Uma introdução a problemas de sequenciamento em sistemas de produção em fluxo

Nicolau Santos

Faculdade de Ciências da Universidade do Porto

18 de Janeiro de 2012

Sumário

Introdução

Num sistema de produção em fluxo temos n trabalhos para processar em m máquinas.
O tempo de processo do trabalho i na máquina j é conhecido e designado por $P(i,j)$ $(i = 1, \ldots, n;\ j = 1, \ldots, m)$.
Assumimos que:

- cada máquina processa apenas um trabalho de cada vez;
- a um dado tempo, cada trabalho só pode ser processado por uma máquina;
- não é possível interromper um trabalho e retomar mais tarde;
- a sequência de produção dos trabalhos é única.

Introdução

Primeiras referências surgem em Johnson (1954) onde é apresentado um algoritmo ótimo que resolve o problema em tempo polinomial para $m = 2$, contudo a resolução é fortemente NP-Difícil para $m \geq 3$ (Garey et al, 1976).
Para simular sistemas de produção é acrescentada uma restrição adicional:

- os trabalhos seguem a mesma ordem de processamento em todas as máquinas.

O problema resultante é o chamado Permutation Flowshop Problem, cada solução pode ser representada por uma permutação π e existem $n!$ possíveis soluções para o problema.

Tomando $C(i,j)$ como o tempo em que terminamos o trabalho i na máquina j o fluxo de uma sequência de trabalhos π pelo sistema de produção pode ser representado do seguinte modo:

