UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO

Técnicas de Seqüenciamento e Programação de Operações em Máquinas

João Vitor Moccellin

EESC-USP
Biblioteca
CAMPUS DE S CARLOS

São Carlos, agosto de 2005

TÉCNICAS DE SEQÜENCIAMENTO E PROGRAMAÇÃO DE OPERAÇÕES EM MÁQUINAS

SUMÁRIO

1 - INTRODUÇÃO

1.1 - Considerações Gerais 1

1.2 - O processo geral de seqüenciamento e programação de operações em máquinas 4

1.3 - Medidas de desempenho 7

1.4 - Classificação de problemas 13

2 - SEQÜENCIAMENTO DE OPERAÇÕES EM UMA MÁQUINA

2.1 - Considerações Gerais 16

2.2 - Alguns Problemas Estáticos de Seqüenciamento de Operações em uma Máquina 18

I) $n/1/\overline{F}$ 19

II) $n/1/T_{max}$ 21

III) $n/1/\overline{F} \mid n(T) = 0$ 23

IV) $n/1/\overline{F} \mid \min T_{max}$ 24

V) $n/1/n(T)$ 25

VI) $n/1/n(T) \mid E$ 26

VII) $n/1/\overline{F} \mid E$ 28

VIII) $n/1/\overline{F} \mid \min n(T)$ 29

IX) $n/1/\overline{F}$ e $n(T)$ 29

3 - PROGRAMAÇÃO DE MÁQUINAS PARALELAS
3.1 - Considerações Gerais 36
3.2 - Tarefas Independentes 37
3.3 - Tarefas Dependentes 46

4 - PROGRAMAÇÃO DE SISTEMAS FLOW SHOP E JOB SHOP
4.1 - Considerações Gerais 49
4.2 - Algoritmo de Johnson 50
4.3 - Métodos *Branch and Bound* 52
4.4 - Formulações em Programação Linear (Inteira Mista) 60
4.5 - Métodos Heurísticos 64

1 - INTRODUÇÃO

1.1 - Considerações Gerais

- **PROBLEMAS DE SEQÜENCIAMENTO**

  - Serviços em uma fábrica (Oficina de Manufatura: *JOB SHOP*)
  - Transportes
  - Comunicações
  - Serviços

- A utilização de MODELOS

- **PROBLEMAS DE SEQÜENCIAMENTO PURO**

Geralmente, em situações reais, a seqüência (programação) de execução das tarefas tem alguma INTERAÇÃO com:

a) QUAIS tarefas devem ser executadas.

b) COMO as tarefas devem ser executadas (método, processo a ser utilizado).

Exemplo: tarefa = produção de 100 unidades de um produto.
decisão = iniciar já ou aguardar um período de tempo.

- Efeitos (conseqüências):

I) A encomenda poderia ser cancelada.

II) Uma encomenda (ordem de serviço) adicional poderia ser recebida, de forma a aumentar a quantidade a ser produzida.

III) Poderia ocorrer uma alteração nas especificações do projeto do produto.

IV) Um determinado tipo de matéria-prima que hoje é disponível, poderia não ser após o período de tempo considerado, necessitando ser substituída.

V) Uma máquina-"chave", hoje disponível, poderia estar em reparos após o período. Em conseqüência, algum equipamento substituto (eventualmente similar) precisaria ser utilizado.

VI) Uma nova máquina (ou ferramenta), já encomendada, poderia chegar nesse período, possibilitando uma melhoria no processo de execução do produto.

VII) Os operadores, durante o período, poderiam adquirir uma maior habilidade pela execução de tarefas semelhantes. Isso poderia melhorar a eficiência da mão-de-obra por ocasião da execução da tarefa.

Usualmente, nos problemas (modelos) de programação da execução de tarefas, supõe-se que todas as decisões referentes a O QUE e COMO fazer já tenham sido tomadas.

- HIPÓTESES GERAIS dos problemas de seqüenciamento e programação de operações em máquinas

1) As tarefas (*jobs*) a serem executadas são perfeitamente conhecidas e determinadas. Uma tarefa corresponde a um determinado conjunto de operações.

2) Todas as tarefas deverão ser executadas.

3) Os recursos (mão-de-obra, máquinas, equipamentos, ferramentas, etc.) a serem utilizados na execução das tarefas são previamente especificados.

4) A seqüência das operações necessárias para execução de cada tarefa é conhecida. O processo de execução também é perfeitamente conhecido e deve existir pelo menos um conjunto de recursos capaz de executar cada operação.

- CONSIDERAÇÕES SOBRE A TERMINOLOGIA

- SEQÜENCIAMENTO (*SEQUENCING*): refere-se à ordenação de tarefas (operações) em uma certa máquina.

- PROGRAMAÇÃO (*SCHEDULING*): refere-se à ordenação, em uma base de tempo, de tarefas (ou operações) em uma ou diversas máquinas.

1.2 - O PROCESSO GERAL DE SEQÜENCIAMENTO E PROGRAMAÇÃO DE OPERAÇÕES EM MÁQUINAS (*Job Shop process*)

- conjunto J de n tarefas (*jobs*).

$$J = \{J_1, J_2, ..., J_i, ..., J_n\}$$

- tarefa $J_i = \{op_{i1}, op_{i2}, ..., op_{ij}, ..., op_{ig_i}\}$

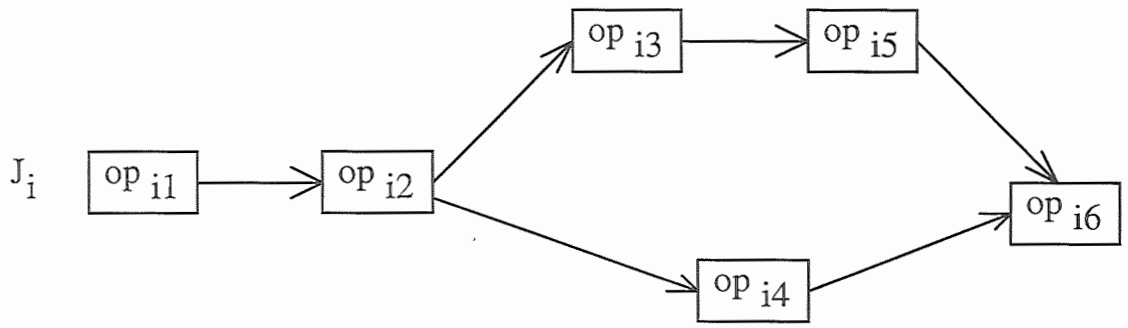
  conjunto de g_i operações.

- precedências tecnológicas entre operações de uma tarefa J_i

.

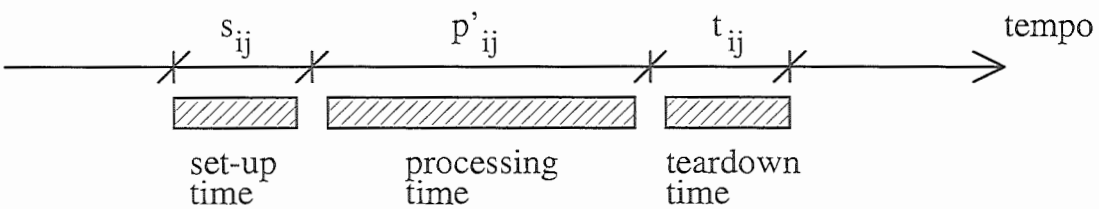
$op_{ia} > op_{ib}$ op_{ia} PRECEDE op_{ib}

$op_{ia} >> op_{ib}$ op_{ia} PRECEDE DIRETA / op_{ib}



- <u>Tempo de processamento de uma operação</u> $op_{ij} \rightarrow p_{ij}$

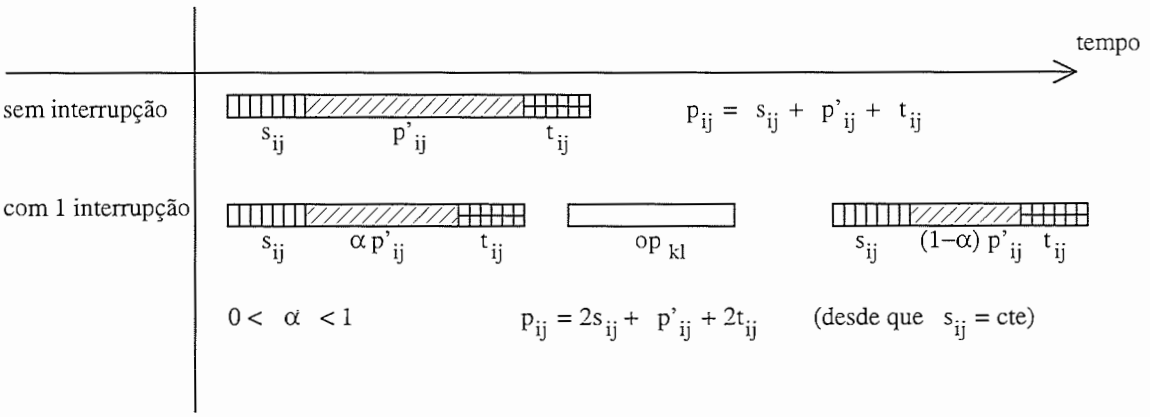
  - Componentes do tempo de processamento de uma operação.



- quando s_{ij} <u>independe</u> da operação anterior (na mesma máquina):

$$p_{ij} = s_{ij} + p'_{ij} + t_{ij} = cte.$$

- quando a interrupção (*preemption*) de uma operação for permitida (ou for possível):

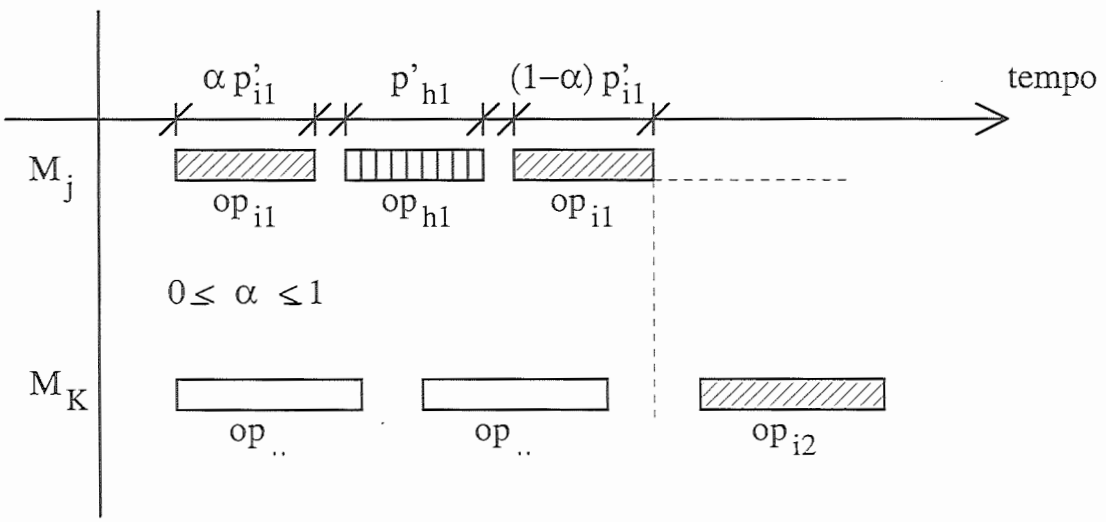


- Conjunto de m máquinas

$$M = \{M_1, M_2, ..., M_m\}$$

Cada máquina ε M é capaz de executar um sub-conjunto de operações das diversas tarefas, sendo identificada através desse sub-conjunto de operações.

- <u>PROGRAMAÇÃO DAS OPERAÇÕES</u>

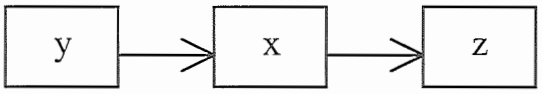


- RESTRIÇÕES ADICIONAIS AO PROCESSO GERAL

→ *Job Shop* Básico
(*Simple Job Shop process*)

a) Cada máquina é disponível continuamente durante um período de tempo suficientemente grande. Não são consideradas as indisponibilidades temporárias devido a quebras ou manutenção.

b) Para uma dada operação x , existe no máximo uma operação y , tal que y >> x , e no máximo uma operação z, tal que x >> z .
x, y, z são operações de uma mesma tarefa



c) Cada operação pode ser executada por somente uma máquina.

d) Existe somente 1 máquina de cada tipo.

e) Não é permitida a interrupção (*preemption*) da execução de qualquer operação.

f) Em um instante qualquer, uma determinada tarefa poderá ter somente uma de suas operações em execução.

g) Uma máquina não pode executar duas operações simultaneamente.

1.3 - MEDIDAS DE DESEMPENHO

- VARIÁVEIS QUE DEFINEM UM PROBLEMA DE PROGRAMAÇÃO (parâmetros)

n tarefas $J = \{J_1, J_2, ..., J_n\}$

m máquinas $M = \{M_1, M_2, ..., M_m\}$

r_i = data de liberação da tarefa J_i, a partir da qual ela pode ser executada (*release time*).

d_i = data de entrega da tarefa J_i.
Data limite estabelecida para término da tarefa (*due date*).

$a_i = d_i - r_i$ = período permitido de permanência da tarefa J_i na oficina.

- tarefa $J_i = \{op_{i1}, op_{i2}, ..., op_{ig_i}\}$
conjunto de g_i operações ordenadas.

  - para cada operação op_{ij}, tem-se:

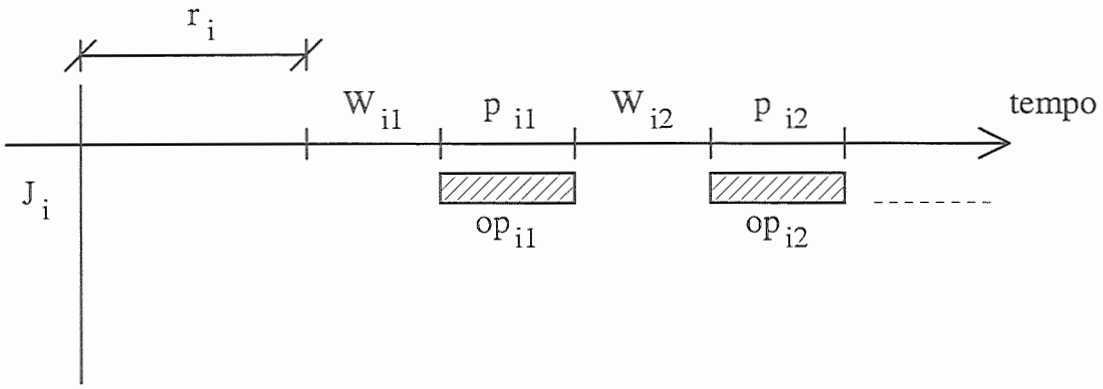
m_{ij} = número inteiro que identifica a máquina que deverá executar a j-ésima operação (op_{ij}) da tarefa J_i.

p_{ij} = tempo de processamento da operação op_{ij}, na máquina m_{ij}.

$p_i = \sum_{j=1}^{g_i} p_{ij}$ = tempo total de processamento de J_i.

VARIÁVEIS QUE DESCREVEM A SOLUÇÃO DE UM PROBLEMA DE PROGRAMAÇÃO

W_{ij} = tempo de espera da operação op_{ij}, após ter sido completada a operação $op_{i(j-1)}$.



$$W_i = \sum_{j=1}^{g_i} W_{ij} = \text{tempo total de espera da tarefa } J_i.$$

Um determinado conjunto viável de W_{ij}

$$\{W_{ij} \; ; \; i=1,2,\ldots,n \; ; \; j=1,2,\ldots,g_i\}$$

determina uma SOLUÇÃO VIÁVEL para o problema de programação.

[solução viável: satisfaz todas as restrições do problema (modelo)].

1.3.1 - MEDIDAS DE DESEMPENHO PROVENIENTES (DEPENDENTES) DE W_{ij}

. C_i = data de término da tarefa J_i (*completion time*)

$$C_i = r_i + \sum_{j=1}^{g_i} W_{ij} + \sum_{j=1}^{g_i} p_{ij} = r_i + W_i + p_i$$

. F_i = tempo de fluxo ou tempo de permanência da tarefa J_i (*flow time*).

$$F_i = C_i - r_i \quad \text{ou} \quad F_i = W_i + p_i$$

. $L_i = C_i - d_i$ = "*Lateness*" da tarefa J_i .

$$(L_i = F_i - a_i)$$

. T_i = ATRASO na execução da tarefa J_i (*Tardiness*).

$$T_i = \max(0, L_i).$$

. $\overline{C} = \frac{1}{n} \sum_{i=1}^{n} C_i$ = data média de término do conjunto de n tarefas.

. $\overline{F} = \frac{1}{n} \sum_{i=1}^{n} F_i$ = tempo médio de fluxo (*mean flow time*).

. $\overline{L} = \frac{1}{n} \sum_{i=1}^{n} C_i - \frac{1}{n} \sum_{i=1}^{n} d_i = \overline{C} - \overline{d}$.

