

7. PROBLEMAS DE FLOWSHOP E JOB SHOP SOLÚVEIS POR ALGORITMOS DE COMPLEXIDADE POLINOMIAL

Hipótese

$$r_i = 0, \quad i=1,2,\dots,n$$

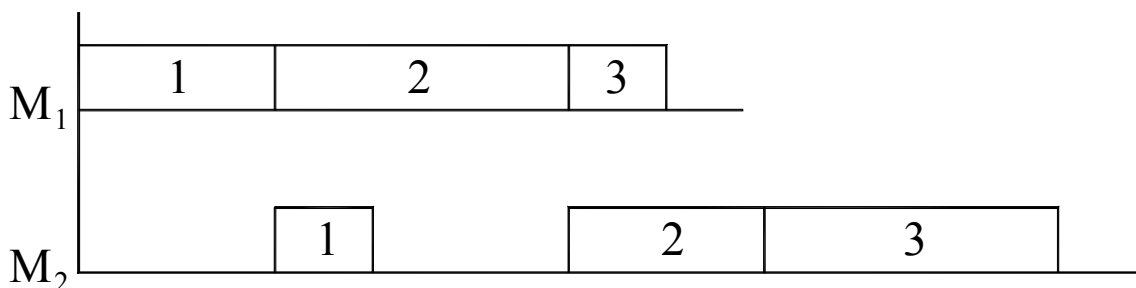
Flowshop

Restrições tecnológicas

Tarefa	Ordem de Processamento			
1	M_1	M_2	...	M_m
2	M_1	M_2	...	M_m
⋮				
n	M_1	M_2	...	M_m

Programa de permutação em flowshop: mesma sequência de processamento em todas as máquinas.

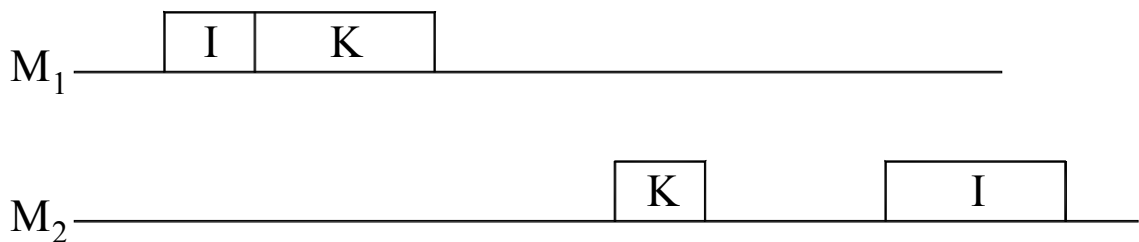
Exemplo



Teorema 7.1 : Para o problema $n/m/F/B$, onde B é uma medida regular de desempenho, é suficiente considerar programas com a mesma sequência de processamento nas duas primeiras máquinas M_1 e M_2 .

