hipótese segue que:

$$\sum_{j=1}^{m} q_{\sigma(j)} \leq \sum_{j=1}^{m} q_{\tau(j)} \ , \quad \text{para qualquer permutação } \tau$$

Como $n \cdot \alpha_1 > 0$, tem-se:

$$n \cdot \alpha_1 \cdot \sum_{j=1}^{m} q_{\sigma(j)} \leq n \cdot \alpha_1 \cdot \sum_{j=1}^{m} q_{\tau(j)} \qquad (i)$$

(b) Do Lema V.3, para qualquer permutação τ :

$$\sum_{j=m+1}^{n-2} (n+m-j) \cdot q_{\sigma(j)} \leq \sum_{j=m+1}^{n-2} (n+m-j) \cdot q_{\tau(j)}$$

Como $\alpha_1 > 0$, tem-se:

$$\alpha_1 \cdot \sum_{j=m+1}^{n-2} (n+m-j) \cdot q_{\sigma(j)} \leq \alpha_1 \cdot \sum_{j=m+1}^{n-2} (n+m-j) \cdot q_{\tau(j)} \qquad (ii)$$

(c) Por hipótese:

$$q_{\sigma(n-1)} \leq q_{\sigma(n)} \quad e$$

$$\alpha_1 \cdot (m+1) \geq \sum_{i=1}^{m} (m+1-i) \cdot \alpha_i$$

Logo, do Lema V.3, para qualquer permutação τ :

$$\alpha_1 \cdot (m+1) \cdot q_{\sigma(n-1)} + q_{\sigma(n)} \cdot \sum_{i=1}^{m} (m+1-i) \cdot \alpha_i \leq$$

$$\alpha_1 \cdot (m+1) \cdot q_{\tau(n-1)} + q_{\tau(n)} \cdot \sum_{i=1}^{m} (m+1-i) \cdot \alpha_i \qquad \text{(iii)}$$

(d) Somando-se (i), (ii) e (iii):

$$n \cdot \alpha_1 \cdot \sum_{j=1}^{m} q_{\sigma(j)} + \alpha_1 \cdot \sum_{j=m+1}^{n-1} (n+m-j) \cdot q_{\sigma(j)} + q_{\sigma(n)} \cdot \sum_{i=1}^{m} (m+1-i) \cdot \alpha_i \leq$$

$$n \cdot \alpha_1 \cdot \sum_{j=1}^{m} q_{\tau(j)} + \alpha_1 \cdot \sum_{j=m+1}^{n-1} (n+m-j) \cdot q_{\tau(j)} + q_{\tau(n)} \cdot \sum_{i=1}^{m} (m+1-i) \cdot \alpha_i$$

De (V.16), segue que:

$\Sigma C_\sigma \leq \Sigma C_\tau$, para qualquer permutação τ

Logo: $\Sigma C_\sigma = \text{Min } \Sigma C_\sigma$ Δ

Finalizando, mostra-se que uma permutação de tarefas com quantidade de processamento em ordem não crescente, minimiza o tempo total de processamento e consequentemente, o problema FC/no-wait/ΣP pode ser resolvido em tempo O(nlogn).

Teorema V.4: Se σ é uma permutação de tarefas tal que

$$q_{\sigma(j)} \geq q_{\sigma(j+1)} \quad , \quad j = 1, \ldots, n-1$$

então $\Sigma P_\sigma = \text{Min } \Sigma P$

prova: Para toda permutação τ de tarefas, pode-se reescrever as equações (V.18) e (V.19) de forma simplificada como segue:

$$\Sigma P_\tau = \sum_{j=1}^{n} a_j \cdot q_{\tau(j)} \ , \ 0 < a_j \leq a_{j+1} \ , \ j = 1,\ldots, n-1 \qquad (i)$$

Por hipótese: $q_{\sigma(j)} \geq q_{\sigma(j+1)}$, $j = 1,\ldots, n-1$ (ii)

De (i) e (ii), utilizando-se o resultado do Lema V.2, tem-se:

$$\sum_{j=1}^{n} a_j \cdot q_{\sigma(j)} \leq \sum_{j=1}^{n} a_j \cdot q_{\tau(j)} \ , \text{ para qualquer permutação } \tau$$

Logo: $\Sigma P_\sigma = \text{Min } \Sigma P \quad \Delta$

CAPITULO VI

PROBLEMAS EM SCHEDULING DO TIPO NO-WAIT FLOW-SHOP COM COLISÕES OCORRENDO APENAS NOS PROCESSADORES EXTREMOS

VI.1 - CONSIDERAÇÕES INICIAIS

Neste capítulo estuda-se problemas em scheduling do tipo no-wait flow-shop com a particularidade das colisões ocorrerem apenas no primeiro ou no último processador, isto é, nos processadores extremos. Um sistema de processamento com tal característica é denotado por FE.

Com a finalidade de introduzir o problema FE/no-wait/C_{max}, estudado na seção VI.5, apresenta-se inicialmente os problemas de Johnson e de Mitten, nas seções VI.2 e VI.3, respectivamente. Na seção VI.4 considera-se o resultado de Mitten aplicado no problema F/no-wait, p_j/C_{max}.

VI.2 - O PROBLEMA DE JOHNSON

Um dos problemas fundamentais em flow-shop scheduling é devido a Johnson [1954] e relaciona-se com problema em flow-shop com n tarefas (J_1, ... , J_n), dois processadores (M_1 e M_2) e cujo objetivo é minimizar a máxima data de término. Esse problema, denotado por F2// C_{max}, é denominado problema de Johnson e pode ser resolvido através da seguinte regra:

"Uma tarefa J_j precede uma tarefa J_k numa sequência ótima se $\min\{p_{1j}, p_{2k}\} \leq \min\{p_{2j}, p_{1k}\}$".

A seguir apresenta-se um algoritmo, O(nlogn), que implementa essa regra.

Algoritmo VI.1: JOHNSON

```
1   entrada: (p_{1,j}, p_{2j}),  j = 1,..., n
2   U := ∅ ; V := ∅  // conjuntos de índices, vazios //
3   para j=1 até n efetuar
4       se p_{1j} < p_{2j} então  U := U + {j} // incluir j em U //
5       caso contrário            V := V + {j} // incluir j em V //
6   ordenar U em ordem não decrescente de p_{1j}
        // j, k ∈ U, se j precede k então p_{1j} ≤ p_{1k} //
7   seja U = {u_1, ... , u_{|U|}}
8   ordenar V em ordem não crescente de p_{2j}
9   seja V = {v_1, ... v_{|V|}}
10  para i=1 até |U| efetuar
11      σ(i) := u_i
12  para i=1 até |V| efetuar
13      σ(i+|U|) := v_i
14  saída: S = (J_{σ(1)}, ... , J_{σ(n)})  Δ
```

Diz-se que um processador M_i é dominado quando, para qualquer permutação de tarefas, não ocorrer nenhuma colisão em M_i. Por outro lado, denomina-se gargalo a todo processador não dominado.

Considere o problema de Johnson estendido para 3 processadores, sendo o segundo dominado. Uma sequência ótima para esse problema obedece a seguinte regra, também devida a Johnson [1954]:

"Se o $\min \{p_{1k}, p_{3k}\} \geq \max \{p_{2k}\}$, $k=1,\ldots, n$

então uma tarefa J_i precede uma tarefa J_j numa sequência ótima se

$$\min \{p_{1i} + p_{2i} , p_{2j} + p_{3j}\} \leq \min \{p_{2i} + p_{3i} , p_{1j} + p_{2j}\}" .$$

Para implementar essa extensão deve-se efetuar as seguintes alterações no Algoritmo VI.1:

linha 4 : se $p_{1j} + p_{2j} < p_{2j} + p_{3j}$ então ...

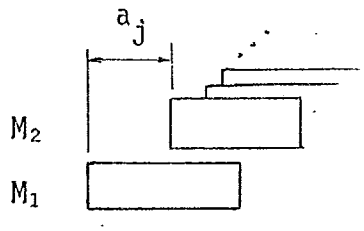
linha 6 : ordenar U em ordem não decrescente de $p_{1j} + p_{2j}$

linha 8 : ordenar V em ordem não crescente de $p_{2j} + p_{3j}$.

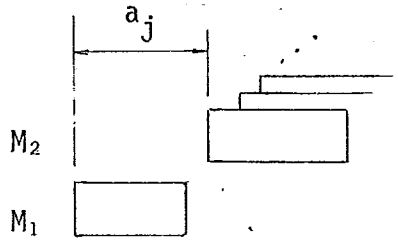
De acordo com Baker [1974], num sistema constituído com 3 processadores não dominados, se a plicação do Algoritmo VI.1 considerando-se separadamente os processadores M_1 e M_2 e os processadores M_2 e M_3, resultar numa mesma sequência S de tarefas, então essa sequência também é ótima para o problema completo, com os 3 processadores.

VI.3 - O PROBLEMA DE MITTEN

Uma extensão do problema de Johnson, devida a Mitten [1959], relaciona-se com a existência de atrasos de partida (a_j) e atrasos de parada (b_j), isto é, a partida da segunda operação de uma tarefa J_j não pode ocorrer antes de a_j unidades do início da primeira operação (atraso de partida) e, por outro lado, o término da segunda operação não pode ocorrer antes de b_j unidades do término da primeira operação (atraso de parada). Esse problema é denominado de Mitten.

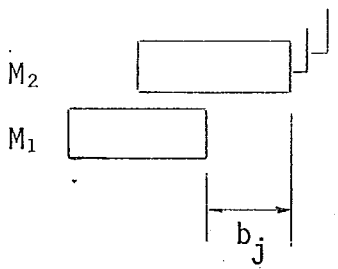


$a_j < p_{1j}$: possibilidade de sobreposição de operações

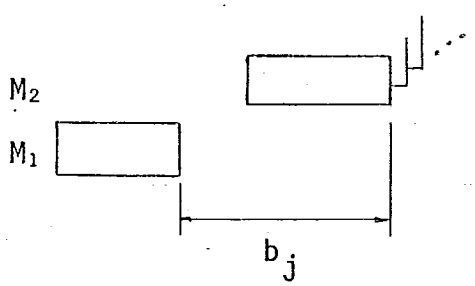


$a_j \geq p_{1j}$: impossibilidade de sobreposição de operações

Fig. VI.1: Atraso de Partida : a_j



$b_j < p_{2j}$: possibilidade de sobreposição de operações



$b_j \geq p_{2j}$: impossibilidade de sobreposição de operações

Fig. VI.2: Atraso de Parada : b_j

O procedimento, devido a Mitten [1959], que constroe uma sequência ótima para o problema de Mitten, pode ser implementado através do seguinte algoritmo de complexidade O(nlogn).

Algoritmo VI.2 : MITTEN

1 entrada: $(p_{1,j}, p_{2,j}, a_j, b_j)$, $j=1,\ldots,n$

2 $U := \emptyset$; $V := \emptyset$ // conjunto de índices, vazios //

3 para $j=1$ até n efetuar

4 $\quad y_j := \max \{a_j - p_{1j}, b_j - p_{2j}\}$

5 $\quad$ se $p_{1j} < p_{2j}$ então $U := U + \{j\}$

6 $\quad$ caso contrário $V := V + \{j\}$

7 ordenar U em ordem não decrescente de $p_{1j} + y_j$

8 seja $U = \{u_1,\ldots, u_{|u|}\}$

9 ordenar V em ordem não crescente de $p_{2j} + y_j$

10 seja $V = \{v_1,\ldots, v_{|v|}\}$

11 para $i=1$ até $|U|$ efetuar $v(i) := u_i$

12 para $i=1$ até $|V|$ efetuar $v(i+|U|) := v_i$

13 saída: $S = (J_{\sigma(1)}, \ldots, J_{\sigma(n)})$ Δ

O resultado de Mitten pode ser estendido para problemas do tipo permutation flow-shop com m processadores ($m > 2$), com colisões ocorrendo apenas nos processadores extremos (M_1 ou M_m).

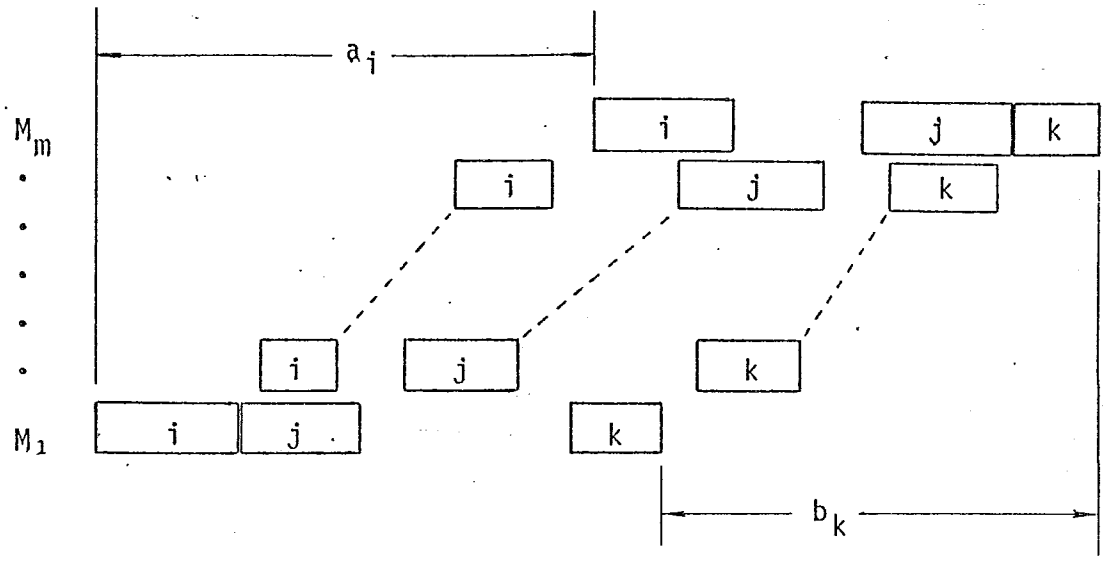


Fig. VI.3 - Colisões nos processadores extremos

VI.4 - O PROBLEMA F/no-wait, p_j/C_{max}

O problema F/no-wait, p_j/C_{max}, estudado por Pinho [1983], o qual caracterizou todo o conjunto de soluções ótimas, considera o tempo de processamento de cada operação de uma tarefa J_j constante para todos os processadores ($p_{1j} = p_{2j} = \ldots = p_{mj} = p_j$).