- <u>Relações entre</u> <u>$\overline{C}$, $\overline{F}$ e $\overline{L}$</u>

$$L_i = C_i - d_i = F_i - a_i$$

$$\left[\begin{array}{l} \overline{L} = \overline{C} - \overline{d} = \overline{F} - \overline{a} \\ \text{onde } \overline{d} \text{ e } \overline{a} \text{ são valores constantes} \end{array} \right.$$

EESC-USP Biblioteca CAMPUS DE SÃO CARLOS

- FUNÇÕES-OBJETIVO usuais:

$$\text{MINIMIZAR} \begin{cases} \overline{F} \text{ (ou } \overline{C}\text{, ou } \overline{L}\text{)} \\ F_{max} = \max_i F_i \\ C_{max} = \max_i C_i \\ T_{max} = \max_i T_i \end{cases}$$

obs.:

I) Se $r_i = r \quad \forall \; i = 1, 2, ..., n$ então $F_{max} = C_{max} - r$

II) MIN $F_{max} \Leftrightarrow$ MIN T_{max} .

1.3.2 - MEDIDAS DE DESEMPENHO DA OFICINA (não relacionadas a tarefas individuais)

- Taxa de Utilização: mede a eficiência de utilização da capacidade produtiva.

$$U = \frac{\text{horas - máquina trabalhadas}}{\text{horas - máquina disponíveis}}$$

para o conjunto de n tarefas.

$$U = \frac{\sum_{i=1}^{n} p_i}{m\,(C_{max} - r_{min})}$$

obs.:

I) Se $\begin{cases} p_{ij} = s_{ij} + p'_{ij} + t_{ij} = cte \\ r_i = r \quad \forall\ i = 1, 2, ..., n \end{cases}$

então $U = \dfrac{\sum_{i=1}^{n} p_i}{m \,.\, F_{max}}$ com $\sum_i p_i = cte$.

Neste caso MAX U $\Leftrightarrow$ MIN F_{max}

II) $\left.\begin{array}{l} \text{Se } s_{ij} \text{ depende da operação anterior} \\ \text{ou} \\ \text{permite - se a interrupção de operações} \end{array}\right\} \sum_i p_i \neq cte$.

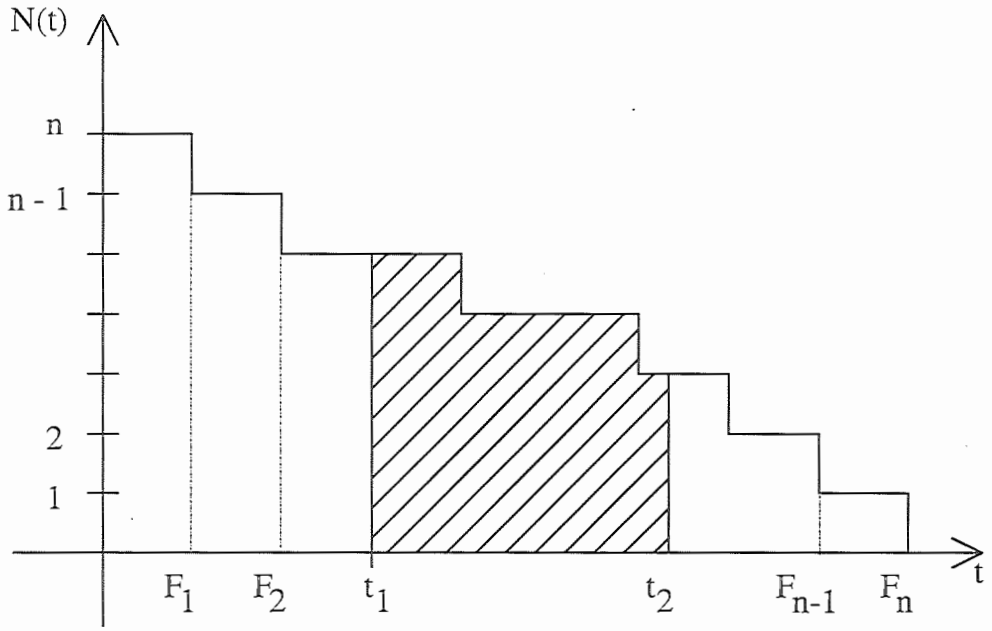
- <u>Estoque em Processamento</u>

N(t) = número de tarefas na oficina, no instante t .

$r_i = 0 \quad \forall\ i = 1, 2, ..., n$

$\overline{N}(t_1, t_2)$ = número médio de tarefas na oficina, durante o intervalo de tempo (t_1, t_2)

$$\overline{N}(t_1, t_2) = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} N(t)\, dt$$



$$N(t) = N_p(t) + N_q(t)$$

$$\begin{cases} N_p(t) = \text{número de tarefas sendo processadas (executadas) no instante t.} \\ N_q(t) = \text{número de tarefas ainda não iniciadas no instante t, ou} \\ \quad \text{não tendo nenhuma de suas operações sendo executadas no instante t.} \end{cases}$$

$$\rightarrow \overline{N}_p(t_1, t_2) \qquad \overline{N}_q(t_1, t_2)$$

- $\overline{N}_p(t_1, t_2)$ torna-se independente da programação à medida que (t_1, t_2) aumenta, sendo função somente do volume de trabalho a ser executado e do número de máquinas disponíveis.

- FUNÇÃO-OBJETIVO: MIN $\overline{N}_q(t_1, t_2)$

1.4 - CLASSIFICAÇÃO DE PROBLEMAS

a) ***Job Shop***: cada tarefa tem sua própria seqüência de processamento nas máquinas.

b) ***Flow Shop***: todas as tarefas têm a mesma seqüência de processamento nas máquinas.

c) ***Open Shop***: não há uma seqüência específica ou preestabelecida para as tarefas serem processadas nas máquinas.

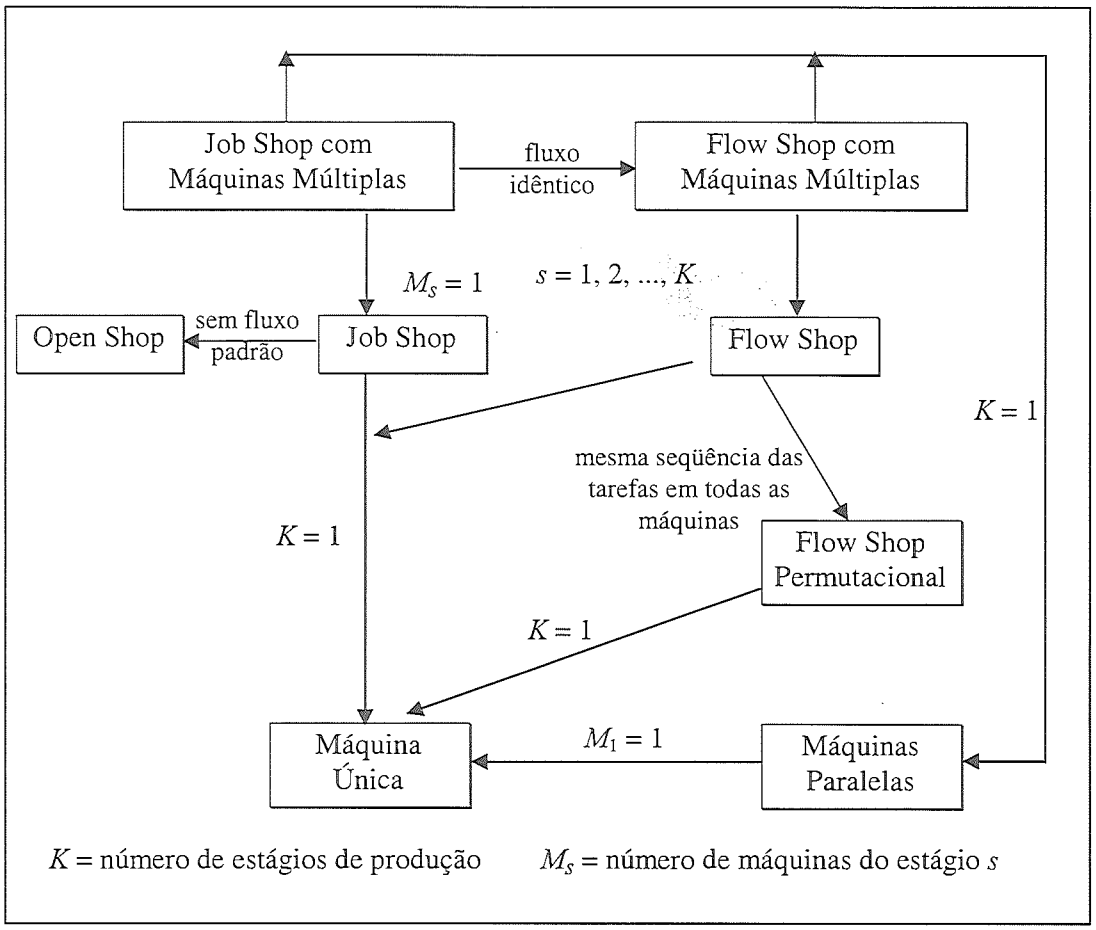
d) ***Flow Shop Permutacional***: é um *Flow Shop* no qual em cada máquina a seqüência das tarefas é a mesma.

e) ***Máquina única***: existe uma única máquina disponível.

f) ***Máquinas paralelas***: são disponíveis diversas máquinas, geralmente idênticas, para as mesmas operações.

g) ***Job Shop com máquinas múltiplas***: é um *Job Shop* no qual existe um conjunto de máquinas paralelas em cada estágio de produção.

h) ***Flow Shop com máquinas múltiplas***: é um *Flow Shop* no qual existe um conjunto de máquinas paralelas em cada estágio de produção.



Um problema de programação de operações em máquinas pode ser identificado por 4 tipos básicos de informações:

I) As tarefas e respectivas operações a serem executadas.

II) O número e tipos de máquinas disponíveis.

III) Regras que disciplinam como as designações OPERAÇÃO-MÁQUINA podem ser feitas.

IV) O critério a ser utilizado para avaliar as alternativas de programação.

Notação de 4 parâmetros A/B/C/D

EXEMPLOS

- $n / 2 / F / F_{max}$: Problema de Johnson

  Programação de $n \geq 2$ tarefas em 2 máquinas, em ambiente *flow shop*, de modo a minimizar o tempo de fluxo máximo (ou duração total da programação).

- $n / m / G / F_{max}$: Problema *Job Shop* geral

  Programação de $n \geq 2$ tarefas em $m \geq 2$ máquinas, com fluxos diferenciados (*Job Shop*), de modo a minimizar o tempo de fluxo máximo (ou duração total da programação).

2 - SEQÜENCIAMENTO DE OPERAÇÕES EM UMA MÁQUINA

2.1 - Considerações Gerais

. APLICAÇÕES
- . Oficinas com uma única máquina.
- . Conjunto de máquinas e equipamentos que opera como se fosse uma única máquina. Ex.: Indústrias Químicas.
- . Máquina - dominante de um processo. Ex.: Indústria de Papel.
- . Máquina - gargalo.

. PARÂMETROS

. $J = \{J_1, J_2, ..., J_n\}$ $n \geq 2$ finito, número de tarefas a serem processadas.

. $m = 1$ uma única máquina relevante.

. $g_i = 1$ uma única operação por tarefa.

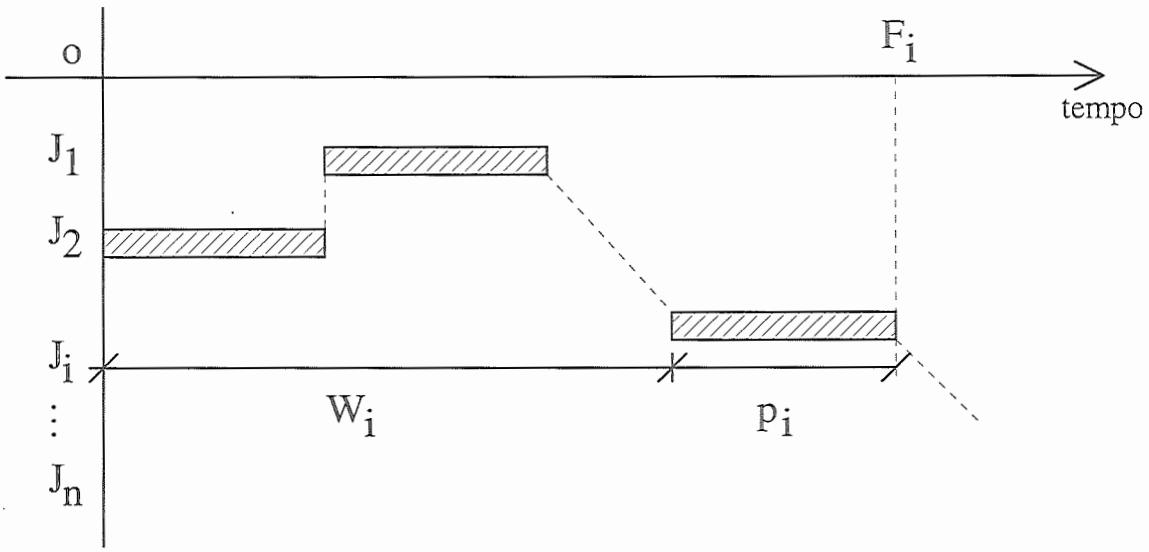
. $r_i = r$ ou simplesmente $r_i = 0$, ou seja, todas as tarefas têm a mesma data de liberação.

. se $r_i = 0 \rightarrow a_i = d_i$

. $p_{i1} = p_i = s_i + p'_i + t_i = cte$

. VARIÁVEIS DE DECISÃO

. $W_{i1} = W_i$ tempo de espera da tarefa (operação) J_i.



. $C_i = F_i = W_i + p_i$ data de término da tarefa J_i = tempo de fluxo de J_i.

. Número de ordenações (seqüências) possíveis das n tarefas = n!
(uma vez que $r_i = 0 \quad \forall\ i = 1, 2, ..., n$).

. Seja uma ordenação qualquer:
$(J_h, J_i, ..., J_k)$ J_k : última tarefa

$$F_k = W_k + p_k \quad \text{onde} \quad \begin{cases} F_k = F_{max} \\ W_k = \sum_{\substack{i=1 \\ i \neq k}}^{n} p_i \end{cases}$$

$$\therefore F_{max} = \sum_{i=1}^{n} p_i = \text{cte.}$$

ou seja, TODAS as n! ordenações têm o mesmo F_{max}.

2.2 - ALGUNS PROBLEMAS ESTÁTICOS DE SEQÜENCIAMENTO DE OPERAÇÕES EM UMA MÁQUINA

I) $n/1/\overline{F}$	(SMITH, 1956)
II) $n/1/T_{max}$	(JACKSON, 1955)
III) $n/1/\overline{F} \mid n(T) = 0$	(SMITH, 1956)
IV) $n/1/\overline{F} \mid \min T_{max}$	(HECK and ROBERTS, 1972)
V) $n/1/n(T)$	(MOORE, 1968)
VI) $n/1/n(T) \mid E$	(SIDNEY, 1973)
VII) $n/1/\overline{F} \mid E$	(EMMONS, 1975)
VIII) $n/1/\overline{F} \mid \min n(T)$	(EMMONS, 1975)
IX) $n/1/\overline{F}$ e $n(T)$	(NELSON *et al.*, 1986)

- SMITH, W.E. "Various optimizers for Single-Stage Production". Naval Research Logistics Quarterly, 3, No.1, March 1956.

- HECK, H. and S. ROBERTS. "A Note on the Extension of a Result on Scheduling with Secondary Criteria". Naval Research Logistics Quarterly, 19, No.2, June 1972.

- MOORE, J.M. "An n Job, one Machine Sequencing Algorithm for Minimizing the Number of Late Jobs". Management Science, 15, No.1, Sept. 1968.

- SIDNEY, J.B. "An Extension of Moore's Due Date Algorithm". Simposium on the Theory of Scheduling and

its Applications, Edited by S.E. Elmaghraby, Springer-Verlag, 1973.

- EMMONS, H. "One Machine Sequencing to Minimize Mean Flow Time with Minimum Number Tardy". Naval Research Logistics Quarterly, 22, No.3, September 1975.

- NELSON, R.T.; R.K. SARIN and R.L. DANIELS. "Scheduling with Multiple Performance Measures: the One-Machine Case". Management Science 32, No.4, April 1986.

I) $n/1/\overline{F}$

J_i	J_1	J_2	...	...	J_n
p_i	p_1	p_2	...	...	p_n

<u>TEOREMA</u>: "No problema $n/1/\overline{F}$, o tempo médio de fluxo (permanência) $\overline{F}$ é minimizado pela ordenação das tarefas (operações) segundo a ORDEM NÃO-DECRESCENTE dos tempos de processamento.

