FLOW SHOP COM MÁQUINAS PARALELAS GENÉRICAS

João Vitor Moccellin
Escola de Engenharia de São Carlos, Universidade de São Paulo
Av. Trabalhador São-Carlense, 400, CEP 13566-590 São Carlos-SP
E-mail: jvmoccel@sc.usp.br

Marcelo Seido Nagano
Faculdade de Economia, Administração e Contabilidade, Universidade de São Paulo
Av. dos Bandeirantes, 3900, CEP 14040-901 Ribeirão Preto-SP
E-mail: drnagano@usp.br

Resumo
Este artigo trata de um problema de Programação de Operações em Flow Shop, com duas máquinas paralelas genéricas em cada estágio de produção, tendo como objetivo minimizar a Duração Total da Programação. São propostos métodos heurísticos alternativos com o objetivo de avaliar, na etapa de melhoria da solução inicial, a eficácia de um movimento condicional de tarefas utilizando um limitante inferior para o tempo de espera das tarefas entre o final de sua operação no estágio s e o seu início no estágio (s+1). Os desempenhos de tais métodos são avaliados e comparados por meio de uma experimentação computacional.
Palavras-chave: *programação da produção, flow shop híbrido, métodos heurísticos.*

Abstract
This paper deals with a flow shop scheduling problem, where each production stage is composed of two arbitrary parallel machines. The performance measure is the total time to complete the schedule (makespan). Alternative improvement heuristics are proposed in order to evaluate, in the improvement step, the effectiveness of a conditional job-move that is related to a lower bound for the waiting time of any job between the end of its operation on stage s and the beginning on stage (s+1). Results from computational experience are presented.
Keywords*: production scheduling, hybrid flow shop, heuristics.*

1. Introdução

Indústrias têxteis e de vestuário geralmente não podem ser representadas pelos modelos clássicos de programação da produção. Tais sistemas de produção apresentam uma estrutura característica que combina alguns aspectos dos problemas de programação em ambiente *flow shop* com estágios de produção apresentando máquinas paralelas. Esse modelo de combinação é conhecido na literatura como o Problema *Flow Shop* Híbrido (FSH), Gupta (1988), Lee & Vairaktarakis (1994).

Deve-se notar, porém, que tal problema também é do tipo *Flow Shop*, tendo em vista o característico fluxo de produção. A multiplicação das máquinas nos estágios de produção leva a uma maior flexibilidade do processo produtivo.

A programação FSH consiste em programar tarefas tendo como objetivo otimizar alguma medida de desempenho , usualmente relacionada ao fator tempo. Por exemplo, minimizar a duração total da programação (*makespan*), o tempo médio de fluxo, o atraso médio, e o tempo ocioso de máquina. A pesquisa nessa área tem sido beneficiada em função do interesse crescente de diversos pesquisadores. Tal interesse surge devido às várias possibilidades de aplicação do problema e também em função dos recursos computacionais hoje disponíveis que fornecem resultados em tempo hábil, em ambientes cada vez mais automatizados.

Na literatura encontram-se relativamente poucas pesquisas que reportam o desenvolvimento de métodos heurísticos para a programação FSH. Considerando tal fato, a pesquisa relatada neste artigo objetivou explorar a viabilidade de utilização de métodos heurísticos, os quais têm mostrado um bom desempenho na minimização do *makespan* para o problema tradicional em ambiente *flow shop*.

2. Uma revisão sobre Programação FSH

O problema de programação *Flow Shop* Híbrido (FSH) consiste de um *flowshop* de múltiplos estágios de produção, onde cada estágio $s \varepsilon \{1, ..., K\}$, $K \geq 2$, é composto de M_s máquinas paralelas, as quais podem processar uma única tarefa de cada vez. Um *flow shop* multi-estágios é um FSH se pelo menos em um dos estágios existirem duas ou mais máquinas.

O problema consiste em programar um conjunto de tarefas $I = \{1, ..., n\}$, onde cada tarefa tem uma única operação em cada estágio. As operações de uma tarefa devem ser efetuadas seqüencialmente, passando por todos os estágios. Além disso, as operações uma vez iniciadas não devem ser interrompidas e também não podem ser subdivididas em operações simultâneas e independentes. Cada tarefa i tem um tempo de processamento conhecido p_{is} no estágio s, $i \varepsilon I$ e $s \varepsilon \{1, ..., K\}$.

Como já salientado, o ambiente FSH pode ser visto como uma combinação do *flowshop* clássico ($M_s = 1$, $K > 1$) com o problema de máquinas paralelas ($M \geq 2$, $K= 1$), os quais têm sido extensivamente estudados. Estudos gerais sobre Programação de Operações em Máquinas podem ser encontrados em Baker (1974) e Pinedo (1995).

A seguir, são apresentados alguns trabalhos que tratam do problema FSH com 2 estágios e do caso geral, com 3 ou mais estágios.

2.1 O problema FSH com 2 estágios

Gupta (1988) estudou o problema de minimizar o *makespan* no caso de $M_1 \geq 2$ e $M_2 = 1$. Ele constatou que o problema é *NP-hard* quando $\max (M_1, M_2) > 1$. Este resultado é importante pelo fato de mostrar que qualquer problema FSH com K estágios, objetivando a minimização do *makespan*, é *NP-hard*.

Sriskandarajah & Sethi (1989) estudaram os desempenhos de algumas heurísticas, quanto ao *makespan*, para problemas com $M_1 \geq 2$ e $M_2 = 1$, e com $M_1 \geq 2$ e $M_2 \geq 2$. As soluções são obtidas baseadas na "regra de Johnson" (1954). Gupta & Tunc (1991) estudaram o problema com $M_1 = 1$ e $M_2 \geq 2$. Eles mostraram que tal problema é o inverso daquele apresentado em Gupta (1988) e que também é *NP-hard*.

Guinet et al. (1992), propuseram inicialmente uma formulação em Programação Inteira Mista para o problema geral com $K \geq 2$, e posteriormente baseado na "regra de Johnson" foi proposto um método heurístico para o problema de minimização do *makespan* em um FSH com 2 estágios. Eles compararam tal método com as regras de programação conhecidas por SPT (*Shortest Processing Time*) e LPT (*Longest Processing Time*), concluindo que esta última fornece boas soluções para o problema. Lee & Vairaktarakis (1994) estudaram o problema de minimização do *makespan* em um FSH com 2 estágios quando é permitido, no primeiro estágio, a subdivisão de operações (*splitting*).

Elmaghraby & Soewandi (1998a) e (1998b) estudaram os problemas de minimização do *makespan* em FSHs de 2 estágios com máquinas idênticas em um caso, e máquinas "uniformes" (proporcionais) no outro. Eles concluíram que, até aquele momento, o melhor método para solução do problema era um que utilizava uma variação do algoritmo de Johnson.

2.2 O problema FSH com múltiplos estágios

Hunsucker et al. (1989) efetuaram uma simulação computacional com o propósito de avaliar o desempenho de diversas regras de prioridade para os casos em que a função-objetivo refere-se ao Tempo Médio de Fluxo e também para o *makespan*. Hunsucker & Shah (1994) realizaram uma análise comparativa do desempenho de regras de prioridade em problemas multi-estágios FSH com restrições quanto ao número máximo de tarefas a serem programadas.

Com o propósito de obter a solução ótima para o problema de minimização do *makespan* em um ambiente multi-estágios FSH, Brah & Hunsucker (1991) apresentaram um algoritmo *branch-and-bound*, adaptando o "Método de Enumeração" proposto por Bratley et al. (1975) para a programação do problema clássico de máquinas paralelas. O procedimento de ramificação consiste em enumerar todas as seqüências possíveis das tarefas nas máquinas de todos os estágios de produção. Como esperado, o método proposto teve um interesse somente teórico, uma vez que o tempo de CPU mesmo para problemas de pequeno porte alcança valores excessivos.

Em Vignier et al. (1995) pode-se encontrar, para aquele momento, uma descrição do estado-da-arte para o problema geral de programação FSH.

Portmann et al. (1996) utilizaram um Algoritmo Genético para minimizar o *makespan* no problema multi-estágios FSH. Nesse trabalho, o algoritmo genético foi usado para calcular Limitantes Superiores para os nós de um algoritmo *branch-and-bound* usando o esquema de ramificação de Brah & Hunsucker (1991). Os autores testaram o algoritmo para problemas com 5, 10 e 15 tarefas em *flowshops* com 2, 3 e 5 estágios. A maioria dos exemplos testados apresentava "gargalos", ou seja: em alguns estágios o número de máquinas era inferior ao dos demais estágios, mantendo para todos os estágios de produção a mesma distribuição dos tempos de processamento das operações. Todos os testes foram interrompidos após 2 horas de tempo computacional em um microcomputador 486 / 33MHz. Como esperado, somente os problemas de menor porte (5 tarefas) foram resolvidos fornecendo a solução ótima.

Guinet & Solomon (1996) foram os primeiros a adaptar métodos heurísticos do *Flow Shop* tradicional para programar um *Flow Shop* Híbrido. Assim como no trabalho de Portmann et al. (1996), a maioria dos problemas testados (90%) apresentava estágios de produção com gargalos.

Nagano & Moccellin (2000) e Moccellin & Nagano (2001) propuseram métodos heurísticos para o problema FSH com múltiplos estágios e duas máquinas em cada um deles, sendo que no primeiro trabalho as máquinas são idênticas e no segundo elas são máquinas proporcionais (uniformes). Neste artigo, é tratado um problema similar onde em cada estágio de produção existem duas máquinas paralelas genéricas.

3. Métodos heurísticos para o problema FSH com máquinas genéricas

O problema de programação *Flow Shop* Híbrido, tratado neste artigo, apresenta as seguintes hipóteses:

- O ambiente FSH é suposto funcionar como 2 *flowshops* tradicionais "paralelos";
- Existem $K \geq 3$ estágios de produção com 2 máquinas paralelas em cada um deles. Tais máquinas podem ser, em um estágio qualquer, idênticas, uniformes (proporcionais), ou não-relacionadas.

No caso de máquinas uniformes, existentes em um estágio *s*, é conhecido um multiplicador C_s para os tempos de processamento das operações no segundo *flowshop* (máquinas mais lentas) em relação aos tempos de processamento no primeiro, com máquinas mais rápidas.

Máquinas não-relacionadas, em um mesmo estágio de produção, podem executar as mesmas operações das diversas tarefas, porém com tempos de processamento desiguais e independentes;

- As operações uma vez iniciadas não devem ser interrompidas e também não podem ser subdivididas em operações simultâneas e independentes;
- A medida de desempenho é o *makespan*.

Os dois métodos heurísticos obtêm soluções iniciais para cada um dos *flowshops* paralelos utilizando o método heurístico construtivo N&M (Nagano & Moccellin, 2002), e a seguir procuram melhorar tais soluções iniciais por meio de busca na vizinhança das soluções correntes. Para determinar o conjunto de tarefas para cada um dos *flowshops* paralelos, resolve-se um problema de partição de conjunto, de forma que os sub-conjuntos de tarefas designados a

cada *flowshop* tenham suas respectivas somas dos tempos de processamento das tarefas o mais próximo possível, entre si. A estrutura de vizinhança é a de Inserção, onde uma seqüência vizinha é obtida da seqüência corrente de tarefas retirando-se uma tarefa de sua posição e inserindo-a em outra posição (a escolha da tarefa e da posição a ser ocupada é feita aleatoriamente). A principal diferença entre os métodos refere-se ao procedimento de busca de novas soluções ser ou não restrito. O primeiro método (denominado **Algoritmo 1**) gera aleatoriamente a tarefa e a sua nova posição, inserindo-a nessa nova posição sem quaisquer restrições. No segundo, porém, (denominado **Algoritmo 2**), a tarefa e sua nova posição também são geradas aleatoriamente, porém o **movimento de inserção** somente é efetuado se uma condição pré-estabelecida for satisfeita.

Condição de Parada

Os métodos heurísticos encerram a busca por melhores soluções quando o tempo de computação (CPU) atinge um determinado limite superior, o qual foi estabelecido por experimentação prévia tendo como objetivo resolver o problema dentro de um esforço computacional não excessivo. Essa Condição de Parada, em função do número de tarefas a serem programadas, é apresentada na Figura 1.

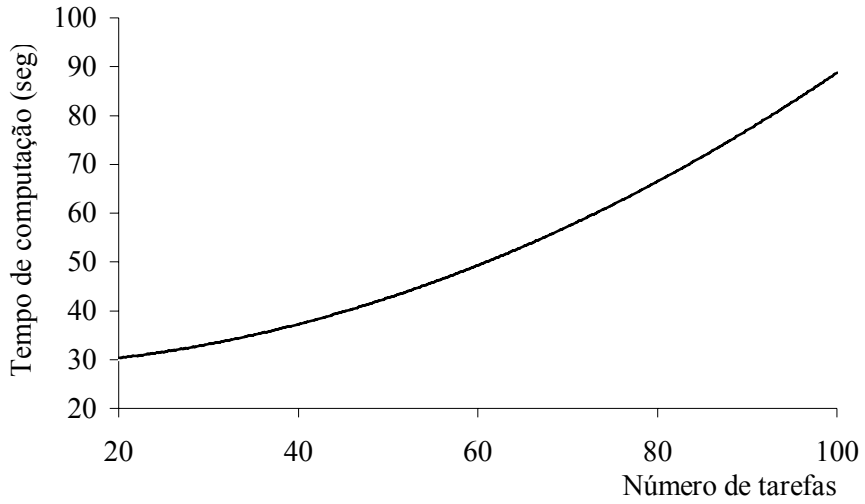

Figura 1 - Condição de parada do procedimento de melhoria da solução inicial.

Algoritmo 1

Passo 1 - Determine os sub-conjuntos de tarefas I_1 e I_2, onde $I_1 \cup I_2 = I = \{1, ..., n\}$,

de forma a minimizar $\left| \sum_{i \in I_1} \sum_{s=1}^{K} p_{is} - \sum_{i \in I_2} \sum_{s=1}^{K} p_{is} \right|$.

Passo 2 - Faça FS_1 ser o 1º *Flowshop* relacionado ao sub-conjunto I_1 com n_1 tarefas, e FS_2 o 2º *Flowshop* referente ao sub-conjunto I_2 com n_2 tarefas. Resolva os *Flowshops* utilizando o método heurístico construtivo N&M. Seja M_h o melhor *makespan* referente ao *Flowshop* FS_h, para h = 1, 2. O *makespan* da solução inicial do problema FSH é:
$M = \max (M_1, M_2)$.

Passo 3 - Enquanto o tempo de CPU $\leq$ **condição de parada**:

- Gere números inteiros aleatórios (*u*, *j*), ambos pertencentes ao conjunto { 1, 2, ..., n_1}. Coloque a tarefa *u* na *j-ésima* posição da melhor seqüência atual de tarefas referente a FS_1.
- Faça o mesmo procedimento para FS_2.
- Guarde a melhor solução atual para o problema FSH.

Algoritmo 2
Passos 1 e 2 - São os mesmos do **algoritmo 1**.

Passo 3 - Enquanto o tempo de CPU $\leq$ **condição de parada**:
- Gere números inteiros aleatórios (u, j), ambos pertencentes ao conjunto $\{ 1, 2, ..., n_1\}$.
- Coloque a tarefa u na *j-ésima* posição da melhor seqüência atual de tarefas referente a FS_1, precedendo a tarefa v na posição $(j + 1)$, **se e somente se** a seguinte desigualdade for verdadeira:

$$2\sum_{s=1}^{K-1} LBY_{uv}^{s} < \text{MAX}\sum LBY_u + \text{MAX}\sum LBY_v$$

onde

$$\text{MAX}\sum LBY_i = \max_{\substack{g=1 \\ g\neq i}}^{n_1}\left\{\sum_{s=1}^{K-1} LBY_{g\,\text{i}}^{\,\text{s}}\right\}$$

$LBY_{g\,\text{i}}^{\text{s}}$ = um **limitante inferior** para o tempo de espera da tarefa i entre o final de sua operação no estágio s e o início no estágio $(s+1)$, quando a tarefa g precede imediatamente a tarefa i (Nagano & Moccellin, 2002).

- Faça o mesmo procedimento para FS_2.
- Guarde a melhor solução atual para o problema FSH.

4. Experimentação computacional

Os métodos heurísticos foram avaliados utilizando-se uma amostra de 750 problemas com o número de estágios $K \in \{4, 7, 10\}$ e o número de tarefas $n \in \{20, 40, 60, 80, 100\}$. Para cada combinação ($K$ x n) foram resolvidos 50 problemas. Os tipos de estágios de produção foram gerados aleatoriamente, com 25% de probabilidade de serem constituídos de máquinas idênticas, 50% para os estágios com máquinas uniformes e probabilidade 25% para o caso de máquinas não-relacionadas. Para os estágios com máquinas uniformes os multiplicadores C_s assumiram valores equiprováveis no conjunto de números inteiros {2, 3}. Os tempos de processamento das operações foram números inteiros gerados aleatoriamente a partir de uma distribuição uniforme no intervalo [1, 10]. Nos testes computacionais, os métodos heurísticos foram codificados em linguagem *Delphi* e processados em um microcomputador *Pentium III 550 Mhz*.

Os resultados principais da experimentação computacional são ilustrados nas Figuras 2, 3 e 4.

A Figura 2 mostra a **Porcentagem de Sucesso** de cada método. Tal porcentagem é definida pelo quociente entre o número total de problemas para os quais o método obteve o melhor *makespan* e o número total de problemas resolvidos. Obviamente, quando os dois métodos obtêm o melhor *makespan* para o mesmo problema, suas Porcentagens de Sucesso são simultaneamente melhoradas.

A Figura 3 apresenta a **Melhoria Relativa**, ou seja, a porcentagem de redução do *makespan* em comparação com a solução inicial. A Melhoria Relativa (MR) é dada por:

$MR = (M_I - M_F) / M_I$, onde

M_I = *makespan* da solução inicial, e
M_F = *makespan* da melhor solução que foi encontrada pelo método heurístico.

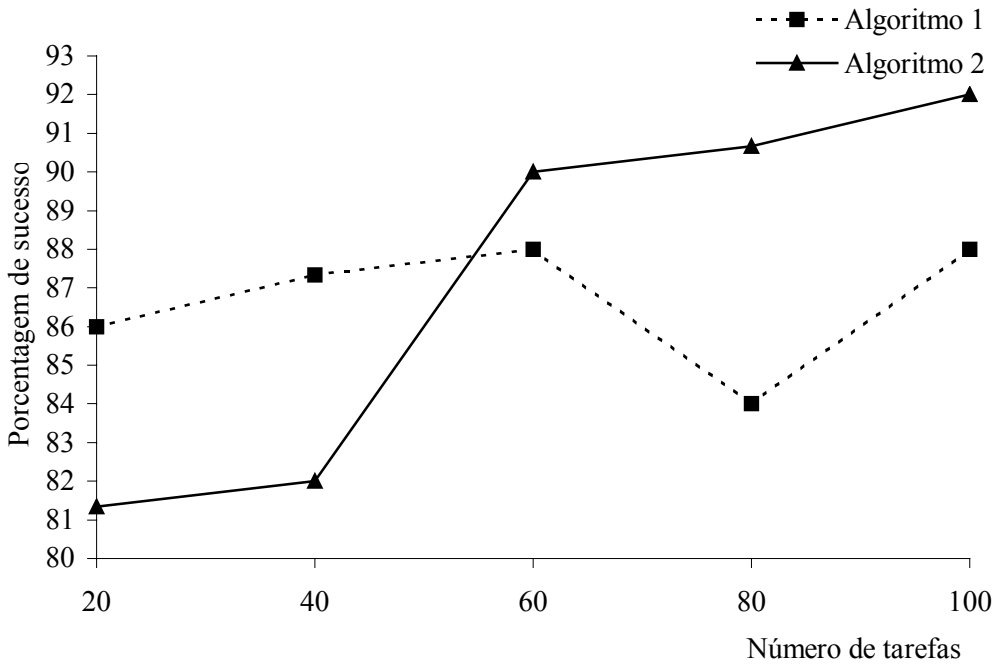

Figura 2 - Porcentagem média de sucesso, de acordo com o número de tarefas.

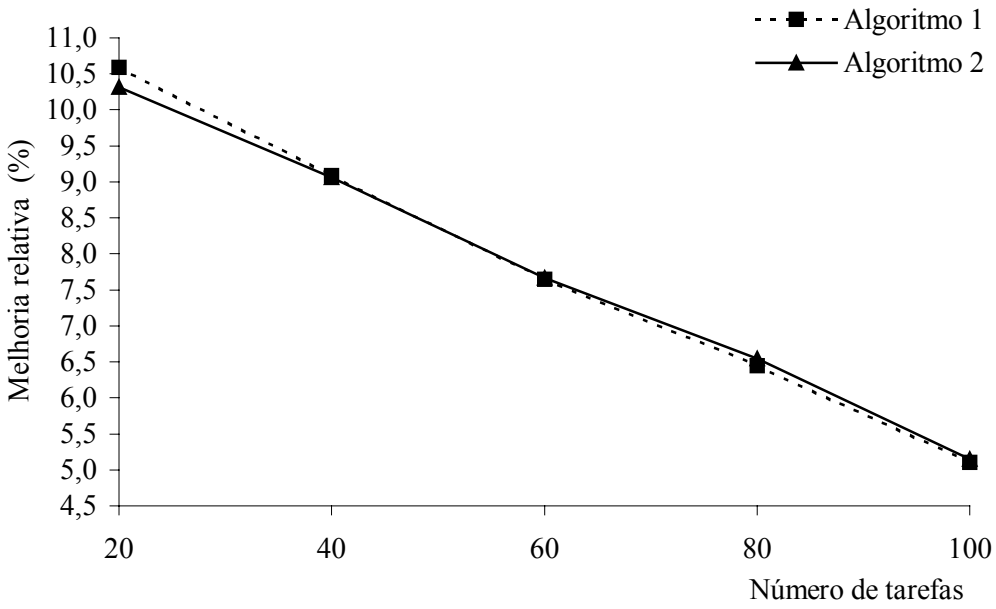

Figura 3 - Melhoria relativa média, de acordo com o número de tarefas.

As Porcentagens de Sucesso da Figura 2 mostram que o Algoritmo 1 tem um melhor desempenho para problemas de pequeno e médio porte, quanto ao número de tarefas, enquanto que o Algoritmo 2 é mais eficaz para problemas maiores.

A Figura 3 mostra que as Melhorias Relativas, em média, são praticamente iguais para os dois métodos heurísticos.

Tais resultados experimentais sugerem a utilização de um método "adaptativo" cujo parâmetro básico seria o porte do problema em termos do número de tarefas. O valor de transição N_t do número de tarefas seria $40 < N_t < 60$, o qual pode ser determinado por meio de uma experimentação computacional com pequenos intervalos de variação do número de tarefas, entre 40 e 60. Os resultados da Figura 3, referentes à Melhoria Relativa sobre a solução inicial, indicam para a mesma direção quanto ao método adaptativo. O valor de transição N_t foi determinado conforme mostra a Figura 4.

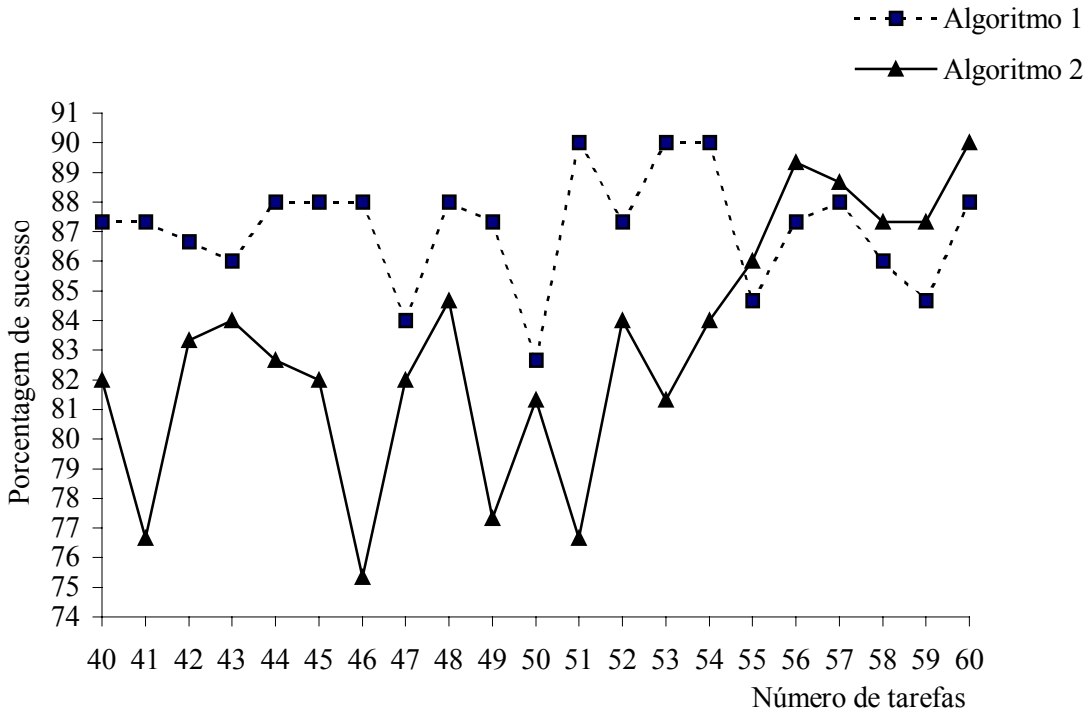

Figura 4 - Porcentagem de Sucesso para $40 \leq n \leq 60$

5. Considerações finais

Os resultados experimentais mostram que o algoritmo 2 tem um melhor desempenho para problemas com o número de tarefas entre 55 e 100, considerando tanto a porcentagem de sucesso quanto a melhoria relativa. Entretanto, para problemas de menor porte o algoritmo 2 é superado pelo algoritmo 1.

Estes são resultados parciais, uma vez que estão sendo testadas outras buscas de inserção na vizinhança das soluções correntes para outros tipos de movimento condicional utilizando o limitante inferior $LBY^{s}_{g\,i}$. De qualquer forma, Busca Restrita em Vizinhança (ou seja, não-aleatória) apresenta-se como uma idéia promissora para a minimização do *makespan* na programação *Flow Shop* com máquinas paralelas, tendo em vista os resultados relatados em trabalhos anteriores e aqueles apresentados neste artigo.

Bibliografia

BAKER K.R., 1974. *Introduction to Sequencing and Scheduling*, John Wiley & Sons Inc, New York.

BRAH S. A. and J.L. HUNSUCKER, 1991. Branch and bound algorithm for the flow shop with multiprocessors, *European Journal of Operational Research*, vol. 51, 88-89.

BRATLEY P., M. FLORIAN and P. ROBILLARD, 1975. Scheduling with earliest start and due date constraints on multiple machines, *Naval Research Logistics Quarterly*, vol. 22, n. 1, 165-173.

ELMAGHRABY S.E. and H. SOEWANDI, 1998a. Sequencing Jobs on Two-Stage Hybrid Flowshop with Uniform Machines to Minimize makespan, *Technical Report*, Department of Industrial Engineering, North Carolina State University, Raleigh, NC, USA.

ELMAGHRABY S.E. and H. SOEWANDI, 1998b. Sequencing Jobs on Two-Stage Hybrid Flowshop with Identical Machines to Minimize makespan, *Technical Report*, Department of Industrial Engineering, North Carolina State University, Raleigh, NC, USA.

GUINET A., F. ECHALIER and A. DUSSAUCHOY, 1992. Scheduling jobs on parallel machines: a survey, *EURO 12*, *TIMS XXXI*, Helsinki, Finland.

GUINET A. and M. SOLOMON, 1996. Scheduling hybrid flow shops to minimize maximum tardiness or maximum completion time, *Int. J. Prod. Res*., vol. 34, 1643-1654.

GUPTA J.N.D., 1988. Two-stage, hybrid flowshop scheduling problem, *J. Op. Res. Soc*., vol. 39, 359-364.

GUPTA J.N.D. and E.A. TUNC, 1991. Schedules for a two-stage hybrid flowshop scheduling with parallel machines at the second stage, *Int. J. Prod. Res*., vol. 29, 1489-1502.

HUNSUCKER J.L., S.A. BRAH and D.L. SANTOS, 1989. Simulation study of a flow shop with multiple processors scheduling, *TIMS/ORSA joint National Meeting in New York City*, 16-18.

HUNSUCKER J.L. and J.R. SHAH, 1994. Comparative performance analysis of priority rules in a constrained flow shop with multiple processor environment, *European Journal of Operational Research*, vol. 72, 102-104.

JOHNSON S.M., 1954. Optimal two and three stage production schedules with setup time included, *Naval Res. Logist. Q*., vol. 1, 61-68.

LEE C.Y. and G.L. VAIRAKTARAKIS, 1994. Minimizing makespan in hybrid Flow Shops, *Operational Research Letter*, vol. 16, 149-158.

MOCCELLIN J.V. and M.S. NAGANO, 2001. Flow Shop Heuristic Scheduling with Parallel Uniform Processors, *INFORMS Annual Meeting*, Miami Beach-FL, USA, Bulletin p.118 (http://www.informs.org/Conf/Miami2001/TALKS/TD25.html).

NAGANO M.S. and J.V. MOCCELLIN, 2002. A High Quality Solution Constructive Heuristic for Flow Shop Sequencing, *Journal of The Operational Research Society*, vol. 53, 1374-1379.

NAGANO M.S. and J.V. MOCCELLIN, 2000. Heuristics for Hybrid Flow Shop Scheduling, *Brazilian Symposium on Operations Research*, Proceedings, 326-337.

PINEDO M., 1995. *Scheduling: Theory, Algorithms, and Systems*, Prentice-Hall, Englewood Cliffs, New Jersey.

PORTMANN M-C, A. VIGNIER, D. DARDILHAC and D. DEZALAY, 1996. Some Hybrid flowshop scheduling by crossing branch and bound and genetic algorithms, *5th International Workshop on Project Management and Scheduling*, Poznan-PL, 186-189.

SRISKANDARAJAH C. and S. P. SETHI., 1989. Scheduling algorithms for flexible flowshops: worst and average case performance, *European Journal of Operational Research*, vol. 43, 143-160.

VIGNIER A., J-C BILLAUT and C. PROUST, 1995. Les problèmes de type flow shop hybride : état de l'art, in *Journée d'études : Affectation et Ordonnancement*, CNRS / GdR Automatique/ Pôle SED /GT3, Tours, France, 7-47.

A pesquisa relatada neste artigo teve o apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq .