MÉTODOS HEURÍSTICOS PARA PROGRAMAÇÃO EM SISTEMAS *FLEXIBLE FLOW LINE* COM *SETUP* DEPENDENTE DA SEQUÊNCIA

Hélio Yochihiro Fuchigami
Programa de Doutorado em Engenharia de Produção
Escola de Engenharia de São Carlos, Universidade de São Paulo
Av. Trabalhador São-Carlense, 400, CEP 13566-590, São Carlos/SP

Departamento de Engenharia de Produção
Universidade Federal de Goiás – Campus Catalão
Av. Dr. Lamartine Pinto de Avelar, 1120, CEP 75704-020, Catalão/GO
heliofuchigami@yahoo.com.br

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

RESUMO

Neste artigo foram propostos e avaliados sete métodos heurísticos construtivos para programação em sistemas *flexible flow line* com tempos de *setup* dependentes da sequência de processamento das tarefas. Neste ambiente de produção pode haver uma ou mais máquinas paralelas idênticas em cada estágio. Além disso, foram consideradas algumas características realísticas: a possibilidade das tarefas saltarem os estágios de produção e a presença de tempos de *setup* antecipados e não antecipados. O critério de desempenho utilizado foi a minimização da duração total da programação (*makespan*).
PALAVRAS CHAVE. Programação da produção. *Flexible flow line*. *Setup* dependente. AD&GP

ABSTRACT

This paper presents seven constructive heuristic algorithms for scheduling on flexible flow line with sequence-dependent setup times. In this production system, may there be one or more identical parallel machines at each stage. Moreover, some realistic characteristics were considered: jobs can skip stages and the setup times may be anticipatory or non-anticipatory. The objective is to minimize the total time to complete the schedule (makespan).
KEYWORDS. Production scheduling. Flexible flow line. Sequence-dependent setup times.

1. Introdução

Esta pesquisa propõe sete heurísticas construtivas para o problema de programação da produção em sistemas *flexible flow line* com tempos de *setup* dependentes da sequência de execução das tarefas. O critério de desempenho considerado é a minimização da duração total da programação (*makespan*).

O ambiente *flexible flow line* é uma generalização do *flow shop* híbrido (ou *flow shop* com múltiplas máquinas), removendo a restrição de que as tarefas precisam passar por todos os estágios. Assim, o problema tratado é multiestágio com fluxo unidirecional onde as tarefas podem saltar estágios. Em cada estágio pode haver uma ou mais máquinas paralelas idênticas.

A configuração do *flexible flow line* pode ser encontrada em um vasto número de indústrias: química, eletrônica, de empacotamento, farmacêutica, automotiva, fabricação de embalagens de vidro, madeireira, têxtil, herbicidas, alimentícia, cosméticos e de semicondutores QUADT; KUHN, 2007a).

A Figura 1 ilustra um *flexible flow line* com g estágios de produção e m_k máquinas no estágio k. Tipicamente, estoques intermediários (*buffers*) são representados entre os estágios para armazenamento temporário das tarefas.

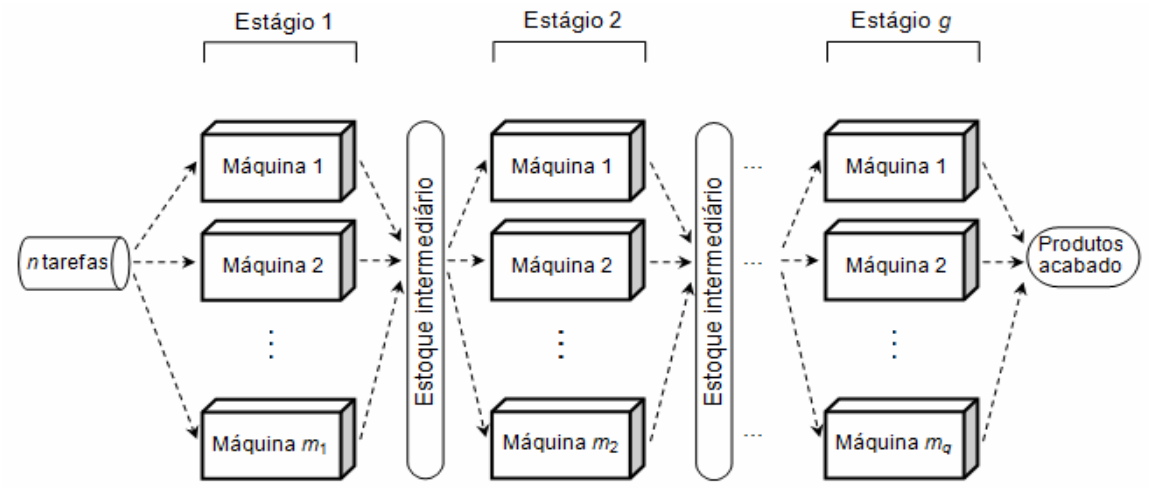

FIGURA 1 – Esquema ilustrativo de um *flexible flow line*

As operações de *setup* incluem todo trabalho de preparação da máquina ou da oficina para a fabricação dos produtos, como a obtenção e ajuste de ferramentas, inspeção e posicionamento de materiais e processos de limpeza.

Os tempos de *setup* dependem tanto da tarefa a ser processada como também da que foi executada imediatamente antes na mesma máquina (*setup* dependente da sequência). Esta abordagem é necessária em indústrias químicas, por exemplo, em que o processo de limpeza entre diferentes componentes é diferenciado para assegurar os baixos níveis toleráveis de impureza. Situações semelhantes podem ser encontradas na produção de diferentes cores de tinta, concentrações de detergente e misturas de combustível. No problema tratado, o tempo de *setup* da primeira tarefa programada em cada máquina também foi considerado.

Além disso, é mais realista considerar que alguns tipos de *setup* podem ser realizados antecipadamente, ou seja, antes da liberação da tarefa no estágio anterior. Como outros tipos de *setup* podem requer que o produto esteja presente na máquina onde será processado, como por exemplo operações de ajuste da peça, ambas as possibilidades foram consideradas neste trabalho: *setup* antecipado e não antecipado.