- Regra <u>SPT</u> (*Shortest Processing Time*)

$$(J_{[1]}, J_{[2]}, \ldots, J_{[i]}, \ldots, J_{[n]})$$

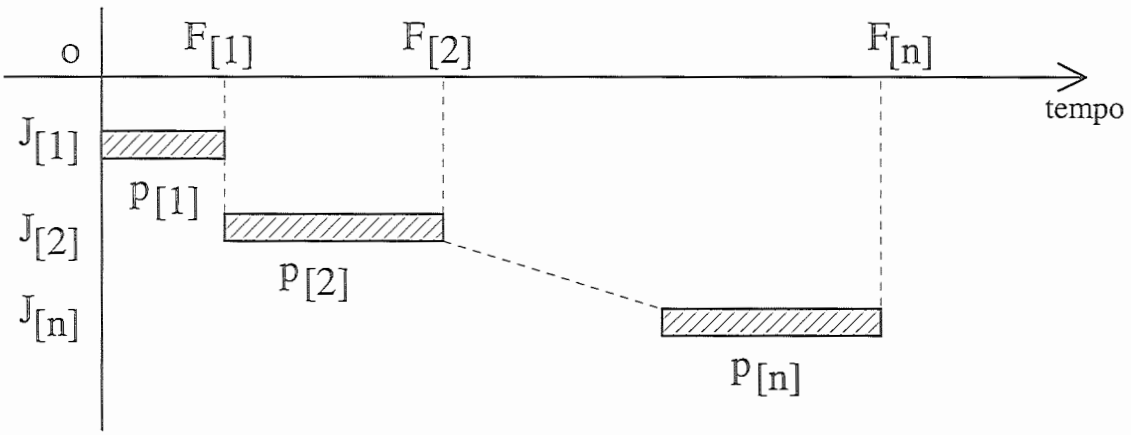
$$\begin{cases} J_{[i]} = \text{tarefa colocada na i-ésima posição} \\ p_{[i]} = \text{tempo de processamento da tarefa } J_{[i]} \end{cases}$$

onde: $p_{[1]} \leq p_{[2]} \leq \ldots \leq p_{[n]}$

EESC-USP Biblioteca CAMPUS DE S CARLOS

- PROVA DO TEOREMA:

$F_{[i]}$: tempo de fluxo (permanência) da tarefa $J_{[i]}$.



$$F_{[1]} = p_{[1]}$$

$$F_{[2]} = p_{[1]} + p_{[2]}$$

$$\vdots$$

$$F_{[i]} = p_{[1]} + p_{[2]} + \ldots + p_{[i]}$$

$$\vdots$$

$$F_{[n]} = p_{[1]} + p_{[2]} + \ldots + p_{[i]} + \ldots + p_{[n]}$$

$$\sum_{i=1}^{n} F_{[i]} = n\,p_{[1]} + (n-1)\,p_{[2]} + \ldots + (n-i+1)\,p_{[i]} + \ldots + 1\,p_{[n]}$$

$$\begin{cases} n : n-1 : \ldots n-i+1 : \ldots : 1 \text{ ordenação DECRESCENTE} \\ \text{se } p_{[1]} : p_{[2]} : \ldots : p_{[i]} : \ldots : p_{[n]} \text{ for CRESCENTE ou NÃO DECRESCENTE} \end{cases}$$

$$\xrightarrow{\text{CONWAY (Pag. 27)}} \sum_{i=1}^{n} F_{[i]} \text{ é MÍNIMA, ou seja}$$

$$\overline{F} = \frac{1}{n} \sum_{i=1}^{n} F_{[i]} \text{ é MÍNIMO.}$$

II) $n/1/T_{max}$ (ou $n/1/L_{max}$)

- <u>Teorema de JACKSON (1955)</u>:

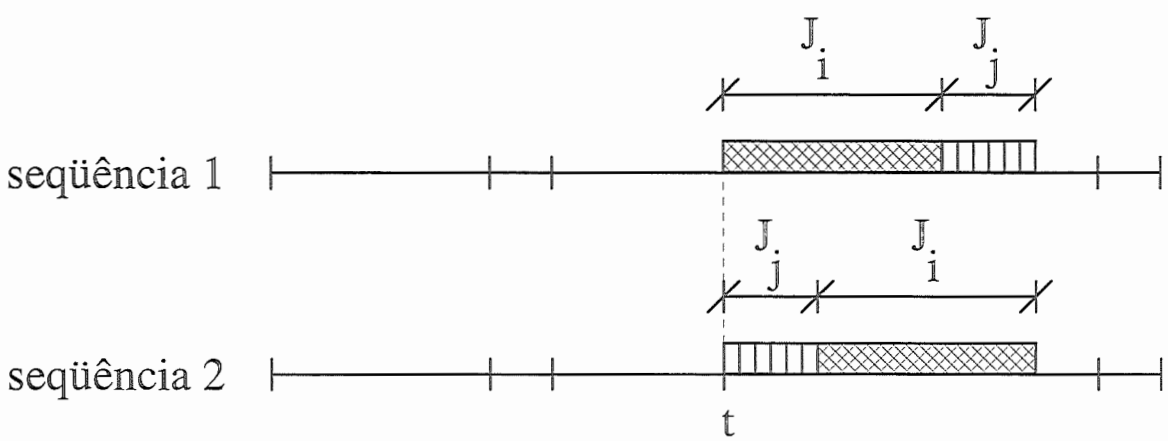
"No problema $n/1/T_{max}$ ou $n/1/L_{max}$, os valores de T_{max} e L_{max} são minimizados pela ordenação das tarefas (operações) segundo a ORDEM NÃO-DECRESCENTE das datas de entrega".

- regra <u>EDD</u> (*Earliest Due Date*)

$(J_{[1]}, J_{[2]}, ..., J_{[n]})$ tal que
$d_{[1]} \leq d_{[2]} \leq ... \leq d_{[n]}$.

- <u>PROVA DO TEOREMA</u>:

. para $L_{\underline{max}}$:



. As seqüências 1 e 2 são idênticas exceto que as tarefas J_i e J_j estão em posições invertidas.

. Seja $d_i < d_j$.

. Se $L_{max,\ seq.\ 1} \leq L_{max,\ seq.\ 2}$ então a regra EDD minimiza L_{max} .

. Seja $\begin{cases} L_{i1} = \text{lateness de } J_i \text{ na seq.1}, \\ \text{analogamente } L_{i2}, L_{j1}, L_{j2} . \\ L = \text{maior lateness das demais (n - 2) tarefas.} \end{cases}$

. Tem-se que:

$$\begin{cases} L_{i1} = t + p_i - d_i \\ L_{j1} = t + p_i + p_j - d_j \end{cases} \qquad \begin{cases} L_{i2} = t + p_j + p_i - d_i \\ L_{j2} = t + p_j - d_j \end{cases}$$

e obviamente, $\underbrace{L_{i1} < L_{i2}}_{*}$ e $L_{j2} < L_{j1}$.

. Sendo $d_i < d_j$, segue que $\underbrace{L_{j1} < L_{i2}}_{**}$.

. portanto $L_{j2} < L_{j1} < L_{i2}$.

. Logo, pode-se escrever

$$L_{max,\ seq.1} = \max\{L, L_{i1}, L_{j1}\}$$

$$L_{max,\ seq.2} = \max\{L, L_{i2}\}$$

. Então, de (*) e (**) segue que:

$$L_{max,\ seq.\ 1} \leq L_{max,\ seq.\ 2}$$

. para T_{max} :

$$\begin{cases} T_{max,\,seq.1} = \max\{0, L_{max,\,seq.1}\} \leq \max\{0, L_{max,\,seq.2}\} \\ T_{max,\,seq.2} = \max\{0, L_{max,\,seq.2}\} \end{cases}$$

. Logo $T_{max,\ seq.\ 1} \leq T_{max,\ seq.\ 2}$

III) $n/1/\overline{F}\,|\,n(T) = 0$ (ou $n/1/\overline{F}\,|\,T_{max} = 0$)

n(T) = número de tarefas programadas com atraso.

- <u>Teorema de SMITH (1956)</u>:

Se todas as tarefas puderem ser programadas sem atraso, então existe uma ordenação das tarefas com a tarefa J_k sendo a última, a qual minimiza $\overline{F}$, sujeito a $n(T) = 0$ (ou $T_{max} = 0$), SE E SOMENTE SE a tarefa J_k tiver as seguintes propriedades:

a) $d_k \geq \sum_{i=1}^{n} p_i$ e

b) $p_k \geq p_i$ para todo i tal que $d_i \geq \sum_{j=1}^{n} p_j$.

- Exemplo

J_i	J_1	J_2	J_3	J_4	J_5
p_i	4	2	3	5	3
d_i	10	10	12	18	6

IV) $n/1/\overline{F}\,|\,\min T_{max}$

Este problema pode ser considerado como uma extensão do problema de SMITH (1956), ou seja, $n/1/\overline{F}\,|\,T_{max} = 0$.

- Teorema de HECK and ROBERTS (1972):

"No problema n/1 existe uma ordenação das tarefas com a tarefa J_k na última posição, a qual minimiza $\overline{F}$ sujeito à condição de mínimo T_{max}, SE E SOMENTE SE:

a) $d_K \geq \sum_{i=1}^{n} p_i - T^*_{max}$ e

b) $p_K \geq p_i$ para todo i tal que $d_i \geq \sum_{j=1}^{n} p_j - T^*_{max}$ "

Obs.: 1) notar que $T_{max} = 0$ leva ao teorema de Smith (problema III).

2) $T^*_{max} = \min T_{max}$

- Exemplo

J_i	J_1	J_2	J_3	J_4
p_i	4	3	2	2
d_i	3	4	5	6

V) n /1/ n(T)

"Minimizar o número de tarefas a serem programadas com atraso".

- ALGORITMO DE MOORE (1968):

. PASSO 1: Ordene as tarefas segundo a regra SPT. Esta ordenação é chamada de seqüência atual.

. PASSO 2: Usando a seqüência atual, encontre a primeira tarefa com atraso $J_{[q]}$ e vá para o passo 3.
Se tal tarefa não for encontrada, o algoritmo está terminado, obtendo-se uma seqüência ótima com a ordenação das tarefas da seqüência atual, de acordo com a regra EDD, e as demais tarefas (removidas em 3.2) em uma ordem qualquer.

. PASSO 3: Reordene as tarefas $J_{[1]}, J_{[2]}, ..., J_{[q]}$ segundo a regra EDD.
Duas situações podem ocorrer:

3.1) Se todas as tarefas da nova subseqüência $J_{[1]}, J_{[2]}, ..., J_{[q]}$ não estiverem atrasadas, defina a seqüência total das tarefas como a seqüência atual e vá para o passo 2.

3.2) Caso contrário, rejeite a tarefa $J_{[q]}$ escolhida no passo 2, removendo-a da seqüência atual para a seqüência das tarefas removidas (parte final da seqüência total). Vá para o passo 2.

- Versão de HODGSON do Algoritmo de MOORE:

. PASSO 1: Ordene as tarefas segundo a regra EDD. Esta ordenação é denominada seqüência atual.

. PASSO 2: Na seqüência atual, identifique a primeira tarefa com atraso. Esta tarefa e as suas precedentes determinam uma subseqüência de confronto. Se tal tarefa for identificada, vá para o passo 3. Caso contrário, a seqüência ótima é a seqüência atual seguida da seqüência das tarefas removidas (no passo 3).

. PASSO 3: Remova a tarefa com maior tempo de processamento entre as tarefas da subseqüência de confronto e coloque-a na seqüência das tarefas removidas. A seqüência total é formada pela seqüência atual e seqüência das tarefas removidas. Vá para o passo 2.

- Exemplo: 8 /1/ n(T)

J_i	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
p_i	10	6	3	1	4	8	7	6
d_i	35	20	11	8	6	25	28	9

VI) n /1/ n(T)| E

E: conjunto de tarefas que não devem ter atraso.

$E \subset \{J_1, J_2, ..., J_n\}$

OBS.: Se $E = \phi \rightarrow$ problema de MOORE (prob. V).

Portanto, este problema pode ser considerado como uma extensão do problema n/1/n(T).

- Teorema de SIDNEY (1973), na versão de EMMONS (1975):

"Para o problema $n/1/n(T)|E$, se todas as tarefas do conjunto E puderem ser programadas sem atraso, então existe uma seqüência ótima das tarefas na qual a tarefa J_K é a última, onde, com as tarefas enumeradas segundo a regra EDD:

a) A tarefa $J_{[j]}$ é a primeira tarefa com atraso, na seqüência EDD;

b) $J' = \{J_{[1]}, J_{[2]}, \ldots, J_{[j]}\}$ sub-seqüência de confronto;

c) $p_k = \max_{i \in \{J' \cap \overline{E}\}} p_i$ onde $\overline{E} = \{J_i \mid i \notin E\}$

OU SEJA:

"Entre as tarefas da sub-seqüência de confronto J' que podem ter atraso ($\notin$ E), escolhe-se a de maior tempo de processamento para ser colocada na última posição (isto é, na seqüência das tarefas removidas)".

Exemplo: Resolver o ex. do problema anterior (prob. V) 8/1/n(T) com $E = \{J_4, J_6\}$.

VII) $n/1/\overline{F} \mid E$

- Teorema de EMMONS (1975)

"Para o problema $n/1/\overline{F} \mid E$, se todas as tarefas do conjunto E puderem ser programadas sem atraso, então existe uma seqüência ótima das tarefas na qual a tarefa J_K é a última, onde:

$$p_k = \max_{i \in \{\overline{E} \cup M\}} p_i$$

onde $\{\overline{E} \cup M\}$ é o conjunto das tarefas que podem ter atraso ou que não tem atraso mesmo sendo a última".

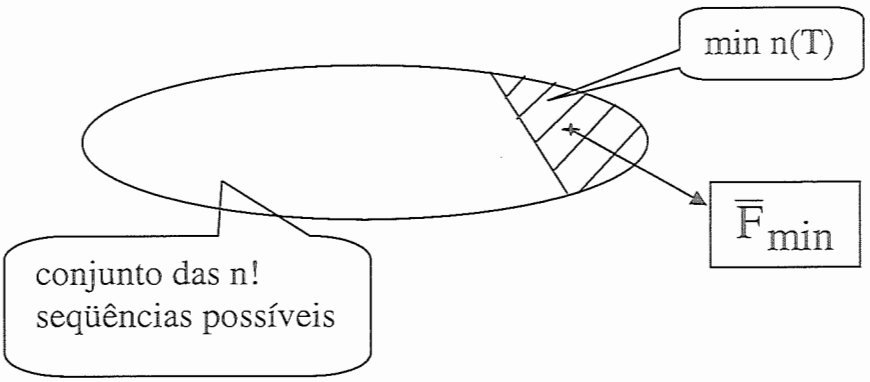
OBS.: Se $E = \phi \rightarrow$ problema $n/1/\overline{F}$ (resolvido pela regra SPT)

- Exemplo: Resolver o ex. do problema V com $E = \{J_4, J_6\}$.

J_i	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
p_i	10	6	3	1	4	8	7	6
d_i	35	20	11	8	6	25	28	9

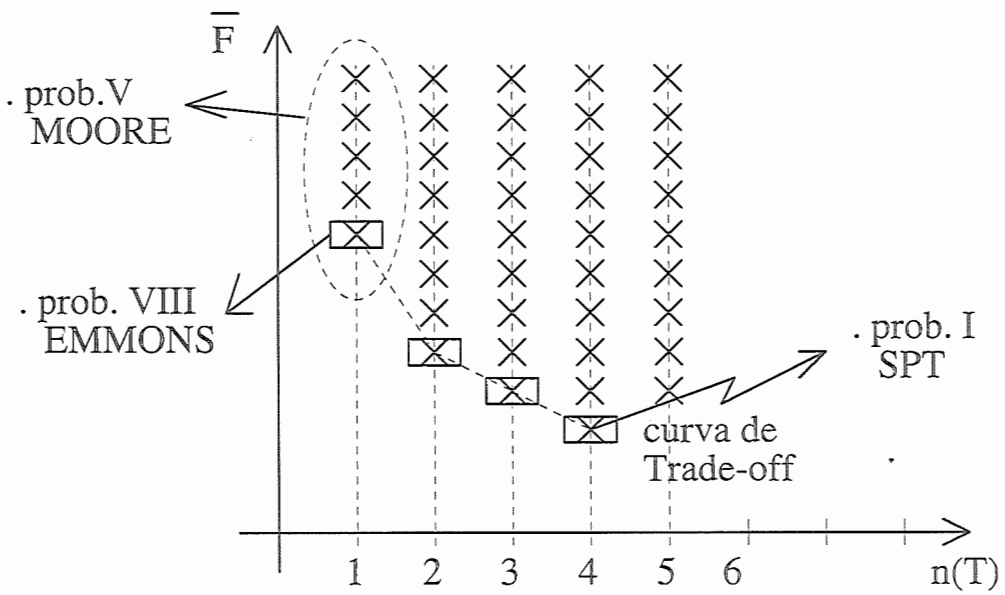
VIII) n /1/ $\overline{F}$ | min n(T)

- Algoritmo de MOORE → min n(T)



- EMMONS (1975) propõe a solução do problema por meio de um algoritmo "*BRANCH and BOUND*".