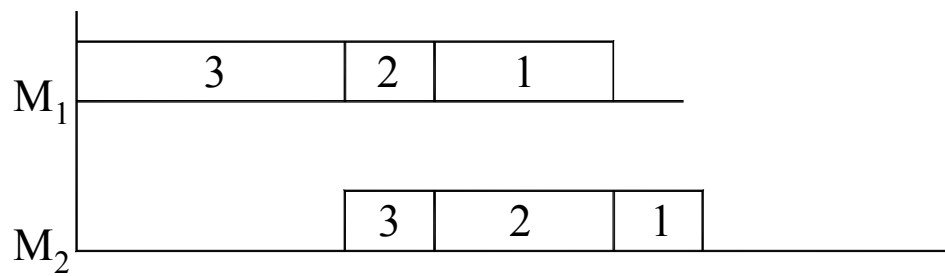
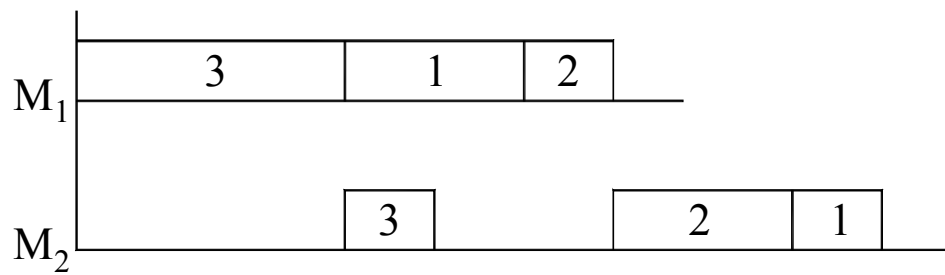
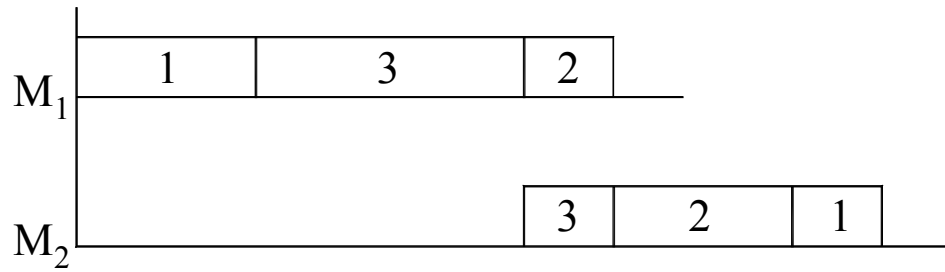
Demonstração

Seja S um programa com sequências distintas em M_1 e M_2 .



Trocando a ordem de I e K em M_1 não aumenta o início de processamento de I e K em M_2 . Repita as trocas em M_1 até obter a mesma sequência de M_2 .

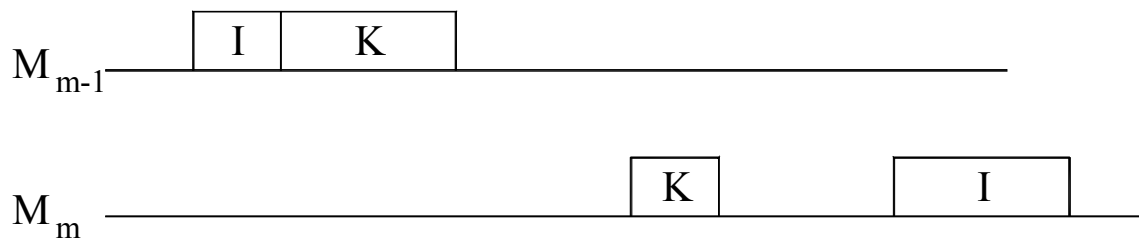
Exemplo



Teorema 7.2 : Para o problema $n/m/F/C_{\max}$ basta considerar programas com a mesma seqüência de processamento nas duas últimas máquinas M_{m-1} e M_m .

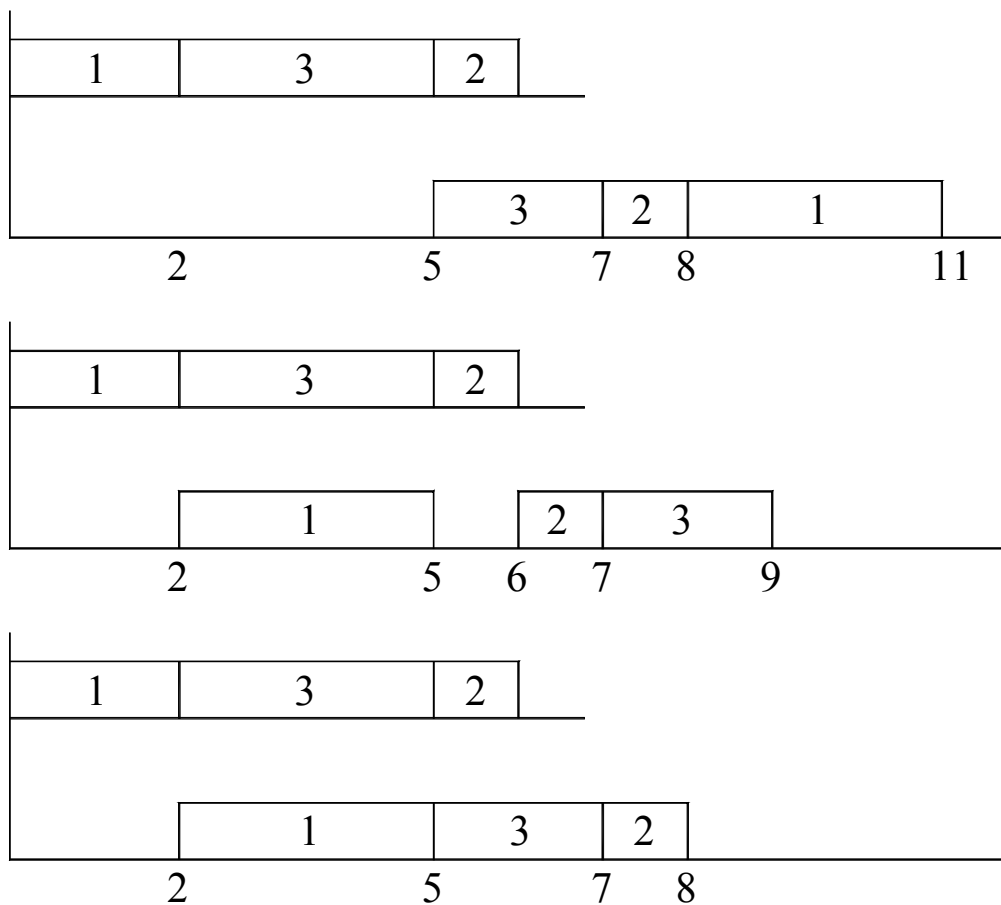
Demonstração

Seja S um programa com seqüências distintas em M_{m-1} e M_m .



A troca da ordem de I e K em M_m não aumenta C_{\max} pois I precede K em M_{m-1} .

EXEMPLO



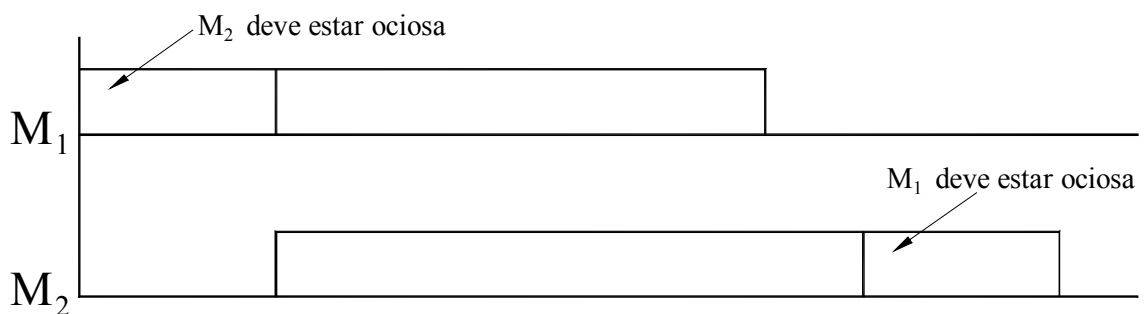
Corolário 7.3 : É suficiente considerar programas de permutação para o problema $n/2/F/B$, onde B é regular.

Corolário 7.4 : É suficiente considerar programas de permutação para o problema $n/3/F/C_{\max}$.

Algoritmo de Johnson para o problema $n/2/F/C_{\max}$

Idéia intuitiva do algoritmo

- 1) Iniciar a sequência com a tarefa que tem o menor tempo de processamento em M_1 ; isto permite que o processamento em M_2 comece o mais cedo possível.
- 2) Terminar a sequência com a tarefa que tem o menor tempo de processamento em M_2 ; isto porque M_1 deve estar ociosa durante esse intervalo de tempo.



Notação

$a_i = p_{i1}$, tempo de processamento da tarefa i em M_1

$b_i = p_{i2}$, tempo de processamento da tarefa i em M_2

Algoritmo de Johnson

Passo 1. $k = 1$; $\ell = n$

Passo 2. Lista corrente das tarefas não sequenciadas = $\{1, 2, \dots, n\}$.

Passo 3. Encontre o mínimo dos tempos a_i, b_i associados às tarefas não sequenciadas.

Passo 4. Se o tempo mínimo é a_i ,

- (i) Coloque a tarefa i na k -ésima posição da sequência
- (ii) Retire a tarefa i da lista corrente de tarefas não sequenciadas
- (iii) $k = k + 1$
- (iv) Vá para o passo 6

Passo 5. Se o tempo mínimo é b_i ,

- (i) Coloque a tarefa i na ℓ -ésima posição da sequência
- (ii) Retire a tarefa i da lista corrente de tarefas não sequenciadas
- (iii) $\ell = \ell - 1$
- (iv) Vá para o passo 6

Passo 6. Se ainda existem tarefas não sequenciadas, vá para o passo 3. Caso contrário, pare.

Nota : Se o tempo mínimo ocorre para mais de uma tarefa no passo 3, escolha a tarefa i arbitrariamente.

Exemplo

Seja o problema $7/2/F/C_{\max}$

Tarefa	1	2	3	4	5	6	7
Tempo de Processamento em M_1	6	2	4	1	7	4	7
Tempo de Processamento em M_2	3	9	3	8	1	5	6

Aplicando o algoritmo

4 - - - - -
4 - - - - 5
4 2 - - - 5
4 2 - - 3 5
4 2 - - 1 3 5
4 2 6 - 1 3 5
4 2 6 7 1 3 5 → sequência ótima

Teorema 7.5 : O algoritmo de Johnson gera um programa ótimo para o problema $n/2/F/C_{\max}$.

Algoritmo de Johnson para o problema $n/2/G/C_{\max}$

Passo 1.

Particione o conjunto de tarefas 1, 2, ..., n em 4 classes:

Classe A : tarefas processadas somente na máquina M_1

Classe B : tarefas processadas somente na máquina M_2

Classe C : tarefas processadas primeiro em M_1 e depois em M_2

Classe D : tarefas processadas primeiro em M_2 e depois em M_1

Passo 2.

- Sequencie as tarefas das classes A e B em qualquer ordem, obtendo as sequências S_A e S_B
- Sequencie as tarefas das classes C e D de acordo com o algoritmo de Johnson para o problema $n/2/F/C_{\max}$

Passo 3.

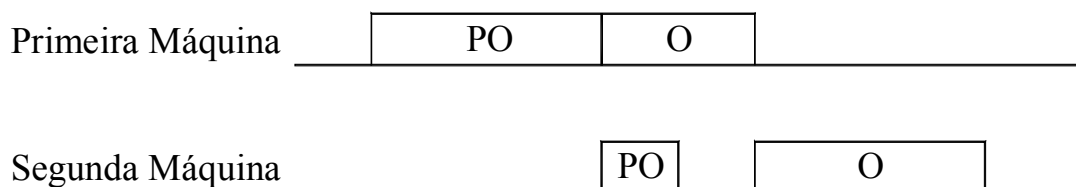
Crie o seguinte programa nas máquinas M_1 e M_2

Máquina	Ordem de Processamento
M_1	(S_C, S_A, S_D)
M_2	(S_D, S_B, S_C)

Teorema 7.6 : O programa do passo 3 é ótimo para o problema $n/2/G/C_{\max}$.

Demonstração

O único tempo ocioso entre operações em M_1 ou M_2 é aquele gerado por uma operação O com início na segunda máquina bloqueada pelo término da operação predecessora PO na primeira máquina (ver figura). Este tempo ocioso é mínimo pelo algoritmo de Johnson para $n/2/F/C_{\max}$.



Exemplo

Seja o problema $9/2/G/C_{\max}$

Ordem e Tempos de Processamento

Tarefa	1	2	3	4	5	6
1ª Máquina	(M ₁ ,8)	(M ₁ ,7)	(M ₁ ,9)	(M ₁ ,4)	(M ₂ ,6)	(M ₂ ,5)
2ª Máquina	(M ₂ ,2)	(M ₂ ,5)	(M ₂ ,8)	(M ₂ ,7)	(M ₁ ,4)	(M ₁ ,3)

Tarefa	7	8	9
1ª Máquina	(M ₁ ,9)	(M ₂ ,1)	(M ₂ ,5)

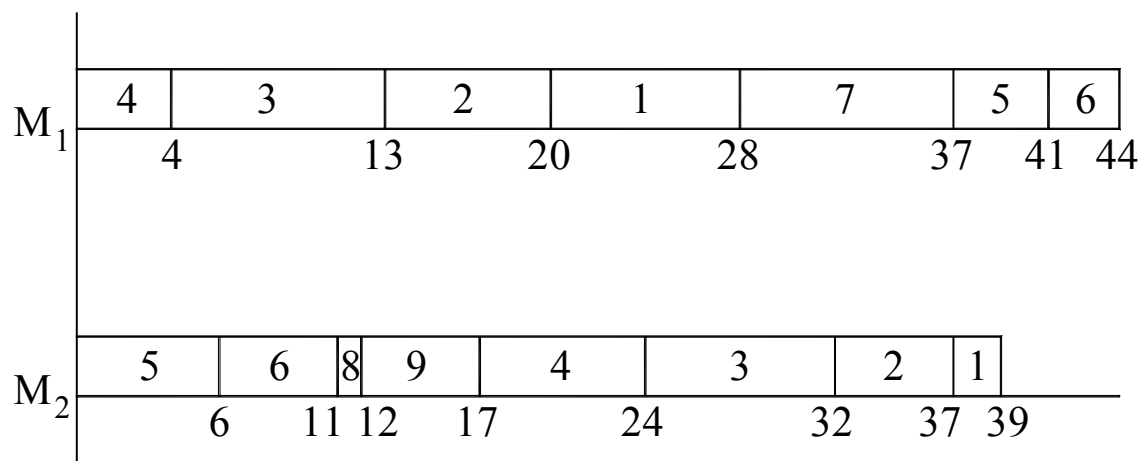
$$S_A = 7 \quad ; \quad S_B = (8, 9)$$

$$S_C = (4, 3, 2, 1) \quad ; \quad S_D = (5, 6)$$

Programa Ótimo

$$M_1(4, 3, 2, 1, 7, 5, 6)$$

$$M_2(5, 6, 8, 9, 4, 3, 2, 1)$$



$$C_{\max} = 44$$

Casos Especiais para o problema $n/3/F/C_{\max}$

O algoritmo de Johnson pode ser estendido para os seguintes casos especiais de $n/3/F/C_{\max}$

$$\text{a) } \min_{i=1}^n \{p_{i1}\} \geq \max_{i=1}^n \{p_{i2}\}$$

$$\text{b) } \min_{i=1}^n \{p_{i3}\} \geq \max_{i=1}^n \{p_{i2}\}$$

Se a) ou b) é verdadeiro, faça para cada tarefa i

$$a_i = p_{i1} + p_{i2} \quad ; \quad b_i = p_{i2} + p_{i3}$$

e obtenha um programa ótimo ao se aplicar o algoritmo de Johnson, onde a_i é o tempo de processamento da tarefa i na primeira máquina e b_i é o tempo de processamento da tarefa i na segunda máquina.

Exemplo

Seja o problema $6/3/F/C_{\max}$

Tarefa	Tempos de Processamento			Tempos de Processamento (a_i, b_i)	
	M_1	M_2	M_3	1ª Máquina	2ª Máquina
1	4	1	3	5	4
2	6	2	9	8	11
3	3	1	2	4	3
4	5	3	7	8	10
5	8	2	6	10	8
6	4	1	1	5	2

$$\min_{i=1}^6 \{p_{i1}\} = 3 \quad ; \quad \max_{i=1}^6 \{p_{i2}\} = 3 \quad ; \quad \min_{i=1}^6 \{p_{i3}\} = 1$$

Portanto, a) é verdadeiro.

Aplicando algoritmo de Johnson para (a_i, b_i) obtém-se o programa ótimo (2, 4, 5, 1, 3, 6).