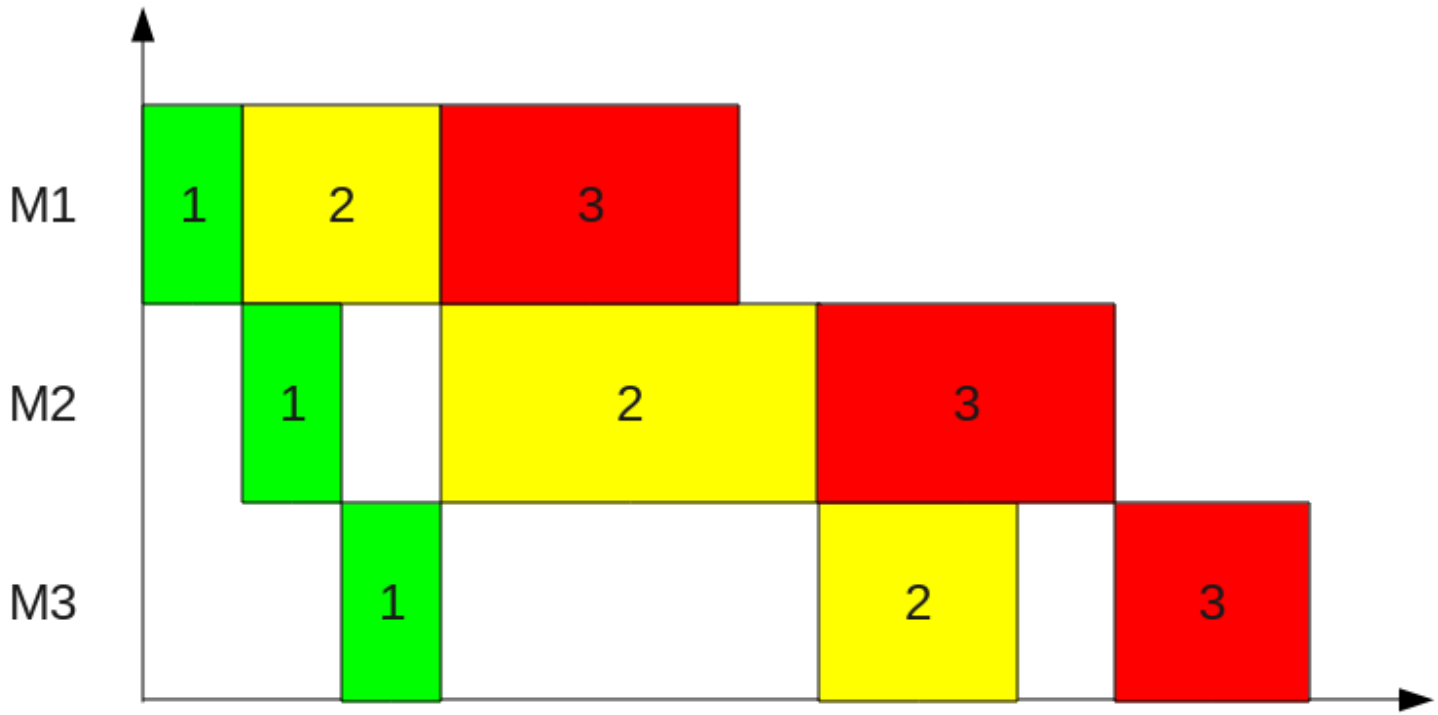
$$C(\pi_1, 1) = P(\pi_1, 1)$$

$$C(\pi_1, j) = C(\pi_1, j-1) + P(\pi_1, j), j = 2 \ldots m$$

$$C(\pi_i, 1) = C(\pi_{i-1}, 1) + P(\pi_i, 1), i = 2 \ldots n$$

$$C(\pi_i, j) = \max\{C(\pi_{i-1}, j), C(\pi_i, j-1)\} + P(\pi_i, j),$$

$$i = 2 \ldots n,\ j = 2 \ldots m$$


M1
M2
M3
1
2
3
1
2
3
1
2
3

Existem vários objectivos analisados na literatura, os mais comuns são:

- minimizar o tempo em que terminamos o último trabalho na última máquina (makespan)

$$C_{max} = C(\pi_n, m)$$

- minimizar o tempo total de fluxo (total flowtime)

$$C_{sum} = \sum_{i=1}^{n} C(\pi_i, m)$$

- minimizar o atraso total (total tardiness)

$$T = \sum_{i=1}^{n} \max\{C(\pi_i, m) - d(\pi_i), 0\}$$

onde $d(i)$ representa a data de entrega do trabalho i.

Literatura (Makespan)

Métodos exactos:

- Programação Inteira: Tseng et al (2004)
- Branch & Bound: Carlier & Rebai (1996)

Heurísticas:

- Regras de despacho: Panwalkar & Iskander (1977)
- NEH: Nawaz, Enscore & Ham (1983)

Meta-heurísticas:

- Tabu Search: Taillard (1990), Nowicki & Smutnicki (1996)
- Iterated Greedy: Ruiz & Stützle (2007)
- Algoritmos populacionais: Nowicki & Smutnicki (2006), Tasgetiren et al (2007)

Literatura (Makespan)

Métodos exactos:

- Programação Inteira: Tseng et al (2004)
- Branch & Bound: Carlier & Rebai (1996)

Heurísticas:

- Regras de despacho: Panwalkar & Iskander (1977)
- NEH: Nawaz, Enscore & Ham (1983)

Meta-heurísticas:

- Tabu Search: Taillard (1990), Nowicki & Smutnicki (1996)
- Iterated Greedy: Ruiz & Stützle (2007)
- Algoritmos populacionais: Nowicki & Smutnicki (2006), Tasgetiren et al (2007)

Literatura (Makespan)

Métodos exactos:

- Programação Inteira: Tseng et al (2004)
- Branch & Bound: Carlier & Rebai (1996)

Heurísticas:

- Regras de despacho: Panwalkar & Iskander (1977)
- NEH: Nawaz, Enscore & Ham (1983)

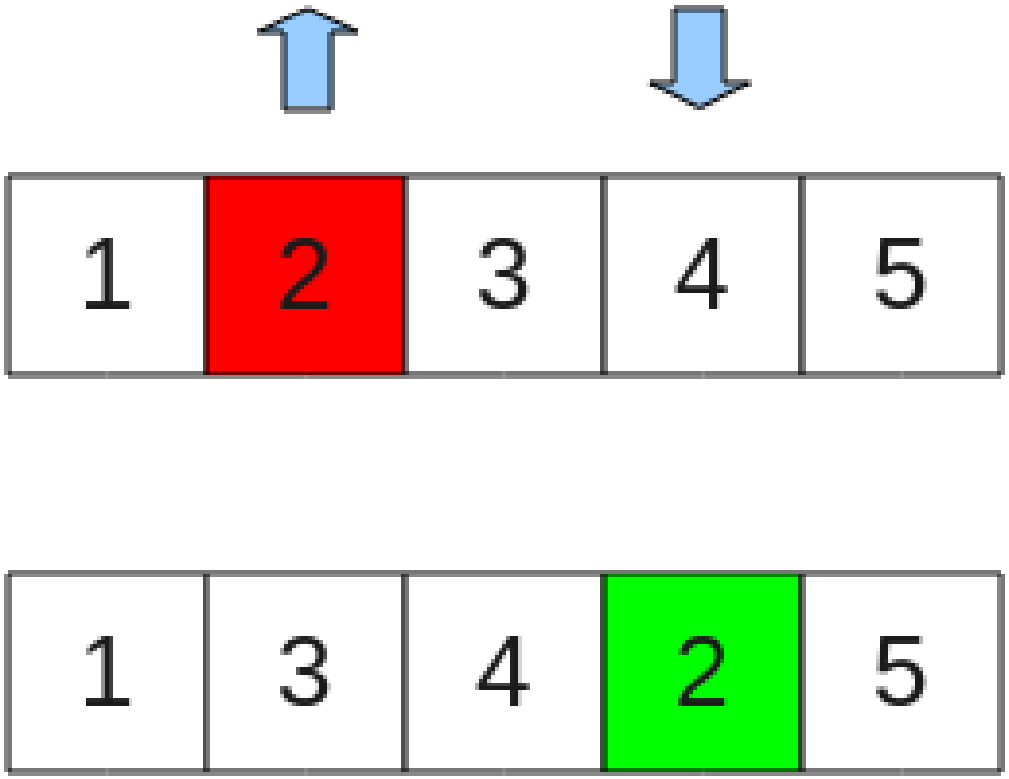
Meta-heurísticas:

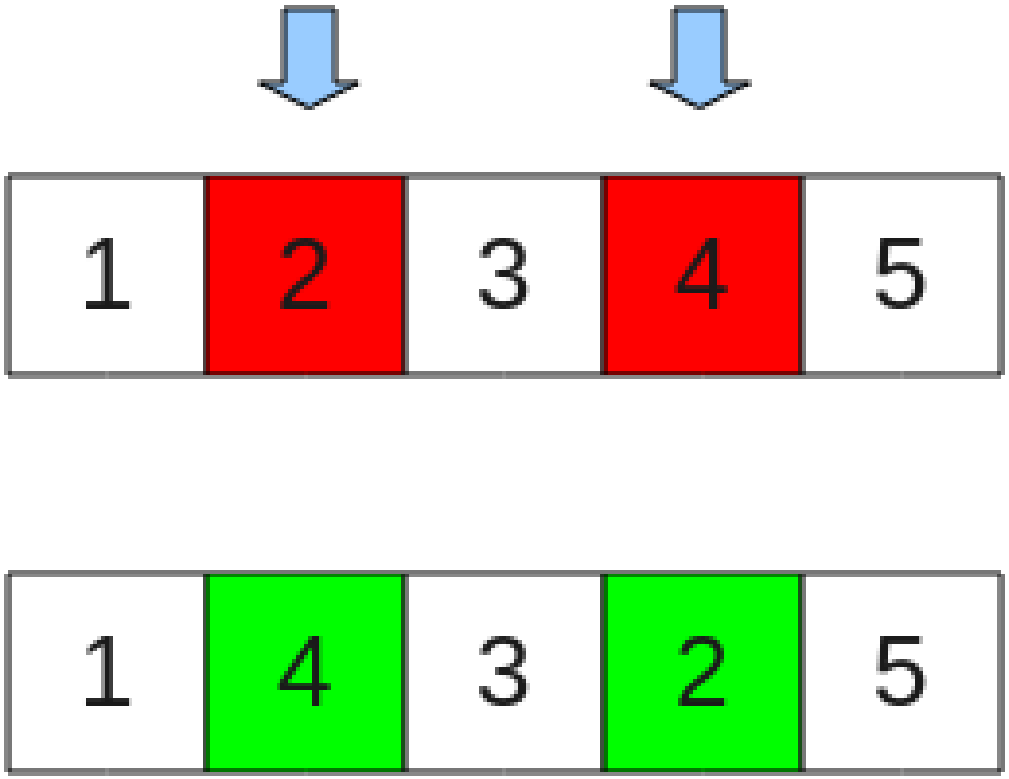
- Tabu Search: Taillard (1990), Nowicki & Smutnicki (1996)
- Iterated Greedy: Ruiz & Stützle (2007)
- Algoritmos populacionais: Nowicki & Smutnicki (2006), Tasgetiren et al (2007)

Pesquisa Local

Processo iterativo em que modificamos a solução actual para obter novas soluções.
Dada uma solução π geramos uma vizinhança $N(\pi)$ através de movimentos.
Os movimentos mais comuns são:

- Inserção: o trabalho da posição i é retirado e inserido na posição j.
- Troca: trocamos o trabalho da posição i com o da posição j.


1
2
3
4
5
1
3
4
2
5


1
2
3
4
5
1
4
3
2
5

Pesquisa Local

procedure local_search(π)
    while π is not locally optimal **do**
        Find $s \in N(\pi)$ with $f(s) < f(\pi)$;
        $\pi \leftarrow s$;
    end do
    return π;
end procedure

Pesquisa Tabu

Método desenvolvido por Glover (1986) para guiar heurísticas pelo espaço de soluções.
A cada iteração é efectuado um novo movimento (mesmo que para um valor pior que o actual).
O movimento é efectuado com base na lista tabu (memória de movimentos efectuados recentemente).

Pesquisa Tabu

procedure tabu_search(π)

$\pi_{best} \leftarrow \pi$

$\pi^* \leftarrow \pi$

while termination criteria not met **do**

Find best $s \in N(\pi^*)$;

$\pi^* \leftarrow s$;

update tabu list

if $\pi^* < \pi_{best}$ **then**

$\pi_{best} \leftarrow \pi^*$;

end if

end do

return π;

end procedure

Pesquisa Tabu

Algoritmo de Taillard:

Vizinhança de Pesquisa:

- são analisadas todas as possíveis inserções de todos os trabalhos numa estratégia first improve

Lista Tabu:

- composta pelos k objetivos mais recentes
- um movimento é tabu se o seu objetivo estiver na lista

Algoritmo de Nowicki & Smutnicky:
Vizinhança de Pesquisa:

- são analisadas apenas as inserções de operações no caminho crítico correspondente ao grafo de operações.

Lista Tabu:

- composta por pares (a,b) correspondentes aos trabalhos e à posição que ocupavam antes do movimento
- um movimento é tabu se o trabalho se encontra lista numa posição inferior à analisada

Critério de aspiração activo (um movimento tabu é realizado se melhorar a melhor solução encontrada).
Contem uma segunda lista de soluções elite, após *ret* iterações sem melhorar o objectivo reinicia a procura partindo de uma destas soluções.

Pesquisa Tabu

Porquê um novo algoritmo de Pesquisa Tabu?

- a adaptação do algoritmo de Taillard (1990) a outros objectivos não é competitiva
- o algoritmo de Nowicki & Smutnicki (1996) não é extensível a outros problemas

Pesquisa Tabu

Sugestões:

- restringir o tamanho da vizinhança a analisar em cada iteração
- estudar o comportamento do algoritmo com listas tabu dinâmicas

Resumo

- Foi efectuada uma breve introdução ao "Permutation Flowshop Problem" e métodos utilizados na sua resolução.
- O problema continua em aberto apesar de mais de 50 anos de estudo.
- Foram definidas linhas de exploração de vizinhança a estudar futuramente.

Obrigado!!!

Referências

- Tseng, F.T. and Stafford, E.F. and Gupta, J.N.D. and others, 2004. An empirical analysis of integer programming formulations for the permutation flowshop.
- Carlier, J. and Rebai, I., 1996. Two branch and bound algorithms for the permutation flow shop problem.
- Panwalkar, SS and Iskander, W., 1977. A survey of scheduling rules.
- Nawaz, M. and Enscore Jr, E.E. and Ham, I., 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem.
- Taillard, E., 1990. Some efficient heuristic methods for the flow shop sequencing problem.
- Nowicki, E. and Smutnicki, C., 1996. A fast tabu search algorithm for the permutation flow-shop problem.

Referências

- Ruiz, R. and Stutzle, T., 2007. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem.
- Glover, F., 1986. Future paths for integer programming and links to artificial intelligence.
- Nowicki, E. and Smutnicki, C., 2006. Some aspects of scatter search in the flow-shop problem.
- Tasgetiren, M.F. and Liang, Y.C. and Sevkli, M. and Gencyilmaz, G., 2007. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem.
- Johnson, S.M., 1954. Optimal two-and three-stage production schedules with setup times included.
- Garey, M.R. and Johnson, D.S. and Sethi, R., 1976. The complexity of flowshop and jobshop scheduling.