IX) n /1/ $\overline{F}$ e n(T)



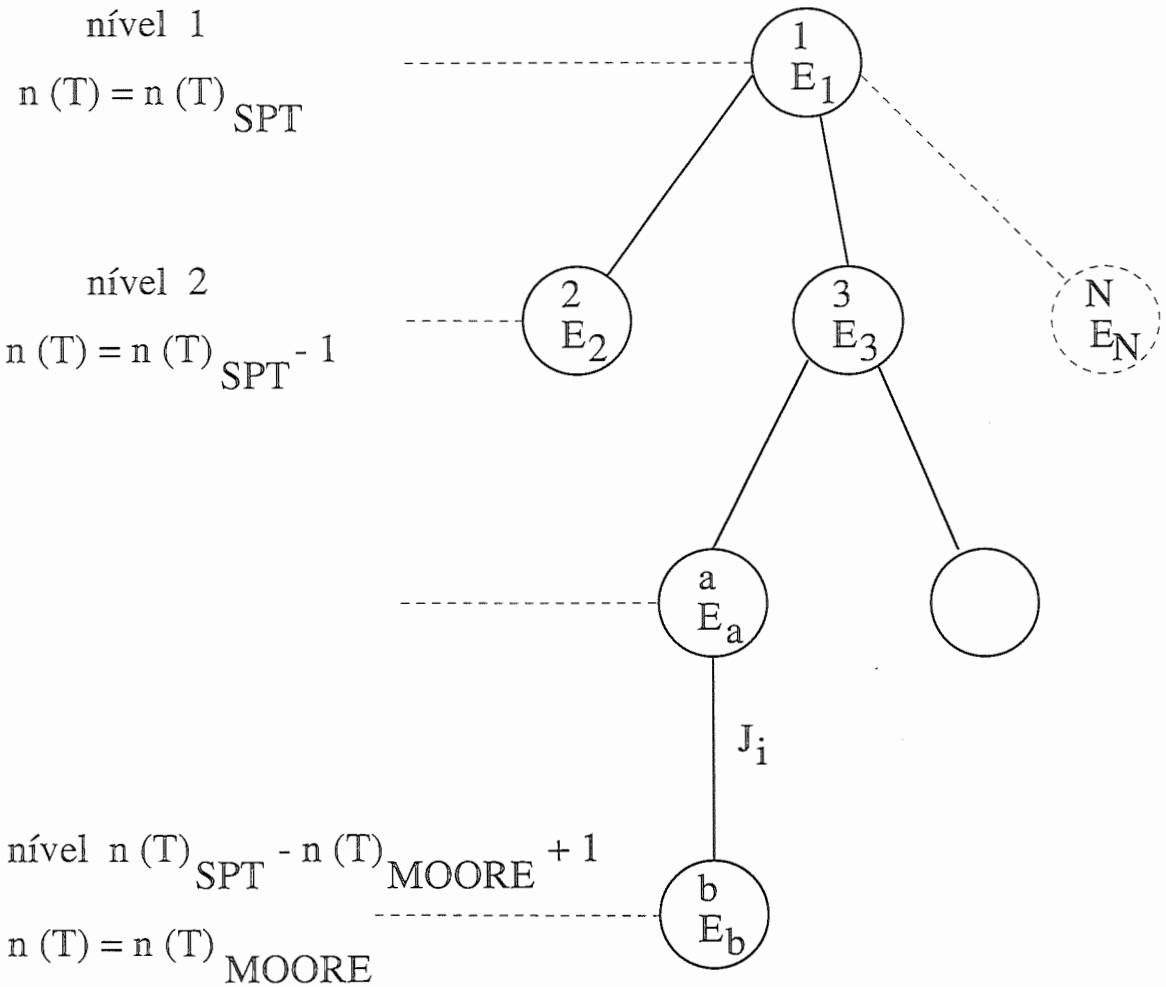
⊠ = Soluções EFICIENTES

. As <u>soluções eficientes</u> são obtidas a partir da solução do problema:

$$\text{MIN } \overline{F} \quad \text{sujeito a} \quad n(T) \leq K$$

$$\text{onde } n(T)_{MOORE} \leq K \leq n(T)_{SPT} \; .$$

. NELSON *et al.* (1986) propõem a solução do problema por meio de um procedimento em árvore:



$$E_b = E_a \cup \{J_i\}$$
$$J_i \notin E_a$$

NA ÁRVORE:

- Um nó ⓐ é definido pelo conjunto de tarefas E_a, as quais não podem ter atraso no correspondente nó.
- Um ramo que parte de um nó ⓐ para um nó ⓑ, corresponde à designação de alguma tarefa J_i para ser incluída no conjunto E_a formando o conjunto E_b.
- Assume-se que as tarefas estejam enumeradas segundo a regra SPT:

$J = \{J_1, J_2, ..., J_n\}$ onde $p_1 \leq p_2 \leq ... \leq p_n$.

- TEOREMA (para redução do tamanho da árvore):

(1) Se a tarefa J_i não tem atraso ($T_i = 0$) na programação obtida pela regra SPT, então existe uma programação que minimiza $\overline{F}$ sujeito a $n(T) \leq K$, na qual a tarefa J_i não tem atraso.

(2) Suponha que somente uma das tarefas J_i ou J_j podem não ter atraso em uma programação das n tarefas que minimiza $\overline{F}$ sujeito a $n(T) \leq K$.
Se $p_i \geq p_j$ e $p_i - p_j \geq d_i - d_j$, então a tarefa J_i não precisa ser considerada para inclusão no conjunto de tarefas que não devem ter atraso.

- Algoritmo de NELSON *et al.*

- PASSO 1: Determine a seqüência de Moore e a seqüência SPT. A árvore completa consistirá de $n(T)_{SPT} - n(T)_{MOORE} + 1$ níveis.

- PASSO 2: No nó a = 1, faça E_1 = conjunto das tarefas sem atraso na seqüência SPT.

- PASSO 3: Expanda para novos nós 2, 3, ..., N onde $N = n - |E_1| + 1$. Estes nós são gerados pela inclusão de uma tarefa $\notin E_1$ no conjunto das tarefas sem atraso. As tarefas são incluídas em E_1 em ordem não decrescente dos tempos de processamento. Os nós 2, 3, ..., N, constituem o 2o. nível da árvore.

EESC-USP Biblioteca CAMPUS DE SÃO CARLOS

- PASSO 4: Use a regra de dominação dada na parte (2) do teorema, para exclusão de possíveis nós.

- PASSO 5: Avalie cada um dos nós remanescentes, resolvendo o problema $n/1/\overline{F} \mid E_a$ e registre os valores de $\overline{F}$ e n(T) de cada seqüenciamento resultante. Se existir uma solução eficiente com $n(T)_{SPT} - 1$ tarefas com atraso, ela será identificada como a de menor $\overline{F}$ entre os nós desse nível da árvore.

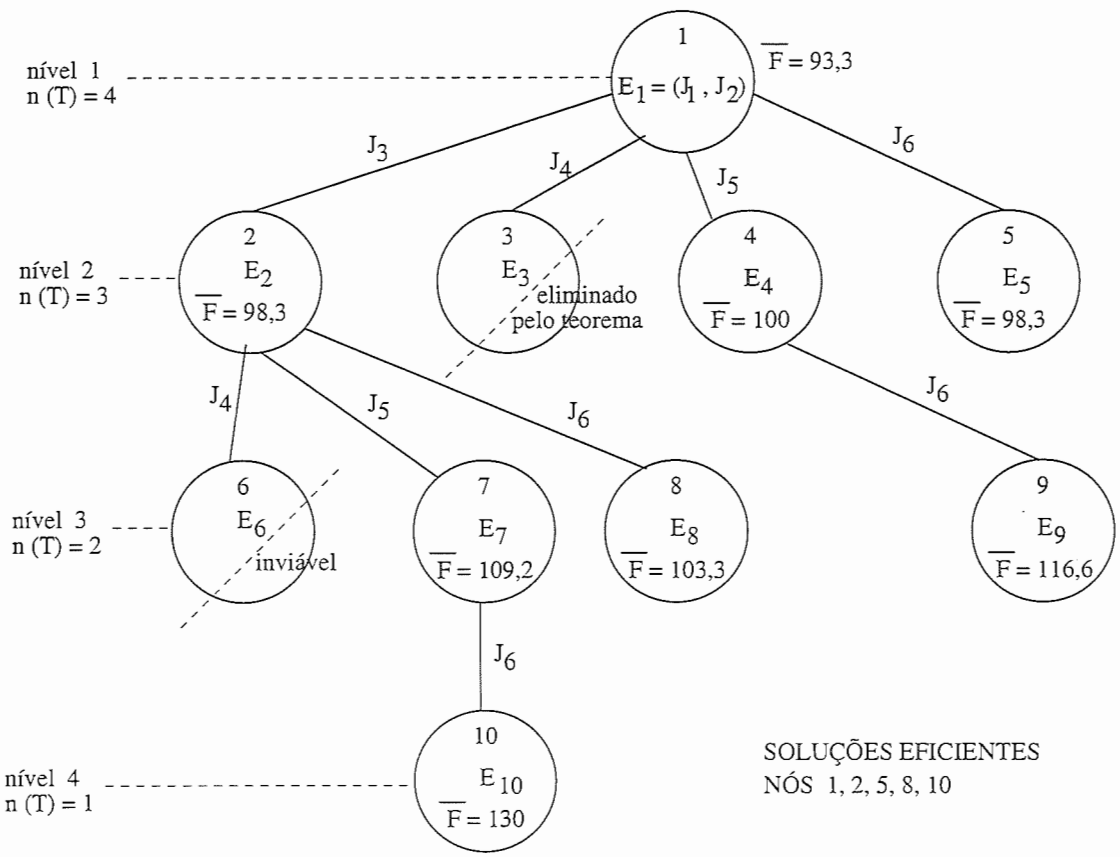
- PASSO 6: Expanda a partir de cada nó não eliminado do nível 2, pela inclusão de uma tarefa adicional em E_a. Para evitar a geração dupla de conjuntos idênticos de tarefas sem atraso (conjuntos E_a para a = 1, 2, ...), a expansão de E_a inclui apenas aquelas tarefas que tenham numeração mais alta do que a da tarefa colocada em E_a, no nível 2. Os nós do nível 3 são enumerados consecutivamente na ordem de sua geração, iniciando com (N+1).

- PASSO 7: Use a regra de dominação da parte (2) do teorema para eliminar possíveis nós do nível 3. Avalie os nós remanescentes do nível 3, resolvendo o problema $n/1/\overline{F} \mid E_a$. Novamente, se existir uma solução eficiente com $n(T) = n(T)_{SPT} - 2$, ela será encontrada no nível 3 como o seqüenciamento com o menor $\overline{F}$, neste nível.
  O procedimento termina quando a solução eficiente para $n(T) = n(T)_{MOORE}$ é obtida no nível $n(T)_{SPT} - n(T)_{MOORE} + 1$ da árvore.

- <u>Exemplo</u>: 6 /1/ $\overline{F}$ e n (T)

J_i	J_1	J_2	J_3	J_4	J_5	J_6
p_i	5	10	30	50	60	90
d_i	245	15	40	55	100	190

CRITÉRIO	ALGORITMO	SEQÜÊNCIA	$\overline{F}$	n (T)
$\overline{F}$	SPT	J_1 - J_2 - J_3 - J_4 - J_5 - J_6	93,3	4
n (T)	Moore	J_2 - J_3 - J_5 - J_6 - J_1 - J_4	130,0	1



- nó ①: $\begin{cases} E_1 = \{J_1, J_2\} \\ \text{seq. } J_1 - J_2 - J_3 - J_4 - J_5 - J_6 \\ \overline{F} = 93{,}3 \qquad n(T) = 4 \end{cases}$

- nó ②: $\begin{cases} E_2 = \{J_1, J_2, J_3\} \\ \text{seq. } J_2 - J_3 - J_1 - J_4 - J_5 - J_6 \\ \overline{F} = 98{,}3 \qquad n(T) = 3 \end{cases}$

- nó ③: $\begin{cases} E_3 = \{J_1, J_2, J_4\} \\ \text{eliminado pelo teorema} \end{cases}$

- nó ④: $\begin{cases} E_4 = \{J_1, J_2, J_5\} \\ \text{seq. } J_1 - J_2 - J_5 - J_3 - J_4 - J_6 \\ \overline{F} = 100 \qquad n(T) = 3 \end{cases}$

- nó ⑤: $\begin{cases} E_5 = \{J_1, J_2, J_6\} \\ \text{seq. } J_1 - J_2 - J_3 - J_4 - J_6 - J_5 \\ \overline{F} = 98{,}3 \qquad n(T) = 3 \end{cases}$

- nó ⑥: $\begin{cases} E_6 = \{J_1, J_2, J_3, J_4\} \\ \text{seq. INVIÁVEL} \end{cases}$

- nó ⑦: $\begin{cases} E_7 = \{J_1, J_2, J_3, J_5\} \\ \text{seq. } J_2 - J_3 - J_5 - J_1 - J_4 - J_6 \\ \overline{F} = 109{,}2 \qquad n(T) = 2 \end{cases}$

- nó 8 : $\begin{cases} E_8 = \{J_1, J_2, J_3, J_6\} \\ \text{seq. } J_2 - J_3 - J_1 - J_4 - J_6 - J_5 \\ \overline{F} = 103{,}3 \qquad n(T) = 2 \end{cases}$

- nó 9 : $\begin{cases} E_9 = \{J_1, J_2, J_5, J_6\} \\ \text{seq. } J_1 - J_2 - J_5 - J_6 - J_3 - J_4 \\ \overline{F} = 116{,}6 \qquad n(T) = 2 \end{cases}$

- nó 10 : $\begin{cases} E_{10} = \{J_1, J_2, J_3, J_5, J_6\} \\ \text{seq. } J_2 - J_3 - J_5 - J_6 - J_1 - J_4 \\ \overline{F} = 130{,}0 \qquad n(T) = 1 \end{cases}$

3 - PROGRAMAÇÃO DE MÁQUINAS PARALELAS

3.1 - Considerações Gerais

. $J = \{J_1, J_2, ..., J_n\}$ $n \geq 2$ finito, número de tarefas a serem processadas.

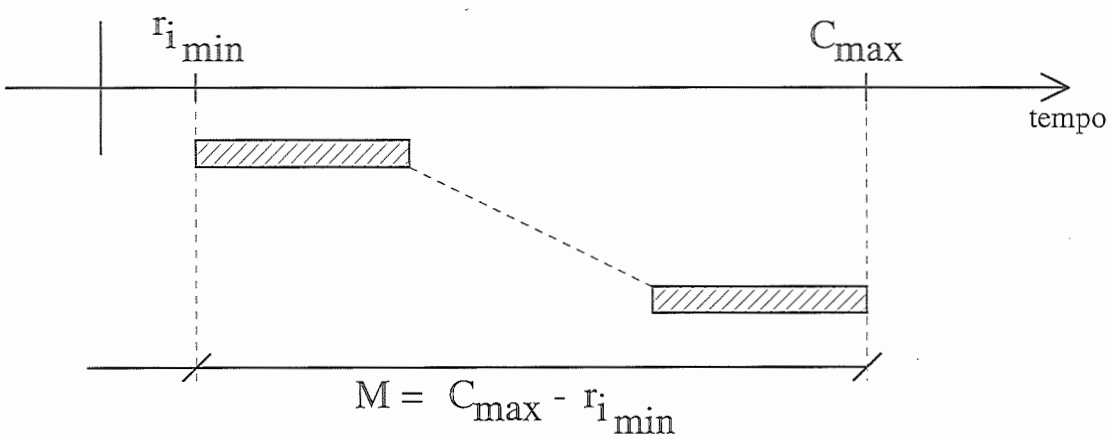
. $M = \{M_1, M_2, ..., M_m\}$ $2 \leq m < n$ finito, conjunto de máquinas <u>idênticas</u>.

. $g_i = 1$ $i = 1, 2, ..., n$ uma única operação por tarefa.

. $r_i = r$ ou $r_i = 0$ todas as tarefas têm a mesma data de liberação.

. tempo de processamento $p_i = cte$ $i = 1, 2, ..., n$.

. <u>Def.</u> MAKESPAN (M) = duração total da programação.



. se $r_i = r = cte \rightarrow C_{max} - r_{i\,min} = F_{max}$

ou seja $\boxed{M = F_{max}}$

3.2 - TAREFAS INDEPENDENTES

• PROBLEMA n /m, ID/ $\overline{F}$

• ALGORITMO 3.2.1 - Para obtenção de uma solução ótima.

• PASSO 1: Ordene todas as tarefas segundo a regra SPT.

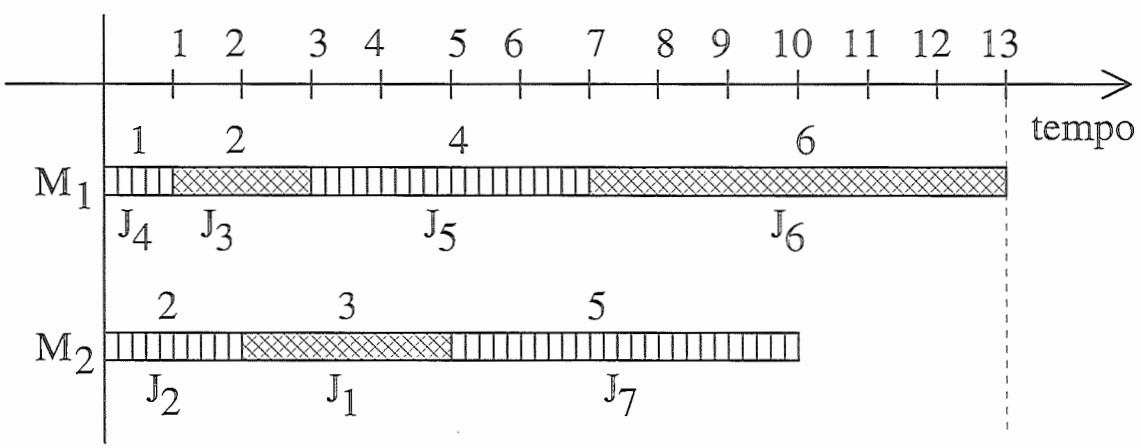
• PASSO 2: Designe à máquina com a menor quantidade de processamento já alocado, a próxima tarefa da seqüência SPT (desempates são arbitrários). Repita este procedimento até que todas as tarefas sejam designadas.

• Exemplo: m = 2

J_i	J_1	J_2	J_3	J_4	J_5	J_6	J_7
p_i	3	2	2	1	4	6	5

• seqüência SPT : J_4, J_2, J_3, J_1, J_5, J_7, J_6 .

• solução:



- $$\overline{F} = \frac{1}{n} \sum_{j=1}^{m} \sum_{i=1}^{n_j} (n_j - i + 1)\, p_{[i]j}$$

  n_j = número de tarefas designadas à máquina M_j (a serem processadas por M_j).

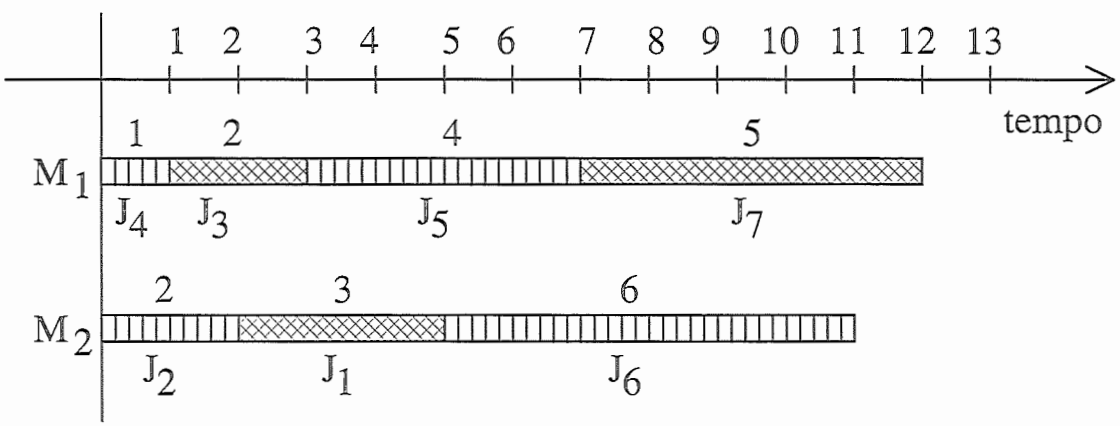
  $p_{[i]j}$ = tempo de processamento da tarefa na i-ésima posição da máquina M_j.

- <u>No exemplo</u>:

$$\overline{F} = \frac{1}{7}(4 + 6 + 8 + 6 + 6 + 6 + 5) = \frac{41}{7} = \min \overline{F}$$

$$M = F_{max} = 13$$

- Considere a seguinte programação:



$$\overline{F} = \frac{1}{7}(1 + 3 + 7 + 12 + 2 + 5 + 11) = \frac{41}{7} = \min \overline{F}$$

$$M = F_{max} = 12.$$

- ALGORITMO 3.2.2 - Generalização do algoritmo 3.2.1.

- PASSO 1: Ordene todas as tarefas segundo a regra SPT.

- PASSO 2: Designe as m tarefas com os maiores tempos de processamento a cada uma das m máquinas, colocando-as na última posição.

- PASSO 3: Designe, dentre as (n - m) tarefas remanescentes, as m tarefas (desde que $n \geq 2m$) com os maiores tempos de processamento a cada uma das m máquinas colocando-as na penúltima posição.

e assim, sucessivamente, até a n-ésima tarefa ser designada.

- PROBLEMA $n/m, ID/F_{max}$ ou M

- ALGORITMO 3.2.3 - Um procedimento heurístico baseado na regra LPT (*Longest Processing Time*)

- PASSO 1: Ordene todas as tarefas segundo a regra LPT.

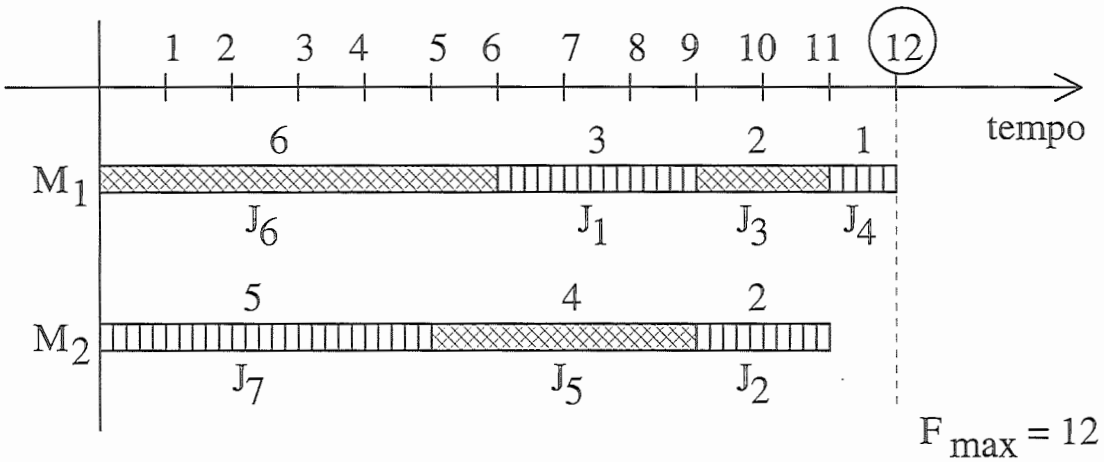
- PASSO 2: Programe as tarefas segundo a ordem LPT, designando, de cada vez, uma tarefa à máquina com a menor quantidade de processamento já alocado.

- Exemplo: $m = 2$

J_i	J_1	J_2	J_3	J_4	J_5	J_6	J_7
p_i	3	2	2	1	4	6	5

- seqüência LPT : J_6, J_7, J_5, J_1, J_3, J_2, J_4.

- solução:



- Para avaliação da eficácia do procedimento (qualidade da solução), compara-se o F_{max} obtido com o valor M*, o qual é um Limitante Inferior para o min F_{max} (ou min M).

  - $\min F_{max} \geq M^*$
  - $M^* = \max \left\{ \frac{1}{m} \sum_{i=1}^{n} p_i, \max_i p_{.i} \right\}$.

- No caso do exemplo:

$$M^* = \max \left\{ \frac{23}{2}, 6 \right\} = 11{,}5$$

Min $F_{max} \geq 11{,}5$ e portanto, nota-se que a solução fornecida pelo procedimento heurístico é uma solução ótima ($F_{max} = 12$), uma vez que não é permitida a subdivisão na execução das tarefas.

- UMA FORMULAÇÃO EM PROGRAMAÇÃO LINEAR INTEIRA PARA SOLUÇÃO EXATA DO PROBLEMA n /m, ID/ M.

- MODELO GERAL DE PROGRAMAÇÃO LINEAR:

Determinar $(x_1, x_2, ..., x_n)$

$$\begin{cases} \text{de modo que} \left\{ \sum_{j=1}^{n} a_{ij} x_j \otimes b_i \quad i = 1, 2, ..., m. \right. \\ \text{e tal que otimizem } z = \sum_{j=1}^{n} c_j x_j \end{cases}$$

$\otimes \in \{\leq, \geq, =\}$ otimizar $\begin{cases} \text{minimizar} \\ \text{maximizar} \end{cases}$ ou

a) $x_j \geq 0 \quad (x_j \in R^+) \rightarrow$ PROG. LINEAR

b) $x_j \geq 0$ inteira $\rightarrow$ PROG. LINEAR INTEIRA

c) $\left.\begin{array}{l} x_j \geq 0 \quad j = 1, 2, ..., K \\ x_j \geq 0 \quad \text{inteira} \quad j = K+1, ..., n \end{array}\right\}$ PROGR. LINEAR INTEIRA MISTA

Seja:

M = duração total da programação (F_{max})

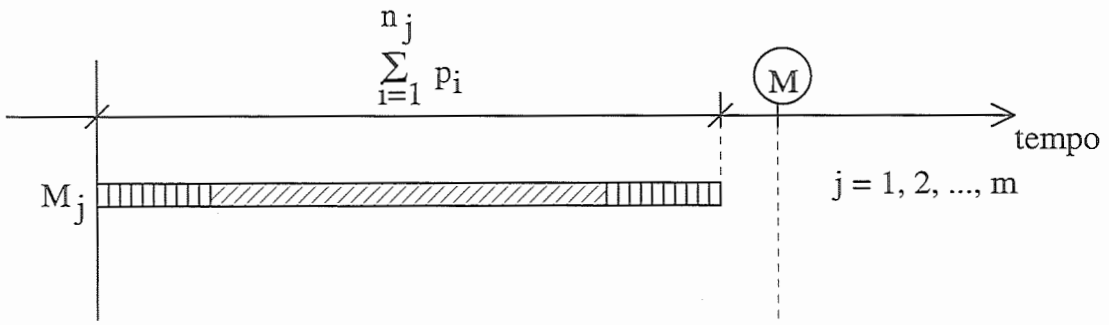
$$\begin{cases} x_{ij} = 1 \text{ , se a tarefa } J_i \text{ for designada à máquina } M_j \\ x_{ij} = 0 \text{ , caso contrário} \end{cases}$$

- CONDIÇÕES A SEREM SATISFEITAS (Restrições):

a) $J_i \rightarrow$ única M_j $\quad$ $i = 1, 2, ..., n$ $\quad$ $j = 1, 2, ..., m$

- $\sum_{j=1}^{m} x_{ij} = 1 \qquad i = 1, 2, ..., n$

b) Cargas das máquinas $\leq$ duração total da programação



$$\sum_{i=1}^{n_j} p_i \leq M \qquad \text{onde} \qquad \sum_{i=1}^{n_j} p_i = \sum_{i=1}^{n} p_i \, x_{ij}$$

- FORMULAÇÃO:

Minimizar M

$$\text{sujeito a} \begin{cases} \sum_{j=1}^{m} x_{ij} = 1 & i = 1, 2, \ldots, n \\ M - \sum_{i=1}^{n} p_i \, x_{ij} \geq 0 & j = 1, 2, \ldots, m \end{cases}$$

com $M \geq 0$

$x_{ij} \in \{0, 1\}$

$$\text{OBS.:} \begin{cases} \text{No. de restrições} = m + n \\ \text{No. de variáveis} = m \,.\, n + 1 \end{cases}$$

• PROBLEMA n /m, ID/ $\overline{F}$	min F_{max}

- ALGORITMO 3.2.4

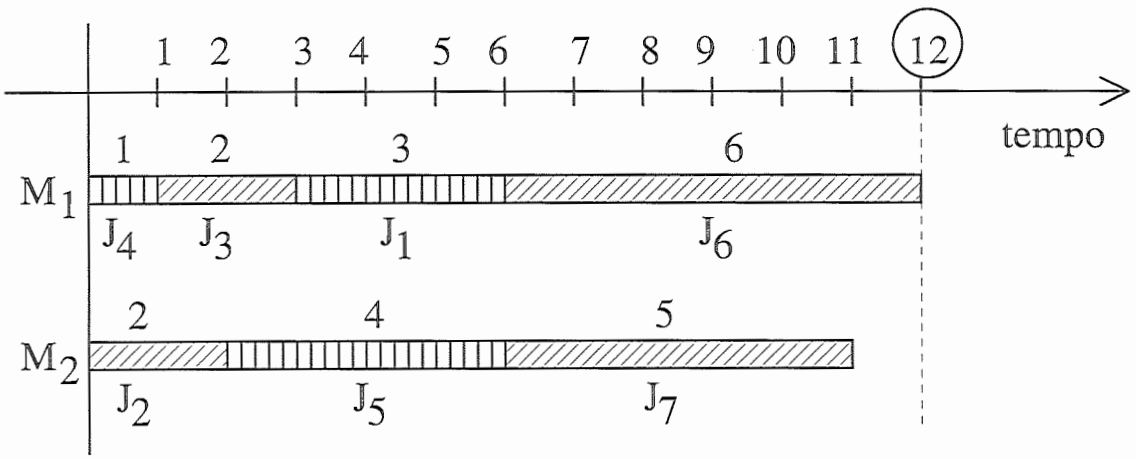
- PASSO 1: Resolva o problema n /m, ID/ F_{max}, usando o algoritmo 3.2.3 ou o modelo de Programação Linear Inteira.

- PASSO 2: Reordene as tarefas designadas a cada máquina, segundo a regra SPT.

- Exemplo: $m = 2$

J_i	J_1	J_2	J_3	J_4	J_5	J_6	J_7
p_i	3	2	2	1	4	6	5

- Solução utilizando no passo 1 o algoritmo 3.2.3:



- $F_{max} = 12$

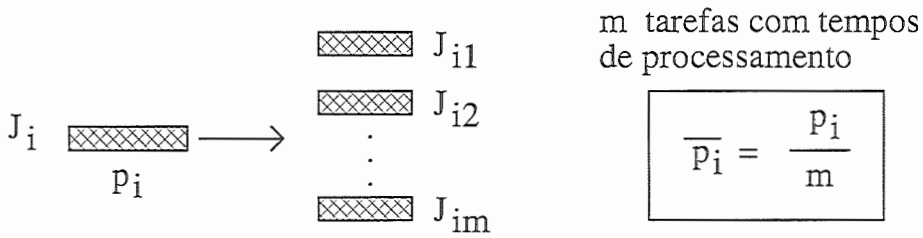
$$\overline{F} = \frac{1}{7}\,(1+3+6+12+2+6+11) = \frac{41}{7} = \min \overline{F}$$

- Caso Particular: $\boxed{\text{Problema n, SPLIT / m, ID}}$

  - <u>HIPÓTESES ADICIONAIS</u>:

. Cada tarefa J_i pode ser dividida em um conjunto de **m** tarefas independentes e com possibilidade de processamento simultâneo.

. Tempo de processamento $p_i = s_i + p'_i + t_i = cte$ com s_i e t_i "nulos" ou não significativos.



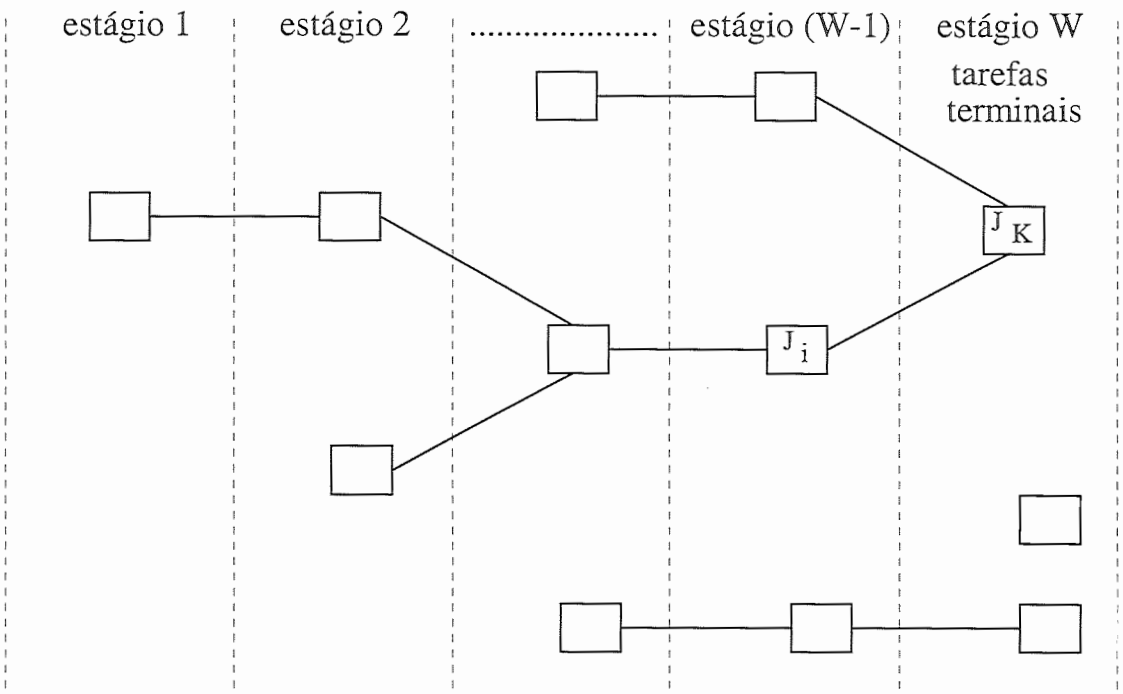
- Neste caso:

Problema n, SPLIT/m, ID $\rightarrow$ problema n/1 .

