PRODUÇÃO ONLINE
REVISTA CIENTÍFICA ELETRÔNICA DE ENGENHARIA DE PRODUÇÃO
ISSN 1676-1901

EFEITOS DE REGRAS DE PRIORIDADE PARA PROGRAMAÇÃO DA PRODUÇÃO EM SISTEMAS INDUSTRIAIS COMPLEXOS

EFFECTS OF PRIORITY RULES FOR PRODUCTION SCHEDULING ON COMPLEX INDUSTRIAL SYSTEMS

Hélio Yochihiro Fuchigami* E-mail: heliofuchigami@ufg.br
João Vitor Moccellin** E-mail: jvmoccel@sc.usp.br
*Universidade Federal de Goiás, Campus Aparecida de Goiânia (UFG), Goiânia, GO
**Universidade Federal do Ceará, Campus do Pici (UFC), Fortaleza, CE

Resumo: Neste artigo foi discutida a eficiência da utilização de regras de prioridade para programação de sistemas produtivos conhecidos como flexible flow line com tempos de setup dependentes da sequência de processamento das tarefas. Este ambiente é caracterizado pela presença de múltiplas máquinas idênticas em cada estágio e a da possibilidade das tarefas saltarem um ou mais estágios de produção. Foram considerados tanto tempos de setup antecipados como não antecipados. O objetivo do problema foi a minimização da duração total da programação (makespan). Este estudo pode ser classificado como “pesquisa aplicada” quanto à natureza, “pesquisa exploratória” quanto aos objetivos e “pesquisa experimental” quanto aos procedimentos, além da abordagem “quantitativa”. Os resultados foram analisados por meio da porcentagem de sucesso, desvio relativo, desvio-padrão do desvio relativo e tempo de computação. As duas regras que obtiveram os melhores desempenhos foram a LPT3 e a LPT5, as únicas que consideram a ordenação decrescente da carga de trabalho em todos os estágios, ou seja, no sistema produtivo tratado é mais vantajoso priorizar as tarefas com as maiores cargas de trabalho.

Palavras-chave: Programação da produção. *Flexible flow line*. *Setup* dependente. Sequenciamento. Heurísticas.

Abstract: This paper discusses the efficiency of priority rules for the flexible flow line scheduling problem. This production environment is characterized by multiple identical machines at each stage and the possibility of jobs skipping one or more stages. Sequence-dependent setup times, which can be anticipatory or non-anticipatory, are also considered. The objective of the problem was the makespan minimization (total time to complete the schedule). This study can be classified as “applied research” regarding the nature, “exploratory” about the objectives and “experimental” as to procedures, besides the “quantitative” approach. The statistics used to evaluate the priority rules’ performances were the percentage of success (in finding the best solution), relative deviation, standard deviation of relative deviation and CPU average time. The priority rules LPT3 and LPT5 reached the best performances, considering both the descending order of the workload in all stages, in other words, on the production system treated, it's more advantageous priorizing the jobs with the biggest workload.

Keywords: Production scheduling. Flexible flow line. Priority rules. Sequence-dependent setup times. Sequencing. Heuristics.

1 INTRODUÇÃO

Devido à crescente necessidade de sistemas produtivos mais flexíveis e a grande carga de trabalho exigida pelas tarefas, os sistemas denominados *flexible*

flow line estão se tornando cada vez mais comuns na indústria. Neste trabalho são apresentadas e comparadas treze regras de sequenciamento, com a respectiva política de alocação, para o problema de programação em *flexible flow line* com *setup* dependente da sequência de execução das tarefas.

Este ambiente 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 de produção. Portanto, o ambiente tratado neste trabalho é multiestágio com fluxo unidirecional onde as tarefas podem saltar estágios. Em cada estágio pode haver uma ou mais máquinas paralelas idênticas.

Os tempos de *setup* são explícitos e separados dos tempos de processamento das tarefas, sendo assimétricos (quando o *setup* da tarefa *i* para a *j* é diferente do *setup* da tarefa *j* para *i*) e dependentes da sequência de execução das tarefas. A medida de desempenho é a duração total da programação ou *makespan*.

A solução do problema de programação em sistemas *flexible flow line* é *NP-hard* para todos os critérios de otimização tradicionais, mesmo quando o *setup* não é considerado (QUADT e KUHN, 2007a). A programação com tempos de *setup* separados dos tempos de processamento é um problema difícil de ser resolvido. Ambientes com múltiplas máquinas e vários estágios de produção são ainda mais desafiadores (LIU e CHANG, 2000). Entretanto, é mais realista assumir que em cada estágio de produção, certo número de máquinas idênticas disponíveis possa operar de forma paralela (LOW, 2005).

Ambientes com tempos de *setup* dependentes da sequência são encontrados em indústrias químicas, por exemplo, onde 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.

Além disso, também 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. Ou seja, de acordo com uma determinada probabilidade, o *setup* para uma tarefa pode ser antecipado ou não antecipado.

Uma importante implicação dos tempos de *setup* antecipados é que a operação de *setup* na máquina ou estágio subsequente pode ser iniciada enquanto a tarefa ainda está sendo executada (ALDOWAISAN, 2001). Por exemplo, se há tempo ocioso na segunda máquina, o que geralmente acontece, então o *setup* nesta máquina pode ser realizado antes do término do processamento da tarefa na primeira máquina. Isto significa que uma medida de desempenho regular, como no caso do *makespan*, pode ser melhorada considerando os tempos de *setup* separados dos tempos de processamento (ALLAHVERDI, 2000).

As regras de prioridade são procedimentos de grande importância prática por serem tecnicamente simples, fáceis de compreender e por requererem pouco esforço para aplicá-las. Geralmente a utilização de regras de prioridade é suficiente para a programação em diversos ambientes de produção. Além disso, tais regras são fáceis de codificar em linguagens de programação modernas e seus cálculos são bastante rápidos.

Por todas estas razões, a pesquisa com regras de prioridade em ambientes complexos de programação da produção, como no caso deste estudo, é um tópico importante e exige cuidadosa atenção.

2 REVISÃO DA LITERATURA

Os ambientes industriais complexos, com a presença de várias máquinas em paralelo nos diversos estágios de produção, são denominados “sistemas híbridos”, como é o caso do *flow shop* híbrido, também referenciado na literatura como *flow shop* com múltiplas máquinas, *flow shop* com múltiplos processadores e *flexible flow shop*. Além destes, há os *flexible flow lines* que se distinguem dos demais pela possibilidade das tarefas saltarem estágios.

Muitos trabalhos publicados já abordaram o problema de programação em *flow shop* híbrido, desde que foi introduzido por Salvador (1973). Vignier, Billaut e Proust (1999) apresentaram o estado da arte para aquele momento para *flow hops* híbridos. Linn e Zhang (1999) propuseram uma classificação das pesquisas em três categorias: problemas com dois estágios, três estágios e *g* estágios ($g > 3$).

Variações flexíveis de *flow shops* híbridos podem ser encontradas em Vairaktarakis (2004). Kis e Pesch (2005) atualizaram o estado da arte com trabalhos

posteriores a 1999, enfocando métodos de solução exata para minimização do *makespan* e tempo médio de fluxo. Wang (2005) fez uma revisão da literatura, classificando em métodos de solução ótima, heurística e de inteligência artificial. Quadt e Khun (2007a) publicaram uma taxonomia para *flow shop* híbridos, porém denominando-o de *flexible flow line*.

Os mais recentes trabalhos de revisão da literatura sobre *flow shop* híbridos foram apresentados por Ruiz e Vázquez-Rodríguez (2010) e Ribas, Leisten e Framiñan (2010).

Como observou Quadt e Kuhn (2007a), o sistema *flexible flow line* pode ser encontrado em um vasto número de indústrias: química, eletrônica, empacotamento, farmacêutica, automotiva, fabricação de embalagens de vidro, madeireira, têxtil, herbicidas, alimentícia, cosméticos e de semicondutores.

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 Carlier e Néron (2000), Moursli e Pochet (2000), Néron, Baptiste e Gupta (2001) e Azizoglu, Cakmak e Kondakci (2001). Devido à complexidade do problema, os métodos de solução ótima têm maior interesse teórico do que prático, uma vez que somente apresentam viabilidade computacional em problemas de pequeno porte.

Alguns trabalhos utilizaram a técnica de decomposição por estágio, reduzindo a complexidade do problema: Soewandi e Elmaghraby (2001) e Riane, Artiba e Iassinovski (2001). Por outro lado, outras pesquisas abordaram o método de decomposição por tarefa, onde a cada iteração uma tarefa é selecionada e alocada a uma máquina em cada estágio de produção, como no caso de Sawik (1995) e Gupta *et al*. (2002).

Häyrinen *et al*. (2000) propuseram heurísticas para programação de *flexible flow line* em indústrias eletrônicas. O problema com máquinas paralelas uniformes nos estágios foi abordado por Sethanan (2001) com o objetivo de minimizar o *makespan*. Kurz e Askin (2001a) apresentaram um limitante inferior para o problema de minimização do *makespan* em *flexible flow line* com *setup* dependente.

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. Chang, Hsieh e Wang (2003) examinaram

uma fábrica de filme polipropileno, propondo um algoritmo genético com taxas de mutação variáveis cuja solução foi melhor do que a prática adotada pela empresa.

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. Para o mesmo problema, Oğuz *et al.* (2004) mostraram que, sem restrições de precedência, a minimização do *makespan* é alcançada em tempo polinomial.

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) empregaram um algoritmo genético para resolver o problema de minimização do *makespan* em que uma operação pode ser processada simultaneamente em várias máquinas de um estágio. O problema de programação em indústria de móveis foi estudado por Wilson, King e Hodgson (2004).

Low (2005) enfocou a minimização do tempo total de fluxo em problemas em que os estágios possuíam várias máquinas paralelas não relacionadas, tempos de *setup* independentes da sequência e tempos de remoção. Tang e Zhang (2005) propuseram uma nova heurística combinada com método de redes neurais artificiais para o problema com tempos de *setup* dependentes da sequência.

Heurísticas construtivas para minimização do *makespan* foram apresentadas por Logendran, Carson e Hanson (2005). 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. Um algoritmo imunológico (*immune algorithm*) foi apresentado por Zandieh, Ghomi e Husseini (2006) para *flow shop* híbrido com *setup* dependente.

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*). 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.

Um problema real de *flexible flow line* com máquinas paralelas não relacionadas e tempos de *setup* dependentes da sequência foi abordado por Ruiz, Şerifoğlu e Urlings (2008). 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. Jungwattanakit *et al.* (2008) propuseram algoritmos heurísticos para o problema *flexible flow line* com *setup* dependente da sequência e da máquina. O objetivo era minimizar uma combinação convexa do *makespan* e do número de tarefas atrasadas.

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

Várias estratégias de alocação foram apresentadas por Weng, Wei e Fujimura (2012) visando auxiliar regras de prioridade na programação de *flow shop* híbridos com chegadas dinâmicas de tarefas e filosofia *just-in-time* (JIT), ou seja, reduzir adiantamentos e atrasos das tarefas. E um estudo de caso em uma indústria de células de painéis solares, identificada como um *flow shop* híbrido, foi desenvolvido por Chen *et al.* (2013). Os tempos de *setup* foram considerados explícitos, havendo tanto dependentes como independentes da sequência. O objetivo do estudo foi propor uma programação que minimizasse o *makespan*.

Trabalhos recentes abordaram problemas relacionados ao tratado nesta pesquisa, porém com tempos de *setup* independentes da sequência. Um caso específico de *flow shop* híbrido conhecido como *flow shops* paralelos proporcionais foi abordado por Fuchigami (2014), por meio da proposição de algoritmos *Simulated Annealing* para a minimização do *makespan*. Fuchigami, Moccellin e Ruiz (2015) apresentaram métodos heurísticos construtivos para programação em *flexible flow line*, caracterizado pela possibilidade das tarefas saltarem estágios, com *setup*

antecipado ou não (de acordo com determinada probabilidade). E Fuchigami e Moccellin (2015) propuseram e avaliaram regras de prioridade para o problema de *flow shop* híbrido com minimização do *makespan*.

Como pode ser observado, embora o ambiente *flow shop* híbrido tenha sido extensivamente estudado, em geral as pesquisas se restringiram a situações ou configurações específicas de máquinas nos estágios, ou com os tempos de *setup* incluídos nos tempos de processamento das tarefas. Especificamente com o sistema *flexible flow line* existem poucas publicações. Além disso, não foi encontrado nenhum trabalho abordando problema exatamente com as mesmas características estudadas nesta pesquisa.

3 DEFINIÇÃO DO PROBLEMA

Segundo a notação de três campos introduzida por Graham et al. (1979), o problema pode ser representado por $\boldsymbol{FFg|s_{ijk}|C_{max}}$. No primeiro campo, "*FFg*" se refere ao *flexible flow line* com *g* estágios de produção; o campo das restrições indica a presença de *setup* dependente da sequência (s_{ijk}) da tarefa *i* para a tarefa *j* em cada estágio *k*; e o terceiro campo apresenta a medida de desempenho, o *makespan* ($C_{\max}$).

O problema consiste em programar um conjunto de *n* tarefas, definido como $J = \{1, ..., n\}$, que será processado em uma série de *g* estágios, indicada por $G = \{1, ..., g\}$. Em cada estágio *k*, $k \in G$, existe um conjunto de m_k máquinas paralelas idênticas, onde $m_k \geq 1$, que estão disponíveis e podem processar qualquer tarefa. Todas as tarefas já estão liberadas na data zero da programação ($r_j = 0$) e possuem o mesmo fluxo, passando pelos estágios na mesma ordem e sendo processadas por apenas uma máquina em cada estágio. As tarefas podem saltar estágios, ou seja, podem não ser processadas em alguns dos estágios. O conjunto de tarefas que visitam o estágio *k*, com $k \in G$, é indicado por V_k, com $V_k \subseteq J$, e o conjunto de estágios visitados pela tarefa *j*, com $j \in J$, é representado por G_j, com $G_j \subseteq G$. O tempo de processamento da tarefa *j*, $j \in J$, no estágio *k*, $k \in G$, é indicado por p_{jk}. Sem perda de generalidade, o tempo de processamento das tarefas que não visitam um estágio foi considerado igual a zero, ou seja, $p_{jk} = 0$ se $j \notin V_k$.

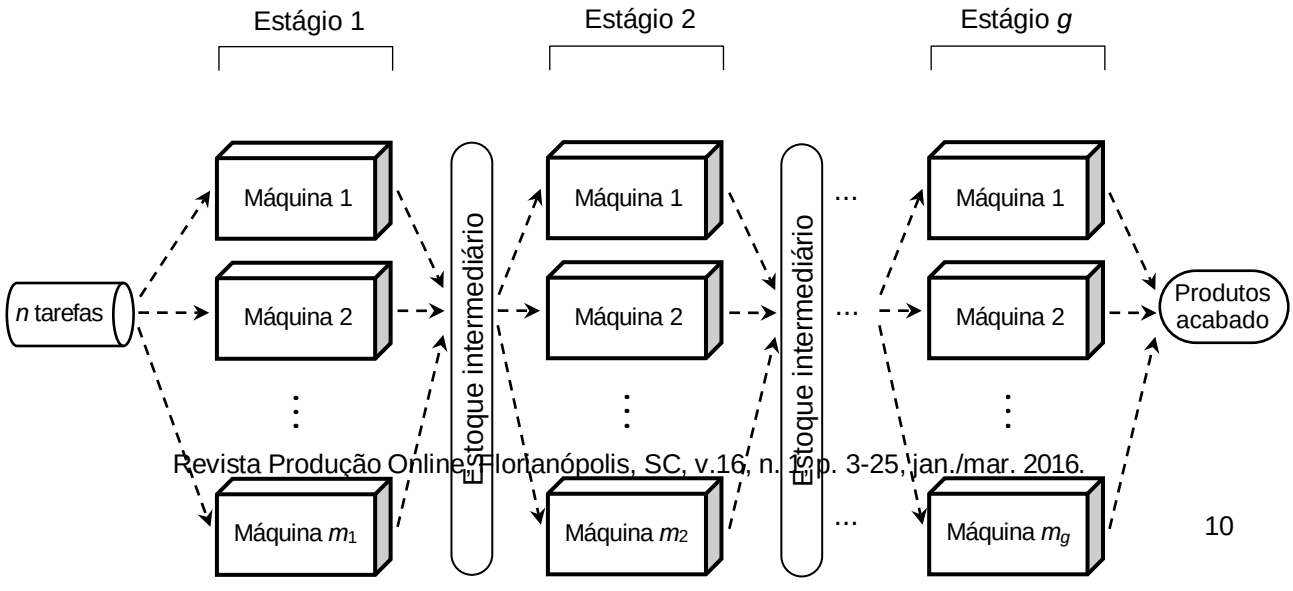
Da mesma forma que os tempos de processamento, os tempos de *setup* das tarefas que não visitam um estágio são considerados iguais a zero, ou seja, $s_{ijk} = 0$ se *i* ou *j* não pertencem a V_k. O tempo de *setup* da primeira tarefa programada em cada máquina também foi considerado. Assim, foi utilizada uma tarefa fictícia, que não requer tempo de processamento.

Além disso, o *setup* pode ser antecipado ou não. O *setup* antecipado é aquele que pode ser realizado antes da liberação da operação no estágio anterior. Conforme observado por Aldowaisan (2001), uma importante implicação dos tempos de *setup* antecipados é que a operação de *setup* na máquina ou estágio subsequente pode ser iniciada enquanto a tarefa ainda está sendo executada. Como exemplificado por Allahverdi (2000), se há tempo ocioso na segunda máquina, o que geralmente acontece, então o *setup* nesta máquina pode ser realizado antes do término do processamento da tarefa na primeira máquina. Isto significa que quando o *setup* pode ser antecipado, uma medida de desempenho regular, como no caso do *makespan*, pode ser melhorada considerando os tempos de *setup* separados dos tempos de processamento.

Entretanto, há casos que requerem que a peça ou o produto que será processado esteja presente na máquina para que o *setup* seja concluído, como por exemplo operações de posicionamento e ajuste. Assim, para que o problema tratado seja mais realista foram considerados ambos os casos de *setup* antecipado e não antecipado.

A Figura 1 ilustra um sistema *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*



A medida de desempenho utilizada foi o *makespan*, ou seja, a duração total da programação. As hipóteses consideradas nos problemas tratados são as seguintes: a produção é do tipo “fazer-para-estoque” (*make-to-stock*), assim, não há prazos de entrega (*due dates*) associados aos produtos; não há parada de máquina e todas estão continuamente disponíveis desde o início da programação (data zero); as tarefas têm a mesma data de liberação, considerada igual a zero, sem perda de generalidade; as tarefas podem esperar entre dois estágios de produção e o armazenamento intermediário (*buffer*) é ilimitado; não há tempos de transporte das tarefas entre os estágios; as operações de cada tarefa não podem ser subdivididas (*no splitting*) nem interrompidas (*no preemption*).

4 MÉTODOS DE PROGRAMAÇÃO DA PRODUÇÃO PROPOSTOS

As regras de prioridade são procedimentos de solução classificadas por Quadt e Kuhn (2007a) como métodos holísticos. Assim, é importante salientar que os métodos de solução apresentados neste trabalho são heurísticas construtivas, pois incluem tanto o sequenciamento como a política de alocação adotada nos estágios.

A necessidade de definir a sequência de tarefas é mais evidente em sistemas de produção empurrados, em que se utilizam critérios pré-determinados para emitir ordens de compra, fabricação e montagem dos itens. Já nos sistemas puxados, normalmente são implementados *kanbans* para gerenciar a produção.

Foram definidas treze regras aplicadas no primeiro estágio, propostas com base nas bem conhecidas regras *SPT* (*Shortest Processing Time*) e *LPT* (*Longest Processing Time*), bem como em adaptações de ideias e de heurísticas apresentadas por Gupta e Tunc (1994), Li (1997), Kurz e Askin (2004), Jungwattanakit *et al.* (2008), Urlings, Ruiz e Şerifoğlu (2008) e Ruiz, Şerifoğlu e Urlings (2008).

Do segundo estágio de produção em diante, os métodos utilizam a regra *ERD* (*Earliest Release Date*), que é equivalente à ordenação *FIFO* (*First In First Out*), em que as tarefas são programadas na ordem em que são liberadas no estágio anterior, ou seja, em fila.

No ambiente em estudo, a aplicação da *ERD* foi motivada pela flexibilidade do ambiente e pela possibilidade de, num determinado estágio, antecipar o processamento das tarefas que não visitaram o anterior. Segundo Jungwattanakit *et al.* (2008), esta regra minimiza a variação nos tempos de espera das tarefas em uma máquina.

A política de alocação empregada em todos os estágios foi a *ECT* (*Earliest Completion Time*), em que a tarefa é programada na máquina que completa o seu processamento mais cedo.

Os valores considerados para cada tarefa *j* na ordenação são apresentados a seguir. As regras derivadas da *SPT* consideram a ordenação crescente (não decrecente, no caso de empates) e as baseadas na *LPT* fazem a ordenação decrescente (ou não crescente).

- **SPT1** e **LPT1:** média dos tempos de *setup* das tarefas *i* para a tarefa *j* no primeiro estágio somada ao tempo de processamento de *j*. Estas regras focalizam a carga de trabalho do estágio inicial.

- **SPT2** e **LPT2:** média dos tempos de *setup* das tarefas *i* para a tarefa *j* no próximo estágio somada ao tempo de processamento também do próximo estágio. Como as regras serão aplicadas somente no primeiro estágio, este valor considerará os tempos de *setup* e de processamento do segundo estágio. Estas regras priorizam a carga de trabalho a ser realizada a seguir.

- **SPT3** e **LPT3:** 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*. Estas regras consideram a carga total de trabalho.

- **SPT4** e **LPT4:** menor tempo de *setup* das tarefas *i*, que visitam o primeiro estágio, para a tarefa *j*, somado ao tempo de processamento de *j* no primeiro estágio. São regras otimistas, considerando que a carga de trabalho do primeiro estágio será a menor possível.

- **SPT5** e **LPT5:** 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*. São regras extremamente otimistas, considerando que a carga total de trabalho será a menor possível.

- **SPT6** e **LPT6:** menor tempo de *setup* para a tarefa *j* no próximo estágio, caso tal *setup* não seja antecipado, somado ao tempo de processamento também no próximo estágio; se tal *setup* for antecipado, considera-se apenas o tempo de processamento de *j*. Estas são as únicas regras que incorporam a característica de antecipação do *setup*. Como nenhum *setup* do primeiro estágio pode ser antecipado, estas regras consideram os tempos de *setup* e de processamento do segundo estágio.

- **RAND:** ordenação aleatória, considerada para efeito de comparação.

A Tabela 1 apresenta um resumo das treze regras de sequenciamento propostas, com seus respectivos critérios de ordenação, em notação matemática. A variável v_k indica o número de *setups* válidos no estágio *k* para o cálculo da média dos tempos de *setup*, ou seja, contabiliza o número de tarefas que visitam o estágio *k* e a tarefa fictícia e não considera os elementos da diagonal principal da matriz de *setup* (elementos em que *i* = *j*, isto é, o *setup* s_{jj}). A matriz A_{ijk} indica com valores 1 os *setup* antecipados e com valores 0 os não antecipados.

Tabela 1 – Critérios de ordenação das regras de prioridade propostas

Regra	Ordenação inicial (estágio *k*=1)	Valor para ordenação inicial (estágio *k*=1)	Valor para ordenação nos estágios *k*=2 a *g*
SPT1	Crescente	$\frac{\sum_{i \in V_1 \cup \{0\}, i \neq j} s_{ij1}}{v_1} + p_{j1}$	$r_{jk} = C_{j(k-1)}$ (Crescente)
LPT1	Decrescente		
SPT2	Crescente	$\sum_{i \in V_{n+1} \cup \{0\}, i \neq j} s_{ij2}$	

LPT2	Decrescente	
SPT3	Crescente	$\sum_{k\in G}\left[\frac{\sum_{i\in V_k\cup\{0\},i\neq j} s_{ijk}}{v_k}+p_{jk}\right]$
LPT3	Decrescente	
SPT4	Crescente	$\min_{i\in V_1\cup\{0\},i\neq j} s_{ij1}+p_{j1}$
LPT4	Decrescente	
SPT5	Crescente	$\sum_{k\in G}\left[\min_{i\in V_k\cup\{0\},i\neq j} s_{ijk}+p_{jk}\right]$
LPT5	Decrescente	
SPT6	Crescente	$(1-A_{ij2})\min_{i\in V_2\cup\{0\},i\neq j} s_{ij2}+p_{j2}$
LPT6	Decrescente	
RAND	Crescente	Aleatório

A seguir é apresentado o algoritmo para a programação das tarefas utilizando as regras de prioridade descritas nesta seção.

ALGORITMO PARA FLEXIBLE FLOW LINE COM SETUP DEPENDENTE

PASSO 1. (*ORDENAÇÃO INICIAL*) No estágio *k*=1, sequencie as tarefas pela regra de prioridade escolhida no conjunto: {SPT1, LPT1, SPT2, LPT2, SPT3, LPT3, SPT4, LPT4, SPT5, LPT5, SPT6, LPT6, RAND}.

PASSO 2. (*ALOCAÇÃO INICIAL*) Aloque sequencialmente as tarefas no estágio *k*=1 na máquina que completa o processamento mais cedo (regra *ECT*). Faça *k*=2.

PASSO 3. (*ATUALIZAÇÃO*) Atualize as datas de liberação do estágio *k* como as datas de término do estágio *k*–1.

PASSO 4. (*ORDENAÇÃO DOS ESTÁGIOS k≥2*) Sequencie as tarefas pela regra *ERD.*

PASSO 5. (*ALOCAÇÃO DOS ESTÁGIOS k≥2*) Aloque sequencialmente as tarefas no estágio *k* na máquina que completa o processamento mais cedo (regra *ECT*), respeitando as datas de liberação das tarefas e desconsiderando a carga das máquinas do estágio anterior. Se for o último estágio (*k*=*g*), PARE; senão, faça *k*=*k*+1 e vá para o PASSO 3.

5 EXPERIMENTAÇÃO COMPUTACIONAL

Segundo as definições de Martins (2010, p.45-49) e Nakano (2010, p.64), esta pesquisa possui abordagem quantitativa, pois se preocupa com mensurabilidade, causalidade, generalização e replicação. Pode também ser classificada como experimento, uma vez que testa o relacionamento entre as variáveis da pesquisa operacionalizada, manipulando variáveis independentes (neste caso, a programação do problema) para se observar o resultado na variável dependente (representada aqui pela medida de desempenho *makespan*).

Além disso, de acordo com Jung (2004), este trabalho se classifica como pesquisa aplicada quanto à natureza, por gerar conhecimento com finalidades de aplicação prática, pesquisa exploratória quanto aos objetivos, pois visa melhoria teórico-prática de sistemas, processos e produtos, e inovação pela proposição de novos modelos, além de ser feita a partir de impulsos criativos, simulações e experimentações, podendo originar novos modelos destinados a invenções, inovações e a otimização, e pesquisa experimental quanto aos procedimentos, para a obtenção de novos conhecimentos e produtos tecnológicos, requerendo uma manipulação de variáveis detalhada e sistemática, e originando inovações a partir de ensaios e estudos dinâmicos em laboratório.

Foram delineados como parâmetros da experimentação computacional o 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*).

Os valores desses parâmetros foram definidos da seguinte forma: 10, 30, 50, 80 e 100 tarefas; 3, 5 e 7 estágios; dois intervalos de distribuição dos 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]%; e duas opções de probabilidade de uma tarefa saltar um estágio, 10% e 50%.

Como em Logendran, Carson e Hanson (2005), os três níveis de flexibilidade determinados para o número de máquinas por estágio são: 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.

Para o cálculo do número de estágios com máquinas paralelas, foi utilizado neste trabalho o arredondamento. Por exemplo, para o nível médio de flexibilidade em um sistema com 5 estágios, o número de estágios com máquinas paralelas é

(2/3)*5 = 3,33. Assim, arredondando, tem-se 3 estágios com máquinas paralelas e 2 estágios com uma máquina. Para definir quais estágios terão as máquinas paralelas, foram utilizados valores inteiros uniformemente distribuídos. Em tais estágios, o número de máquinas paralelas é de 2 a 5.

Foi considerado um intervalo fixo para os tempos de processamento, com valores inteiros uniformemente distribuídos entre 1 e 99. Para cada classe de problemas, foram geradas aleatoriamente 100 instâncias visando reduzir o erro amostral.

Desta forma, o número de classes de problemas é 5 (*n*) * 3 (*g*) * 3 (*f*) * 2 (*s*) * 2 (*a*) * 2 (*l*) = 360, perfazendo um total de 36.000 instâncias.

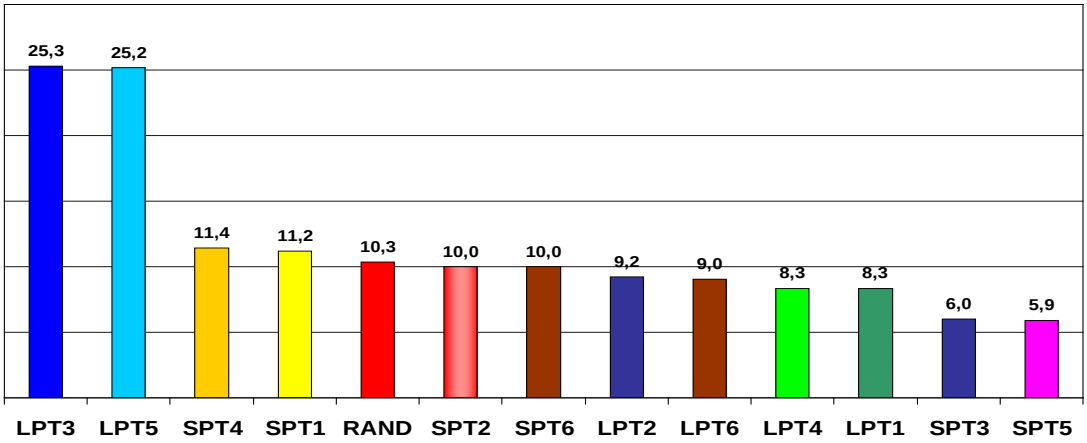
No desenvolvimento do software que gerou e resolveu os problemas, foi utilizado o sistema operacional Windows e o ambiente de programação Delphi. As configurações da máquina são as seguintes: processador AMD Turion com 1.8 GHz de frequência e 512 MB de memória RAM.

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 (medido em milissegundos).

A porcentagem de sucesso é número de vezes que o método forneceu a melhor solução (empatando ou não), dividido pelo número de problemas da classe. O desvio relativo (DR) é o percentual da variação correspondente à melhor solução obtida pelos métodos. O desvio igual a zero indica o melhor método. E o desvio-padrão do desvio relativo mede a variação dos desvios relativos de uma classe de problemas em torno do desvio relativo médio. Quanto menor o desvio-padrão, mais estável é o desempenho do método.

A análise global dos 36.000 problemas mostrou que as duas melhores regras de prioridade são a LPT3 (25,3% de sucesso) e a LPT5 (25,2% de sucesso), como pode ser visto na Figura 2. Isto indica que no ambiente tratado é mais vantajoso priorizar as tarefas com maiores cargas de trabalho.

Figura 2 – Comparação de desempenho das regras de prioridade (em % de sucesso)



O bom desempenho das regras LPT3 e LPT5 mostram que quando se consideram os dados de todos os estágios na ordenação, é melhor sequenciar pela regra *LPT*. No caso de se considerar apenas os dados do primeiro estágio, é melhor fazer o sequenciamento pela regra *SPT*, como no caso das regras SPT4 e SPT1, que ficaram em terceiro e quarto lugares (respectivamente, com 11,4% e 11,2% de sucesso).

As quatro melhores regras juntas, LPT3, LPT5, SPT4 e SPT1, obtiveram 73,1% de sucesso (empatando ou não). Todas as outras regras obtiveram desempenho inferior à regra RAND, que ficou em quinto lugar, com 10,3% de sucesso.

As piores regras foram a SPT5, que obteve 5,9% de sucesso, e a SPT3, com 6,0% de sucesso, correspondendo à ordenação oposta das duas melhores regras. Analogamente, a terceira e quarta piores regras foram a LPT4 e LPT1, cada uma com 8,3% de sucesso, também correspondendo ao oposto da ordenação da terceira e da quarta melhores regras.

A Tabela 2 apresenta o desempenho de cada regra de prioridade para cada opção de porte do problema. Os melhores resultados estão em negrito.

Tabela 2 – Desempenho das regras por porte do problema (em % de sucesso)

g	*n*	SPT1	LPT1	SPT2	LPT2	SPT3	LPT3	SPT4	LPT4	SPT5	LPT5	SPT6	LPT6	RAND
3	10	24,7	23,0	21,7	28,0	17,3	**39,7**	25,1	22,8	16,5	38,3	21,3	27,5	24,0
	30	8,7	6,3	6,5	8,1	3,8	19,8	10,1	5,6	2,9	**21,4**	7,0	8,2	7,5
	50	6,9	5,5	5,0	8,2	2,8	**20,8**	7,7	5,6	2,0	19,8	5,4	7,7	7,0
	80	6,8	4,5	5,6	7,3	2,8	**20,2**	6,7	6,0	3,1	18,2	5,0	7,3	7,8
	100	7,2	5,1	5,3	7,7	3,3	16,4	8,0	5,0	3,0	**20,0**	5,5	7,2	7,3
5	10	27,6	22,5	24,6	23,8	18,6	**38,1**	27,0	22,8	19,0	37,8	23,8	23,5	25,0
	30	7,9	4,9	7,1	5,8	3,3	24,7	7,7	5,8	3,1	**24,8**	7,6	5,0	7,0
	50	7,4	3,8	6,8	4,1	2,3	**23,5**	7,9	4,7	2,8	22,7	7,5	4,2	6,2

	80	6,3	4,7	6,2	3,6	2,4	23,7	7,1	4,4	2,0	**24,3**	5,8	4,4	6,5
	100	7,5	4,4	6,2	4,7	2,4	**23,3**	7,0	4,2	2,5	22,2	6,0	4,5	6,2
7	10	28,0	22,0	24,6	22,6	18,5	35,3	27,1	22,2	18,8	**35,9**	25,7	22,0	24,7
	30	8,3	5,1	8,9	4,2	3,2	**23,8**	8,6	5,0	3,3	22,4	9,0	4,8	6,8
	50	7,0	4,5	7,3	3,5	3,5	22,5	7,6	4,0	3,3	**23,8**	6,6	2,8	7,0
	80	6,9	4,2	7,2	3,3	3,2	23,3	7,1	3,2	2,8	**23,8**	7,2	3,3	6,3
	100	6,9	4,1	7,5	3,4	2,8	**24,6**	6,8	3,3	3,0	22,5	6,5	3,4	5,8

Em problemas com 10 tarefas, todas as regras obtiveram desempenho muito superior do que outras opções do número de tarefas, chegando a uma diferença de 22 pontos percentuais (atingida pela LPT3 comparando com problemas com 100 tarefas e 3 estágios). Isto comprova o pressuposto bom desempenho das regras para problemas de pequeno porte. O desempenho é estável para problemas de médio e grande porte, de 30 a 100 tarefas nas três opções de número de estágios.

Ambientes com flexibilidade alta têm uma leve melhoria no desempenho das regras, principalmente em problemas com 10 tarefas. Esta diferença é ainda mais perceptível com as duas melhores regras. Isto significa que quanto mais estágios contendo máquinas paralelas, melhor é o desempenho das regras, sobretudo da LPT3 e da LPT5.

Em relação aos dois intervalos de tempos de *setup* considerados, não houve diferenças significativas no desempenho de cada regra, indicando que este parâmetro não influencia no desempenho das regras de forma relevante.

`Na análise dos intervalos de porcentagem de antecipação do *setup*, houve apenas variações pouco relevantes em relação aos dois intervalos estabelecidos. As regras SPT6 e LPT6, que consideram a característica de antecipação do *setup* no valor da ordenação, não obtiveram resultados expressivos.

A probabilidade de salto foi o parâmetro que ocasionou maior variação no desempenho das regras dependendo do valor considerado (10% ou 50% de probabilidade de salto). Isto mostra a importância do estudo da possibilidade das tarefas saltarem estágios, uma situação presente em muitos tipos de indústrias.

Quando a probabilidade de saltar foi de 10%, houve certa estabilidade no desempenho das regras: LPT3 e LPT5 em torno de 20% de sucesso e as outras, na faixa até 10% de sucesso. Com 50% de probabilidade de salto, o desempenho da LPT3 e da LPT5 ficou perto dos 60% de sucesso para problemas com 10 tarefas e na faixa de 20% a 30% para problemas de 30 a 100 tarefas. Os outros métodos

também demonstraram melhor desempenho para 50% de probabilidade de salto e problemas com 10 tarefas, ficando na faixa de 30% a 45%.

Na análise dos desvios relativos, os menores valores foram apresentados pelas duas melhores regras, sendo praticamente iguais (diferenças de no máximo 0,01%). Da mesma forma, os maiores valores ocorreram com as duas priores regras (SPT5 e SPT3), também com praticamente o mesmo valor.

Nas três configurações do número de estágios, com o aumento do número de tarefas, os valores dos desvios relativos de cada regra reduzem-se consideravelmente. Por outro lado, para cada opção do número de tarefas, com o aumento do número de estágios, os valores dos desvios relativos permanecem constantes. Isto significa que a carga de trabalho afeta mais a estabilidade do resultado dos métodos do que o layout do sistema.

Na análise detalhada por característica do problema, os maiores valores dos desvios relativos ocorram em problemas com 10 tarefas e 3 estágios. Isto indica que problemas de pequeno porte causam instabilidade na solução. Com exceção desta configuração, os desvios relativos mantiveram-se estáveis na faixa de 2% a 14%.

Confirmando os resultados verificados anteriormente, as duas melhores regras também tiveram os menores valores de desvio-padrão e as duas piores regras ficaram entre as que tiveram os mais altos valores. Os valores médios do desvio-padrão de todas as regras foram muito baixos e com uma amplitude pouco relevante: na faixa de 0,03 a 0,05.

O custo de CPU foi desprezível, pois a média dos tempos de computação foi de 0,27 ms e o maior tempo gasto na solução de um problema foi de 16 ms.

6 CONSIDERAÇÕES FINAIS

Este trabalho apresentou a importância prática de se utilizar regras de prioridade para programação em sistemas complexos reais, como é o caso do *flexible flow line* com *setup* dependente. A grande vantagem das regras de prioridade em relação a outros métodos é a facilidade de implementação e o pequeno esforço computacional requerido no processamento do procedimento.

Para a solução do problema, foram propostos métodos heurísticos contrutivos compostos por regras de ordenação inicial e política de alocação. Dentre as treze

regras de prioridade analisadas, houve grande destaque no desempenho da LPT3 (que considera a soma de todos os estágios da média dos tempos de *setup* e dos tempos de processamento) e da LPT5 (que usa a soma de todos os estágios do menor tempo de *setup* e dos tempos de processamento), com resultados muito superiores às demais. Este resultado, principalmente em problemas com alta flexibilidade e com 50% da probabilidade de salto (a maior entre as analisadas), mostra a eficácia de regras que consideram dados de todos os estágios e a priorização das maiores cargas de trabalho em ambientes flexíveis.

Alguns parâmetros considerados não afetaram de forma significativa o resultado dos métodos, como os intervalos de *setup* e a porcentagem de antecipação do *setup*. Já a probabilidade de salto foi a característica que mais influenciou na solução. Além disso, o número de tarefas afetou mais do que o número de estágios. Ou seja, os atributos das tarefas demonstraram-se mais relevantes do que os das máquinas (tempos de *setup* e layout).

As curvas dos desvios relativos e dos desvios-padrão apresentaram comportamento similiar e confirmaram a análise feita com base na porcentagem de sucesso. A informação adicional foi que quanto menor é o número de tarefas, maior é a instabilidade das soluções. O custo CPU foi desprezível. Portanto, parece plausível aplicar as regras de prioridade propostas como método de solução do problema tratado.

Para trabalhos futuros, sugere-se a implementação de diferentes técnicas de solução para o problema tratado, como as metaheurísticas, e a consideração de diferentes medidas de desempenho, como tempo médio de fluxo e medidas de atraso.

Notas

A pesquisa relatada neste artigo teve o apoio da CAPES, CNPq e FAPEG. Os autores agradecem a estas agências e também aos revisores da revista pelas sugestões e comentários.

REFERÊNCIAS

ALDOWAISAN, T. A new heuristic and dominance relations for no-wait flowshop with setups. **Computer and Operations Research**, v. 28, p. 563-584, 2001. http://doi:10.1016/S0305-0548(99)00136-7

ALLAHVERDI, A. Minimizing mean flowtime in a two-machine flowshop with sequence-independent setup times. **Computers and Operations Research**, v. 27, p. 111-127, 2000. http://doi:10.1016/S0305-0548(99)00010-6

ALLAOUI, H., ARTIBA, A. Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. **Computers and Industrial Engineering**, v. 47, p. 431-450, 2004. http://10.1016/j.cie.2004.09.002

AZIZOGLU, M.; CAKMAK, E.; KONDAKCI, S. A flexible flowshop problem with total flow time minimization. **European Journal of Operational Research**, v. 132, n. 3, p. 528-538, 2001. http://doi:10.1016/S0377-2217(00)00142-9

CARLIER, J.; NÉRON, E. An exact method for solving the multi-processor flow-shop. **RAIRO Operations Research**, v. 34, p. 1-25, 2000. http://eudml.org/doc/197780

CHANG, P.-C.; HSIEH, J.-C.; WANG, Y.-W. Genetic algorithms applied in BOPP film scheduling problems: minimizing total absolute deviation and setup times. **Applied Soft Computing**, v. 3, p. 139-148, 2003. http://doi:10.1016/S1568-4946(03)00009-7

CHEN, Y.-Y.; CHENG, C-Y.; WANG, L.-C.; CHEN, T.-Z. A hybrid approach based on the variable neighborhood search and particle swarm optimization for parallel machine scheduling problems – A case study for solar cell industry. **International Journal of Production Economics**, v. 141, p.66-78, 2013. http://dx.doi.org/10.1155/2014/596850

GRAHAM, R. L.; LAWLER, E. L.; LENSTRA, J.K.; RINNOOY KAN, A.H.G. Optimization and approximation in deterministic sequencing and scheduling: a survey. **Annals of Discrete Mathematics**, v. 5, p. 287-326, 1979. http://doi:10.1016/S0167-5060(08)70356-X

FUCHIGAMI, H. Proposição de algoritmo *Simulated Annealing* para programação em *flow shops* paralelos proporcionais com tempos de *setup* explícitos. **Produção Online**, v. 14, n. 3, p. 997-1023, 2014. http://dx.doi.org/10.14488/1676-1901.v14i3.1631

FUCHIGAMI, H.; MOCCELLIN, J.V. Desempenho de regras de prioridade para programação de *flow shop* híbrido com tempos de *setup*. **Produção Online**, v. 15, n. 4, p. 1174-1194, 2015. http://dx.doi.org/10.14488/1676-1901.v15i4.1791

FUCHIGAMI, H.; MOCCELLIN, J.V.; RUIZ, R. Novas regras de prioridade para programação em *flexible flow line* com tempos de *setup* explícitos. **Production**, v. 25, n. 4, p. 779-790, 2015. http://dx.doi.org/10.1590/0103-6513.089212

GUPTA, J.N.D.; KRUGER, K.; LAUFF, V.; WERNER, F.; SOTSKOV, Y.N. Heuristics for hybrid flow shops with controllable processing times and assignable due dates. **Computers and Operations Research**, v. 29, n. 10, p. 1417-1439, 2002. http:// doi:10.1016/S0305-0548(01)00040-5

GUPTA, J.N.D., TUNC, E.A. Scheduling a two-stage hybrid flowshop with separable setup and removal times. **European Journal of Operational Research**, v. 77, p. 415-428, 1994. http://doi:10.1016/S0895-7177(97)00258-6

HÄYRINEN, T.; JOHNSSON, M.; JOHTELA, T.; SMED, J.; NEVALAINEN, O. Scheduling algorithm for computer-aided line balancing in printed circuit board assembly. **Production Planning and Control**, v. 11, n. 5, p. 497-510, 2000. http://dx.doi.org/10.1080/09537280050051997

JENABI, M.; FATEMI GHOMI, S.M.T.; TORABI, S.A.; KARIMI, B. Two hybrid meta-heuristics for the finite horizon ELSP in flexible flow lines with unrelated parallel machines. **Applied Mathematics and Computation**, v. 186, p. 230-245, 2007. http://doi:10.1016/j.amc.2006.06.121

JUNGWATTANAKIT, J.; REODECHA, M.; CHAOVALITWONGSE, P.; WERNER, F. Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. **International Journal of Advanced Manufacturing Technology**, v. 37, p. 354-370, 2008. http://10.1007/s00170-007-0977-0

KIS, T.; PESCH, E. A review of exact solution methods for the non-preemptive multiprocessor flowshop problem. **European Journal of Operational Research**, v. 164, p. 592-608, 2005. http://doi:10.1016/j.ejor.2003.12.026

KURZ, M.E. **Scheduling flexible flow line with sequence dependent setup times**. 2001. 266p. Tese (Doutorado) – Departamento de Sistemas e Engenharia Industrial, Universidade do Arizona, Tucson, 2001.

KURZ, M.E.; ASKIN, R.G. An adaptable problem-space-based search method for flexible flow line scheduling. **IIE Transactions**, v. 33, p. 691-693, 2001. http://dx.doi.org/10.1023/A:1010943502591

KURZ, M.E.; ASKIN, R.G. Comparing scheduling rules for flexible flow lines. **International Journal of Production Economics**, v. 85, p. 371-388, 2003. http://dx.doi.org/10.1016/S0925-5273(03)00123-3

KURZ, M.E.; ASKIN, R.G. Scheduling flexible flow lines with sequence-dependent setup times. **International Journal of Operational Research**, v. 159, p. 66-82, 2004. http://dx.doi.org/10.1016/S0377-2217(03)00401-6

LI, S. A hybrid two-stage flowshop with part family, batch production, and major and minor set-ups. **European Journal of Operational Research**, v. 102, p. 142-156, 1997. http://doi:10.1016/S0377-2217(96)00213-5

LIN, H.T.; LIAO, C.J. A case study in a two-stage hybrid flow shop with setup time and dedicated machines. **International Journal of Production Economics**, Amsterdam, v. 86, n. 2, p. 133-143, 2003. http://doi:10.1016/S0925-5273(03)00011-2

LINN, R.; ZHANG, W. Hybrid flow shop scheduling: a survey. **Computers & Industrial Engineering**, v. 37, n. 1-2, p. 57-61, 1999. http://doi:10.1016/S0360-8352(99)00023-6

LIU, C.-Y.; CHANG, S.-C. Scheduling flexible flow shops with sequence-dependent setup effects. **IEEE Transactions on Robotics and Automation**, v.16, n.4, p.408-419, 2000. http://dx.doi.org/10.1109/70.864235

LOGENDRAN, R.; CARSON, S.; HANSON, E. Group scheduling in flexible flow shops. **International Journal of Production Economics**, v. 96, p. 143-155, 2005. http://doi:10.1016/j.ijpe.2004.03.011

LOGENDRAN, R.; deSZOEKE, P.; BARNARD, F. Sequence-dependent group scheduling problems in flexible flow shops. **International Journal of Production Economics**, v. 102, p. 66-86, 2006. http://dx.doi.org/10.1016/j.ijpe.2005.02.006

LOW, C. Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. **Computers and Operations Research**, v. 32, p. 2013-2025, 2005. http://doi:10.1016/j.cor.2004.01.003

MARTINS, R.A. Abordagens Quantitativa e Qualitativa. In: MIGUEL, P.A.C.(Org.). **Metodologia de pesquisa em Engenharia de Produção e Gestão de Operações**. Rio de Janeiro: Elsevier, 2010. p. 45-61, cap. 3.

MOURSLI, O.; POCHET, Y. A branch-and-bound algorithm for the hybrid flowshop. **International Journal of Production Economics**, Amsterdam, v. 64, n. 1-3, p. 113-125, 2000. http://dx.doi.org/10.1016/S0925-5273(99)00051-1

NADERI, B.; RUIZ, R.; ZANDIEH, M. Algorithms for a realistic variant of flowshop scheduling. **Computers & Operations Research**, v. 37, p. 236-246, 2010. http://doi:10.1016/j.cor.2009.04.017

NAKANO, D. Métodos de Pesquisa Adotados na Engenharia de Produção e Gestão de Operações. In: MIGUEL, P.A.C.(Org.). **Metodologia de pesquisa em Engenharia de Produção e Gestão de Operações**. Rio de Janeiro: Elsevier, 2010. p. 63-72, cap. 4.

NÉRON, E.; BAPTISTE, P.; GUPTA, J.N. Solving hybrid flow shop problem using energetic reasoning and global operations. **Omega – The International Journal of Management Science**, v. 29, n. 6, p. 501-511, 2001. http://dx.doi.org/10.1016/S0305-0483(01)00040-8

OĞUZ, C.; ERCAN, M.F.; CHENG, T.C.E.; FUNG, Y.F. Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop. **European Journal of Operational Research**, v. 149, n. 2, p. 390-403, 2003. http://dx.doi.org/10.1016/S0377-2217(02)00766-X

OĞUZ, C.; ZINDER, Y.; DO, V.H.; JANIAK, A. LICHTENSTEIN, M. Hybrid flow-shop scheduling problems with multiprocessor task systems. **European Journal of Operational Research**, v. 152, n. 1, p. 115-131, 2004. http://dx.doi.org/10.1016/S0377-2217(02)00644-6

QUADT, D.; KUHN, H. A taxonomy of flexible flow line scheduling procedures. **European Journal of Operational Research**, v. 178, p. 686-698, 2007. http://dx.doi.org/10.1016/j.ejor.2006.01.042

QUADT, D.; KUHN, H. Batch scheduling of jobs with identical process times on flexible flow lines. **International Journal of Production Economics**, v. 105, p. 385-401, 2007. http://doi:10.1016/j.ejor.2006.01.042

RIANE, F.; ARTIBA, A.; IASSINOVSKI, S. An integrated production planning and scheduling system for hybrid flowshop organizations. **International Journal of Production Economics**, v. 74, p. 33-48, 2001. http://dx.doi.org/10.1016/S0925-5273(01)00105-0

RIBAS, I.; LEISTEN, R.; FRAMIÑAN, J.M. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. **Computers & Operations Research**, v. 37, p. 1439-1454, 2010. http://doi:10.1016/j.cor.2009.11.001

RUIZ, R.; MAROTO, C. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. **European Journal of Operational Research**, v. 169, p. 781-800, 2006. http://doi:10.1016/j.ejor.2004.06.038

RUIZ, R.; ŞERIFOĞLU, F.S.; URLINGS, T. Modeling realistic hybrid flexible flowshop scheduling problems. **Computers & Operations Research**, v. 35, p. 1151-1175, 2008. http://doi:10.1016/j.cor.2006.07.014

RUIZ, R.; VÁZQUEZ-RODRÍGUEZ, J.A. The hybrid flow shop scheduling problem. **European Journal of Operational Research**, v. 205, p. 1-18, 2010. http://10.1016/j.ejor.2009.09.024

SALVADOR, M.S. A solution to a special case of flow shop scheduling problems. In: ELMAGHRABY, S.E. (Ed.), **Symposium on the Theory of Scheduling and its Applications**, Springer, Berlin, p. 83-91, 1973.

SAWIK, T.. Scheduling flexible flow lines with no in-process buffers. **International Journal of Production Research**, v. 33, n. 5, p. 1357-1367, 1995. http://dx.doi.org/10.1080/00207549508930214

SETHANAN, K. **Scheduling flexible flowshops with sequence dependent setup times**. 2001. 181 p. Tese (Doutorado em Ciências de Decisão e Sistemas de Produção) – College of Engineering and Mineral Resources, West Virginia University, Morgantown, 2001.

SOEWANDI, H.; ELMAGHRABY, S.E. Sequencing on two-stage hybrid flowshops with uniform machines to minimize makespan. **IIE Transactions**, v. 35, p. 467-477, 2003. http://doi:10.1080/07408170304391

ŞERIFOĞLU, F.S.; ULUSOY, G. Multiprocessor task scheduling in multistage hybrid flow-shops: A genetic algorithm approach. **Journal of the Operational Research Society**, v. 55, p. 504-512, 2004. http://doi:10.1080/00207540500536939

TANG, L.X.; ZHANG, Y.Y. Heuristic combined artificial neural networks to schedule hybrid flow shop with sequence dependent setup times. **Lecture Notes in Computer Science**, v. 3496, p. 788-793, 2005. http://doi:10.1007/11427391_126

URLINGS, T.; RUIZ, R.; ŞERIFOĞLU, F.S. Genetic algorithms for complex hybrid flexible flow lines problems. **International Journal of Metaheuristics**, v. 1, n. 1, p. 30-54, 2010. http://dx.doi.org/10.1504/IJMHEUR.2010.033122

VAIRAKTARAKIS, G. Flexible Hybrid Flowshops. *In:* LEUNG, J.Y-T. **Handbook of Scheduling:** algorithms, models, and performance analysis. San Diego: Chapman & Hall/CRC Press, 2004. p. 5.1-5.33.

VIGNIER, A.; BILLAUT, J.C.; PROUST, C. Les problèmes d'ordonnancement de type flow-shop hybride: état de l'art. **RAIRO – Recherche Opérationnelle**, v. 33, n. 2, p. 117-183, 1999. http://eudml.org/doc/116592

WANG, H. Flexible flow shop scheduling: optimum, heuristic and artificial intelligence solutions. **Expert Systems**, v. 22, n. 2, p. 78-85, 2005.

WENG, W.; WEI, X.; FUJIMURA, S. Dynamic routings strategies for JIT production in hybrid flow shops. **Computers & Operations Research**, v. 39, p. 3316-3324, 2012. http://doi:10.1016/j.cor.2012.04.022

WILSON, A.D.; KING, R.E.; HODGSON, T.J. Scheduling non-similar groups on a flow line: multiple group setups. **Robotics and Computer-Integrated Manufacturing**, v. 20, p. 505-515, 2004. http://dx.doi.org/10.1016/j.rcim.2004.07.002

ZANDIEH, M.; FATEMI GHOMI, S.M.T.; MOATTAR HUSSEINI, S.M. An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. **Applied Mathematics and Computation**, v. 180, p. 111-127, 2006.
http://dx.doi.org/10.1016/j.amc.2005.11.136

Artigo recebido em 27/12/2013 e aceito para publicação em 02/12/2015
DOI: http://dx.doi.org/ 10.14488/1676-1901.v16i1.1720