Segundo a conhecida notação de três campos e as adaptações de Vignier, Billaut e Proust (1999), Allahverdi, Gupta e Aldowaisan (1999) e Lin e Cheng (2005), o problema estudado pode ser representado por $FFSg,((PM^{(k)})_{k=1}^{g}) \mid s_{ijk}, as, ns \mid C_{\max}$.

O problema de programação em ambientes *flexible flow line* é *NP-hard* para todos os critérios de otimização tradicionais, mesmo quando o *setup* não é considerado explicitamente (QUADT; KUHN, 2007a). Pela complexidade intrínseca ao problema, um dos principais objetivos deste trabalho foi desenvolver um conjunto de heurísticas eficazes e computacionalmente eficientes.

2. Revisão bibliográfica

Muitas publicações já abordaram o problema de programação em *flow shop* híbrido, desde o trabalho de Salvador (1973). Linn e Zhang (1999) propuseram uma classificação das pesquisas em três categorias: problemas com dois estágios, três estágios e g estágios ($g > 3$). Definições de variações flexíveis deste ambiente podem ser encontradas em Vairaktarakis (2004) e uma taxonomia para *flow shop* híbridos foi publicada por Quadt e Khun (2007a). O estado da arte para os respectivos momentos foi apresentado por Vignier, Billaut e Proust (1999), Kis e Pesch (2005) e Wang (2005).

A maioria dos algoritmos que fornecem solução ótima de problemas sem *setup* separado baseia-se no método *branch-and-bound*, como é o caso de Brah e Hunsucker (1991), Rajendran e Chaudhuri (1992), Moursli e Pochet (2000), Néron, Baptiste e Gupta (2001) e Azizoglu, Cakmak e Kondakci (2001).

O ambiente com dois estágios, sendo uma máquina no primeiro e várias máquinas paralelas idênticas no segundo, com o critério de minimização do *makespan*, foi estudado por Gupta e Tunc (1994) e Huang e Li (1998). O mesmo problema com máquinas paralelas não relacionadas no segundo estágio foi analisado por Riane, Artiba e Elmaghraby (2002).

Alguns trabalhos utilizaram a técnica de decomposição por estágio: Wittrock (1988), Guinet e Solomon (1996), Brah e Loo (1999), Koulamas e Kyparisis (2000), Soewandi e Elmaghraby (2001) e Riane, Artiba e Iassinovski (2001). Por outro lado, outras pesquisas abordaram o método de decomposição por tarefa: Sawik (1995), Gupta *et al*. (2002).

Häyrinen *et al*. (2000) propuseram heurísticas para programação de *flexible flow line* em indústrias eletrônicas. As tarefas eram agrupadas em famílias e os tempos de *setup* eram dependentes da sequência e da máquina. Kurz e Askin (2001) propuseram um limitante inferior para o ambiente *flexible flow line* e minimização do *makespan*.

Lin e Liao (2003) enfocaram um problema real com tempos de *setup* dependentes da sequência no primeiro estágio, máquinas dedicadas no segundo e dois prazos de entrega. O objetivo era minimizar o atraso máximo ponderado na programação de um dia de produção. Oğuz *et al.* (2003) estudaram o problema de multiprocessamento de tarefas, que permite o processamento simultâneo das operações. Foram apresentados algoritmos construtivos para a minimização do *makespan* em ambientes com dois estágios.

Kurz e Askin (2003) compararam regras de sequenciamento para minimização do *makespan* em *flexible flow line* com tempos de *setup* não antecipados e dependentes da sequência de tarefas. Kurz e Askin (2004) estudaram o mesmo problema e apresentaram várias heurísticas de inserção e algoritmos genéticos *random keys*.

Várias medidas de desempenho foram analisadas por Allaoui e Artiba (2004), entre elas a minimização do *makespan* e do atraso máximo, considerando tempos de transporte. Şerifoğlu e Ulusoy (2004) revolveram o problema de minimização do *makespan* em que uma operação pode ser processada simultaneamente em várias máquinas de um estágio. Foi empregado um algoritmo genético para o sequenciamento do primeiro estágio.

O problema de programação em indústria de móveis foi estudado por Wilson, King e Hodgson (2004). Havia várias máquinas paralelas idênticas em cada estágio. As heurísticas desenvolvidas baseiam-se em um algoritmo genético e demonstraram eficiência na minimização do tempo de *setup* (independente da sequência) e do *makespan*.

Fuchigami (2005) propuseram quatro métodos heurísticos construtivos para programação em *flow shop* híbridos com tempos de *setup* assimétricos e dependentes da sequência. O objetivo era minimizar o *makespan*.

Heurísticas construtivas para minimização do *makespan* foram apresentadas por Logendran, Carson e Hanson (2005). As tarefas foram agrupadas em famílias e foram considerados tempos de *setup* dependentes das máquinas e independentes das famílias de tarefas. Logendran, deSzoeke e Barnard (2006) estudaram o mesmo problema e apresentaram três diferentes algoritmos baseados na busca tabu.

Ruiz e Maroto (2006) propuseram algoritmos genéticos para problemas com máquinas paralelas não relacionadas em cada estágio, tempos de *setup* dependentes e elegibilidade de máquina. Foram realizados experimentos com dados simulados e reais de indústrias de cerâmica.

Um algoritmo imunológico foi apresentado por Zandieh, Ghomi e Husseini (2006). Os resultados apresentaram desempenho superior ao algoritmo genético *random key* de Kurz e Askin (2004), principalmente para problemas de médio e grande porte.

Jenabi *et al*. (2007) consideraram o sistema *flexible flow line* com máquinas paralelas não relacionadas. O objetivo era minimizar a soma do custo de *setup* e de armazenamento sem ruptura (*stock-out*). Duas heurísticas construtivas foram propostas para fornecer a solução inicial a um algoritmo genético híbrido e um *simulated annealing*.

O problema de dimensionamento de lote com custos de *setup* foi considerado por Quadt e Kuhn (2007b). O objetivo era minimizar os custos de *setup* e o tempo médio de fluxo.