3.3 - TAREFAS DEPENDENTES

- RESTRIÇÕES ADICIONAIS: As n tarefas são parcialmente ordenadas segundo precedências tecnológicas.

- Caso de ÁRVORES DE MONTAGEM:



- PROBLEMA n, DEP-T /m, ID/ M

- ALGORITMO DE HU (procedimento heurístico)

- Fase de cálculo dos índices de prioridade:

- PASSO 1: Designe a cada tarefa terminal J_k (estágio W) um índice de prioridade α_k igual ao seu tempo de processamento p_k .

- PASSO 2: Designe a cada tarefa J_i do estágio (W - 1) um índice de prioridade $\alpha_i = \alpha_k + p_i$ onde $J_i >> J_k$. Repita o procedimento para todos os estágios, sucessivamente, até o estágio 1.

Fase de programação:

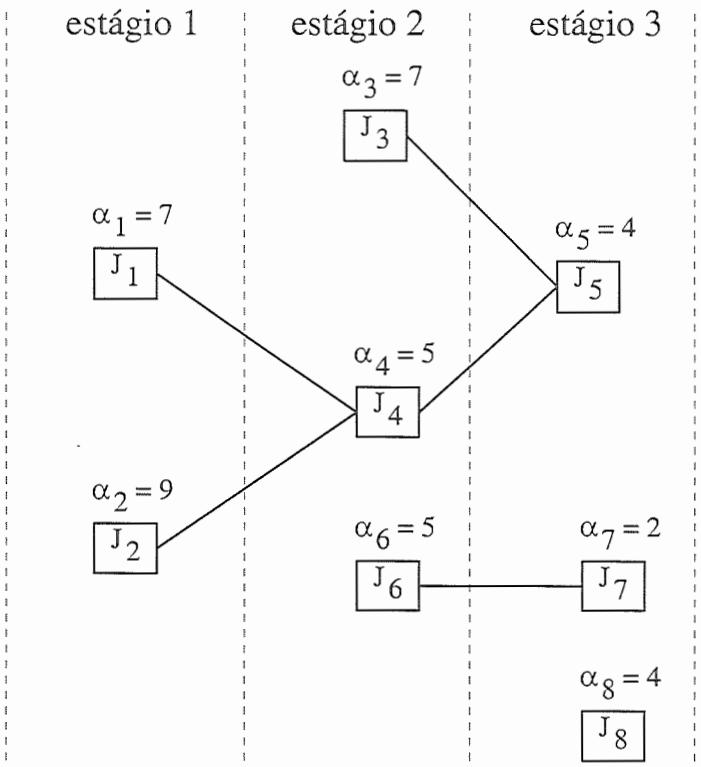
- PASSO 3A: Se o número de tarefas sem tarefas precedentes for menor ou igual a m, programe tais tarefas simultaneamente observando-se as restrições de precedência, deixando as máquinas restantes sem utilização. Vá para o passo 4.

- PASSO 3B: Se o número de tarefas sem tarefas precedentes for maior que m, programe simultaneamente as m tarefas com os maiores índices de prioridade (desempates são arbitrários), observando-se as restrições de precedência. Vá para o passo 4.

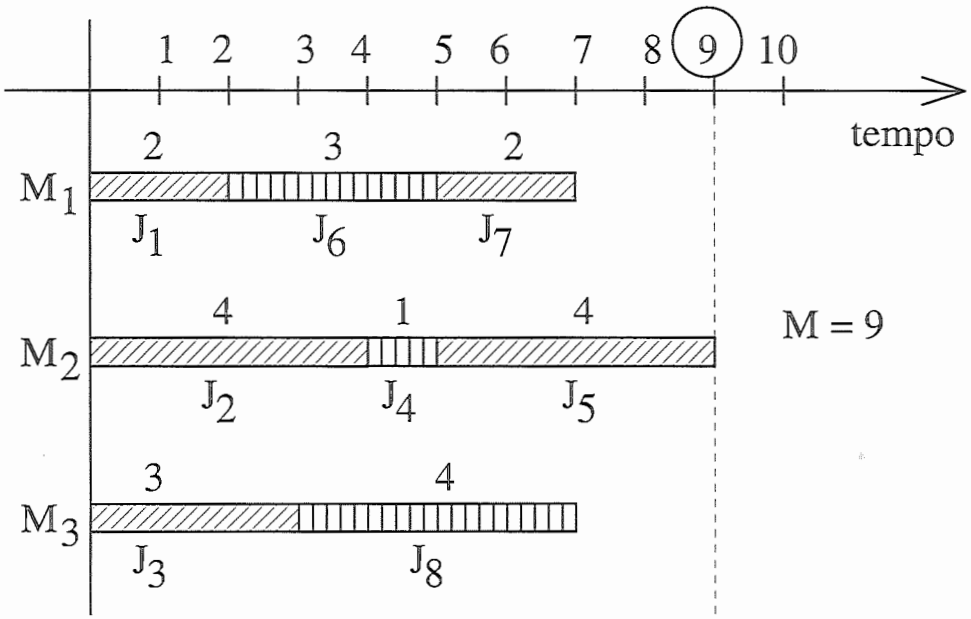
- PASSO 4: Remova as tarefas já programadas do conjunto de n tarefas e volte ao passo 3, até que todas as tarefas sejam programadas.

- Exemplo: m = 3

J_i	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
p_i	2	4	3	1	4	3	2	4


estágio 1
estágio 2
estágio 3
$\alpha_3 = 7$
J_3
$\alpha_1 = 7$
J_1
$\alpha_5 = 4$
J_5
$\alpha_4 = 5$
J_4
$\alpha_2 = 9$
J_2
$\alpha_6 = 5$
J_6
$\alpha_7 = 2$
J_7
$\alpha_8 = 4$
J_8

- <u>solução</u>:


1 2 3 4 5 6 7 8 9 10
tempo
M_1
2 3 2
J_1 J_6 J_7
M_2
4 1 4
J_2 J_4 J_5
M_3
3 4
J_3 J_8
M = 9

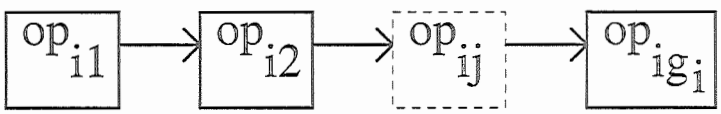
4 - PROGRAMAÇÃO DE SISTEMAS FLOW SHOP E JOB SHOP

4.1 - Considerações Gerais

- $J = \{J_1, J_2, ..., J_i, ..., J_n\}$ $n \geq 2$ finito, número de tarefas a serem processadas.

- $J_i = \{op_{i1}, op_{i2}, ..., op_{ij}, ..., op_{ig_i}\}$ conjunto de g_i operações ($1 \leq g_i \leq m$).

- Precedências tecnológicas entre as operações de uma tarefa J_i (estrutura linear):



- $r_i = r$ ou $r_i = 0$ $i = 1, 2, ..., n.$

- $p_{ij} = cte$ tempo de processamento da op_{ij} .

- $M = \{M_1, M_2, ..., M_m\}$ $m \geq 2$ finito, conjunto de máquinas <u>distintas</u>.

Cada máquina $\in M$ é capaz de executar um sub-conjunto de operações das diversas tarefas, sendo identificada através desse sub-conjunto de operações.

. Demais hipóteses do Job Shop básico.

4.2 - Algoritmo de Johnson

Para a solução do problema

$$\boxed{n / 2 / F / F_{max} \text{ ou } M}$$

•PROGRAMAÇÃO PERMUTACIONAL
(*PERMUTATION SCHEDULE*)

"Em um ambiente Flow Shop, uma programação permutacional é simplesmente uma programação na qual a seqüência das tarefas é a mesma em todas as máquinas".

• TEOREMA DE JOHNSON:

"No problema n/ 2/ F/ M, a programação ótima é uma programação permutacional onde: A tarefa J_i precede a tarefa J_k se

$$\min\{p_{i1}, p_{k2}\} \leq \min\{p_{i2}, p_{k1}\}" .$$

• ALGORITMO DE JOHNSON:

• PASSO 1: Determine $\min_i \{p_{i1}, p_{i2}\}$ $i = 1, 2, ..., n$.

• PASSO 2A: Se o mínimo tempo de processamento refere-se a uma operação op_{i1} a ser processada na máquina M_1, coloque a tarefa J_i na primeira posição disponível da seqüência de tarefas. Vá para o passo 3.

• PASSO 2B: Se o mínimo tempo de processamento refere-se a uma operação op_{i2} a ser processada na máquina M_2, coloque a tarefa J_i na última posição disponível da seqüência de tarefas. Vá para o passo 3.

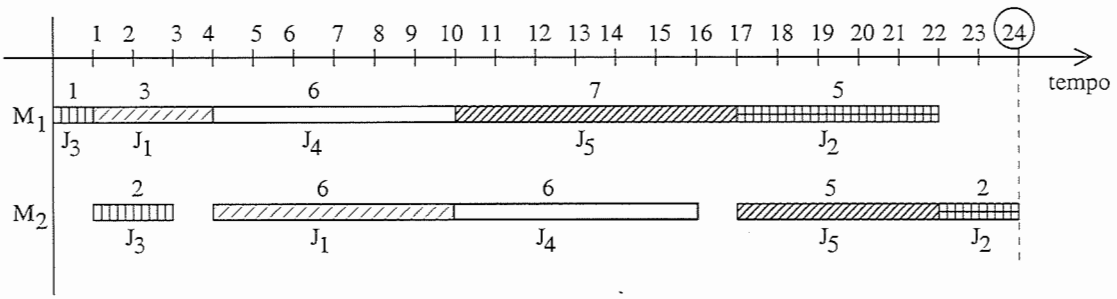
- PASSO 3: Remova a tarefa designada do conjunto de n tarefas e volte ao passo 1, até que todas as posições na seqüência de tarefas sejam ocupadas (os desempates podem ser feitos arbitrariamente).

- Exemplo:

J_i	J_1	J_2	J_3	J_4	J_5
p_{i1}	3	5	1	6	7
p_{i2}	6	2	2	6	5

ETA-PA	TAREFAS AINDA NÃO PROGRAMADAS	MÍNIMO p_{ij}	DESIGNAÇÃO	SEQÜÊNCIA PARCIAL
1	J_1, J_2, J_3, J_4, J_5	p_{31}	$J_3 = J_{[1]}$	J_3
2	J_1, J_2, J_4, J_5	p_{22}	$J_2 = J_{[5]}$	$J_3 - - - J_2$
3	J_1, J_4, J_5	p_{11}	$J_1 = J_{[2]}$	$J_3 J_1 - - J_2$
4	J_4, J_5	p_{52}	$J_5 = J_{[4]}$	$J_3 J_1 - J_5 J_2$
5	J_4	$p_{41} = p_{42}$	$J_4 = J_{[3]}$	$J_3 J_1 J_4 J_5 J_2$

- Programação



$$\min M = \min F_{max} = 24$$

4.3 - MÉTODOS *BRANCH AND BOUND*

- Procedimentos fundamentais:

I) RAMIFICAÇÃO (*BRANCHING*): processo de partição de um problema em dois ou mais sub-problemas.

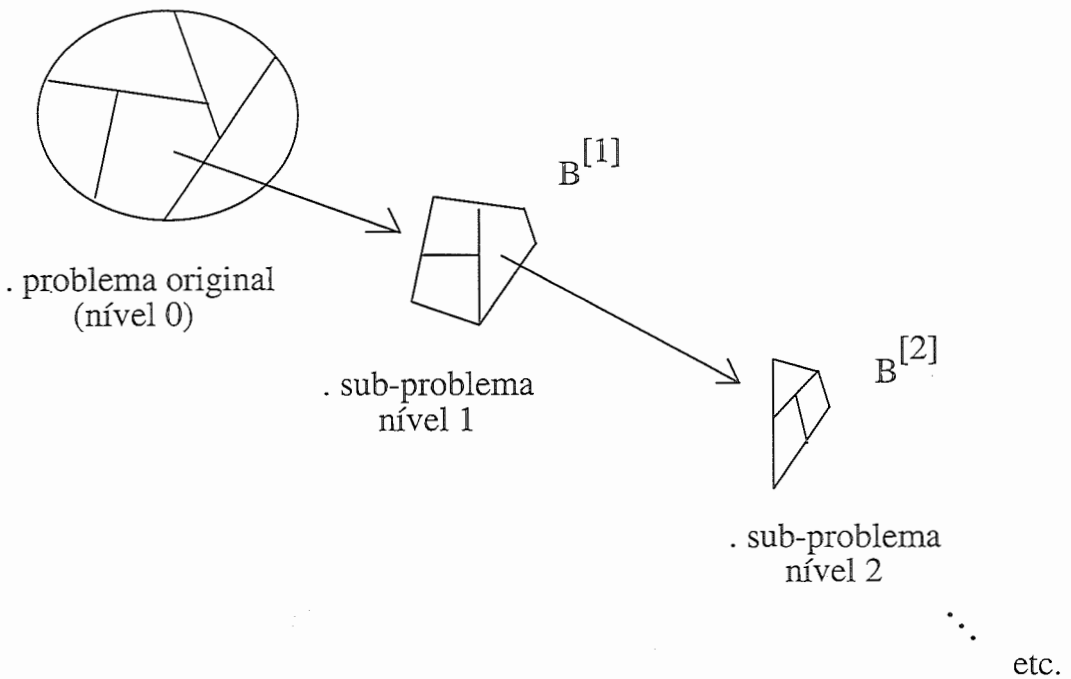
II) LIMITAÇÃO (*BOUNDING*): processo de cálculo de um valor limite (inferior e/ou superior) para a medida de desempenho adotada, referente à solução ótima de um dado sub-problema.

O procedimento de RAMIFICAÇÃO substitui um problema original por um conjunto de novos problemas, os quais são:

a) Sub-problemas mutuamente exclusivos e exaustivos do problema original.

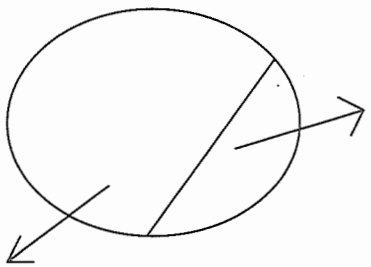
b) Versões parcialmente resolvidas do problema original.

c) Problemas menores do que o problema original.



• PROBLEMA n/3/F/ M ou F_{max}

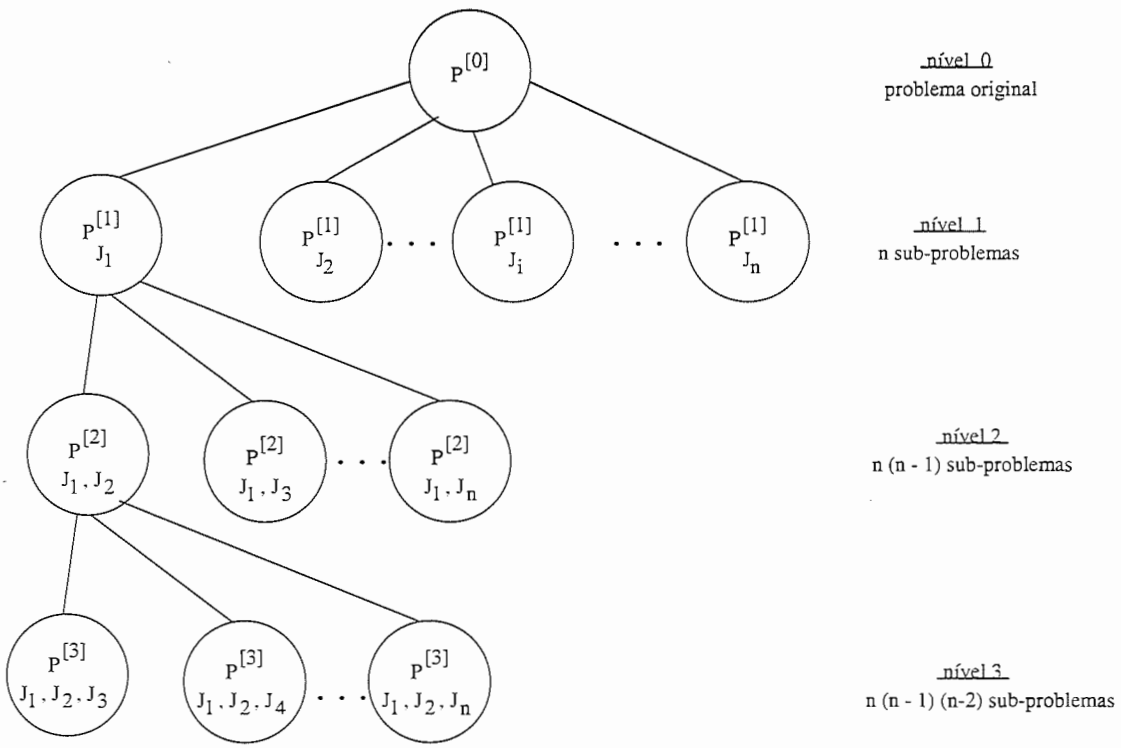
• $J = \{J_1, J_2, \ldots, J_i, \ldots, J_n\}$ $\qquad J_i = \{op_{i1}, op_{i2}, op_{i3}\}$



n! = No. de programações permutacionais

$(n!)^m$ = No. total de programações viáveis $\begin{bmatrix} g_i = m \\ r_i = r = cte \end{bmatrix}$

• <u>ÁRVORE de geração de sub-problemas (procedimento de ramificação) para programações permutacionais</u>:



<u>nível n</u>: n! sub-problemas

- Na árvore:

$P^{[\ell]}_{J_r,\ldots,J_s}$ = sub-problema do nível ℓ, no qual as ℓ tarefas $\{J_r,\ldots,J_s\}$ ocupam, nesta ordem, as ℓ primeiras posições da seqüência de tarefas em cada máquina e as demais (n - ℓ) tarefas ocupando posições quaisquer.

- obs: Cada sub-problema do nível n, corresponde a uma solução completa (programação viável) do problema original $P^{[0]}$.

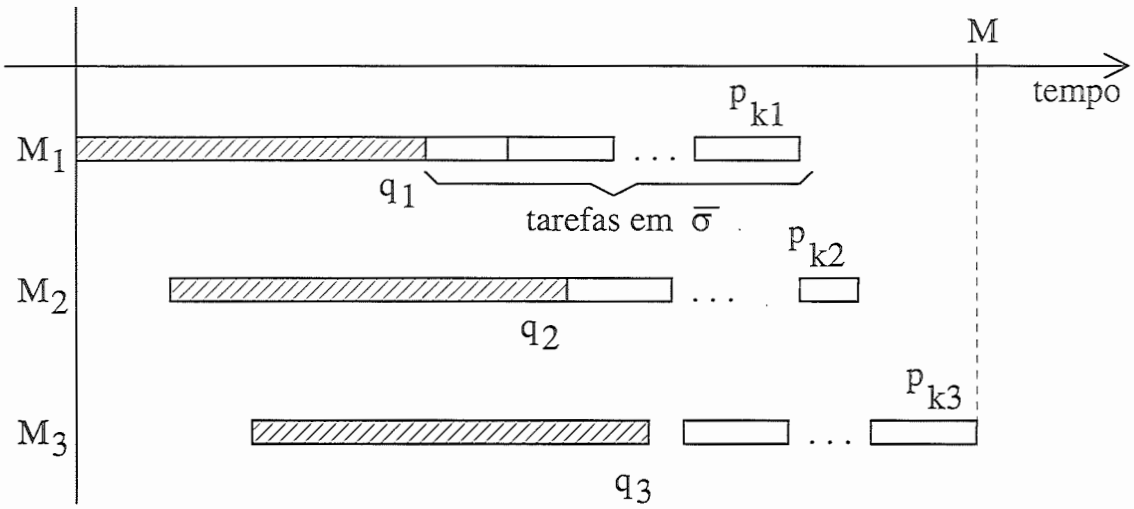
Seja $\begin{cases} \sigma = \{J_r,\ldots,J_s\} \text{ sub-seqüência das tarefas já designadas às máquinas, no sub-problema correspondente.} \\ \overline{\sigma} = \text{conjunto complementar de } \sigma \text{ em relação a } J\text{, ou seja, } J = \sigma \cup \overline{\sigma}. \end{cases}$

- No problema em que a duração total da programação (M) é a medida de desempenho, o procedimento de limitação (*BOUNDING*) consiste em associar a cada sub-problema $P^{[\ell]}_{\sigma}$ um LIMITANTE INFERIOR $LB^{[\ell]}_{\sigma}$ tal que o valor de M para TODAS as (n - ℓ)! soluções completas que são geradas a partir do sub-problema $P^{[\ell]}_{\sigma}$ seja $\geq LB^{[\ell]}_{\sigma}$.

- ALGORITMO DE IGNALL E SCHRAGE para solução ótima do $\boxed{\text{prob. } n/3/F/M}$.

- <u>Cálculo de</u> $LB_{\sigma}^{[\ell]}$ <u>para o sub-problema</u> $P_{\sigma}^{[\ell]}$:

seja: q_j = data de término da operação correspondente à última tarefa da sub-seqüência σ , na máquina M_j .



- Podemos definir os seguintes limitantes inferiores para a duração total da programação (M):

$$\begin{cases} b_1 = q_1 + \sum_{i \in \overline{\sigma}} p_{i1} + \min_{i \in \overline{\sigma}} (p_{i2} + p_{i3}) \\ b_2 = q_2 + \sum_{i \in \overline{\sigma}} p_{i2} + \min_{i \in \overline{\sigma}} p_{i3} \\ b_3 = q_3 + \sum_{i \in \overline{\sigma}} p_{i3} \end{cases}$$

- Uma vez que $M \geq b_j \qquad j = 1, 2, 3$

adota-se $LB_{\sigma}^{[\ell]} = \max\{b_1, b_2, b_3\}$.

- PROPRIEDADE entre sub-problemas utilizada para diminuir o tamanho da árvore (TESTE DE DOMINÂNCIA)

"Suponha que as sub-seqüências σ e σ' contenham as mesmas tarefas (em ordem diferente). Se $q_2 \leq q'_2$ e $q_3 \leq q'_3$, então σ DOMINA σ' e portanto σ' não precisa ser considerada na busca de uma solução ótima".

- ALGORITMO:

- PASSO 1: A partir do problema original $P^{[0]}$, gere os n sub-problemas $P_{\sigma}^{[1]}$ do nível 1 e calcule os respectivos limitantes inferiores $LB_{\sigma}^{[1]}$. Ordene tais sub-problemas segundo a ordem não-decrescente dos $LB_{\sigma}^{[1]}$. Esta ordenação é denominada Lista dos Sub-problemas Ativos (LSA).

- PASSO 2: Considere o primeiro sub-problema $P_{\sigma}^{[\ell]}$ da LSA.
  - Se $\ell = n$, a seqüência σ corresponde à seqüência ótima das tarefas nas máquinas e $LB_{\sigma}^{[\ell]} = \min M$.
  - Se $\ell < n$, remova o sub-problema $P_{\sigma}^{[\ell]}$ da LSA e gere os $(n - \ell)$ sub-problemas

$P^{[\ell+1]}_{\sigma J_i}$ a partir de $P^{[\ell]}_{\sigma}$. Calcule os respectivos $LB^{[\ell+1]}_{\sigma J_i}$.

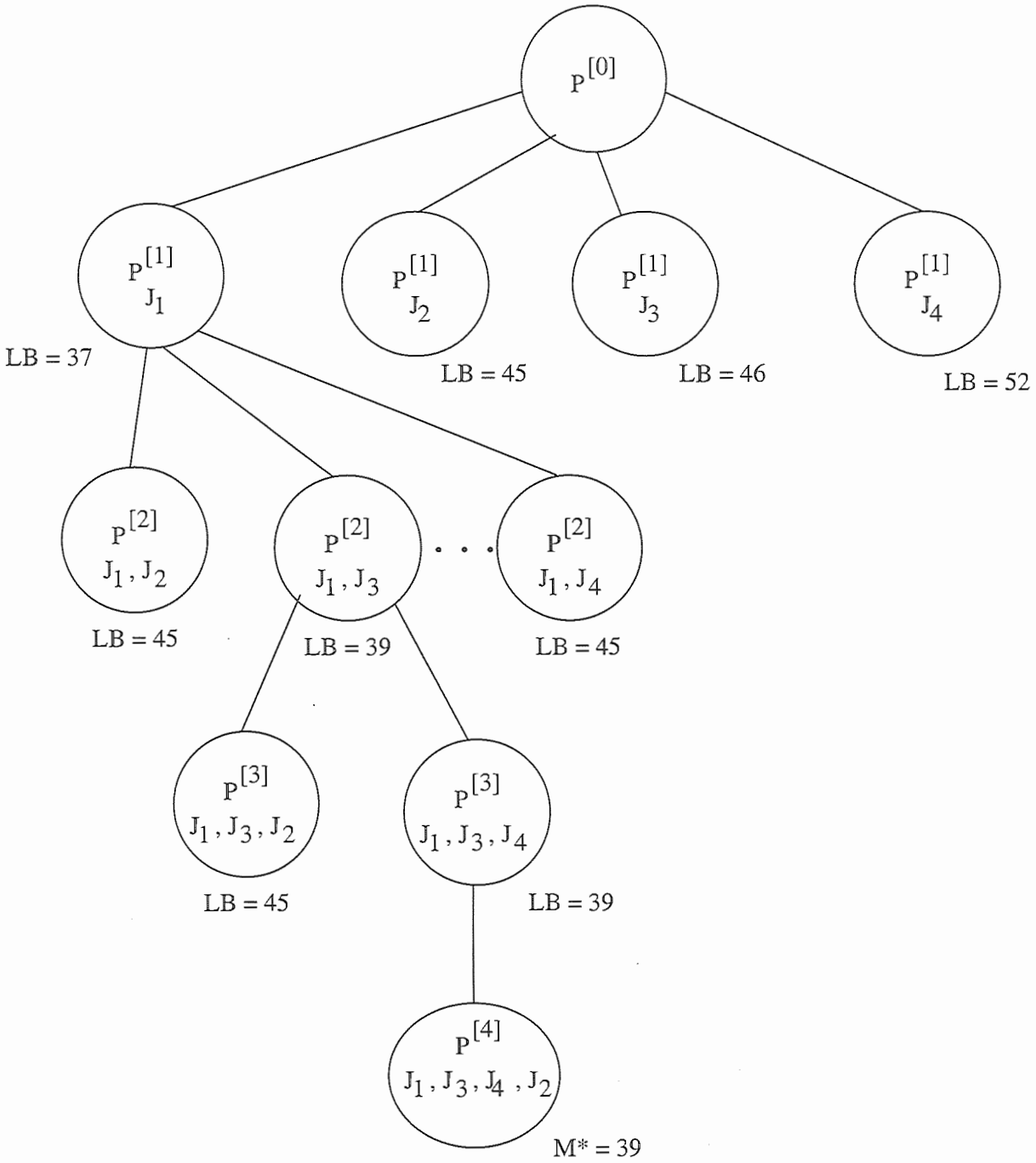
- PASSO 3: Coloque os sub-problemas gerados no passo 2 na LSA, reordenando-a segundo a ordem não-decrescente dos LB_{σ}.
- PASSO 4: Aplique o teste de dominância entre os sub-problemas da LSA, removendo aqueles que forem dominados. Volte ao passo 2.

- Exemplo: 4 / 3 / F / M

J_i	J_1	J_2	J_3	J_4
p_{i1}	3	11	7	10
p_{i2}	4	1	9	12
p_{i3}	10	5	13	2

sub-seqüências σ	(q_1, q_2, q_3)	(b_1, b_2, b_3)	LB_{σ}
J_1	3, 7, 17	37, 31, 37	37
J_2	11, 12, 17	45, 39, 42	45
J_3	7, 16, 29	37, 35, 46	46
J_4	10, 22, 24	37, 41, 52	52
J_1, J_2	14, 15, 22	45, 38, 37	45
J_1, J_3	10, 19, 32	37, 34, 39	39
J_1, J_4	13, 25, 27	37, 40, 45	45
J_1, J_3, J_2	21, 22, 37	45, 36, 39	45
J_1, J_3, J_4	20, 32, 34	37, 38, 39	39
J_1, J_3, J_4, J_2	31, 33, 39	31, 33, 39	39

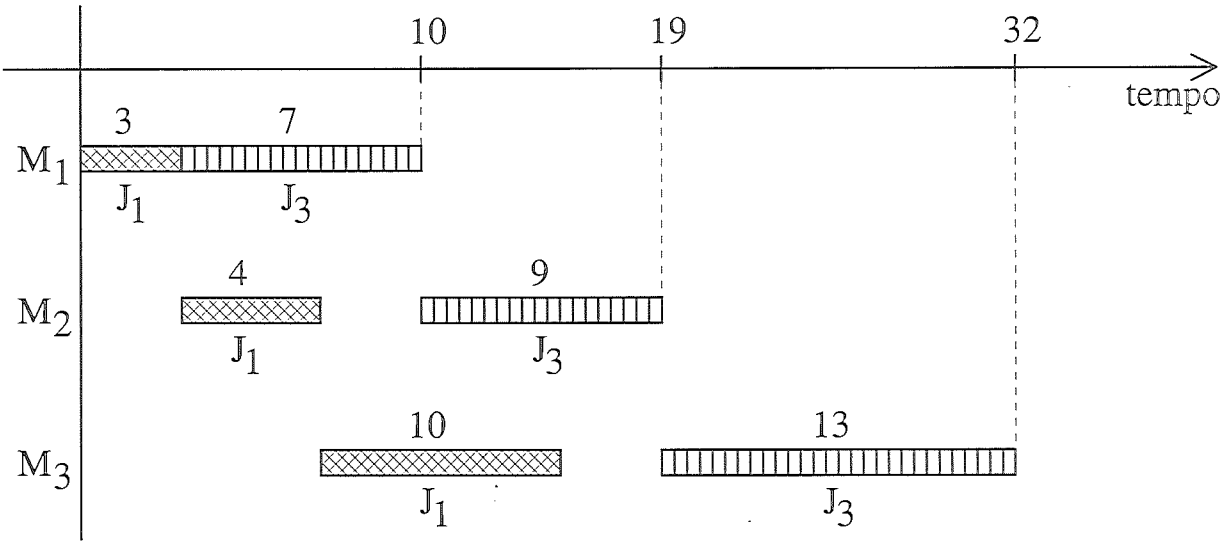
- ÁRVORE de sub-problemas gerados:



- SEQÜÊNCIA ÓTIMA: J_1, J_3, J_4, J_2

- <u>Exemplo de cálculo dos</u> $LB_{\sigma}^{[\ell]}$

sub-problema $P_{J_1, J_3}^{[2]}$ $\quad \sigma = \{J_1, J_3\} \quad \overline{\sigma} = \{J_2, J_4\}$

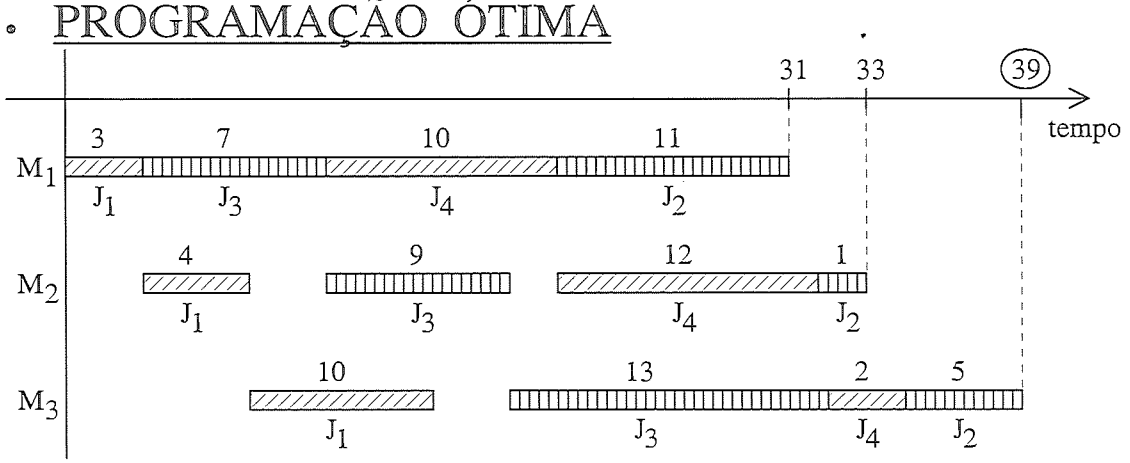


$q_1 = 10 \qquad q_2 = 19 \qquad q_3 = 32$

$$\left\{\begin{array}{l} b_1 = 10 + (11+10) + 6 = 37 \\ b_2 = 19 + (1+12) + 2 = 34 \\ b_3 = 32 + (5+2) = 39 \end{array}\right\rangle \quad LB_{J_1, J_3}^{[2]} = \max\{37, 34, 39\}$$

$$LB_{J_1, J_3}^{[2]} = 39$$

- <u>PROGRAMAÇÃO ÓTIMA</u>



M* = 39

- <u>EXERCÍCIO</u>: Elaborar um procedimento para cálculo dos LIMITANTES INFERIORES de maneira cumulativa, ou seja,

$$LB_{\sigma J_i}^{[\ell+1]} \text{ em função de } LB_{\sigma}^{[\ell]}.$$

4.4 - FORMULAÇÕES EM PROGRAMAÇÃO LINEAR (INTEIRA MISTA)

- PROBLEMA n/m/G/M

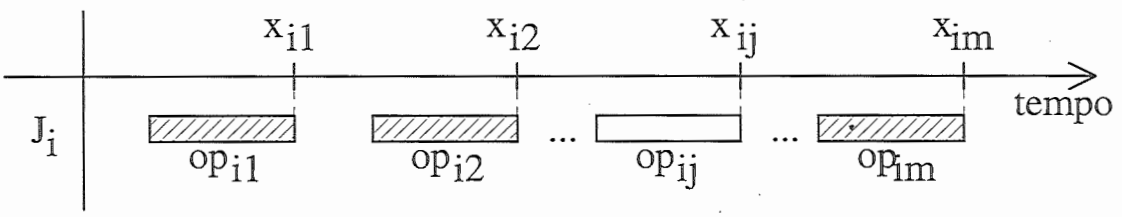
Seja: $g_i = m$ (número de operações de cada tarefa).

x_{ij} = data de término da j-ésima operação op_{ij} da tarefa J_i, a ser executada por uma determinada máquina.

M= duração total da programação.

- CONDIÇÕES A SEREM SATISFEITAS (Restrições)

a) <u>Precedências tecnológicas entre as operações de uma tarefa (estrutura linear)</u>



$$\left\{\begin{array}{l} x_{i1} \geq p_{i1} \\ x_{i2} \geq x_{i1} + p_{i2} \\ \vdots \\ x_{im} \geq x_{i(m-1)} + p_{im} \end{array}\right\}$$

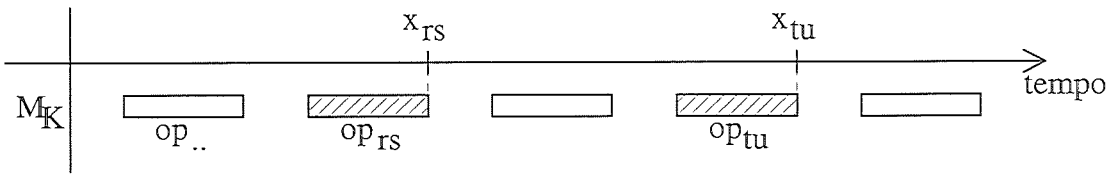
conjunto de m restrições

$$x_{ij} \geq x_{i(j-1)} + p_{ij} \qquad j = 1, 2, \ldots, m$$

$$\text{com } x_{i0} = 0$$

Considerando todas as n tarefas, ter-se-á um conjunto de mn restrições de precedência tecnológica entre as operações.

b) Seqüência entre as operações a serem executadas por uma mesma máquina



. seqüência LINEAR de n operações → n! seqüências alternativas possíveis.

Seja op_{rs} e op_{tu} duas operações a serem executadas pela máquina M_k . Em uma seqüência qualquer, devemos ter uma das duas situações:

$\boxed{op_{rs} \text{ PRECEDE } op_{tu}}$ ou $\boxed{op_{tu} \text{ PRECEDE } op_{rs}}$

$$\left[\begin{array}{l} \text{Na 1a. situação}: x_{tu} \geq x_{rs} + p_{tu} \\ \text{Na 2a. situação}: x_{rs} \geq x_{tu} + p_{rs} \end{array}\right\} \begin{array}{l}\text{RESTRIÇÕES} \\ \text{DISJUNTIVAS}\end{array}$$

Para levar em conta as duas situações possíveis, define-se:

$y_{rtk} = 1$, se a operação op_{rs} preceder a operação op_{tu}, na máquina M_k;

$y_{rtk} = 0$, caso contrário, ou seja: se op_{tu} preceder op_{rs}, na máquina M_k.

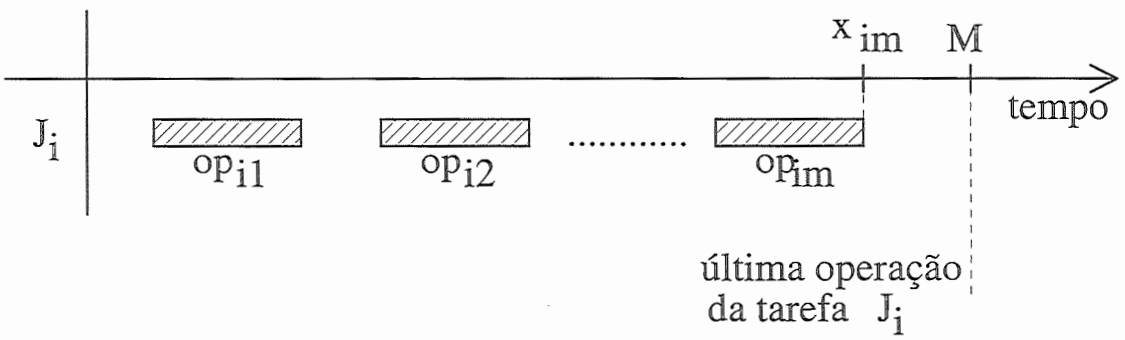
Desta forma, as restrições (não mais disjuntivas) envolvendo as operações op_{rs} e op_{tu} ficam:

$$\begin{cases} x_{tu} - x_{rs} + H(1 - y_{rtK}) \geq p_{tu} \\ x_{rs} - x_{tu} + H \cdot y_{rtK} \geq p_{rs} \end{cases}$$

onde H é um número positivo suficientemente grande, por exemplo, $H = \sum_i \sum_j p_{ij}$.

As duas restrições acima deverão ser consideradas para todas as combinações de 2 operações do conjunto de <u>n</u> operações a serem executadas pela máquina M_k $(k = 1, 2, ..., m)$, o que leva a um conjunto de $mn(n-1)$ restrições.

c) <u>Duração total da programação</u> (M)



- $M \geq x_{im}$ $\quad i = 1, 2, ..., n$ (n restrições)

- <u>FORMULAÇÃO</u>:

Minimizar M

$$\text{Sujeito a} \begin{cases} x_{ij} - x_{i(j-1)} \geq p_{ij} & i = 1, 2, \ldots, n \quad j = 1, 2, \ldots, m \\ x_{tu} - x_{rs} + H(1 - y_{rtK}) \geq p_{tu} & K = 1, 2, \ldots, m \\ & r = 1, 2, \ldots, n-1 \\ x_{rs} - x_{tu} + H \cdot y_{rtK} \geq p_{rs} & t = 2, 3, \ldots, n \\ & t > r \\ M - x_{im} \geq 0 & i = 1, 2, \ldots, n. \end{cases}$$

com $\quad M, x_{..} \geq 0 \qquad H = \sum_i \sum_j p_{ij}$

$\quad y_{rtk} \in \{0, 1\} \qquad x_{io} = 0.$

OBS.:

$$\begin{cases} \text{No. de restrições} = mn + mn(n-1) + n = n(mn+1) \\ \text{No. de variáveis} = 1 + mn + \dfrac{mn(n-1)}{2} = \dfrac{mn(n+1)}{2} + 1 \end{cases}$$

- $n = 4$, $m = 3$ $\longrightarrow$ 52 restrições, 31 variáveis

- $n = 10$, $m = 5$ $\longrightarrow$ 510 restrições, 276 variáveis

<u>Exercício</u>: Formular o problema $n/m/G/\overline{F}$, em termos de Programação Linear.

4.5 - Métodos Heurísticos

PROBLEMA n / m / F / M	(e n / m / PF / M)

PF = programação Flow Shop permutacional.

- <u>Algoritmo de CAMPBELL, DUDEK e SMITH (CDS)</u>

Baseado em duas propriedades:

a) Utiliza a Regra de Johnson de forma heurística.
b) Geralmente obtém diversas programações viáveis, sendo destas escolhida a melhor.

O algoritmo CDS corresponde a um procedimento heurístico de Múltiplos Estágios, onde em cada estágio é aplicada a Regra de Johnson a um novo problema (com m = 2) proveniente do problema original, utilizando tempos de processamento p^*_{i1} e p^*_{i2}, como segue:

- <u>Estágio 1</u>: $p^*_{i1} = p_{i1}$ $\qquad p^*_{i2} = p_{im}$

- <u>Estágio 2</u>: $p^*_{i1} = p_{i1} + p_{i2}$ $\qquad p^*_{i2} = p_{im} + p_{i(m-1)}$

.
.
.

- <u>Estágio s</u>: $p^*_{i1} = \sum_{j=1}^{s} p_{ij}$ $\qquad p^*_{i2} = \sum_{j=1}^{s} p_{i(m-j+1)}$.

Ou seja:

- No estágio 1, aplica-se a Regra de Johnson considerando-se somente a primeira e a última máquina (as demais são ignoradas).

- No estágio 2, aplica-se a Regra de Johnson às somas dos tempos de processamento da primeira com a segunda máquina e da última com a penúltima máquina.

- E assim, sucessivamente, até o estágio (m - 1).

Em cada estágio, a seqüência das tarefas obtida é usada para calcular a correspondente duração total da programação (M), considerando-se todas as m máquinas. A melhor programação é escolhida dentre as (m - 1) programações viáveis obtidas nos (m - 1) estágios.

- EXEMPLO:

	J_i	J_1	J_2	J_3	J_4	J_5
M_1	p_{i1}	6	4	3	9	5
M_2	p_{i2}	8	1	9	5	6
M_3	p_{i3}	2	1	5	8	6

- melhor seqüência obtida: J_3, J_5, J_4, J_1, J_2 .

- Duração total da programação $M = 35$.

PROBLEMAS n/m/G/M e n /m/G/$\overline{F}$

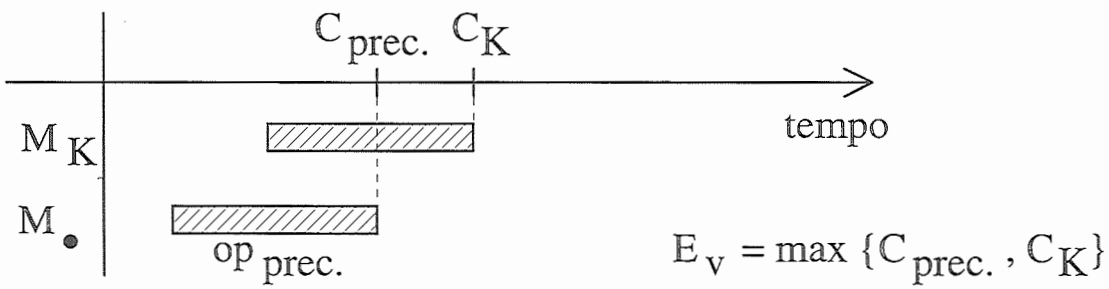
- ALGORITMO GERAL para obtenção de soluções heurísticas (versão modificada do algoritmo proposto por BAKER)

Seja:

PS_t = uma programação parcial na qual estão já programadas t operações;

COC_t = o conjunto de operações CANDIDATAS a serem programadas no estágio t, correspondente a uma dada PS_t ;

E_v = data mais cedo de início da operação $op_v \in COC_t$, a ser executada por uma máquina M_K ;



$C_{prec.}$ = data de término da operação que precede diretamente a op_v (na mesma tarefa);

C_k = data de "liberação" da máquina M_k , a qual deverá executar a op_v .

- ALGORITMO:

- PASSO a: Faça $t = 0$ e inicie com PS_0 como a programação parcial nula. Inicialmente COC_0 inclui todas as operações sem operações precedentes, ou seja, as primeiras operações de todas as tarefas;

- PASSO b: Calcule o valor de E_v das operações op_v pertencentes a COC_t. Determine a MENOR DATA entre os E_v dessas operações. Denote essa data por E^*;

- PASSO c: Identifique as máquinas que executam as operações com $E_v = E^*$;

  - PASSO c.1: Programe todas as operações pertencentes à COC_t com $E_v = E^*$, cujas máquinas têm uma única designação nesse conjunto COC_t;

  - PASSO c.2: Agrupe as operações pertencentes a COC_t com $E_v = E^*$ não programadas no passo anterior, de acordo com a máquina a ser utilizada. Esses grupos constituem os conjuntos de OPERAÇÕES CONFLITANTES. Operações conflitantes são aquelas que têm a mesma data de início E^* e que utilizam uma mesma máquina. Para cada conjunto de operações conflitantes, calcule os ÍNDICES DE

PRIORIDADE de acordo com a Regra de Prioridade (R) adotada.

Para cada conjunto de operações conflitantes, coloque na Programação Parcial PS_t a operação com o MELHOR ÍNDICE DE PRIORIDADE (desempate arbitrário ou de acordo com uma segunda regra de prioridade), programando-a o mais cedo possível;

- PASSO d: Para a nova Programação Parcial criada nos passos (c.1) e (c.2) atualize os dados da seguinte maneira:
  (i) Incremente t de h unidades, onde h é o número de operações programadas nos passos (c.1) e (c.2);
  (ii) Remova de COC_t as operações já programadas nos passos (c.1) e (c.2);
  (iii) Forme o conjunto COC_{t+h} ao adicionar a COC_t as operações imediatamente sucessoras (da mesma tarefa) daquelas que foram programadas nos passos (c.1) e (c.2).

- PASSO e: Retorne ao PASSO b considerando a Programação Parcial PS_{t+h} gerada nos passos (c.1) e (c.2), até que uma Programação Completa tenha sido obtida.

REGRAS DE PRIORIDADE usuais:

* SPT (*Shortest Processing Time*): seleciona a operação com o menor tempo de processamento.

* FCFS (*First Come First Served*): seleciona a operação que entrou no conjunto COC_t antes das demais.

* MWKR (*Most Work Remaining*): seleciona a operação correspondente à tarefa que tem a maior quantidade de trabalho remanescente a ser executado (medida em unidades de tempo).

* MOPNR (*Most Operations Remaining*): seleciona a operação que tem o maior número de operações sucessoras (na mesma tarefa).

* LWKR (*Least Work Remaining*): seleciona a operação correspondente à tarefa que tem a menor quantidade de trabalho remanescente a ser executado (medida em unidades de tempo).

* *RANDOM*: seleciona a operação de maneira aleatória, ou seja, por "sorteio".

OBS: * regras que têm se mostrado relativamente melhores para o problema $n / m / G / \overline{F}$.

* regra que tem se mostrado relativamente melhor para o problema $n / m / G / M$.

• Exemplo: 4 /3 /G /M R = MWKR'
(desempates com SPT)

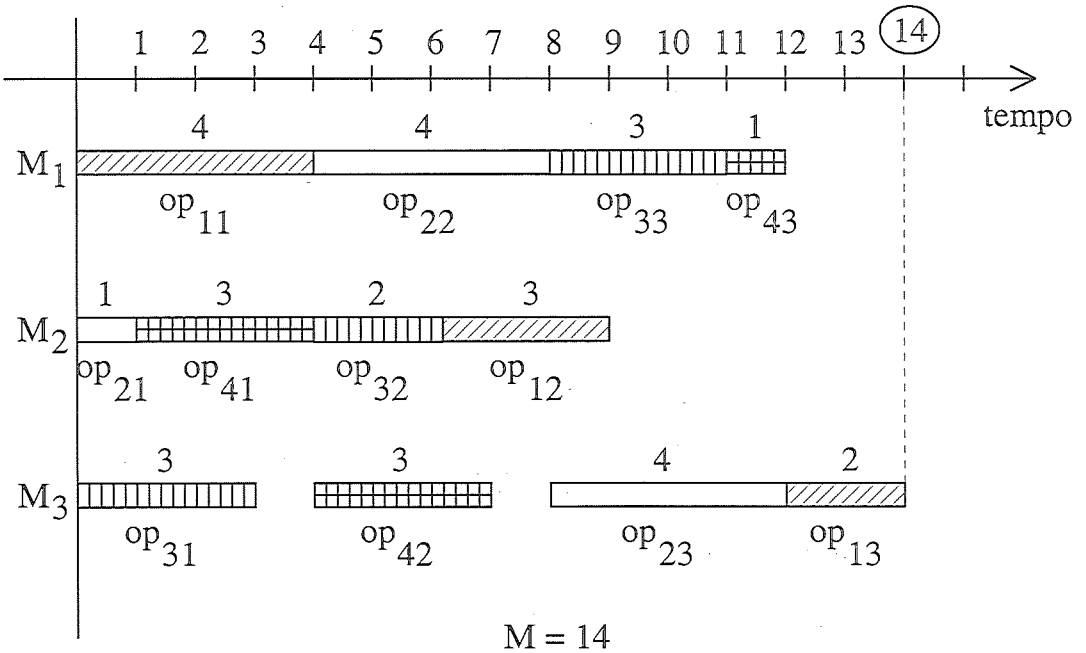
- tempos de processamento

J_i	J_1	J_2	J_3	J_4
p_{i1}	4	1	3	3
p_{i2}	3	4	2	3
p_{i3}	2	4	3	1

- Quadro de precedências (*Routing*)

J_i	J_1	J_2	J_3	J_4
op_{i1}	M_1	M_2	M_3	M_2
op_{i2}	M_2	M_1	M_2	M_3
op_{i3}	M_3	M_3	M_1	M_1

• programação obtida:



M = 14

$\overline{F}$ = 12,25

MWKR' : Inclui o tempo de processamento da respectiva operação.

MWKR : Não inclui o tempo de processamento da respectiva operação.

BIBLIOGRAFIA BÁSICA

BAKER, K.R. Introduction to Sequencing and Scheduling. Wiley, 1974.

CONWAY, R.W., MAXWELL, W.L. and MILLER, L.W. Theory of Scheduling. Addison-Wesley, 1967.

DEMPSTER, M.A.H., LENSTRA, J.K. and RINNOOY KAN, A.H.G. Deterministic and Stochastic Scheduling. D. Reidel, 1982.

FRENCH, S. Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop. Ellis Horwood Ltd., 1982.

MORTON, T.E. and PENTICO, D.W. Heuristic Scheduling Systems with applications to production Systems and Project Management. John Wiley and Sons, Inc., 1993.