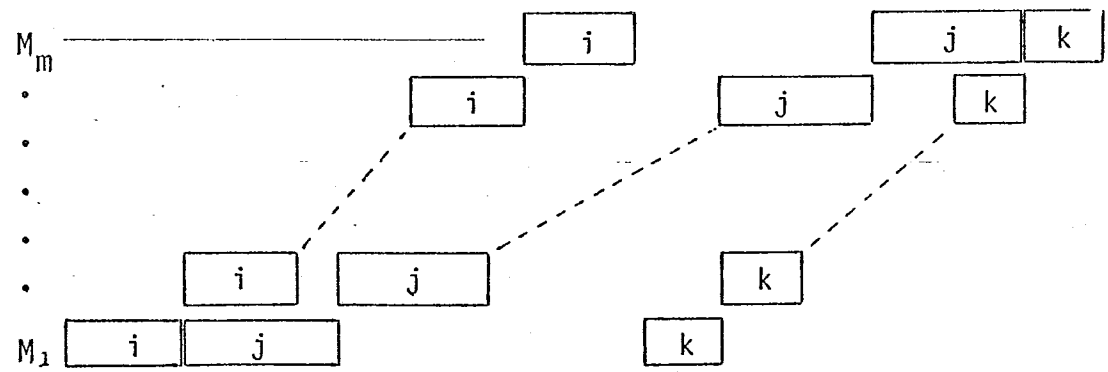


Fig. VI.4 - Schedule tipo no-wait flow-shop, com tempo de processamento constante

Evidentemente, neste problema a colisão ocorre no processador M_1 ou M_m e os schedules são de permutação, podendo-se portanto aplicar o resultado de Mitten, com atrasos de partida (a_j) e de chegada (b_j) determinados como segue:

$$a_j = b_j = (m-1) \cdot p_j \qquad \text{(VI.1)}$$

Assim sendo, considerando-se o processamento do Algoritmo VI.2 (Mitten), tem-se:

linha 4 : $y_j = \max \{a_j - p_{1j} , b_j - p_{2j}\}$

$= \max \{(m-1) \cdot p_j - p_j , (m-1) \cdot p_j - p_j)\}$

$= (m-2) \cdot p_j$

linhas 5 e 6 : $U = \emptyset$

$V = \{1, \ldots , n\}$

linha 9 : ordenação de V em ordem não crescente de

$p_{2j} + y_j = (m-1) \cdot p_j$

linha 13: sequência ótima : $S = (J_{\sigma(1)}, \ldots , J_{\sigma(n)}$ tal que:

$p_{\sigma(1)} \geq p_{\sigma(2)} \geq \cdots \geq p_{\sigma(n)}$

De acordo com o esperado, o resultado obtido a partir do procedimento de Mitten pertence ao conjunto das soluções obtidas por Pinho [1983],

$$\{(J_{\sigma(1)}, \ldots , J_{\sigma(k)}, \ldots , J_{\sigma(n)}) /$$

$$p_{\sigma(1)} \leq p_{\sigma(2)} \leq \cdots \leq p_{\sigma(k)} \geq p_{\sigma(k+1)} \geq \cdots \geq p_{\sigma(n)}\},$$

considerando-se k=1.

VI.5 - O PROBLEMA FE/no-wait/C_{max}

Nesta seção apresentam-se alguns resultados relacionados com o problema em scheduling do tipo no-wait flow-shop, com as colisões ocorrendo apenas no primeiro ou no último processador. Inicialmente, cada um desses casos é analisado.

VI.5.1. Colisão no Processador M_1:

Este caso pode ser visualizado através da seguinte figura:

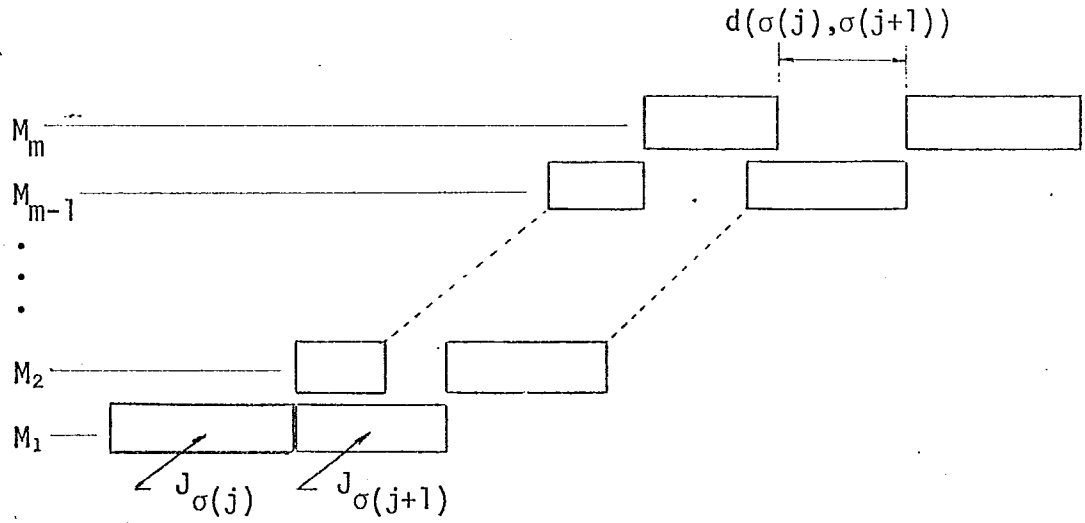


Fig. VI.5 - Colisão no Processador M_1

Claramente:

$$d(\sigma(j),\sigma(j+1)) = \sum_{i=1}^{m-1} p_{i,\sigma(j+1)} - \sum_{i=2}^{m} p_{i,\sigma(j)} \geq 0 \qquad (VI.2)$$

$$\therefore \sum_{i=1}^{m} p_{i,\sigma(j+1)} - p_{m,\sigma(j+1)} \geq \sum_{i=1}^{m} p_{i,\sigma(j)} - p_{1,\sigma(j)} \qquad (VI.3)$$

Por conveniência, considere:

$$\sum_{i=1}^{m} p_{i,\sigma(j)} = \Sigma\sigma(j)$$

$$\sum_{i=1}^{m} p_{i,\sigma(j)} - p_{m,\sigma(j)} = a_{\sigma(j)} \quad \text{(atraso de partida)}$$

$$\sum_{i=1}^{m} p_{i,\sigma(j)} - p_{1,\sigma(j)} = b_{\sigma(j)} \quad \text{(atraso de parada)}$$

Logo:

$$d(\sigma(j),\sigma(j+1) = a_{\sigma(j+1)} - b_{\sigma(j)} \geq 0 \qquad \text{(VI.4)}$$

VI.5.2. Colisão no Processador M_m

A figura abaixo mostra esse caso.

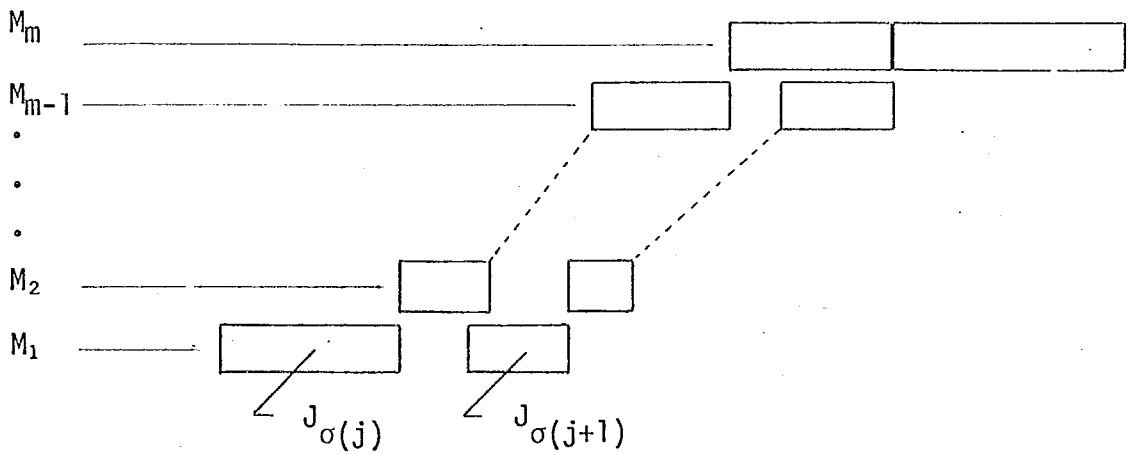


Fig. VI.6 - Colisão no Processador M_m

Claramente:

$$d(\sigma(j),\sigma(j+1) = 0 \qquad \text{(VI.5)}$$

$$a_{\sigma(j+1)} \leq b_{\sigma(j)} \qquad \text{(VI.6)}$$

A partir de (VI.4) - (VI.6) pode-se determinar o intervalo entre tarefas, de uma maneira geral, como segue:

$$d(\sigma(j),\sigma(j+1)) = \max \{0, \ a_{\sigma(j+1)} - b_{\sigma(j)}\} \qquad \text{(VI.7)}$$

VI.5.3. Determinação de $C_{max}(\sigma)$ e $\Sigma C(\sigma)$

É imediato verificar que:

$$C_{max}(\sigma) = \sum_{i=1}^{m-1} p_{i,\sigma(1)} + \sum_{j=1}^{n} p_{m,\sigma(j)} + \sum_{j=1}^{n-1} d(\sigma(j),\sigma(j+1)) \qquad \text{(VI.8)}$$

Como $\sum_{j=1}^{n} p_{m,\sigma(j)} = \sum_{j=1}^{n} p_{m,j}$, segue que:

$$C_{max}(\sigma) = a_{\sigma(1)} + \sum_{j=1}^{n} p_{m,j} + \sum_{j=1}^{n-1} d(\sigma(j),\sigma(j+1)) \qquad \text{(VI.9)}$$

Como $d(\sigma(j),\sigma(j+1)) \geq 0$, $j=1,\ldots, n-1$, segue que:

$$C_{max}(\sigma) \geq a_{\sigma(1)} + \sum_{j=1}^{n} p_{m.j} \qquad \text{(VI.10)}$$

Para a determinação de $\Sigma C(\sigma) = \sum_{j=1}^{n} C_{\sigma(j)}$, deve-se, evidentemente, conhecer os valores de $C_{\sigma(j)}$, $j=1,\ldots, n$:

$$C_{\sigma(1)} = d(\sigma(0),\sigma(1)) + p_{m,\sigma(1)}, \quad \text{onde} \quad d(\sigma(0),\sigma(1)) = \sum_{i=1}^{m-1} p_{i,\sigma(1)}$$

$$C_{\sigma(2)} = d(\sigma(0),\sigma(1)) + d(\sigma(1),\sigma(2)) + p_{m,\sigma(1)} + p_{m,\sigma(2)}$$

$$= \sum_{j=1}^{2} d(\sigma(j-1),\sigma(j)) + \sum_{j=1}^{2} p_{m,\sigma(j)}$$

$$\vdots$$

$$C_{\sigma(n)} = \sum_{j=1}^{n} d(\sigma(j-1),\sigma(j)) + \sum_{j=1}^{n} p_{m,\sigma(j)}$$

De uma forma geral:

$$C_{\sigma(k)} = \sum_{j=1}^{k} d(\sigma(j-1),\sigma(j)) + \sum_{j=1}^{k} p_{m,\sigma(j)} \quad , \quad k=1,\ldots, n \qquad \text{(VI.11)}$$

Logo:

$$\Sigma C(\sigma) = \sum_{k=1}^{n} \left(\sum_{j=1}^{k} d(\sigma(j-1),\sigma(j)) + \sum_{j=1}^{k} p_{m,\sigma(j)} \right)$$

$$\therefore \Sigma C(\sigma) = \sum_{k=1}^{n} \sum_{j=1}^{k} d(\sigma(j-1),\sigma(j)) + \sum_{k=1}^{n} \sum_{j=1}^{k} p_{m,\sigma(j)} \qquad \text{(VI.12)}$$

VI.5.4. Determinação de Permutação que Minimiza C_{max}

Os teoremas VI.1 e VI.2 apresentam condições necessárias para uma permutação de tarefas ser solução ótima do problema FE/no-wait/C_{max} .

Teorema VI.1: Se σ é uma permutação de tarefas tal que

$$b_{\sigma(j)} \geq a_{\sigma(j+1)} \quad , \quad j = 1,\ldots, n-1$$

$$\text{e } a_{\sigma(1)} \leq a_j \quad , \quad j = 1,\ldots, n$$

$$\text{então } C_{max}(\sigma) = \text{Min } C_{max}$$

prova:

De (VI.7):

$$d(\sigma(j),\sigma(j+1)) = \max \{0,\ a_{\sigma(j+1)} - b_{\sigma(j)}\},\quad j=1,\ldots,n-1 \qquad (i)$$

Por hipótese:

$$a_{\sigma(j+1)} - b_{\sigma(j)} \leq 0\ ,\quad j=1,\ldots,n-1 \qquad (ii)$$

De (i) e (ii):

$$d(\sigma(j),\sigma(j+1)) = 0,\quad j=1,\ldots,n-1 \qquad (iii)$$

De (VI.9) e (iii):

$$C_{max}(\sigma) = a_{\sigma(1)} + \sum_{j=1}^{n} p_{m,j} \qquad (iv)$$

De (VI.10), para qualquer permutação τ, tem-se:

$$C_{max}(\tau) \geq a_{\tau(1)} + \sum_{j=1}^{n} p_{m,j} \qquad (v)$$

Por hipótese:

$$a_{\sigma(1)} \leq a_j\ ,\quad j=1,\ldots,n$$

Logo:

$$a_{\sigma(1)} \leq a_{\tau(1)}\ ,\ \text{para qualquer permutação } \tau \qquad (vi)$$

De (iv) - (vi), segue que:

$$C_{max}(\sigma) \leq C_{max}(\tau),\ \text{para qualquer permutação } \tau$$

Logo:

$$C_{max}(\sigma) = \text{Min } C_{max}\quad \Delta$$

Os resultados dos lemas a seguir são utilizados na prova do Teorema VI.2.

Lema VI.1: Se $b_{\sigma(j)} \geq a_{\sigma(j+1)}$, $j=1,\ldots, n-1$

e $p_{1,\sigma(j)} \geq p_{m,\sigma(j)}$, $j=1,\ldots, n-1$

então $b_{\sigma(j)} \geq a_{\sigma(k)}$, $k > j$, $j=1,\ldots, n-1$

prova: (por indução)

(a) Para $k = j+1$:

O resultado é imediato, pois:

por hipótese $b_{\sigma(j)} \geq a_{\sigma(j+1)} = a_{\sigma(k)}$

(b) Supor válido para $k = j+\ell$, $\ell \leq n-j-1$:

$b_{\sigma(j)} \geq a_{\sigma(k)} = a_{\sigma(j+\ell)}$

$\therefore \quad \Sigma\sigma(j) - p_{1,\sigma(j)} \geq \Sigma\sigma(j+\ell) - p_{m,\sigma(j+\ell)}$ (i)

(c) Para $k = j+\ell+1$:

Por hipótese:

$b_{\sigma(j+\ell)} \geq a_{\sigma(j+\ell+1)}$

$\therefore \quad \Sigma\sigma(j+\ell) - p_{1,\sigma(j+\ell)} \geq \Sigma\sigma(j+\ell+1) - p_{m,\sigma(j+\ell+1)}$ (ii)

Por hipótese:

$p_{1,\sigma(j+\ell)} \geq p_{m,\sigma(j+\ell)}$ (iii)

De (i) e (iii)

$\Sigma\sigma(j) - p_{1,\sigma(j)} \geq \Sigma\sigma(j+\ell) - p_{1,\sigma(j+\ell)}$ (iv)

De (ii) e (iv):

$$\Sigma\sigma(j) - p_{1,\sigma(j)} \geq \Sigma\sigma(j+\ell+1) - p_{m,\sigma(j+\ell+1)}$$

$$\therefore \quad b_{\sigma(j)} \geq a_{\sigma(j+\ell+1)}$$

Como $k = j+\ell+1$, segue, finalmente:

$$b_{\sigma(j)} \geq a_{\sigma(k)} \quad \Delta$$

Lema VI.2: Se $b_{\sigma(j)} \geq a_{\sigma(j+1)}$

e $p_{1,\sigma(j)} \geq p_{m,\sigma(j)} \geq p_{m,\sigma(j+1)}$, $j=1,\ldots, n-1$

então $\Sigma\sigma(1) \geq \Sigma\sigma(2) \geq \ldots \geq \Sigma\sigma(n)$

prova:

Do Lema VI.1:

$$b_{\sigma(j)} \geq a_{\sigma(k)} , \quad k > j, \quad j = 1,\ldots, n-1$$

$$\therefore \quad \Sigma\sigma(j) - p_{1,\sigma(j)} \geq \Sigma\sigma(k) - p_{m,\sigma(k)}$$

$$\therefore \quad \Sigma\sigma(j) - \Sigma\sigma(k) \geq p_{1,\sigma(j)} - p_{m,\sigma(k)} \qquad (i)$$

Por nipótese:

$$p_{1,\sigma(j)} - p_{m,\sigma(k)} \geq 0, \quad k > j \qquad (ii)$$

De (i) e (ii):

$$\Sigma\sigma(j) - \Sigma\sigma(k) \geq 0, \qquad j < k, \quad j = 1,\ldots, n-1$$

Logo:

$$\Sigma\sigma(1) \geq \Sigma\sigma(2) \geq \ldots \geq \Sigma\sigma(n) \quad \Delta$$

Lema VI.3: Se $b_{\sigma(j-1)} \geq a_{\sigma(j)}$,

$$p_{1,\sigma(j-1)} \geq p_{m,\sigma(j-1)} \geq p_{m,\sigma(j)} \quad \text{e}$$

$$p_{m,\sigma(j-1)} < p_{1,\sigma(j)} \ , \quad j = 2,\ldots, n$$

então $b_{\sigma(j)} < a_{\sigma(k)}$, $j > k$, $j = 2,\ldots, n$

prova: (por indução)

(a) Para $k = j-1$

Do Lema VI.2:

$$\Sigma\sigma(j-1) \geq \Sigma\sigma(j)$$

$$\therefore \Sigma\sigma(k) - \Sigma\sigma(j) \geq 0 \qquad \text{(i)}$$

Por hipótese:

$$p_{1,\sigma(j)} > p_{m,\sigma(j-1)} = p_{m,\sigma(k)}$$

$$\therefore p_{1,\sigma(j)} - p_{m,\sigma(k)} > 0 \qquad \text{(ii)}$$

Somando-se (i) e (ii):

$$\Sigma\sigma(k) - \Sigma\sigma(j) + p_{1,\sigma(j)} - p_{m,\sigma(k)} > 0$$

$$\therefore \Sigma\sigma(j) - p_{1,\sigma(j)} < \Sigma\sigma(k) - p_{m,\sigma(k)}$$

$$\therefore b_{\sigma(j)} < a_{\sigma(k)} \ , \quad j > k$$

(b) Supor válida para $k = j-\ell$, $\ell < j-1$:

$$b_{\sigma(j)} < a_{\sigma(k)} = a_{\sigma(j-\ell)}$$

$$\therefore \Sigma\sigma(j) - p_{1,\sigma(j)} < \Sigma\sigma(j-\ell) - p_{m,\sigma(j-\ell)} \qquad \text{(iii)}$$

(c) Para $k = j-\ell-1$:

Por hipótese:

$$b_{\sigma(j-\ell-1)} \geq a_{\sigma(j-\ell)}$$

$$\therefore \Sigma\sigma(j-\ell-1) - p_{1,\sigma(j-\ell-1)} \geq \Sigma\sigma(j-\ell) - p_{m,\sigma(j-\ell)} \qquad \text{(iv)}$$

De (iii) e (iv):

$$\Sigma\sigma(j) - p_{1,\sigma(j)} < \Sigma\sigma(j-\ell-1) - p_{1,\sigma(j-\ell-1)} \qquad \text{(v)}$$

Por hipótese:

$$p_{1,\sigma(j-\ell-1)} \geq p_{m,\sigma(j-\ell-1)} \qquad \text{(vi)}$$

De (v) e (vi)

$$\Sigma\sigma(j) - p_{1,\sigma(j)} < \Sigma\sigma(j-\ell-1) - p_{m,\sigma(j-\ell-1)}$$

$$\therefore b_{\sigma(j)} < a_{\sigma(j-\ell-1)}$$

Como, $k = j-\ell-1$, segue, finalmente:

$$b_{\sigma(j)} < a_{\sigma(k)} \quad \Delta$$

Lema VI.4: Se σ é uma permutaçao tal que

$$b_{\sigma(j)} \geq a_{\sigma(j+1)}$$

e $p_{1,\sigma(j)} \geq p_{m,\sigma(j)} \geq p_{m,\sigma(j+1)}$, $j = 1,\ldots, n-1$

então o schedule $S = (J_{\sigma(n)})$

minimiza C^1_{max}

prova:

Seja $S(1) = (J_{\sigma(j)})$, $j = 1,\ldots, n$

Neste caso:

$$C_{max}^{1} = \Sigma\sigma(j) \text{ , } j = 1,\ldots, n$$

Do Lema VI.2:

$$\Sigma\sigma(n) \leq \Sigma\sigma(j) \text{ , } j = 1,\ldots, n$$

Logo:

$$\text{Min } C_{max}^{1} = \Sigma\sigma(n)$$

Portanto:

$$S^*(1) = (J_{\sigma(n)}) \ \Delta$$

Lema VI.5: Se σ é uma permutação tal que

$$b_{\sigma(j)} \geq a_{\sigma(j+1)} \text{ ,}$$

$$p_{1,\sigma(j)} \geq p_{m,\sigma(j)} \geq p_{m,\sigma(j+1)} \quad \text{e}$$

$$p_{1,\sigma(j)} \geq p_{1,\sigma(j+1)} \geq p_{m,\sigma(j)}$$

então o schedule $S = (J_{\sigma(n-1)} , J_{\sigma(n)})$

minimiza C_{max}^{2}

prova:

Seja $S(2) = (J_{\sigma(j)}, J_{\sigma(k)})$, $j \neq k \quad j,k = 1,\ldots, n$

Neste caso:

$$C^{2}_{max} = \Sigma\sigma(j) + d(\sigma(j),\sigma(k)) + p_{m,\sigma(k)}$$

De (VI.7):

$$C^{2}_{max} = \Sigma\sigma(j) + \max\{0,\ a_{\sigma(k)} - b_{\sigma(j)}\} + p_{m,\sigma(k)} \qquad (i)$$

Existem dois casos a considerar:

Caso I: $j < k$

Do Lema VI.1:

$$a_{\sigma(k)} - b_{\sigma(j)} \leq 0$$

De (i):

$$C^{2}_{max}(I) = \Sigma\sigma(j) + p_{m,\sigma(k)} \qquad (ii)$$

Do Lema VI.2:

$$\Sigma\sigma(1) \geq \Sigma\sigma(2) \geq \ldots \geq \Sigma\sigma(n-1) \geq \Sigma\sigma(n) \qquad (iii)$$

Por hipótese:

$$p_{m,\sigma(1)} \geq \cdots \geq p_{m,\sigma(n-1)} \geq p_{m,\sigma(n)} \qquad (iv)$$

De (ii) - (iv), como neste caso $j < k$, segue que:

$$\text{Min } C^{2}_{max}(I) = \Sigma\sigma(n-1) + p_{m,\sigma(n)} \qquad (v)$$

Logo:

$$S^{*}(2)_{I} = (J_{\sigma(n-1)}\ ,\ \sigma(n))$$

Caso II: $j > k$

Do Lema VI.3:

$$a_{\sigma(k)} - b_{\sigma(j)} > 0$$

De (i):

$$C^2_{max}(II) = \Sigma\sigma(j) + a_{\sigma(k)} - b_{\sigma(j)} + p_{m,\sigma(k)}$$

$$\therefore\ C^2_{max}(II) = \Sigma\sigma(j) + \Sigma\sigma(k) - p_{m,\sigma(k)} - \Sigma\sigma(j) + p_{1,\sigma(j)} + p_{m,\sigma(k)}$$

$$\therefore\ C^2_{max}(II) = \Sigma\sigma(k) + p_{1,\sigma(j)} \qquad \text{(vi)}$$

Por hipótese:

$$p_{1,\sigma(1)} \geq \cdots \geq p_{1,\sigma(n-1)} \geq p_{1,\sigma(n)} \qquad \text{(vii)}$$

De (iii), (vi) e (vii), como neste caso $k < j$, segue que:

$$\text{Min } C^2_{max}(II) = \Sigma\sigma(n-1) - p_{1,\sigma(n)} \qquad \text{(viii)}$$

Logo:

$$S^*(2)_{II} = (J_{\sigma(n)}\ ,\ J_{\sigma(n-1)}$$

Comparação do Caso I com o Caso II:

De (v) e (viii), como por hipótese,

$p_{1,\sigma(n)} \geq p_{m,\sigma(n)}$, segue que:

$$\text{Min } C^2_{max}(I) \leq \text{Min } C^2_{max}(II)$$

Portanto:

$$S^*(2) = (J_{\sigma(n-1)}\ ,\ J_{\sigma(n)})\quad \Delta$$

Teorema VI.2: Se σ é uma permutação de tarefas tal que

$$b_{\sigma(j)} \geq a_{\sigma(j+1)} \quad \text{e}$$

$$p_{1,\sigma(j)} \geq p_{1,\sigma(j+1)} \geq p_{m,\sigma(j)} \geq p_{m,\sigma(j+1)}$$

então $C_{max}(\sigma) = \text{Min } C_{max}$

prova: (por indução, no número k de tarefas a serem processadas)

(a) Para k=1

Do Lema VI.4:

$$\text{Min } C^{1}_{max} = \Sigma\sigma(n) \ , \quad S^*(1) = (J_{\sigma(n)})$$

(b) Para k=2

Do Lema VI.5:

$$\text{Min } C^{2}_{max} = \Sigma\sigma(n-1) + p_{m,\sigma(n)} \ , \quad S^*(2) = (J_{\sigma(n-1)}, J_{\sigma(n)})$$

(c) Supor válido para k tarefas, $k < n$:

$$\text{Min } C^{k}_{max} = \Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} \ , \quad 2 < k < n \qquad \text{(i)}$$

$$S^*(k) = (J_{\sigma(n-k+1)} \ , J_{\sigma(n-k+2)} \ , \ \cdots \ , J_{\sigma(n)})$$

(d) Para k+1 tarefas

Existem 3 casos a considerar:

Caso I:

$$S(k+1)_I = (J_{\sigma(j)}, J_{\sigma(n-k+1)}, J_{\sigma(n-k+2)}, \ \cdots \ , J_{\sigma(n)}),$$

$$j = 1, \ldots, n-k$$

Neste caso:

$$C^{k+1}_{max}(I) = \Sigma\sigma(j) + d(\sigma(j),\sigma(n-k+1)) + D(\sigma(n-k+1),\sigma(n)) \qquad \text{(ii)}$$

Do Lema VI.1, como $j < n-k+1$, segue que:

$$a_{\sigma(n-k+1)} - b_{\sigma(j)} < 0$$

Logo, de (VI.7):

$$d(\sigma(j),\sigma(n-k+1)) = 0 \qquad \text{(iii)}$$

De (i):

$$D(\sigma(n-k+1),\sigma(n)) = \sum_{j=n-k+1}^{n} p_{m,\sigma(j)} \qquad \text{(iv)}$$

Substituindo-se (iii) e (iv) em (ii):

$$C_{max}^{k+1}(I) = \Sigma\sigma(j) + \sum_{j=n-k+1}^{n} p_{m,\sigma(j)}, \quad j = 1,\ldots, n-k$$

Do Lema VI.2:

$$\Sigma\sigma(1) \geq \ldots \geq \Sigma\sigma(n-k)$$

Logo:

$$\text{Min } C_{max}^{k+1}(I) = \Sigma\sigma(n-k) + \sum_{j=n-k+1}^{n} p_{m,\sigma(j)} \qquad \text{(v)}$$

Caso II:

$$S(k+1)_{II} = (J_{\sigma(n-k+1)}, J_{\sigma(n-k+2)}, \ldots, J_{\sigma(n)}, J_{\sigma(j)}),$$

$$j = 1,\ldots, n-k$$

Neste caso, de (i), segue que:

$$C_{max}^{k+1}(II) = \Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} + d(\sigma(n),\sigma(j)) + p_{m,\sigma(j)} \qquad \text{(vi)}$$

De (VI.7):

$$d(\sigma(n),\sigma(j)) = \max \{0,\ a_{\sigma(j)} - b_{\sigma(n)}\},\quad j < n$$

Do Lema VI.3:

$$a_{\sigma(j)} - b_{\sigma(n)} > 0$$

Logo:

$$d(\sigma(n),\sigma(j)) = a_{\sigma(j)} - b_{\sigma(n)} \qquad \text{(vii)}$$

Substituindo-se (vii) em (vi):

$$C_{max}^{k+1}(II) = \Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} + a_{\sigma(j)} - b_{\sigma(n)} + p_{m,\sigma(j)}$$

$$\therefore C_{max}^{k+1}(II) = \Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} + \Sigma\sigma(j) - p_{m,\sigma(j)} - \Sigma\sigma(n) +$$

$$+ p_{1,\sigma(n)} + p_{m,\sigma(j)}$$

$$\therefore C_{max}^{k+1}(II) = \underbrace{\Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} - \Sigma\sigma(n) + p_{1,\sigma(n)}}_{\text{constante}} + \Sigma\sigma(j)$$

Do Lema VI.2:

$$\Sigma\sigma(1) \geq \ldots \geq \Sigma\sigma(n-k)$$

Logo:

$$\text{Min } C_{max}^{k+1}(II) = \Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} - \Sigma\sigma(n) + p_{1,\sigma(n)} + \Sigma\sigma(n-k)$$

(viii)

Caso III:

$$S(k+1)_{III} = (J_{\sigma(n-k+1)}, \cdots, J_{\sigma(n-\ell)}, J_{\sigma(j)}, J_{\sigma(n-\ell+1)}, \cdots, J_{\sigma(n)})$$

Neste caso, de (i), segue que:

$$C_{max}^{k+1}(III) = \Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n-\ell} p_{m,\sigma(j)} + d(\sigma(n-\ell),\sigma(j)) + p_{m,\sigma(j)} +$$

$$+ d(\sigma(j),\sigma(n-\ell+1)) + \sum_{j=n-\ell+1}^{n} p_{m,\sigma(j)} \qquad (ix)$$

De (VI.7):

$$d(\sigma(n-\ell),\sigma(j)) = \max\{0, a_{\sigma(j)} - b_{\sigma(n-\ell)}\}, \quad j < n-\ell$$

Do Lema VI.3:

$$a_{\sigma(j)} - b_{\sigma(n-\ell)} > 0$$

Logo:

$$d(\sigma(n-\ell),\sigma(j)) = a_{\sigma(j)} - b_{\sigma(n-\ell)} \qquad (x)$$

De (VI.7):

$$d(\sigma(j),\sigma(n-\ell+1)) = \max\{0, a_{\sigma(n-\ell+1)} - b_{\sigma(j)}\}, \quad j < n-\ell$$

Do Lema VI.1:

$$a_{\sigma(n-\ell+1)} - b_{\sigma(j)} \leq 0$$

Logo:

$$d(\sigma(j),\sigma(n-\ell+1)) = 0 \qquad (xi)$$

Substituindo-se (x) e (xi) em (ix):

$$C_{max}^{k+1}(III) = \Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} + a_{\sigma(j)} - b_{\sigma(n-\ell)} + p_{m,\sigma(j)}$$

$$\therefore \quad C_{max}^{k+1}(III) = \Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} + \Sigma\sigma(j) - p_{m,\sigma(j)} - \Sigma\sigma(n-\ell) +$$

$$+ p_{1,\sigma(n-\ell)} + p_{m,\sigma(j)}$$

$$\therefore \quad C_{max}^{k+1}(III) = \underbrace{\Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} - \Sigma\sigma(n-\ell) + p_{1,\sigma(n-\ell)}}_{\text{constante}} + \Sigma\sigma(j)$$

Do Lema VI.2:

$$\Sigma\sigma(1) \geq \ldots \geq \Sigma\sigma(n-k)$$

Logo:

$$\text{Min } C_{max}^{k+1}(III) = \Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} - \Sigma\sigma(n-\ell) +$$

$$+ p_{1,\sigma(n-\ell)} + \Sigma\sigma(n-k) \qquad (xii)$$

Comparação A: Caso I com Caso II

De (v) e (viii) para provar que

$$\text{Min } C_{max}^{k+1}(II) \geq \text{Min } C_{max}^{k+1}(I)$$

basta mostrar que:

$$\Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} - \Sigma\sigma(n) + p_{1,\sigma(n)} - \sum_{j=n-k+1}^{n} p_{m,\sigma(j)} \geq 0$$

De fato, do Lema VI.3, segue que:

$$a_{\sigma(n-k+1)} > b_{\sigma(n)} \text{ , pois } n-k+1 < n$$

$$\therefore \quad \Sigma\sigma(n-k+1) - p_{m,\sigma(n-k+1)} - \Sigma\sigma(n) + p_{1,\sigma(n)} > 0$$

Como:

$$p_{m,\sigma(n-k+1)} = \sum_{j=n-k+1}^{n} p_{m,\sigma(j)} - \sum_{j=n-k+2}^{n} p_{m,\sigma(j)}$$

segue o resultado esperado:

$$\Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} - \Sigma\sigma(n) - p_{1,\sigma(n)} - \sum_{j=n-k+1}^{n} p_{m,\sigma(j)} > 0$$

Comparação B: Caso I com Caso III

Do Lema VI.1, tem-se:

$$b_{\sigma(n-k+1)} \geq a_{\sigma(n-\ell)} \quad , \quad \text{pois } n-k+1 < n-\ell$$

$$\therefore \Sigma\sigma(n-k+1) - p_{1,\sigma(n-k+1)} - \Sigma\sigma(n-\ell) + p_{m,\sigma(n-\ell)} \geq 0$$

Por hipótese:

$$p_{1,\sigma(j)} \geq p_{m,\sigma(j)} \quad , \quad j = 1,\ldots, n-1$$

Logo:

$$\Sigma\sigma(n-k+1) - p_{m,\sigma(n-k+1)} - \Sigma\sigma(n-\ell) + p_{1,\sigma(n-\ell)} \geq 0$$

$$\therefore \Sigma\sigma(n-k+1) - \sum_{j=n-k+1}^{n} p_{m,\sigma(j)} + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} - \Sigma\sigma(n-\ell) +$$

$$+ p_{1,\sigma(n-\ell)} + \Sigma\sigma(n-k) - \Sigma\sigma(n-k) \geq 0$$

$$\therefore \Sigma\sigma(n-k+1) + \sum_{j=n-k+2}^{n} p_{m,\sigma(j)} - \Sigma\sigma(n-\ell) + p_{1,\sigma(n-\ell)} + \Sigma\sigma(n-k) \geq$$

$$\Sigma\sigma(n-k) + \sum_{j=n-k+1}^{n} p_{m,\sigma(j)}$$

De (v) e (xiii), tem-se, finalmente:

$$\text{Min } C_{max}^{k+1} \text{ (III)} \geq \text{Min } C_{max}^{k+1} \text{ (I)}$$

Das comparações A e B segue que o schedule

$$S = (J_{\sigma(n-k)} , J_{\sigma(n-k+1)} , \cdots , J_{\sigma(n)})$$

minimiza C_{max}^{k+1}

Para k+1 = n, tem-se:

$S = (J_{\sigma(1)}, \cdots , J_{\sigma(n)})$ minimiza C_{max}^{n}

ou seja:

$C_{max}(\sigma) = \text{Min } C_{max}$ Δ

CAPITULO VII

UMA EXTENSÃO DO CONCEITO NO-WAIT FLOW-SHOP

VII.1 - CONSIDERAÇÕES INICIAIS

Conforme definido na seção II.8.4, nos problemas em scheduling do tipo no-wait flow-shop não é permitida a formação de filas de espera de operações aguardando processamento, ou seja, o tempo entre o início (término) de operações sucessivas, $O_{i,j}$ e $O_{i+1,j}$, é igual ao tempo de processamento $p_{i,j}$ ($p_{i+1,j}$), $i = 1,\ldots, m-1$, $j = 1,\ldots, n$.

Por outro lado, o conceito de atraso de partida e de parada apresentado na seção VI.3, considerando-se apenas dois processadores, pode ser estendido para um número m, qualquer, de processadores, como é mostrado a seguir:

Considere uma tarefa J_j, composta das operações $O_{1,j},\ldots, O_{m,j}$. Denomina-se atraso de partida (parada) da operação O_{ij}, denotando-se por $\alpha_{i,j}$ ($\beta_{i,j}$), o intervalo de tempo entre o início (término) da operação $O_{m,j}$ e o início (término)da operação $O_{i,j}$, $i = 1,\ldots, m$. Em ambos os casos, considere positivo o sentido da direita para a esquerda, como mostra a figura a seguir:

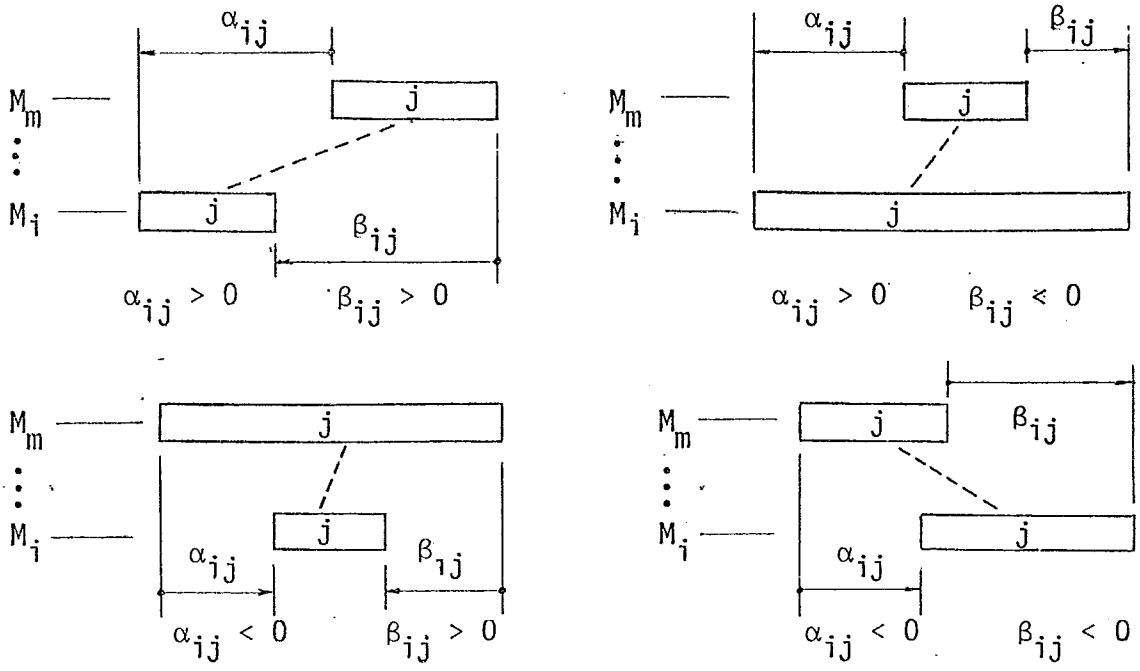


Fig. VII.1 - Atraso de Partida (α_{ij}) e de Parada (β_{ij})

A partir da Fig. acima é imediato verificar que:

$$\alpha_{i,j} + p_{m,j} = \beta_{i,j} + p_{i,j} \qquad \text{(VII.1)}$$

$$\alpha_{i,j} - \beta_{i,j} = p_{i,j} - p_{m,j} \qquad \text{(VII.2)}$$

Manteve-se a denominação atraso de partida e de parada, não obstante algumas situações não resultar propriamente em atraso.

De um certo sentido a extensão acima pode ser considerada também uma extensão do conceito no-wait, podendo este ser definido através do seguinte caso particular:

$$\alpha_{k,j} = \sum_{i=k}^{m-1} p_{i,j} \quad , \quad k = 1,\ldots, m-1 \qquad \text{(VII.3)}$$

ou, equivalentemente:

$$\beta_{k,j} = \sum_{i=k+1}^{m} p_{i,j} \quad , \quad k = 1,\ldots, m-1 \qquad \text{(VII.4)}$$

Considere $d(\sigma(j-1),\sigma(j))$ o intervalo entre duas tarefas consecutivas de uma permutação σ de tarefas.

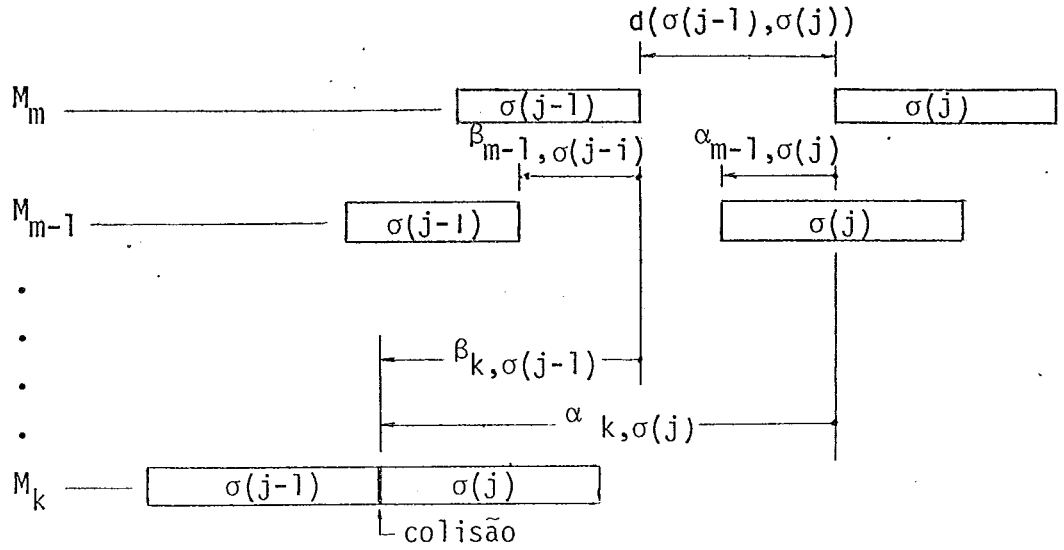


Fig. VII.2 - Intervalo $d(\sigma(j-1),\sigma(j))$

A partir da Fig. VII.2 é possível verificar que se a colisão ocorre num processador M_k então

$$d(\sigma(j-1),\sigma(j)) = \alpha_{k,\sigma(j)} - \beta_{k,\sigma(j-1)}$$

onde $$\alpha_{k,\sigma(j)} - \beta_{k,\sigma(j-1)} = \max_{1 \leq i \leq m} \{\alpha_{i,\sigma(j)} - \beta_{i,\sigma(j-1)}\} \qquad \text{(VII.5)}$$

No caso da colisão ocorrer em M_m, de (VII.5), segue que:

$$d(\sigma(j-1),\sigma(j)) = \alpha_{m,\sigma(j)} - \beta_{m,\sigma(j)} .$$

Como, de acordo com a definição, $\alpha_{m,\sigma(j)} = \beta_{m,\sigma(j)} = 0$ então segue o resultado esperado para esse caso:

$$d(\sigma(j-1),\sigma(j)) = 0$$

Por conveniência considere pseudo-tarefas $J_0 = J_{\sigma(0)}$ e $J_{n+1} = J_{\sigma(n+1)}$, tal que $\beta_{i,\sigma(0)} = \alpha_{i,\sigma(n+1)} = 0$, $i = 1,\ldots, m$. Logo, de acordo com (VII.5), tem-se:

$$d(\sigma(0),\sigma(1)) = \max_{1\leq i\leq m} \{\alpha_{i,\sigma(1)}\} \qquad \text{(VII.6)}$$

$$d(\sigma(n),\sigma(n+1)) = \max_{1\leq i\leq m} \{-\beta_{i,\sigma(n)}\} \qquad \text{(VII.7)}$$

A determinação de $C_{max}(\sigma)$ pode ser feita como segue:

$$C_{max}(\sigma) = \sum_{j=1}^{n} p_{m,\sigma(j)} + \sum_{j=1}^{n+1} d(\sigma(j-1),\sigma(j)) \qquad \text{(VII.8)}$$

Como:

$$\sum_{j=1}^{n} p_{m,\sigma(j)} = \sum_{j=1}^{n} p_{m,j} ,$$

isto é, independe da permutação de tarefas, segue que uma permutação σ^* que minimize C_{max} é tal que

$$\sum_{j=1}^{n+1} d(\sigma^*(j-1),\sigma^*(j)) \leq \sum_{j=1}^{n+1} d(\sigma(j-1),\sigma(j)),$$

para qualquer permutação σ de tarefas.

O problema em scheduling, do tipo flow-shop, considerando-se a extensão do conceito no-wait e cujo objetivo é minimizar a máxima data de término, denotado por F/ext-no-wait/C_{max}, pode ser formulado como um problema de Caixeiro Viajante e portanto pertence à classe NP-Difícil. No entanto, apresenta-se nas seções seguintes dois casos particulares que podem ser resolvidos em tempo polinomial.

VII.2 - O PROBLEMA F2/ext-no-wait/C_{max}

Sabe-se que uma solução para o problema F2/ /no-wait/C_{max} pode ser encontrada utilizando-se um algoritmo devido a Gilmore e Gomory [1964], em tempo O(nlogn).

Mostra-se a seguir que o problema F2/ext-no--wait/C_{max}, portanto uma extensão do problema acima, também pode ser resolvido através do referido algoritmo.

De fato, o algoritmo de Gilmore e Gomory resolve um caso particular do Problema de Caixeiro Viajante, com distâncias c_{jk}, entre a j e a k-ésima cidade, dada por:

$$c_{jk} = \begin{cases} \int_{b_j}^{a_k} f(x)\,dx \;, & \text{se } b_j < a_k \\ \int_{a_k}^{b_j} g(x)\,dx \;, & \text{se } b_j \geq a_k \end{cases} \qquad \text{(VII.9)}$$

onde:
- f e g são funções integráveis, satisfazendo $f(y) + g(x) \geq 0$, para toda variável de estado x;
- a_j, valor da variável de estado, quando se parte da j-ésima cidade;
- b_j, valor da variável de estado, quando se chega a qualquer cidade partindo da j-ésima cidade;
- c_{jk}, valor necessário adicionar à variável de estado com valor b_j, para torná-lo igual a a_k.

No problema F2/ext-no-wait/C_{max}, considere:

$a_j = \alpha_{1,j}$ e $b_j = \beta_{1,j}$, para todo $J_j \in J$.

Considere, também pseudo-tarefas J_0 e J_{n+1} , tais que :

$b_0 = a_{n+1} = 0$.

Assim sendo, segue diretamente de (VII.5) que

$$d(j,k) = \max (0, a_k - b_j), \quad j \neq k, \forall J_j, J_k \in J. \qquad \text{(VII.10)}$$

Por outro lado, considerando-se $f(x) = 1$ e $g(x) = 0$ em (VII.9), tem-se:

$$c_{jk} = \begin{cases} a_k - b_j & , \text{ se } b_j < a_k \\ 0 & , \text{ se } b_j \geq a_k \end{cases} \qquad \text{(VII.11)}$$

De (VII.10) e (VII.11), segue que:

$$d(j,k) = c_{jk} \qquad \text{(VII.12)}$$

Com as considerações acima, fica evidenciada a transformação do problema F2/ext-no-wait/C_{max}, no caso particular do Problema de Caixeiro Viajante que pode ser resolvido pelo algoritmo de Gilmore e Gomory, em tempo O(nlogn).

VII.3 - O PROBLEMA F/ext-no-wait, $|\{C_k\}| < L/C_{max}$

Mostra-se nesta seção que o problema em scheduling do tipo ext-no-wait flow-shop, com n tarefas pertencendo a um número limitado de classes, C_k, e cujo objetivo é minimizar a máxima data de término, denotado por F/ext-no-wait, $|\{C_k\}| < L/C_{max}$, pode ser resolvido em tempo polinomial, utilizando-se algoritmos devidos a Szwarcfiter [1984A].

Diz-se que duas tarefas, J_a e J_b são <u>similares</u>, denotando-se por $J_a \simeq J_b$, quando:

$$\alpha_{i,a} = \alpha_{i,b} \ , \quad i = 1,\ldots, m$$

$$\beta_{i,a} = \beta_{i,b} \ , \quad i = 1,\ldots, m$$

<u>Lema VII.1</u>: Se J_a e J_b são tarefas similares então

$$p_{i,a} - p_{i,b} = p_{m,a} - p_{m,b} \ , \quad i = 1,\ldots, m$$

<u>prova</u>:

De (VII.2), para i=1,..., m:

$$\alpha_{i,a} - \beta_{i,a} = p_{i,a} - p_{m,a} \qquad (i)$$

$$\alpha_{i,b} - \beta_{i,b} = p_{i,b} - p_{m,b} \qquad (ii)$$

Por hipótese, para i=1,..., m:

$$\alpha_{i,a} = \alpha_{i,b} \qquad (iii)$$

$$\beta_{i,a} = \beta_{i,b} \qquad (iv)$$

De (i) - (iv), segue o resultado:

$$p_{i,a} - p_{i,b} = p_{m,a} - p_{m,b} \ , \quad i=1,\ldots, m \quad \Delta$$

Considere uma partição de $J = \{J_1, \ldots, J_n\}$ em classes C_k, k = 1,..., q de tarefas similares, isto é:

- $C_1 \cup \ldots \cup C_q = J$ (VII.13)
- $C_i \cap C_j = \emptyset \ , \ i \neq j, \ j \leq i, j \leq q$ (VII.14)
- $J_a \in C_i \ , J_b \in C_j$ e $J_a \simeq J_b \Longrightarrow C_i = C_j \ , \forall J_a, J_b \in J$ (VII.15)

A determinação das classes de tarefas similares, satisfazendo (VII.13) - (VII.15), pode ser feita utilizando-se o seguinte algoritmo:

Algoritmo VII.1: Determinação das Classes

entrada: • tarefas $J_1, \ldots, J_n$ e m processadores

• $(\alpha_{ij}, \beta_{ij})$, $1 \leq i \leq m$, $1 \leq j \leq n$

• inteiro L, limitante superior

$q := 1$

desmarque todas as tarefas

// O(L)// enquanto $q \leq L$ e existe tarefas desmarcadas efetue

escolha arbitrariamente uma tarefa J_j desmarcada
marque J_j
$C_q := \{J_j\}$

//O(n)// para cada $J_k \in J$, desmarcada efetue

//O(m)// se $J_k \simeq J_j$ então

$C_q := C_q \cup \{J_k\}$
marque J_k

$q := q+1$

se $q > L$ então saída :// insucesso//

caso contrário saída : $C_1, \ldots, C_q$ Δ

A complexidade do algoritmo acima é $O(L \cdot n \cdot m)$.

Denota-se por $C(J_j)$ a classe que contém a tarefa $J_j \in J$.

Considere um schedule $S = (J_{\sigma(1)}, \ldots, J_{\sigma(n)})$. Duas tarefas em S, J_a e J_b, são denominadas ligadas, ou diz-se existir um elo ligando J_a à J_b, denotando-se por $J_a \rightarrow J_b$ ou (J_a, J_b), quando

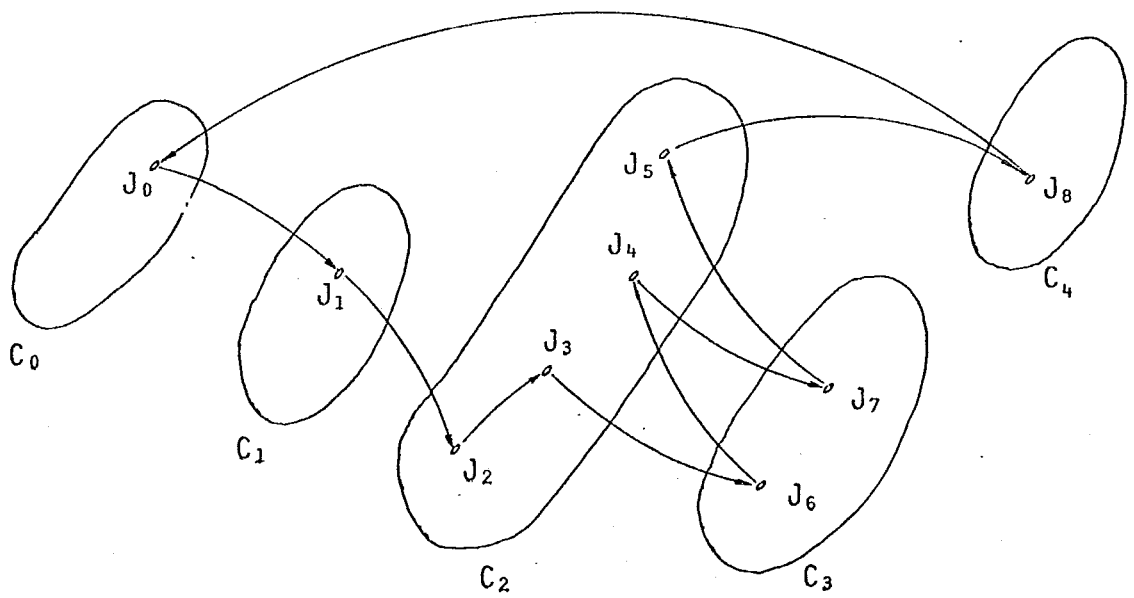
$$J_a = J_{\sigma(j-1)} \Longrightarrow J_b = J_{\sigma(j)}, \quad 1 < j \leq n ,$$

Considere, por conveniência:

- duas classes adicionais: $C_0 = \{J_{\sigma(0)}\}$ e $C_{q+1} = \{J_{\sigma(n+1)}\}$;
- os elos: $(J_{\sigma(0)}, J_{\sigma(1)})$, $(J_{\sigma(n)}, J_{\sigma(n+1)})$ e $(J_{\sigma(n+1)}, J_{\sigma(0)})$;
- $d(\sigma(n+1), \sigma(0)) = 0$.

Considere E(S) o conjunto de elos associado a um schedule S.

Exemplo VII.1: Representação de um schedule $\tilde{S}$
(n=7, q=3)



$E(\tilde{S}) = \{ (J_0, J_1), (J_1, J_2), (J_2, J_3), (J_3, J_6), (J_6, J_4), (J_4, J_7), (J_7, J_5), (J_5, J_8), (J_8, J_0) \}$ Δ

Seja x_{ij} o número de elos ligando tarefas da classe C_i à tarefas de C_j em S. Evidentemente, x_{ii} é o número de elos ligando tarefas dentro da classe C_i.

Denomina-se <u>perfil</u> de um schedule S, denotando-se por P(S), a matriz $X = [x_{ij}]$, $0 \leq i,j \leq q+1$:

$$P(S) = X = [x_{ij}], \quad 0 \leq i,j \leq q+1 \tag{VII.16}$$

Exemplo VII.2: Perfil de um schedule $\tilde{S}$
(considere $\tilde{S}$ do exemplo anterior)

$$P(\tilde{S}) = \begin{array}{c|ccccc|c} & & & & & & \Sigma \\ \hline & 0 & 1 & 0 & 0 & 0 & 1 \\ & 0 & 0 & 1 & 0 & 0 & 1 \\ & 0 & 0 & 1 & 2 & 1 & 4 \\ & 0 & 0 & 2 & 0 & 0 & 2 \\ & 1 & 0 & 0 & 0 & 0 & 1 \\ \hline \Sigma & 1 & 1 & 4 & 2 & 1 & 9 \end{array} \quad \Delta$$

É imediato verificar que para todo schedule S tem-se:

$$\text{(i)} \quad \sum_{j=0}^{q+1} x_{ij} = \sum_{j=0}^{q+1} x_{ji} = |C_i|, \quad i=0,\ldots,q+1 \tag{VII.17}$$

Por outras palavras, a todo schedule S pode-se associar um ciclo hamiltoniano.

$$\text{(ii)} \quad \Sigma x_{ij} = n+2 \ , \quad 0 \leq i,j \leq q+1 \tag{VII.18}$$

(iii) $\sum_{i,j \in I} x_{ij} < \sum_{i \in I} |C_i|$, para todo subconjunto I propriamente contido em $\{0,1,\ldots, q+1\}$. Ou seja, não é permitida a formação de sub-ciclos, ligando todas as tarefas de um subconjunto próprio de classes.

Sejam C_i e C_j duas classes não necessariamente distintas, $J_a \in C_i$ e $J_b \in C_j$, $a \neq b$. Define-se distância δ_{ij} da classe C_i à classe C_j, por:

$$\delta_{ij} = d(a, b) \qquad \text{(VII.19)}$$

Evidentemente, como as tarefas de uma classe são similares, a definição acima independe do par de tarefas escolhido.

De acordo com o acima exposto segue que o perfil de um schedule ótimo deve ser solução do seguinte problema de Programação Linear Inteira (PLI):

$$\min \sum \delta_{ij} \cdot x_{ij}, \quad 0 \leq i,j \leq q+1 \qquad \text{(VII.20)}$$

$$\text{s.a.:} \quad \sum_{j=0}^{q+1} x_{ij} = \sum_{j=0}^{q+1} x_{ji} = |C_i|, \quad i = 0,\ldots, q+1 \qquad \text{(VII.21)}$$

$$\sum_{i,j \in I} x_{ij} < \sum_{i \in I} |C_i|, \quad I \text{ propriamente contido em } \{0,\ldots, q+1\} \qquad \text{(VII.22)}$$

$$x_{i,j} \geq 0 \quad \text{e} \quad \text{inteiro} \qquad \text{(VII.23)}$$

Considerando-se o número de classes q limitado por uma constante L, o problema PLI acima pode ser resolvido em tempo polinomial, como segue: considerar todos os $O(n^{q^2})$ perfis; cada um contendo $O(q^2)$ variáveis; checar as $O(2^q)$ restrições e selecionar um perfil viável que minimize a função objetivo. Claramente esse processo encontra uma solução ótima em tempo $O(q^2 \cdot 2^q \cdot n^{q^2})$.

Considerando-se os resultados devidos a Lenstra [1983], pode-se resolver o problema PLI acima em tempo

$$O(2^{q^4} \cdot (q^2 \cdot 2^q \cdot \log (n + \delta_{max}))^{c^{q^2}})$$

onde: • $\delta_{max} = \max \{\delta_{ij} / 0 \leq i,j \leq q+1\}$

• c é uma constante apropriada de acordo com Blazewicz e Ecker [1983].

O problema remanescente pode ser assim formulado: dado um perfil $X = [x_{ij}]$, $0 \leq i,j \leq q+1$, encontrar um schedule S tal que $P(S) = X$.

Considere J o conjunto de todas as n tarefas e J(S) o conjunto de tarefas de um schedule S.

Diz-se que S' é um schedule parcial quando:

(i) $J(S') \subset J$

(ii) $J(S') \neq \emptyset$.

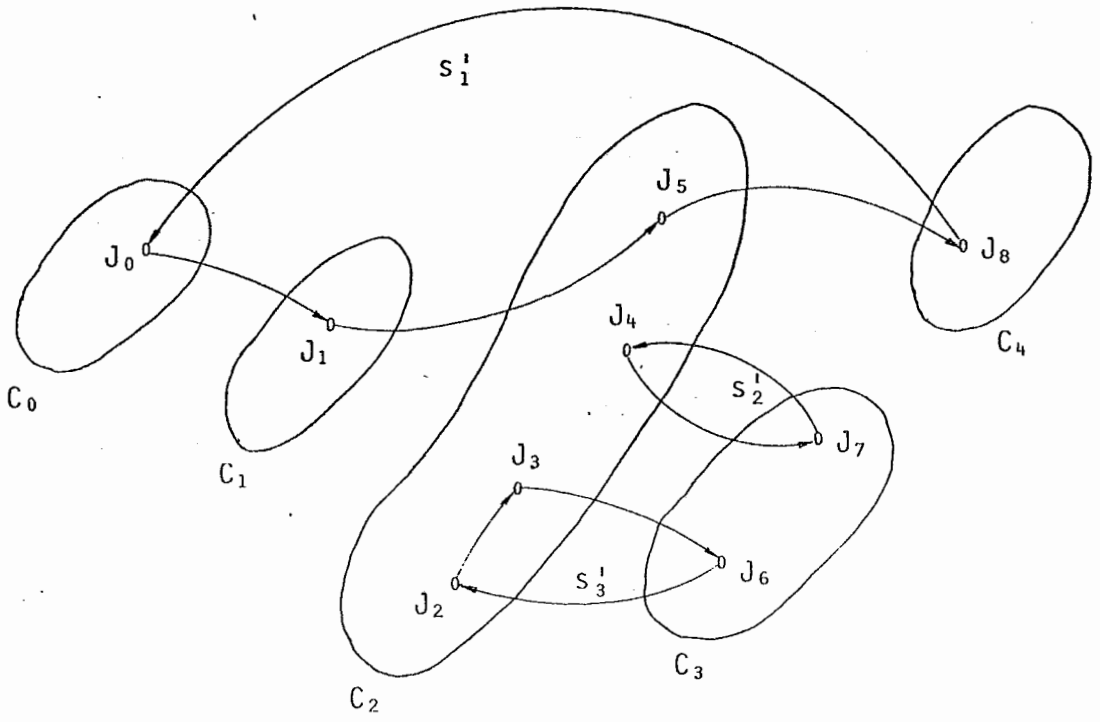
Um schedule $\bar{S} = S'_1 \cup \ldots \cup S'_t$, com perfil $P(\bar{S}) = [x_{ij}]$, $0 \leq i,j \leq q+1$, satisfazendo (VII.20) - (VII.23), é denominado partido quando:

(i) $J(\bar{S}) = J$

(ii) $J(S'_a) \cap J(S'_b) = \emptyset$, $\forall a \neq b$, $1 \leq a,b \leq t$.

Exemplo VII.3: Schedule Partido $\bar{S}$

(considere o perfil do Exemplo VII.2)



$$\bar{S} = S_1' \cup S_2' \cup S_3'$$

$$E(S_1') = \{(J_0, J_1), (J_1, J_5), (J_5, J_8), (J_8, J_0)\}$$

$$E(S_2') = \{(J_4, J_7), (J_7, J_4)\}$$

$$E(S_3') = \{(J_2, J_3), (J_3, J_6), (J_6, J_2)\} \quad \Delta$$

Dado um perfil $X = [x_{ij}]$, $0 \leq i,j \leq q+1$, satisfazendo (VII.20) - (VII.23), pode-se construir um schedule partido $\bar{S}$ para X, utilizando-se o seguinte algoritmo de complexidade O(n):

Algoritmo VII.2: Construção de schedule partido $\bar{S}$

entrada: . classes de tarefas $C_1, \ldots, C_q$

. perfil $X = [x_{ij}]$, $0 \leq i,j \leq q+1$

$E(\bar{S}) := \emptyset$

$e(J_j) := s(J_j) := 0, \quad \forall J_j \in J$

seja $g(J_j) = e(J_j) + s(J_j)$, $J_j \in J$

enquanto $\exists x_{ij} \neq 0$ efetue

escolha arbitrariamente i,j tal que $x_{ij} > 0$

selecione $J_a \in C_i$ e $J_b \in C_j$, $a \neq b$ tal que

$s(J_a) = e(J_b) = 0$

$e(J_a) \neq 0 \Longrightarrow g(J_{a'}) \neq 0, \quad \forall J_{a'} \in C_i, \quad a' \neq b$

$s(J_b) \neq 0 \Longrightarrow g(J_{b'}) \neq 0, \quad \forall J_{b'} \in C_j, \quad b' \neq a$

$E(\bar{S}) := E(\bar{S}) \cup \{(J_a, J_b)\}$

$s(J_a) := e(J_b) := 1$

$x_{ij} := x_{ij} - 1$

saída: $E(\bar{S})$ Δ

A construção de um schedule $S = (J_{\sigma(0)}, J_{\sigma(1)}, \ldots, J_{\sigma(n)}, J_{\sigma(n+1)})$ a partir de um schedule partido $\bar{S}$ pode ser feita em tempo $O(n)$ através do algoritmo a seguir:

Algoritmo VII.3: Construção de schedule S

entrada: • schedule partido $\bar{S} = S'_1 \cup \ldots \cup S'_t$

• $E(S'_a)$, $1 \leq a \leq t$

• classes de tarefas C_k, $0 \leq k \leq q+1$

seja $E(S) = E(S'_1) \cup \ldots \cup E(S'_t)$

construa um grafo bipartite $G(V_1 \cup V_2, A)$, tal que

$V_1 = \{C_k \mid 0 \leq k \leq q+1\}$

$V_2 = \{S'_a \mid 1 \leq a \leq t\}$

$(C_i, S'_a) \in A \Longleftrightarrow C_i \cap J(S'_a) \neq 0$, $0 \leq i \leq q+1$, $1 \leq a \leq t$

enquanto $t > 1$ efetue

escolha uma classe $C_i \in V_1$ que tenha grau maior ou igual a 2

seja: $\{(C_i, S'_a), (C_i, S'_b)\} \subset A$, $S'_a \neq S'_b$

$J_j \in C_i \cap J(S'_a)$, $(J_j, J_{j'}) \in E(S'_a)$

$J_k \in C_i \cap J(S'_b)$, $(J_k, J_{k'}) \in E(S'_b)$

$E(S'_a) := \{\{E(S'_a) \cup E(S'_b)\} - \{(J_j, J_{j'}), (J_k, J_{k'})\}\} \cup$

$\cup \{(J_j, J_{k'}), (J_k, J_{j'})\}$

para $(C_i, S'_b) \in A$ e $(C_i, S'_a) \notin A$ efetue

$A := A \cup \{(C_i, S'_a)\}$

$V_2 := V_2 - \{S'_b\}$ // exclue S'_b de G com suas arestas //

$E(S'_b) := \emptyset$

$t := t - 1$

saída: $E(S)$ Δ

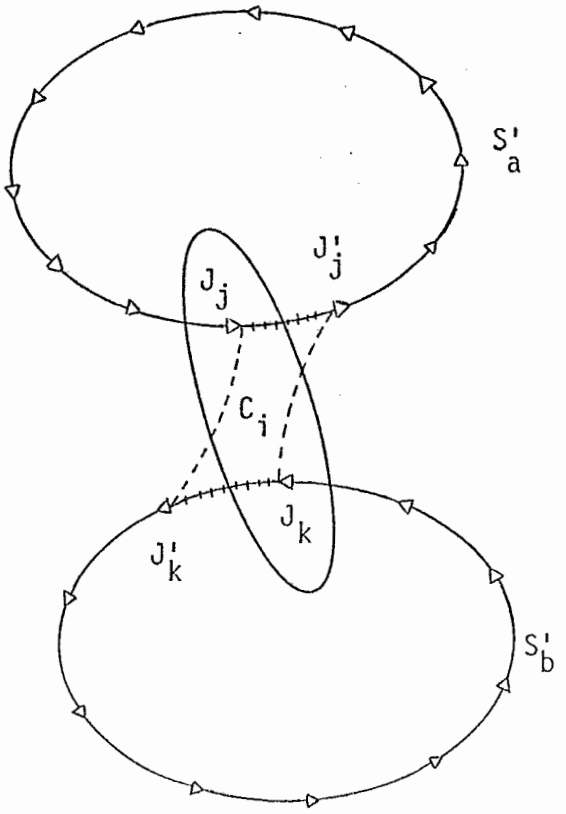


Fig. VII.3 - União de Schedules Parciais

A figura acima mostra a maneira pela qual ocorre a união de schedules parciais, através do Algoritmo VII.3. O valor do schedule S obtido a partir deste algoritmo é igual ao do schedule partido $\bar{S}$ pelo fato de

$$d(j, j') = d(k, j') \quad e$$

$$d(k, k') = d(j, k').$$

Considerando que o schedule partido $\bar{S}$ é obtido a partir de um perfil satisfazendo (VII.20) - (VII.23) então seu valor é ótimo, e consequentemente o schedule S é uma solução ótima para o problema F/ext-no-wait, $|\{C_k\}| < L/C_{max}$.

Finalizando, apresenta-se um algoritmo que sintetiza os resultados apresentados nesta seção.

Algoritmo VII.4: Problema F/ext-no-wait, $|\{C_k\}| < L/C_{max}$

1 entrada: • tarefas $J_1, \ldots, J_n$ e m processadores

• $(\alpha_{ij}, \beta_{ij})$, $1 \leq i \leq m$, $1 \leq j \leq n$

• inteiro L, limitante superior

2 determine as classes $C_1, \ldots, C_q$ // Algoritmo VII.1, O(L.n.m)//

3 resolva o problema PLI // (VII.20) - (VII.23) //

4 construa um schedule partido $\bar{S}$ // Algoritmo VII.2, O(n) //

5 construa um schedule S // Algoritmo VII.3, O(n) //

6 saída: Schedule S Δ

A complexidade do algoritmo acima é $O(L.n.m + 2^{q^4} \cdot (q^2 \cdot 2^q \cdot \log(n + \delta_{max}))^{cq^2})$. Logo, considerando-se o número de classes limitado ($q \leq L$), ele encontra uma solução para o problema F/ext-no-wait, $|\{C_k\}| < L/C_{max}$ em tempo polinomial.

CAPITULO VIII

CONCLUSÕES

Tendo em vista os resultados apresentados neste trabalho aponta-se a seguir algumas linhas para pesquisas posteriores.

No capítulo IV mostra-se que o problema FQ/ /no-wait/ΣC_j pode ser resolvido em tempo polinomial. Neste problema o tempo de processamento é uma função linear da quantidade de processamento ($p_{ij} = \alpha_i \cdot q_j + \beta_i$) e as constantes de proporcionalidade (α_i e β_i) são não-crescentes. Sugere-se os estudos considerando constantes de proporcionalidade quaisquer, outros tipos de funções e também processadores ordenados (FL/ /no-wait/ΣC_j). Outras sugestões para pesquisas futuras se referem a associação de custos w_j às tarefas (FQ/no-wait/$\Sigma w_j C_j$) e a consideração de datas de chegadas r_j distintas (FQ/no-wait, $r_j/\Sigma C_j$) e datas previstas para término d_j (FQ/no-wait/ΣT_j).

No capítulo V apresenta-se resultados para os problemas FC/no-wait/C_{max}, FC/no-wait/ΣC_j e FC/no-wait/ΣP_j. Sugere-se estudos dos seguintes problemas: FC/no-wait, r_j/C_{max}, FC/no-wait, r_j/F_{max}, FC/no-wait, $r_j/\Sigma C_j$ e FC/no-wait/$\Sigma w_j c_j$.

No capítulo VI estuda-se problemas em scheduling do tipo no-wait flow-shop com colisões ocorrendo apenas nos processadores extremos, especialmente o problema FE/no-wait/

/C_{max}. Sugere-se o estudo dos problemas FE/no-wait/ΣC_j e FE/ /no-wait, r_j/C_{max}.

No capítulo VII mostra-se que um algoritmo de complexidade O(nlogn) pode ser utilizado na solução do problema F2/ext-no-wait/C_{max}. Será que o problema F2/ext-no-wait, r_j/ /C_{max} também pode ser resolvido em tempo polinomial? Ainda neste capítulo apresenta-se resultados para o problema F/ext-no--wait, $|\{C_k\}| < L/C_{max}$. Sugere-se estudos com outros critérios de otimização, por exemplo, do problema F/ext-no-wait, $|\{C_k\}| < L/\Sigma C_j$.

Algumas novas direções interessantes de pesquisa em scheduling se referem a análise de problemas com multi-critérios de otimização, a análise de sensibilidade, a consideração de recursos escassos adicionais e a utilização do enfoque estocástico.

Finalizando, sugere-se pesquisas no sentido de se determinar as interfaces entre as teorias de Scheduling, Estoque, Filas, Planejamento e Controle da Produção, visando uma síntese na teoria da Produção.

BIBLIOGRAFIA

01 ACHUGBUE, J.O. e CHIN, F.Y., "Complexity and Solutions of Some Three-stage Flow-shop Scheduling Problems", Mathematics of Operations Research, 7(4), 532-544 (1982).

02 ARORA, R.K. e RANA, S.P., "Scheduling in a Semi-ordered Flow-shop Without Intermediate Queues", American Institute of Industrial Engineers Transactions, 12 (3), 263-272 (1980).

03 ASHOUR, S., "A Branch-and-Bound Algorithm for Flow-Shop Scheduling Problems", American Institute of Industrial Engineers Transactions, 2, 172-176 (1970).

04 ASHOUR, S., "An Experimental Investigation and Comparative Evaluation of Flow Shop Scheduling Techniques", Operations Research, 18(3), (1970A).

05 AXSÄTER, S., "On Scheduling in a Semi-ordered Flow-shop Without Intermediate Queues", Institute of Industrial Engineers Transactions, 14(2), 128-130 (1982).

06 BAKER, K.R., "Introduction to Sequencing and Scheduling", J. Wiley & Sons, New York (1974).

07 BAKER, K.R., "A Comparative Study of Flow Shop Algorithms", Operations Research, 23, 62-73 (1975).

08 BARANY, I., "A Vector-sum Theorem and its Application to Improving Flow-shop Guarantees", Mathematics of Operations Research, 6(3), 445-452 (1981).

09 BELLMAN, R., ESOGBUE, A.O. e NABESHIMA, J., "Mathematical Aspects of Scheduling and Applications", Perganon Press (1982).

10 BLAZEWICZ, J. e ECKER, K., "A Linear Time Algorithm for Restricted Bin Packing and Scheduling Problems", Operations Research Letters, 2, 80-83 (1983).

11 BLAZEWICZ, J. KUBIAK, W., RÖCK, H. e SZWARCFITER, J.L., "Flow Shop Scheduling Under Resource Constraints", aguardando publicação (1985).

12 BLAZEWICZ, J., LENSTRA, J.K. e RINNOOY-KAN, A.H.G., "Scheduling Subject to Resource Constraints : Classification and Complexity", Discrete Applied Mathematics, 5, 11-24 (1983).

13 BROWN, A. e LOMNICKI, Z., "Some Applications of the Branch and Bound Algorithm to the Machine Scheduling Problem", Operations Research Quarterly,17(2),(1966).

14 BRUMELLE, S., VANCOUVER, B.C. e SIDNEY, J.B., "The Two Machine Flow-shop Scheduling Problem with Stochastic Job Times", TIMS/ORSA Joint National Meeting, Chicago (1983).

15 BRUNO, J., COFFMAN JR., E.G. e SETHI, R., "Scheduling Independent Tasks to Reduce Mean Finishing Time", Communications of the Association for Computing Machenery, 17, 382-387 (1974).

16 CAMPBELL, H.G., DUDEK, R.A. and SMITH, M.L., "A Heuristic Algorithm for the n-Job m-Machine Sequencing Problem", Management Science, 16, 630-637 (1970).

17 CHIN, F.Y. e TSAI, L., "On J-maximal and J-minimal Flow--shop Schedules", Journal of the Association for Computing Machenery, 28(3), 462-476 (1981).

18 CHO, Y. and SAHNI, S., "Preenptive Scheduling of Independent Jobs with Release and Due Times on Open, Flow and Job Shops", Operations Research, 29, 511-522 (1981).

19 CONWAY, R.W., MAXWELL, W.L. and MILLER, L.W., "Theory of Scheduling", Addison-Wesley, Reading, Mass (1967).

20 DANNENBRING, D.G., "An Evaluation of Flow Shop Sequencing Heuristics", Management Science, 23, 1174-1182 (1977).

21 DEMPSTER, M.A.H., FISHER, M.L., JANSEN, L., LAGEWEG, B.J., LENSTRA, J.K. e RINNOOY-KAN, A.H.G., "Analysis of Heuristics for Stochastic Programming : Results for Hierarchical Scheduling Problems", Mathematics Operations Research, 8, 525-537 (1983).

22 DUTTA, S.K. e CUNNINGHAM, A.A., "Sequencing Two-machine Flow-shops with Finite Intermediate Storage", Management Science, 21, 989-996 (1975).

23 ENSCORE, E.E., NAWAZ, N. e HAM, I., "A Heuristic Algorithm for the m-Machine, n-Job Flow-shop Sequencing Problem", OMEGA-Pergamon Press, 11(1), 91-95 (1983).

24 FORST, F.G., "Minimizing Total Expected Costs in the Two-machine, Stochastic Flow-shop", Operations Research Letters, 2(2), 58-61 (1983).

25 FRENK, J.B.G. e RINNOOY-KAN, A.H.G., "The Asymptotic Optimality of the LPT Scheduling Heuristic", Report Erasmus University, Rotterdam (1984).

26 GAREY, M.R. e JOHNSON, D.S., "Approximation Algorithm for Combinatorial Problems : an Annotated Bibliography", in: J.F. Traub,(ed.), Algorithms and Complexity : New Directions and Recent Results, Academic Press, New York, 41-52 (1976).

27 GAREY, M.R. e JOHNSON, D.S., "Computers and Intractability : A Guide to the Theory of NP-Completeness", W.H. Freeman Press, San Francisco (1979).

28 GAREY, M.R., JOHNSON, D.S. e SETHI, R., "The Complexity of flow shop and jobshop scheduling", Mathematics of Operations Research, 1, 117-129 (1976).

29 GILMORE, P.C. e GOMORY, R.E., "Sequencing a One-state Variable Machine : a Solvable Case of the Traveling Salesman Problem", Operations Research, 12, 655--679 (1964).

30 GONZALES, T. e SAHNI, S., "Flowshop and Job Shop Schedules : Complexity and Approximation", Operations Research, 26, 36-52 (1978).

31 GOYAL, S.K., "A Note on the Paper : on the Flow-shop Sequencing Problem with No-wait in Process", Operational Research Quarterly, 24, 130-133 (1973), Operations Research, 20, 687-697 (1973).

32 GRABOWSKI, J., "A New Algorithm of Solving the Flow-shop Problem", G. Feichtinger, P.Kall (eds.), Operations Research in Progress, Reidel, Dordrecht, 57--75 (1982).

33 GRABOWSKI, J., SKUBALSKA, E. e SMUTNICKI, C., "On Flow Shop Scheduling with Release and Due Dates to Minimize Maximum Lateness", Journal of the Operational Research Society, 34(7), 615-620 (1983).

34 GRAHAN, R.L., LAWLER, E.L., LENSTRA, J.K. e RINNOOY-KAN, A.H.G., "Optimization and Approximation in Deterministic Sequencing and Scheduling : A Survey" , Annals of Discrete Mathematics, 5, 287-326 (1979).

35 GUPTA, J.N.D., "A Functional Heuristic Algorithm for the Flow Shop Scheduling Problem", Operational Research Quarterly,22, 39-47 (1971).

36 GUPTA, J.N.D., "Heuristic Algorithms for Multistage Flow Shop Problem", American Institute of Industrial Engineers Transactions, 4(1), (1972).

37 GUPTA, J.N.D., "Optimal Flow-shop Schedules with no Intermediate Storage Space", Naval Research Logistics Quarterly, 23, 235-243 (1976).

38 HELLER, J., "Some Numerical Experiments for an M x J Flow Shop and its Decision-Theoretical Aspects", Operations Research, 8, 178-184 (1960).

39 IGNALL, E.J. e SCHRAGE, L., "Application of the Branch--and-bound Technique to some Flow-shop Scheduling Problems", Operations Research, 13, 400-412 (1965).

40 JACKSON, J.R., "Scheduling a Production Line to Minimize Maximum Tardiness", Research Report 43, Management Science Research Project, University of California, Los Angeles (1955).

41 JOHNSON, S.M., "Optimal Two and Three Stage Production Schedules with Setup Times Included", Naval Research Logistics Quarterly, 1(1), 61-68 (1954).

42 JOHNSON, S.M., "Discussion : Sequencing n Jobs on Two Machines with Arbitrary Time Lags", Management Science, 5, 299-303 (1958).

43 KALRA, K.R. e BAGGA, P.C., "Flow-shop Sequencing with no Intramachine Waiting : Minimization of Total Waiting Cost", Journal of Information and Optimization Sciences, 2(3), 267-272 (1981).

44 KANG, S.H. e KOOK, K.H., "The Correctiye Heuristic Algorithm Analysis of the N x 3 Flow-shop Problem and Comparative Study with Multi-model", Journal of Korean Operational Research Society, 6(2), 13-19 (1981).

45 KRONE, M.J. e STEIGLITZ, K., "Heuristic-Programming Solution of Flowshop-Scheduling Problem", Operations Research, 22, 3, 629-638 (1974).

46 KURISU, T., "Two-machine Scheduling Under Required Precedence Among Jobs", Journal of the Operational Research Society of Japan, 19 (1976).

47 LAGEWEG, B.J., LENSTRA, J.K., LAWLER, E.L. e RINNOOY-KAN, A.H.G., "Computer-aided Complexity Classification of Combinatorial Problems", Communications of Association for Computing Machenery, 25(11), 817--822 (1982).

48 LAGEWEG, B.J., LENSTRA, J.K. e RINNOOY-KAN, A.H.G., "A General Bounding Scheme for the Permutation Flow--shop Problem", Operations Research, 26, 53-67 (1978).

49 LAWLER, E.L., "A Pseudopolynomial Algorithm for Sequencing Jobs to Minimize Total Tardiness", Annals of Discrete Methematics, 1, 331-342 (1977).

50 LAWLER, E.L., "Recent Results in the Theory of Machine Scheduling", in Mathematical Programming : The state of the art, edited by A. Bachen, M.Grötschel, B. Korte, Springer-Verlag, 202-234 (1983).

51 LAWLER, E.L., LENSTRA, J.K. e RONNOOY-KAN, A.H.G., "Recent Developments in Deterministic Sequencing and Scheduling : a Survey", in : Deterministic and stochastic scheduling.(ed.),M.A.H. Dempster et alii, Dordrecht, 35-73 (1982).

52 LENSTRA, Jr., H.W., "Integer Programming with a Fixed Number of Variables", Mathematics of Operations Research, 8, 538-548 (1983).

53 LENSTRA, J.K. e RINNOOY-KAN, A.H.G., "Some Simple Applications of the Traveling Salesman Problem", Operations Research Quarterly, 26, 717-733 (1975).

54 LENSTRA, J.K. e RINNOOY-KAN, A.H.G., "New directions in Scheduling Theory", Operations Research Letters, 2, 6, 255-259 (1984).

55 LENSTRA, J.K. e RINNOOY-KAN, A.H.G., "Scheduling Theory Since 1981 : an Annotated Bibliography", Combinatorial Optimization : Annotated Bibliographies, M. O'h Eigeartaigh, J.K. Lenstra e A.H.G. Rinnooy Kan (eds.), Wiley, Chichester (1984A).

56 LENSTRA, J.K., RINNOOY-KAN, A.H.G. e BRUCKER, P., "Complexity of Machine Scheduling Problems", in Annals of Discrete Mathematics, 4, 343-362 (1977).

57 LENSTRA, J.K., RINNOOY-KAN, A.H.G. e STOUGIE, L., "A Framework for the Probabilistic Analysis of Hierarchical Planning Systems", Annals of Operations Research, aguardando publicação (1984).

58 LOMNICKI, Z., "A Branch-and-Bound Algorithm for the Exact Solution of the Three-Machine Scheduling Problem", Operational Research Quarterly, 16(1), 89-100 (1965).

59 McMAHON, G.B., "Optimal Production Schedules for Flow Shops", Canadian Operational Research Journal, 7, 141-151 (1969).

60 McMAHON, G.B., "A Study of Algorithms for Industrial Scheduling Problems", Thesis, University of New South Wales, Kensington (1971).

61 McMAHON, G.B. e BURTON, P.G., "Flow Shop Scheduling with the Branch-and-Bound Method", Operations Research, 15, 473-481 (1967).

62 McNAUGHTON, R., "Scheduling with Deadlines and Loss Functions", Management Science, 6, 1-12 (1959).

63 MITTEN, L.G., "Sequencing n Jobs on Two Machine with Arbitrary Time Lags", Management Science, 5, 293--298 (1958).

64 MITTEN, L.G., "A Scheduling Problem an Analytical Solution Based Upon Two Machines, n-Jobs Arbitrary Start and Stop Lags and Common Sequence", The Journal of Industrial Engineering, 131-135 (1959).

65 MÖHRING, R.H., RADERMACHER, F.J. e WEISE, G., "Stochastic Scheduling Problems I : General Strategies", Zeitschrift für Operations Research, 28, 193-260 (1984).

66 MONMA, C.L., "Optimal m x n Flow Shop Sequencing with Precedence Constraints and Lag Times", School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY (1977).

67 MONMA, C.L., "The two Machine Maximum Flow-Shop Problem with Series-Parallel Precedence Constraints : An Algorithm and Extesions", Operations Research, 27, 792-798 (1979).

68 MONMA, C.L. e SIDNEY, J.B., "A General Algorithm for Optimal Job Sequencing with Series-parallel Precedence Constraints", Technical Report 347, School of Operations Research, Cornell University, Ithaca, NY (1977).

69 NABESHIMA, I., "Sequencing on Two Machines with Start Lag and Stop Lag", Journal of the Operational Research Society of Japan, 5, 97-101 (1963).

70 PALMA, O. I., "Desenvolvimento e Comparação de Algoritmos para Problemas de Ordenamento no Caso de Flow-shop", Tese de Mestrado, Universidade de Brasília (1977).

71 PALMER, D.S., "Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time - A Quick Method of Obtaining a Near Optimum", Operational Research Quarterly, 16(1), 101-107 (1965).

72 PANWALKAR, S.S. e WOOLLAM, C.R., "Flow Shop Scheduling Problems with No In-Process Waiting : A Special Case", Journal of the Operational Research Society, 30, 661-664 (1979).

73 PANWALKER, S.S. e WOOLLAM, C.R., "Ordered Flow Shop Problems with no In-process Waiting : Further Results", Journal of the Operational Research Society, 31, 1039-1043 (1980).

74 PAPADIMITRIOU, C.H. e KANELLAKIS, P.C., "Flow-shop Scheduling with Limited Temporary Storage", Journal of the Association for Computing Machenery, 27(3), 533-549 (1980).

75 PIEHLER, J., "Ein beitrag zum reihenfolgeproblem", Unternehmensforschung, 4, 138-142 (1960).

76 PINEDO, M.L. "Minimizing the Expected Makespan in Stochastic Flow Shops", Operations Research, 30(1), 148--162 (1982).

77 PINEDO, M.L., "Inequalities and Bounds for Stochastic Scheduling Models", TIMS/ORSA Joint National Meeting, Chicago (1983).

78 PINHO, A.A., "Algoritmos Polinomiais Exatos ou Aproximativos para Certos Problemas em Scheduling", Tese de Doutorado, COPPE-SISTEMAS/UFRJ (1983).

79 POTTS, C.N., "Analysis of Heuristics for Two-machine Subject to Release Dates", Report BW 150, Mathematisch Centrum, Amsterdam (1981).

80 REDDI, S.S. e RAMAMOORTHY, C.V., "On the Flow-shop Sequencing Problem with No-wait in Process", Operational Research Quarterly, 23, 323-331 (1972).

81 REJENDRA-PRASAD, V., "n x 2 Flowshop Sequencing Problem with Randon Processing Times", Operational Research Society of India, 18(1), 1-14 (1981).

82 RINNOOY-KAN, A.H.G., "Machine Scheduling Problems: Classification, Complexity and Computations", Nijhoff, The Hague (1976).

83 RÖCK, H., "Three Machine No-wait Flow Shop - Complexity and Approximation", Report 82-07, Fachbereich Informatik, Technical University of Berlin, Berlin (1982).

84 RÖCK, H., "The Three-machine No-wait Flow shop is NP-Complete", Journal of the Association for Computing Machenery, 31(2), 336-345 (1984).

85 RÖCK, H., "Some New Results in Flow Shop Scheduling", Zeitschrift fur Operations Research, 28, 1-16 (1984A).

86 RÖCK, H. e SCHMIDT, G., "Machine Aggregation Heuristics in Shop Scheduling", Methods of Operations Research, 45, 303-314 (1983).

87 SAHNI, S. e CHO, Y., "Complexity of Scheduling Jobs with No-wait in Process", Mathematics of Operations Research, 45, 303-314 (1983).

88 SIDNEY, J.B., "The Two-machine Maximum Flow Time Problem with Series-parallel Precedence Constraints", Operations Research, 27, 782-791 (1979).

89 SILVER, E.A., VIDAL, R.V. e de WERRA, D., "A Tutorial on Heuristic Methods", European Journal of Operational Research, 5, 153-162 (1980).

90 SMITH, W.E., "Various Optimizers for Single Stage Production", Naval Research Logistics Quarterly, 3, 59-66 (1956).

91 STINSON, J.P. e SMITH, A.W., "A Heuristic Programming Procedure for Sequencing the Static Flow Shop", International Journal of Production Research, 20 (6), 753-764 (1982).

92 SZWARC, W., "On Some Sequencing Problems", Naval Research Logistics Quarterly, 15, 127-155 (1968).

93 SZWARC, W., "Elimination Methods in the m x n Sequencing Problem", Naval Research Logistics Quarterly, 18, 295-305 (1971).

94 SZWARC, W., "Optimal Elimination Methods in the m x n Flow shop Scheduling Problem", Operations Research, 21, 1250-1259 (1973).

95 SZWARC, W., "Precedence Relations of the Flow-Shop Problem", Operations Research, 29, 400-410 (1981).

96 SZWARC, W., "A Note on the Flow-shop Problem Without Interruptions in Job Processing", Naval Research Logistics Quarterly, 28 (4), 665-669 (1981A).

97 SZWARC, W., "Flow Shop Problems with Time Lags", Management Science, 29(4), 477-481 (1983).

98 SZWARC, W., "Solvable Cases of the Flow-Shop Problem Without Interruptions in Job Processing", Naval Research Logistics Quarterly, 30(1), 179-183 (1983A).

99 SZWARCFITER, J.L., "Grafos e Algoritmos Computacionais", Editora Campos, Rio de Janeiro (1984).

100 SZWARCFITER, J.L., "Traveling Salesman Problem with Restricted Distances", aguardando publicação (1984A).

101 VAN DEMAN, J.M. e BAKER, K.R., "Minimizing Mean Flowtime in the Flow Shop with No Intermediate Queues", American Institute of Industrial Engineers Transactions, 6, 28-34 (1974).

102 WERMUS, M., "An Example of Getting a Wrong Answer Using Expected Times in Stochastic Flow Shop", TIMS/ORSA Joint National Meeting, Chicago (1983).

103 WISMER, D.A., "Solution of the Flowshop Scheduling Problem with No Intermediate Queues", Operations Research, 20, 689-697 (1972).