Jungwattanakit *et al.* (2008) propuseram algoritmos heurísticos para o problema *flexible flow line* com *setup* dependente da sequência e da máquina. Cada estágio possuía máquinas paralelas não relacionadas. O objetivo era minimizar uma combinação convexa do *makespan* e do número de tarefas atrasadas.

Ruiz, Şerifoğlu e Urlings (2008) abordaram um problema real de *flexible flow line* com máquinas paralelas não relacionadas e tempos de *setup* dependentes da sequência. Várias outras restrições foram consideradas: presença de tempos de *setup* antecipado e não antecipado, diferentes datas de liberação das máquinas, elegibilidade de máquina e restrições de precedência das tarefas.

Fuchigami e Moccellin (2009) propuseram e compararam o desempenho de treze regras de prioridade para o problema de programação em *flexible flow line* com *setup* dependente, identificando que é mais eficaz considerar a ordenação não crescente da carga de trabalho de todos os estágios. Fuchigami (2010) abordou dois problemas relacionados (*flexible flow line* com *setup* dependente e independente da sequência), propondo para cada um deles dois grupos de métodos de solução: regras de prioridade e heurísticas construtivas. O estudo destaca também as características do ambiente de produção mais influentes nos resultados dos métodos.

Naderi, Ruiz e Zandieh (2010) propuseram dois algoritmos para programação em *flexible flow line* com *setup* dependente da sequência e minimização do *makespan*: uma regra de prioridade dinâmica e o uma meta-heurística de busca local iterativa.

Como pode ser observado, embora o ambiente *flow shop* híbrido tenha sido extensivamente estudado há muitos anos, existem poucas publicações com os sistemas *flexible flow line*. Além disso, não foi encontrado nenhum trabalho abordando o mesmo problema proposto nesta pesquisa.

3. Métodos heurísticos construtivos propostos

Os algoritmos propostos atendem a classificação de *métodos de decomposição por estágio*, apresentada por Quadt e Kuhn (2007a), em que o problema o original com g estágios é dividido em g subproblemas iterativos contendo uma única máquina ou múltiplas máquinas paralelas idênticas.

Heurística Hd1

Do primeiro ao penúltimo estágio, a heurística Hd1 verifica todas as possibilidades de alocação tarefa-máquina e escolhe a que leve à menor data de término, ou seja, faz a alocação pela regra *Earliest Completion Time* (*ECT*). Esta ideia foi empregada por Weng, Lu e Ren (2001)

no problema de minimização do tempo de fluxo ponderado no ambiente de máquinas paralelas não relacionadas com tempos de *setup* dependentes da sequência.

No último estágio, a heurística Hd1 emprega o algoritmo *Multiple Insertion* (*MI*), proposto por Kurz e Askin (2001b) para o problema de minimização do *makespan* em máquinas paralelas com datas de liberação diferentes e *setup* dependente.

Algoritmo Hd1

PASSO 1. Do primeiro ao penúltimo estágio, analise todas as possibilidades de alocação tarefa-máquina e escolha a opção com a menor data de término (*ECT*).
PASSO 2. Repita o PASSO 1 até que todas as tarefas estejam programadas.
PASSO 3. No último estágio, aplique o Algoritmo *MI*.

Heurísticas Hd2 e Hd3

As heurísticas Hd2 e Hd3 são adaptações da heurística Hd1 para situações específicas em que existem estágios com apenas uma máquina. Quando *todos* os estágios possuem máquinas paralelas, as heurísticas Hd1, Hd2 e Hd3 são idênticas.

Qualquer que seja o estágio com apenas uma única máquina, as heurísticas Hd2 e Hd3 resolvem-no como um típico Problema do Caixeiro Viajante, aplicando respectivamente os algoritmos *FITSP* (*Farthest Insertion Traveling Salesman Procedure*) e *NITSP* (*Nearest Insertion Traveling Salesman Procedure*), ambos descritos a seguir.

Algoritmos FITSP e *NITSP*

PASSO 1: Escolha arbitrariamente o nó inicial (tarefa fictícia J_0).
PASSO 2: Encontre o próximo nó da sub-rota que tenha a maior distância (ou a menor distância no *FITSP*) do nó escolhido anteriormente.
PASSO 3 (PASSO DE SELEÇÃO): Entre os nós ainda não selecionados, escolha o que tenha a maior distância (ou a menor distância no *FITSP*) de qualquer nó da sub-rota.
PASSO 4 (PASSO DE INSERÇÃO): Examine todas as possibilidades de inserção do nó escolhido na sub-rota atual, adotando aquela que leva ao menor comprimento total.
PASSO 5: Se todos os nós foram inseridos na rota, PARE. Caso contrário, vá para o PASSO 3.

Algoritmo Hd2/Hd3

PASSO 1. Do primeiro ao penúltimo estágio, se o estágio a ser programado possuir máquinas paralelas, analise todas as possibilidades de alocação tarefa-máquina e escolha a opção com a menor data de término.
Senão, se o estágio a ser programado possuir apenas uma máquina, aplique o Algoritmo *FITSP* (ou *NITSP*, na Hd3). No Passo de Seleção, considere na matriz de distâncias a soma $r_{jk}+s_{ijk}$, com $i \in V_k \cup \{0\}$, $j \in V_k$, onde V_k é o conjunto de tarefas que visitam o estágio k a ser programado. No Passo de Inserção, calcule o comprimento total considerando as datas de liberação das tarefas (r_{jk}) e a antecipação ou não do *setup*.
PASSO 2. Repita o passo 1 até que todas as tarefas estejam programadas e vá para o PASSO 3.
PASSO 3. Se o último estágio possuir máquinas paralelas, aplique o Algoritmo *MI*.
Senão, aplique o Algoritmo *FITSP* (ou *NITSP*, na Hd3), tal como no PASSO 1.

Heurística Hd4

A heurística Hd4 utiliza o procedimento desenvolvido por Stinson (STINSON e SMITH, 1982) tanto no sequenciamento como na alocação no *flexible flow line*. A heurística de Stinson é utilizada para fornecer uma sequência de tarefas que minimizam o tempo total de *setup* e já foi aplicado por Simons Jr. (1992) no *flow shop* tradicional, por Fuchigami (2005) na

ordenação inicial para o *flow shop* híbrido e por Chen (2008) como uma etapa de melhoria na sua heurística para o problema de máquina única.

Algoritmo Hd4

PASSO 1. No primeiro estágio, aplique a heurística de Stinson, considerando a matriz formada pelos elementos $\sum_{k=1}^{g}\left[\left(1-a_{ijk}\right)s_{ijk}+p_{jk}\right]$, com $i,j \in V_1$, onde g é o número de estágios do problema, V_1 é o conjunto de tarefas que visitam o estágio 1 e $a_{ijk} = 1$ se o *setup* da tarefa i para a j no estágio k for antecipado e $a_{ijk} = 1$, caso contrário.

PASSO 2. Quebre a sequência circular obtida em cada ponto possível e selecione a sequência com o menor tempo total de fluxo (considerando a tarefa fictícia inicial e os dados do primeiro estágio) como se fosse um problema de máquina única.

PASSO 3. Quebre a sequência obtida em m_1 partes da forma mais equitativa possível* e aloque cada parte a uma máquina do primeiro estágio e vá para o estágio 2.

PASSO 4. Atualize as datas de liberação como as datas de término do estágio anterior.

PASSO 5. Analise todas as possibilidades de alocação tarefa-máquina e escolha a opção com a menor data de término.

PASSO 6. Repita o PASSO 5 até que todas as tarefas do estágio estejam programadas.

PASSO 7. Vá para o próximo estágio e repita os PASSOS 4 a 6 até que todos os estágios estejam programados.

* formação das m_1 partes equitativas: divide-se o tempo total de fluxo pelo número de máquinas do estágio 1, alocando a primeira fração da carga à primeira máquina. Antecedendo a segunda fração da carga, considera-se o *setup* da tarefa fictícia e divide-se o novo tempo total de fluxo pelo número de máquinas não programadas. Este procedimento é repetido até que todas as máquinas tenham recebido a sua fração de carga.

Heurísticas Hd5 e Hd6

As heurísticas Hd5 e Hd6 empregam respectivamente as regras LPT3 e LPT5 na ordenação inicial e na alocação a regra *Next Job Same Machine* (*NJSM*), apresentadas por Urlings, Ruiz e Şerifoğlu (2008). A regra *NJSM* assume que a tarefa q é associada à mesma máquina que a tarefa j e aloca a tarefa j à máquina em que q termina antes. Assim, é escolhida a máquina que otimiza $\min_{m \in M_k}[C_{ik}+s_{ijk}+s_{jqk}]$, onde C_{ik} é a data de término da tarefa i no estágio k, uma vez que p_{jk} e p_{qk} são constantes. Salienta-se que apenas a tarefa j é efetivamente alocada.

Algoritmo Hd5/Hd6

PASSO 1. No primeiro estágio, sequencie as tarefas pela regra LPT3 (ou LPT5, na Hd6). Vá ao PASSO 3.

PASSO 2. Do segundo estágio em diante, sequencie as tarefas pela regra *ERD*.

PASSO 3. Associe sequencialmente cada tarefa de acordo com a regra *NJSM*, ou seja, considerando que as tarefas consecutivas j e q (com $j,q \in V_k$, onde V_k é o conjunto de tarefas que visitam o estágio k a ser programado) sejam alocadas na mesma máquina, aloque somente a tarefa j à máquina que minimiza C_{qk}.

PASSO 4. Repita os PASSOS 2 e 3 até que todos os estágios estejam programados.

Heurística Hd7

Neste estudo foi utilizado também um método de programação aleatório que serve como base para comparar as heurísticas propostas, assim como foi feito por Moursli (1999, p.69).

Algoritmo Hi6

PASSO 1. (INICIALIZAÇÃO) Faça k=1.

PASSO 2. No estágio *k*, associe números aleatórios às tarefas que o visitam e sequencie-as pela ordem não decrescente destes valores.
PASSO 3. Aloque sequencialmente cada tarefa à máquina selecionada por meio da geração de um número aleatório compreendido no intervalo dos índices das máquinas do estágio *k*.
PASSO 4. Repita o PASSO 2 até que todas as tarefas estejam programadas no estágio *k*.
PASSO 5. Se for o último estágio (*k*=*g*), PARE; senão, faça *k*=*k*+1 e vá para o PASSO 1.

4. Experimentação computacional e resultados

4.1. Delineamento do experimento

Na experimentação computacional foram testados e avaliados 36.000 problemas, divididos em 360 classes definidas pelo número de tarefas (*n*), número de estágios de produção (*g*), níveis de flexibilidade (*f*), intervalos de tempos de *setup* (*s*), probabilidade do *setup* ser antecipado (*a*) e probabilidade da tarefa saltar um estágio (*l*).

Para cada classe, foram gerados aleatoriamente 100 problemas visando reduzir o erro amostral. Com base em trabalhos reportados na literatura, foram definidos os seguintes parâmetros de dados dos problemas: 10, 30, 50, 80 e 100 tarefas (KURZ e ASKIN, 2004); 3, 5 e 7 estágios (LOGENDRAN *et al*., 2005); intervalo fixo de tempos de processamento, *U*[1, 99]; dois intervalos de tempos de *setup*, *U*[25, 74] e *U*[75, 125]; dois intervalos de probabilidade do *setup* ser antecipado, *U*[0, 50]% e *U*[50, 100]%; duas opções de probabilidades da tarefa saltar o estágio, 10% e 50% (RUIZ *et al*., 2006).

Tal como Logendran *et al.* (2005), foram determinados três níveis de flexibilidade para o número de máquinas por estágio: baixo, em que 1/3 dos estágios possuem máquinas paralelas; médio, com 2/3 dos estágios; e alto, onde todos os estágios possuem máquinas paralelas.

De acordo com esses parâmetros, todos os problemas foram gerados aleatoriamente e resolvidos por meio de um software construído especificamente para esta finalidade. Este software gera os arquivos de entrada com os dados dos problemas e produz a saída de arquivos comparativos com o *makespan* e o tempo de computação de cada método.

Foi utilizado o sistema operacional Windows e a linguagem de programação Delphi. As configurações da máquina são as seguintes: processador AMD Turion com 1.8 GHz de frequência e 512 MB de memória RAM.

4.2. Método de análise

Os resultados obtidos na experimentação computacional foram analisados por meio da porcentagem de sucesso, desvio relativo médio, desvio-padrão do desvio relativo e tempo médio de computação dos doze métodos desenvolvidos.

A porcentagem de sucesso é calculada pelo número de vezes que o método forneceu a melhor solução (empatando ou não), dividido pelo número de problemas da classe.

O desvio relativo mede a variação correspondente à melhor solução obtida pelos métodos, ou seja, a qualidade da solução. O desvio relativo (R_h) de um método *h* para um determinado problema é assim calculado: $R_h = (C^h_{\max} - C^b_{\max}) / C^b_{\max}$, onde $C^h_{\max}$ é o *makespan* fornecido pelo método *h* e $C^b_{\max}$ é o melhor *makespan* fornecido pelo grupo de métodos analisado.

O desvio-padrão do desvio relativo é o valor da variação dos desvios relativos de uma classe de problemas em torno do desvio relativo médio, ou seja, mede a estabilidade da solução. O desvio-padrão do desvio relativo (S_h) de um método *h* é calculado da seguinte forma: $S_h = \sqrt{\left[\sum_{i=1}^{L} (R_h^i - \overline{R})^2\right] / (L-1)}$, onde *L* é o número de problemas da classe, R_h^i é o desvio relativo da solução do problema *i* fornecida pelo método *h* e $\overline{R}$ é o desvio relativo médio do método *h* para a classe de problemas.

O tempo médio de computação de um método é a média do tempo de CPU medido em milissegundos (ms).

4.3. Análise dos resultados

Os resultados das sete heurísticas propostas foram confrontados com os das duas melhores regras de prioridade propostas por Fuchigami e Moccellin (2009): LPT3, que considera para cada tarefa *j* a soma de todos os estágios da média dos tempos de *setup* das tarefas *i* para a tarefa *j* e dos tempos de processamento de *j*, e LPT5, que utiliza a soma de todos os estágios do menor tempo de *setup* das tarefas *i* para a tarefa *j*, que visitam o estágio, e dos tempos de processamento de *j*.

É importante salientar que as regras LPT3 e LPT5 também constituem métodos heurísticos construtivos, atendendo à definição de *métodos holísticos* de acordo com a classificação de Quadt e Kuhn (2007a).

A Tabela 1 apresenta a comparação do desempenho médio das heurísticas propostas, em porcentagem de sucesso.

TABELA 1 – Comparação de desempenho das heurísticas propostas (em % de sucesso)

Heurística	Hd3	Hd2	Hd1	LPT5	Hd6	Hd5	LPT3	Hd4	Hd7
% Sucesso	64,5	23,5	19,2	10,3	10,2	10,2	10,2	9,1	0,0

Na análise global dos 36.000 problemas resolvidos, a melhor heurística foi a Hd3, com 64,5% de sucesso. Em segundo lugar ficou a heurística Hd2, obtendo 23,5% de sucesso. E em terceiro, a heurística Hd1 atingiu 19,2% de sucesso. Juntas e desconsiderando-se os empates, as heurísticas Hd1, Hd2 e Hd3 obtiveram 75,6% de sucesso. Este resultado mostra a eficácia da aplicação conjunta da alocação *ECT* e do algoritmo *MI* no problema tratado.

Os seguintes métodos tiveram resultados muito próximos: LPT5, com 10,3% de sucesso, Hd6, Hd5 e LPT3, com 10,2%, e Hd4, com 9,1%. Conforme poderia se esperar, devido a sua aleatoriedade, a pior heurística foi a Hd7, com desempenho muito próximo de zero, de 0,0194% de sucesso.

Esperava-se que a heurística Hd4 obtivesse melhores resultados, principalmente com a flexibilidade baixa, em que há maior incidência de problemas com uma máquina no primeiro estágio. Quando há apenas uma única máquina no primeiro estágio, a heurística Hd4 mantém a sequência original obtida pela Heurística de Stinson, constituindo a própria programação do estágio, uma vez que não é necessário fazer alocação de tarefas a diferentes máquinas. Quando há mais de uma máquina no primeiro estágio, ela quebra a sequência obtida.

Enquanto na análise global a heurística Hd4 ficou em penúltimo lugar (8º), com 9,1% de sucesso, perdendo apenas para a heurística aleatória Hd7, nos problemas com flexibilidade baixa ela subiu para o terceiro lugar, com 7,7% de sucesso. Neste caso, a Hd4 superou até a heurística Hd1, demonstrando a eficácia do Algoritmo de Stinson quando há predominância de estágios com uma máquina.

Com a flexibilidade média, a heurística Hd4 ficou em quarto lugar, com 6% de sucesso, e na flexibilidade alta, ficou em penúltimo, com 13,8% de sucesso. Portanto, foram os resultados da flexibilidade alta que reduziram a média do desempenho global da heurística Hd4.

Dos 12.000 problemas gerados com flexibilidade baixa, verificou-se que 7.947 possuem uma única máquina no primeiro estágio, correspondendo a 66,2%. Isto sugere que a força da heurística Hd4 está mesmo na *ordenação* e não na política de alocação utilizada, já que nas outras opções de flexibilidade, quando há predomínio de máquinas paralelas nos estágios, o desempenho deste método foi menos significativo.

A Figura 2 apresenta a porcentagem de sucesso das heurísticas por porte do problema (tarefas x estágios), agregando os resultados dos demais parâmetros.

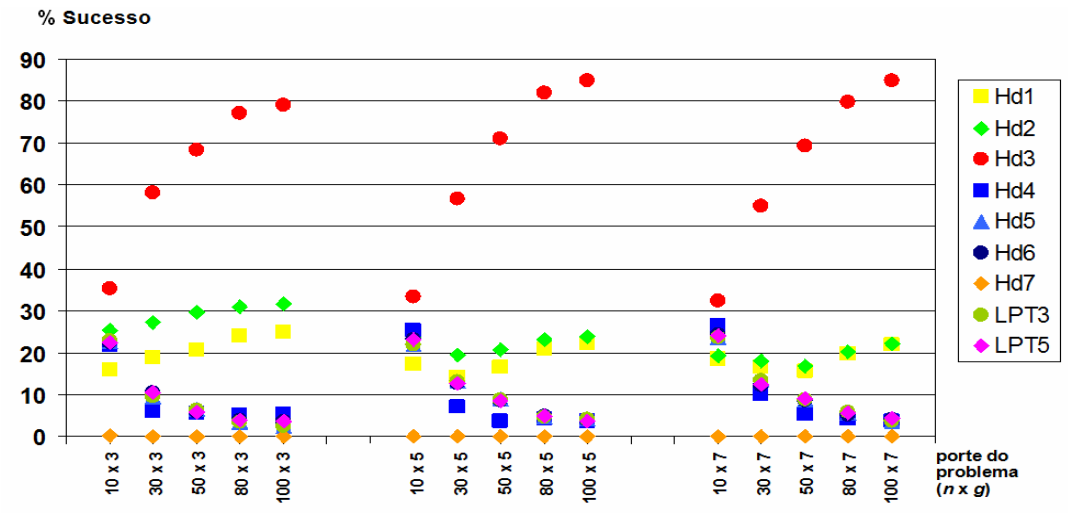

FIGURA 2 – Desempenho das heurísticas por porte do problema (em % de sucesso)

Em relação ao tamanho do problema, com o aumento do número de tarefas, as heurísticas Hd1, Hd2 e Hd3 melhoram o seu desempenho, enquanto que os métodos Hd4, Hd5, Hd6, LPT3 e LPT5 apresentam uma sensível piora. A heurística Hd7 tem desempenho constante e próximo de zero.

A melhoria de Hd1 e Hd2 é suave e deve-se à melhoria na flexibilidade alta. Já o aumento de desempenho da heurística Hd3 é substancial, variando de 32% de sucesso em problemas com 10 tarefas até 85% em problemas com 100 tarefas. Este resultado demonstra a grande eficácia da heurística Hd3 para problemas de grande porte.

Os métodos Hd4, Hd5, Hd6, LPT3 e LPT5 apresentaram resultados muito próximos, com diferença de no máximo 6%.

Como as heurísticas Hd2 e Hd3 aplicam-se somente a problemas que contêm estágios com uma máquina, foi necessário fazer a análise comparativa de forma mais detalhada separando por opção de flexibilidade. Evidentemente, para a flexibilidade alta, os resultados de Hd1, Hd2 e Hd3 foram exatamente os mesmos.

Com as flexibilidades baixa e média, de forma geral, a curva de desempenho dos métodos apresentou comportamento relativamente similar, sugerindo que o *número de estágios com máquina única* não afete significativamente o desempenho de um método.

A única heurística que tem desempenho ascendente na flexibilidade alta é a Hd3. Todas as outras pioram o desempenho com o aumento do porte do problema, nas três opções de flexibilidade.

Não apenas de forma global, mas analisando-se também mais detalhadamente por opção de flexibilidade e porte do problema, ainda há predominância de desempenho da heurística Hd3. Porém, particularmente em problemas com 10 tarefas e flexibilidade alta, a heurística Hd3 obteve o segundo pior resultado, com média de 26,1% de sucesso nas três opções de estágio, depois apenas da heurística aleatória Hd7, com 0,04% de sucesso (embora esta informação não seja perceptível no gráfico da Figura 2 por apresentar de forma agregada os resultados dos níveis de flexibilidade).

Os resultados da LPT3 e da LPT5 ficaram muito próximos, sendo coincidentes na maioria dos casos. As regras obtêm melhor desempenho com a flexibilidade alta e 50% de probabilidade de salto, principalmente em problemas com 10 tarefas.

Em relação aos dois intervalos dos tempos de *setup* considerados, não foram percebidas diferenças significativas no desempenho de cada método. Além disso, não houve variações significativas nos dois intervalos de porcentagem de antecipação do *setup*, indicando que a quantidade de *setups* antecipados não afeta a qualidade da solução das heurísticas.

Na análise das opções de probabilidade das tarefas saltarem estágios, as variações mais perceptíveis ocorreram com as heurísticas Hd2 e Hd3, especialmente para os níveis de

flexibilidade baixa e média. Com o aumento da probabilidade de salto, a heurística Hd2 aumentou o desempenho enquanto a Hd3 reduziu.

Com a flexibilidade alta, as maiores variações ocorreram com as heurísticas Hd6, Hd7, LPT3 e LPT5, ou seja, com os métodos que utilizam ordenação inicial. O desempenho foi 23% melhor com a opção de 50%. As regras de prioridade inicial tornam a programação menos flexível e portanto mais suscetível à variação de desempenho em função das características do ambiente de produção.

Os desvios relativos do método Hd3 foram os menores (2,2%), reafirmando a sua superioridade em termos de qualidade da solução. Além disso, tendem a se reduzir com o aumento do porte do problema, chegando a praticamente zero.

Em geral, todos os outros métodos aumentam os valores dos desvios relativos com o aumento do porte do problema, principalmente Hd5, Hd6, LPT3 e LPT5. A heurística aleatória Hd7 foi a mais instável, com média de 176,5% de desvio relativo.

Em termos do desvio-padrão do desvio relativo, mais uma vez a heurística Hd3 ficou em primeiro lugar (com 0,031). Os valores do desvio-padrão dos desvios relativos dos métodos ficaram na faixa até no máximo 0,10, exceto os da heurística Hd7, que apresentaram grande instabilidade e valores ascendentes com o aumento do porte do problema e da flexibilidade.

O custo de CPU não foi relevante, uma vez que a média dos tempos de computação foi de 3,2 ms. O maior tempo gasto na solução de um problema foi de 110 ms.

5. Considerações finais

Esta pesquisa abordou o problema de minimização do *makespan* em ambientes *flexible flow line* com presença de *setup* antecipado e não antecipado e dependente da sequência. Foram propostas sete métodos de solução heurísticas, comparados com os algoritmos baseados nas regras de prioridade LPT3 e LPT5.

A heurística que apresentou de longe os melhores resultados foi a Hd3, tanto na análise global, com 64,5% de sucesso, como na análise detalhada por flexibilidade e porte do problema. Além disso, com o aumento do tamanho do problema, o desempenho da Hd3 melhora substancialmente, provando a sua eficácia principalmente para problemas de grande porte. Até o penúltimo estágio, a Hd3 faz a alocação pela regra *SCT* se houver máquinas paralelas no estágio ou pelo algoritmo *NITSP* se houve apenas uma máquina; e no último estágio, utiliza o algoritmo *MI*.

Em segundo lugar, ficou a heurística Hd2, com 23,5% de sucesso, que é análoga à Hd3, utilizando o algoritmo *FITSP* no lugar no *NITSP*. As duas melhores heurísticas obtiveram juntas 88% de sucesso. Isto mostra a eficácia dos algoritmos *NITSP* e *FITSP* no ambiente *flexible flow line* e a importância de se tratar de forma particular os estágios com máquina única, pois na prática geralmente constituem gargalos de produção.

A Hd4, que utiliza o Algoritmo de Stinson, melhores resultados em problemas em que há máquina única no primeiro estágio.

A análise por flexibilidade indicou que o número de estágios com máquina única não afeta de forma relevante o desempenho dos métodos.

Os parâmetros relacionados ao *setup* (intervalos dos tempos e intervalo de porcentagem de antecipação) não influenciaram significativamente os resultados. Os parâmetros que mais afetaram o desempenho dos métodos foram o número de estágios, a flexibilidade e a probabilidade de salto. Isto mostra a importância do estudo da configuração do ambiente (*layout*) e das características das tarefas.

O custo de CPU mostrou-se desprezível. E os valores dos desvios relativos e do desvio-padrão do desvio relativo consubstanciaram a análise feita para a porcentagem de sucesso em termos da qualidade e estabilidade da solução dos métodos. Como esperado, os resultados da heurística aleatória Hd7 foram bastante instáveis.

Para o desenvolvimento de futuros trabalhos, sugere-se um estudo mais detalhado das propriedades envolvendo os estágios gargalhos, possivelmente com a proposição de novos métodos heurísticos. O mesmo ambiente de produção pode ser avaliado com outras medidas de desempenho, como *flow time* e atraso médio, ou até mesmo com bicritério. Além disso, o problema pode ser resolvido também por meio de meta-heurísticas.

Referências

Allahverdi, A., Gupta, J.N.D., Aldowaisan, T. (1999), A review of scheduling research involving setup considerations, *Omega - The International Journal of Management Science*, 27, 219-239.

Allaoui, H., Artiba, A. (2004), Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints, *Computers and Industrial Engineering*, 47, 431-450.

Azizoglu, M., Cakmak, E., Kondakci, S. (2001), A flexible flowshop problem with total flow time minimization, *European Journal of Operational Research*, 132, 3, 528-538.

Brah, S.A., Hunsucker, J.L. (1991), Heuristics for scheduling in a flow shop with multiple processors, *European Journal of Operational Research*, 51, 88-99.

Brah, S.A., Loo, L.L. (1999), Heuristics for scheduling in a flow shop with multiple processors, *European Journal of Operational Research*, 113, 113-122.

Chen, W.-J. (2008), Sequencing heuristic for scheduling jobs with dependent setups in a manufacturing system, *International Journal of Advanced Manufacturing Technology*, 38, 176-184.

Fuchigami, H.Y., *Flexible flow line com tempos de setup: métodos heurísticos*, Tese (Doutorado), Universidade de São Paulo, São Carlos, 2010.

Fuchigami, H.Y., *Métodos heurísticos construtivos para o problema de programação da produção em sistemas flow shop híbridos com tempos de preparação das máquinas assimétricos e dependentes da sequência*, Dissertação (Mestrado), Universidade de São Paulo, São Carlos, 2005.

Fuchigami, H.Y., Moccellin, J.V. (2009), Regras de prioridade para programação em sistemas *flexible flow line* com tempos de *setup* dependentes da sequência. *Anais do XVI Simpósio Brasileiro de Engenharia de Produção*. CD-ROM.

Guinet, A.G.P., Solomon, M.M. (1996), Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time, *International Journal of Production Research*, 34, 6, 1643-1654.

Gupta, J.N.D., Tunc, E.A. (1994), Scheduling a two-stage hybrid flowshop with separable setup and removal times, *European Journal of Operational Research*, 77, 415-428.

Gupta, J.N.D., Kruger, K., Lauff, V., Werner, F., Sotskov, Y.N. (2002), Heuristics for hybrid flow shops with controllable processing times and assignable due dates, *Computers and Operations Research*, 29, 10, 1417-1439.

Häyrinen, T., Johnsson, M., Johtela, T., Smed, J., Nevalainen, O. (2000), Scheduling algorithm for computer-aided line balancing in printed circuit board assembly, *Production Planning and Control*, 11, 5, 497-510.

Huang, W., Li, S. (1998), A two-stage hybrid flowshop with uniform machines and setup times, *Mathematical and Computer Modeling*, 27, 27-45.

Jenabi, M., Fatemi Ghomi, S.M.T., Torabi, S.A., Karimi, B. (2007), Two hybrid meta-heuristics for the finite horizon ELSP in flexible flow lines with unrelated parallel machines, *Applied Mathematics and Computation*, 186, 230-245.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., Werner, F. (2008), Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria, *International Journal of Advanced Manufacturing Technology*, 37, 354-370.

Kis, T., Pesch, E. (2005), A review of exact solution methods for the non-preemptive multiprocessor flowshop problem, *European Journal of Operational Research*, 164, 592-608.

Koulamas, C., Kyparisis, G.J. (2000), Scheduling uniform parallel machines to minimize maximum lateness, *Operations Research Letters*, 26, 4, 175-179.

Kurz, M.E.; Askin, R.G. (2001), An adaptable problem-space-based search method for flexible flow line scheduling, *IIE Transactions*, 33, 691-693.

Kurz, M.E.; Askin, R.G. (2003), Comparing scheduling rules for flexible flow lines, *International Journal of Production Economics*, 85, 371-388.

Kurz, M.E.; Askin, R.G. (2004), Scheduling flexible flow lines with sequence-dependent setup times, *International Journal of Operational Research*, 159, 66-82.

Lin, B.M.T., Cheng, T.C.E. (2005), Two-machine flowshop batching and scheduling, *Annals of Operations Research*, 133, 149-161.

Lin, H.T., Liao, C.J. (2003), A case study in a two-stage hybrid flow shop with setup time and dedicated machines, *International Journal of Production Economics*, 86, 2, 133-143.

Linn, R., Zhang, W. (1999), Hybrid flow shop scheduling: a survey, *Computers & Industrial Engineering*, 37, 1-2, 57-61.

Logendran, R., Carson, S., Hanson, E. (2005), Group scheduling in flexible flow shops, *International Journal of Production Economics*, 96, 143-155.

Logendran, R., deSzoeke, P., Barnard, F. (2006), Sequence-dependent group scheduling problems in flexible flow shops, *International Journal of Production Economics*, 102, 66-86.
Moursli, O., Pochet, Y. (2000), A branch-and-bound algorithm for the hybrid flowshop, International Journal of Production Economics, 64, 1-3, 113-125.
Naderi, B., Ruiz, R., Zandieh, M. (2010), Algorithms for a realistic variant of flowshop scheduling, *Computers and Operations Research*, 37, 2, 236-246.
Néron, E., Baptiste, P., Gupta, J.N.D. (2001), Solving hybrid flow shop problem using energetic reasoning and global operations, *Omega - The International Journal of Management Science*, 29, 6, 501-511.
Oğuz, C., Ercan, M.F., Cheng, T.C.E., Fung, Y.F. (2003), Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop, *European Journal of Operational Research*, 149, 2, 390-403.
Quadt, D., Kuhn, H. (2007), A taxonomy of flexible flow line scheduling procedures, *European Journal of Operational Research*, 178, 686-698.
Quadt, D., Kuhn, H. (2007), Batch scheduling of jobs with identical process times on flexible flow lines, *International Journal of Production Economics*, 105, 385-401.
Rajendran, C., Chaudhuri, D. (1992), Scheduling in n-job, m-stage flow shop with parallel processors to minimize makespan, *International Journal of Production Economics*, 27, 137-143.
Riane, F., Artiba, A., Elmaghraby, S.E. (2002), Sequencing a hybrid two-stage flow shop with dedicated machines, *International Journal of Production Research*, 40, 17, 4353-4380.
Riane, F., Artiba, A., Iassinovski, S. (2001), An integrated production planning and scheduling system for hybrid flowshop organizations, *International Journal of Production Economics*, 74, 33-48.
Ruiz, R., Sivrikaya-Şerifğlu, F., Urlings, T. (2008), Modeling realistic hybrid flexible flowshop scheduling problems, *Computers and Operations Research*, 35, 1151-1175.
Ruiz, R., Maroto, C. (2006), A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility, *European Journal of Operational Research*, 169, 781-800.
Ruiz, R., Maroto, C., Alcaraz, J. (2006), Two new robust genetic algorithm for the flowshop scheduling problem, *Omega – The International Journal of Management Science*, 34, 461-476.
Salvador, M.S., A solution to a special case of flow shop scheduling problems. *In*: Elmaghraby, S.E. (ed.) (1973), *Symposium on the Theory of Scheduling and its Applications*, Springer, 83-91, 1973.
Sawik, T.A. (1993), A scheduling algorithm for flexible flow lines with limited intermediate buffers, *Applied Stochastic Models and Data Analysis*, 9, 127-138.
Şerifoğlu, F.S.; Ulusoy, G. (2004), Multiprocessor task scheduling in multistage hybrid flow-shops: A genetic algorithm approach, *Journal of the Operational Research Society*, 55, 504-512.
Simons Jr., J.V. (1992), Heuristics in flow shop scheduling with sequence dependent setup times, *OMEGA – International Journal of Management Science*, 20, 2, 215-225.
Soewandi, H., Elmaghraby, S.E. (2003), Sequencing on two-stage hybrid flowshops with uniform machines to minimize makespan, *IIE Transactions*, 35, 467-477.
Stinson, J.P., Smith, A.W. (1982), A heuristic programming procedure for sequencing the static flowshop, *International Journal of Production Research*, 6, 6, 753-764.
Vairaktarakis, G., Flexible Hybrid Flowshops. *In:* Leung, J.Y-T., *Handbook of Scheduling: algorithms, models, and performance analysis*, Chapman & Hall/CRC Press, San Diego, 5.1-5.33, 2004.
Vignier, A., Billaut, J.C., Proust, C. (1999), Les problèmes d'ordonnancement de type flow-shop hybride: état de l'art, *RAIRO – Recherche Opérationnelle*, 33, 2, 117-183.
Wang, H. (2005), Flexible flow shop scheduling: optimum, heuristic and artificial intelligence solutions, *Expert Systems*, 22, 2, 78-85.
Weng, M.X., Lu, J., Ren, H. (2001), Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective, *International Journal of Production Economics*, 70, 3, 215-226.
Wilson, A.D., King, R.E., Hodgson, T.J. (2004), Scheduling non-similar groups on a flow line: multiple group setups, *Robotics and Computer-Integrated Manufacturing*, 20, 505-515.
Wittrock, R. (1998), An adaptable scheduling algorithm for flexible flow lines, *Operations Research*, 36, 445-453.
Zandieh, M., Fatemi Ghomi, S.M.T., Moattar Husseini, S.M. (2006), An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times, *Applied Mathematics and Computation*, 180, 111-127.

A pesquisa relatada neste artigo tem o apoio da CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior.