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

DESEMPENHO RELATIVO DE REGRAS DE PRIORIDADE PARA PROGRAMAÇÃO DE *FLOW SHOP* HÍBRIDO COM TEMPOS DE *SETUP*

RELATIVE PERFORMANCE OF PRIORITY RULES FOR HYBRID FLOW SHOP SCHEDULING WITH SETUP TIMES

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
**Universidade Federal do Ceará, Campus do Pici

Resumo: Este trabalho aborda o problema de programação de *flow shop* híbridos com tempos de *setup* explícitos e independentes da sequência de tarefas. Este ambiente de produção é um sistema multiestágio com fluxo unidirecional de tarefas, onde cada estágio pode conter múltiplas máquinas disponíveis para processamento. A medida otimizada foi a duração total da programação (*makespan*). O objetivo foi propor novas regras de prioridade e avaliar o seu desempenho relativo no sistema produtivo considerado por meio da porcentagem de sucesso, desvio relativo médio, desvio-padrão do desvio relativo e tempo médio de computação. A experimentação computacional indicou que as regras que utilizam a ordenação crescente dos tempos de processamento e *setup* do primeiro estágio (SPT1 e SPT1_ERD) apresentam melhor desempenho, perfazendo juntas mais de 56% de sucesso.

Palavras-chave: Programação da produção. *Flow shop* híbrido. Regras de prioridade. *Setup* independente. Heurísticas.

Abstract: This paper focuses the hybrid flow shop scheduling problem with explicit and sequence-independent setup times. This production environment is a multistage system with unidirectional flow of jobs, wherein each stage may contain multiple machines available for processing. The optimized measure was the total time to complete the schedule (makespan). The aim was to propose new priority rules to support the schedule and to evaluate their relative performance at the production system considered by the percentage of success, relative deviation, standard deviation of relative deviation, and average CPU time. Computational experiments have indicated that the rules using ascending order of the sum of processing and setup times of the first stage (SPT1 and SPT1_ERD) performed better, reaching together more than 56% of success.

Keywords: Production scheduling. Hybrid flow shop. Priority rules. Sequence-independent setup times. Heuristics.

1 INTRODUÇÃO

A programação da produção pode ser definida genericamente como a alocação de recursos disponíveis para a execução de tarefas em um horizonte de tempo. Esta é uma importante decisão a ser tomada em sistemas de controle da produção. O problema de programação em *flow shops* híbridos tem recebido uma considerável atenção dos pesquisadores. Porém, apenas nos últimos anos alguns

trabalhos têm abordado ambientes com tempos de *setup* explícitos, ou seja, separados dos tempos de processamento das tarefas.

Os *flow shop* híbridos são sistemas produtivos caracterizados por um fluxo unidirecional de tarefas que passam por sucessivos estágios de produção, compostos por uma ou mais máquinas paralelas disponíveis para processamento. Em cada estágio, as tarefas devem ser executadas necessariamente por uma única máquina. Em essência, este sistema consiste em dois subproblemas: associar as operações das tarefas nas máquinas (problema de alocação) e sequenciar as operações em cada máquina (problema de sequenciamento).

Este tipo de ambiente é relativamente comum e pode ser encontrado em indústrias de eletrônicos, semicondutores e na produção petroquímica. Este problema também possui importantes aplicações práticas em manufaturas automatizadas como as indústrias químicas, farmacêuticas, têxteis e de papel (BOTTA-GENOULAZ, 2000).

O **tempo de *setup*** inclui todo trabalho de preparação da máquina ou da oficina para a fabricação de produtos, por exemplo, a obtenção e ajuste de ferramentas, inspeção e posicionamento de materiais e processos de limpeza.

Existem dois tipos de problemas com tempos de *setup* explícitos. No primeiro, o *setup* depende somente da tarefa a ser processada e é chamado **independente da sequência**, como nesta pesquisa. E no segundo caso, o *setup* depende tanto da tarefa a ser processada como também da que foi executada imediatamente antes na mesma máquina, sendo chamado **dependente da sequência** (ALLAHVERDI; GUPTA; ALDOWAISAN, 1999).

Muitas pesquisas em programação da produção desconsideram os tempos de *setup* ou então os incluem nos tempos de processamento das tarefas. Isto simplifica a análise das aplicações, porém afeta a qualidade da solução, principalmente quando tais tempos têm uma variabilidade relevante em função da ordenação das tarefas nas máquinas. No caso do *setup* independente da sequência, o seu tratamento explícito influencia na solução sobretudo quando puder ser antecipado, como é o caso do problema abordado neste trabalho.

É mais realista considerar que algumas operações de *setup* podem ser realizadas antecipadamente, ou seja, antes da liberação da tarefa no estágio

anterior. Como outros tipos de *setup* podem requerer que o produto esteja presente na máquina onde será processado, 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).

Este trabalho objetiva propor regras de prioridade para o problema de minimização da duração total da programação (*makespan*) em *flow shop* híbrido com tempos de *setup* independentes da sequência.

2 PROGRAMAÇÃO DA PRODUÇÃO EM FLOW SHOP HÍBRIDOS

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 shops* híbridos. Linn e Zhang (1999) propuseram uma classificação das pesquisas em três categorias: problemas com dois estágios, três estágios e mais de três estágios.

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 (2007*a*) publicaram uma taxonomia para *flow shop* híbridos, porém denominando-o de *flexible flow line*.

Ruiz e Vázquez-Rodríguez (2010) apresentaram uma revisão da literatura de métodos exatos, heurísticos e meta-heurísticos, discutindo as variações do problema, suas diferentes hipóteses, restrições e funções objetivo, além de apresentar as oportunidades de pesquisa na área. E uma extensa revisão dos trabalhos publicados recentemente (desde 1995) foi elaborada por Ribas, Leisten e Framiñan (2010), com um novo método de classificação dos trabalhos, do ponto de vista da produção, de acordo com as características das máquinas e das tarefas.

Os sistemas *flexible flow shop* estão se tornando gradativamente mais frequentes na indústria, principalmente devido à grande carga de trabalho requerida pelas tarefas nas máquinas (LOGENDRAN; CARSON; HANSON, 2005). Como observou Quadt e Kuhn (2007*a*), o sistema *flow shop* híbrido pode ser encontrado em um vasto número de indústrias, como química, eletrônica, de empacotamento, farmacêutica, automotiva, fabricação de embalagens de vidro, madeireira, têxtil, de herbicidas, alimentícia, de cosméticos e de semicondutores.

Estudos envolvendo *flow shop* híbrido com tempos de *setup* dependentes da sequência de tarefas foram realizados por Kurz e Askin (2003), Lin e Liao (2003), Kurz e Askin (2004), Ruiz e Maroto (2006), Quadt e Kuhn (2007*b*), Jungwattanakit et al. (2008), Ruiz, Şerifoğlu e Urlings (2008), Naderi, Ruiz e Zandieh (2010).

Mais recentemente, estratégias de alocação baseadas em computação distribuída 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*.

Especificamente considerando tempos de *setup* independentes da sequência de execução das tarefas, conforme o tema da presente pesquisa, foram encontrados os seguintes trabalhos. Gupta e Tunc (1994), Li (1997) e Huang e Li (1998) abordaram o ambiente *flow shop* híbrido com dois estágios, sendo uma única máquina no primeiro e várias máquinas paralelas idênticas no segundo, e a minimização do *makespan*.

O problema de minimização do atraso máximo das tarefas sujeito ao mínimo tempo de transporte foi analisado por Botta-Genoulaz (2000), incluindo os tempos de remoção das tarefas. O autor propôs e avaliou o desempenho de seis heurísticas. Allaoui e Artiba (2004) enfocaram o problema *flexible flow shop* com várias medidas de desempenho, entre elas a minimização do *makespan* e do atraso máximo, considerando tempos de transporte.

A minimização do tempo total de fluxo em problemas com estágios contendo máquinas paralelas não relacionadas e tarefas com tempos de remoção foi estudada por Low (2005). Heurísticas construtivas para minimização do *makespan* com tarefas agrupadas em famílias foram apresentadas por Logendran et al. (2005). Foram considerados os tempos de *setup* dependentes das máquinas e independentes das famílias de tarefas.

Um caso específico de *flow shop* híbrido com tempos de *setup* independentes, conhecido como *flow shops* paralelos proporcionais, foi abordado por Fuchigami (2014). Um método de geração da sequência inicial e seis algoritmos *Simulated Annealing* foram propostos para a minimização do *makespan*. E Fuchigami, Moccellin e Ruiz (2015) apresentaram métodos heurísticos construtivos para o problema de programação em *flexible flow line*, caracterizado pela possibilidade das tarefas saltarem estágios, com *setup* independente e antecipado ou não (de acordo com determinada probabilidade).

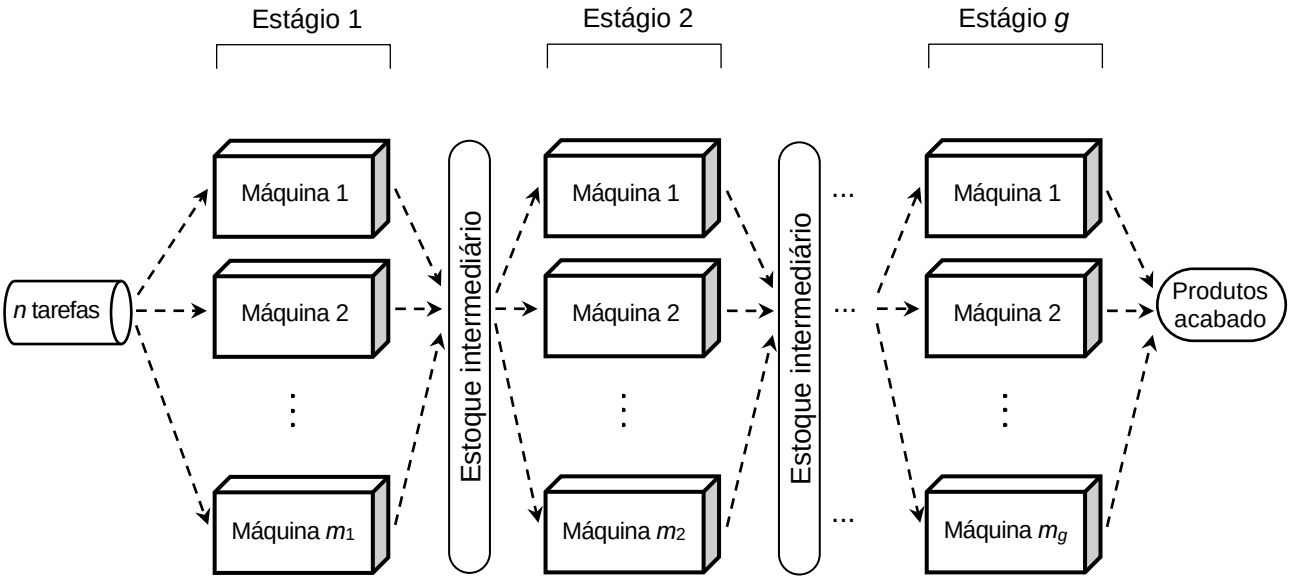
Como pode ser observado, embora o ambiente *flow shop* híbrido tenha sido extensivamente estudado, existem relativamente poucas publicações abordando especificamente sistemas com *setup* independente. Além disso, não foi encontrado nenhum trabalho considerando problema exatamente as mesmas características 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 $\boldsymbol{HFg|s_{jk}|C_{max}}$. No primeiro campo, "*HFg*" se refere ao *flow shop* híbrido com *g* estágios de produção; o campo das restrições indica a presença de *setup* independente da sequência (s_{jk}) para cada tarefa *j* em cada estágio *k*; e o terceiro campo apresenta a medida de desempenho, o *makespan* (C_{max}).

A Figura 1 ilustra um *flow shop* híbrido 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 *flow shop* híbrido



O problema consiste em programar um conjunto de *n* tarefas, definido como $J = \{1, ..., n\}$, que será processado em um conjunto de *G* estágios (1, ..., *g*). Em cada estágio *k*, $k \in G$, existe um conjunto de M_k (1, …, m_k) máquinas paralelas idênticas, onde $m_k \geq 1$, que estão disponíveis e podem processar todas as tarefas. Todas as tarefas têm o mesmo fluxo, passando pelos estágios na mesma ordem e sendo processadas por apenas uma máquina em cada estágio. O tempo de processamento da tarefa *j*, $j \in J$, no estágio *k*, $k \in G$, é indicado por p_{jk}. O tempo de *setup* das máquinas do estágio *k* é independente da sequência de execução das

tarefas e é representado por s_{jk}. O *setup* pode ser antecipado ou não, com determinada probabilidade. Como já salientado, o *setup* antecipado é aquele que pode ser realizado antes da liberação da operação no estágio anterior.

Os pressupostos considerados no problema tratado são os 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*).

A solução consiste em definir para cada estágio de produção quais as tarefas que serão processadas em cada máquina e em que ordem, além dos instantes de início e término de cada uma delas, de forma que minimize a duração total da programação.

4 REGRAS DE PRIORIDADE PROPOSTAS

Regras de prioridade, também conhecidas como *regras de sequenciamento* ou *regras de despacho*, são procedimentos de relevância prática. São tecnicamente simples, fáceis de compreender e requerem pouco esforço para aplicá-las na prática. 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 (FUCHIGAMI; MOCCELLIN; RUIZ, 2015)

É importante salientar que as *regras de prioridade* propostas neste trabalho são também **métodos heurísticos construtivos**, pois incluem em sua definição a política de alocação adotada nos estágios, e são assim aqui referenciadas para enfatizar o estabelecimento de uma ordenação inicial.

A necessidade de se definir uma 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 (SLACK et al., 1999).

Por estas razões, a pesquisa com regras de prioridade para tais problemas complexos de programação da produção é um tópico importante e exige cuidadosa atenção, constituindo o foco deste trabalho.

Para obter a solução deste problema de programação da produção, foram definidas regras de sequenciamento para as tarefas e uma política de alocação nas máquinas. As propostas se baseiam nas conhecidas regras SPT (*Shortest Processing Time*) e LPT (*Longest Processing Time*), e algumas são adaptações de heurísticas apresentadas por Gupta e Tunc (1994) e Li (1997). Além disso, como base de comparação, foi testada também a sequência gerada aleatoriamente.

As variações na forma de aplicação das regras de sequenciamento no primeiro e nos demais estágios de produção perfizeram um total de quatorze métodos de solução, definidos a seguir:

- **SPT1:** sequencia as tarefas pela ordem não-decrescente da soma dos tempos de processamento e de *setup* no primeiro estágio ($p_{j1} + s_{j1}$) e mantém esta mesma sequência em todos os estágios.
- **SPT1_ERD:** no primeiro estágio, sequencia pela ordem não-decrescente da soma dos tempos de processamento e de *setup* ($p_{j1} + s_{j1}$), e nos demais estágios, ordena pela regra ERD (*Earliest Release Date*), ou seja, aloca as tarefas na sequência em que elas são liberadas no estágio anterior.
- **SPT2:** em cada estágio, sequencia as tarefas pela ordem não-decrescente da soma dos tempos de processamento e de *setup* no estágio seguinte ($p_{j(k+1)} + s_{j(k+1)}$). No último estágio, considera a soma dos tempos do próprio estágio.
- **SPT2_ERD:** no primeiro estágio é análoga à regra SPT2 e nos demais, considera a regra ERD.
- **SPT3:** em cada estágio, sequencia as tarefas pela ordem não-decrescente da soma dos tempos de processamento e de *setup* em todos os estágios ($\sum_{k=1}^{g} p_{jk} + s_{jk}$).

- **SPT3_ERD:** no primeiro estágio é análoga à regra SPT3 e nos demais, considera a regra ERD.
- **LPT1:** sequencia as tarefas pela ordem não-crescente da soma dos tempos de processamento e de *setup* no primeiro estágio ($p_{j1} + s_{j1}$) e mantém esta mesma sequência em todos os estágios.
- **LPT1_ERD:** no primeiro estágio é análoga à regra LPT1 e nos demais, considera a regra ERD.
- **LPT2:** em cada estágio, sequencia as tarefas pela ordem não-crescente da soma dos tempos de processamento e de *setup* no estágio seguinte ($p_{j(k+1)} + s_{j(k+1)}$). No último estágio, considera a soma dos tempos do próprio estágio.
- **LPT2_ERD:** no primeiro estágio é análoga à regra LPT2 e nos demais, considera a regra ERD.
- **LPT3:** em cada estágio, sequencia as tarefas pela ordem não-crescente da soma dos tempos de processamento e de *setup* em todos os estágios ($\sum_{k=1}^{g} p_{jk} + s_{jk}$).
- **LPT3_ERD:** no primeiro estágio é análoga à regra LPT3 e nos demais, considera a regra ERD.
- **RAND:** considera em todos os estágios a mesma sequência de tarefas gerada aleatoriamente.
- **RAND_ERD:** no primeiro estágio, utiliza uma sequencia gerada aleatoriamente e nos demais, considera a regra ERD.

Após o estabelecimento da ordenação das tarefas, cada uma é associada sequencialmente à máquina com a menor carga alocada. Nesta máquina, a tarefa terá a menor data de término. Isto não ocorre, por exemplo, no ambiente em que o tempo de *setup* é dependente da sequência, pois a data de término da tarefa é afetada pela variação do tempo de *setup* de acordo com a tarefa anterior.

A Tabela 1 apresenta um resumo das quatorze regras de prioridade propostas, com seus respectivos critérios de ordenação, em notação matemática.

Tabela 1 – Critérios de ordenação das regras de prioridade propostas

Regra de Prioridade	Ordenação inicial (estágio k=1)	Critério para ordenação inicial (estágio k=1)	Critério para ordenação nos estágios k=2 a g–1	Critério para ordenação no último estágio(k=g)
SPT1	Crescente	$p_{j1} + s_{j1}$	$p_{j1} + s_{j1}$	$p_{j1} + s_{j1}$
SPT1_ERD	Crescente	$p_{j1} + s_{j1}$	$r_{jk} = C_{j(k-1)}$ (Crescente)	$r_{jk} = C_{j(k-1)}$ (Crescente)
SPT2	Crescente	$p_{j(k+1)} + s_{j(k+1)}$	$p_{j(k+1)} + s_{j(k+1)}$	$p_{jg} + s_{jg}$
SPT2_ERD	Crescente	$p_{j(k+1)} + s_{j(k+1)}$	$r_{jk} = C_{j(k-1)}$ (Crescente)	$r_{jk} = C_{j(k-1)}$ (Crescente)
SPT3	Crescente	$\sum_{k=1}^{g} p_{jk} + s_{jk}$	$\sum_{k=1}^{g} p_{jk} + s_{jk}$	$\sum_{k=1}^{g} p_{jk} + s_{jk}$
SPT3_ERD	Crescente	$\sum_{k=1}^{g} p_{jk} + s_{jk}$	$r_{jk} = C_{j(k-1)}$ (Crescente)	$r_{jk} = C_{j(k-1)}$ (Crescente)
LPT1	Decrescente	$p_{j1} + s_{j1}$	$p_{j1} + s_{j1}$	$p_{j1} + s_{j1}$
LPT1_ERD	Decrescente	$p_{j1} + s_{j1}$	$r_{jk} = C_{j(k-1)}$ (Crescente)	$r_{jk} = C_{j(k-1)}$ (Crescente)
LPT2	Decrescente	$p_{j(k+1)} + s_{j(k+1)}$	$p_{j(k+1)} + s_{j(k+1)}$	$p_{jg} + s_{jg}$
LPT2_ERD	Decrescente	$p_{j(k+1)} + s_{j(k+1)}$	$r_{jk} = C_{j(k-1)}$ (Crescente)	$r_{jk} = C_{j(k-1)}$ (Crescente)
LPT3	Decrescente	$\sum_{k=1}^{g} p_{jk} + s_{jk}$	$\sum_{k=1}^{g} p_{jk} + s_{jk}$	$\sum_{k=1}^{g} p_{jk} + s_{jk}$
LPT3_ERD	Decrescente	$\sum_{k=1}^{g} p_{jk} + s_{jk}$	$r_{jk} = C_{j(k-1)}$ (Crescente)	$r_{jk} = C_{j(k-1)}$ (Crescente)
RAND	Aleatória	Aleatório	Igual ao estágio k=1	Igual ao estágio k=1
RAND_ERD	Aleatório	Aleatório	$r_{jk} = C_{j(k-1)}$ (Crescente)	$r_{jk} = C_{j(k-1)}$ (Crescente)

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

ALGORITMO PARA FLOW SHOP HÍBRIDO COM SETUP INDEPENDENTE

PASSO 1. (*ORDENAÇÃO INICIAL*) No estágio k=1, sequencie as tarefas pela regra de prioridade escolhida no conjunto: {SPT1, LPT1, SPT2, LPT2, SPT3, LPT3, SPT1_ERD, LPT1_ERD, SPT2_ERD, LPT2_ERD, SPT3_ERD, LPT3_ERD}.

PASSO 2. (*ALOCAÇÃO INICIAL*) Aloque sequencialmente as tarefas no estágio k=1 na máquina de menor carga. 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*)

PASSO 4a. Se a regra de prioridade escolhida estiver entre {SPT1, LPT1, SPT3, LPT3}, considere a mesma sequência de tarefas do estágio k=1.

PASSO 4b. Se a regra de prioridade escolhida for {SPT2}:

– se *não* for o último estágio ($2 \geq k > g$), sequencie as tarefas pela ordem não decrescente da soma ($s_{j(k+1)} + p_{j(k+1)}$);

– se for o último estágio ($k=g$), sequencie as tarefas pela ordem não decrescente da soma ($s_{jg} + p_{jg}$).

PASSO 4c. Se a regra de prioridade escolhida for {LPT2}:

– se *não* for o último estágio ($2 \geq k > g$), sequencie as tarefas pela ordem não crescente da soma ($s_{j(k+1)} + p_{j(k+1)}$);

– se for o último estágio ($k=g$), sequencie as tarefas pela ordem não crescente da soma ($s_{jg} + p_{jg}$).

PASSO 4d. Se a regra de prioridade escolhida estiver entre {SPT1_ERD, LPT1_ERD, SPT2_ERD, LPT2_ERD, SPT3_ERD, LPT3_ERD}, sequencie as tarefas pela regra *ERD*.

PASSO 5. (*ALOCAÇÃO DOS ESTÁGIOS k≥2*) Aloque sequencialmente as tarefas no estágio k na máquina de menor carga, 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 E RESULTADOS

5.1 Método da pesquisa

Segundo as definições de Martins (2010, p.45-49) e Nakano (2010, p.64), esta pesquisa possui *abordagem quantitativa*, pois há preocupação 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.

5.2 Planejamento do experimento

Na experimentação computacional foram testados e avaliados 16.200 problemas, divididos em 162 classes definidas pelo número de tarefas (*n*), número de estágios de produção (*g*), níveis de flexibilidade (*f*), intervalos de tempos de *setup* (*s*) e probabilidade do *setup* ser antecipado (*a*).

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 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]; três opções de probabilidades do *setup* ser antecipado, 0%, 50% e 100% (RUIZ et al., 2006).

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 (LOGENDRAN et al., 2005). Para o cálculo do número de estágios com máquinas paralelas, foi utilizado o arredondamento. E para definir quais estágios teriam máquinas paralelas, foram utilizados valores inteiros uniformemente distribuídos.

Desta forma, o número de classes de problemas é 3 (*n*) * 3 (*g*) * 3 (*f*) * 2 (*s*) * 3 (*a*) = 162. Sendo 100 problemas por classe, a quantidade total de problemas é 16.200.

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 o ambiente de programação Delphi. As configurações da máquina são as seguintes: processador Pentium 4 da Intel com 3 GHz de frequência e 512 MB de memória RAM.

5.3 Análise dos resultados

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 quatorze métodos propostos.

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** (DR) mede a variação correspondente à melhor solução obtida pelos métodos. O melhor algoritmo é aquele que apresenta o menor valor de desvio relativo médio (a média aritmética dos desvios relativos) para uma determinada classe de problemas.

O desvio relativo (DR_h) de um método *h* para um determinado problema é assim calculado:

$$DR_h = \frac{C_{\max}^{h} - C_{\max}^{best}}{C_{\max}^{best}} \quad (5.1)$$

onde $C_{\max}^{h}$ é o *makespan* obtido pelo método *h* e $C_{\max}^{best}$ é o melhor *makespan* obtido pelos quatorze métodos.

O **desvio-padrão** de uma amostra mede o grau de dispersão dos elementos em torno da média. Neste trabalho, 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. Quanto menor for o valor do desvio-padrão, melhor é o método de solução quando comparado com outro, no caso em que ambos apresentarem desvios relativos médios com diferença não significativa. Além disso, quanto menor o desvio-padrão, mais estável é o método quanto ao seu desempenho.

O desvio-padrão (S_h) do desvio relativo de um método (*h*) é calculado da seguinte forma:

$$S_h = \sqrt{\frac{\sum_{i=1}^{L} (DR_{h_i} - DRM)^2}{L-1}} \quad (5.2)$$

onde *L* é o número de problemas da classe, DR_{h_i} é o desvio relativo da solução do problema *i* e *DRM* é o desvio relativo médio da classe de problemas.

O **tempo médio de computação** de um método é a média aritmética do tempo consumido para fornecer a solução, medido em milissegundos (ms).

Como pode ser observado na Tabela 2, na totalidade dos problemas resolvidos, o método SPT1_ERD obteve desempenho superior aos demais, atingindo 32,2% de sucesso. Em seguida, o método SPT1 alcançou 24,3% de sucesso.

Estas duas regras, além da grande diferença de desempenho em relação à ordenação aleatória (RAND), atingiram juntas 56,5% de sucesso, indicando a eficiência do sequenciamento pela ordem não-decrescente (SPT) da soma dos tempos de processamento e de *setup* do primeiro estágio. Esta análise conjunta das duas regras indica que, dentre as quatorze soluções encontradas, em 56,5% dos

problemas resolvidos, o melhor resultado foi obtido pela regra SPT1_ERD **ou** pela SPT1.

Tabela 2– Total das porcentagens de sucesso dos métodos

Método	% Sucesso
SPT1_ERD	32,2
SPT1	24,3
SPT2_ERD	19,3
LPT3_ERD	17,0
LPT3	9,7
SPT3_ERD	9,1
RAND_ERD	8,5
LPT2_ERD	7,2
RAND	5,0
SPT3	4,6
LPT1_ERD	1,4
LPT1	0,6
LPT2	0,5
SPT2	0,0

Como esperado, o desempenho do método RAND_ERD (8,5%) foi superior ao do RAND (5,0%) por considerar a sequência em que as tarefas são liberadas no estágio anterior. Todos os métodos que utilizam a ordenação baseada na regra ERD possuem melhor desempenho em relação ao método análogo que não utiliza tal regra. Principalmente na significativa diferença no resultado dos métodos SPT2_ERD e SPT2, isso demonstra a eficiência de se aplicar a regra ERD. Enquanto o método SPT2_ERD ficou em terceiro lugar com 19,3% de sucesso, o método SPT2 ficou em último lugar, fornecendo a melhor solução em apenas 3 classes do total de 162, ou seja, com 0,019% de sucesso.

O método LPT2 apresentou o segundo pior desempenho com apenas 0,5% de sucesso. Este resultado, aliado ao do método SPT2, indica a ineficiência de se ordenar pela soma dos tempos de processamento e de *setup* do estágio seguinte. Vale ressaltar a diferença no desempenho dos métodos quando aplicada a regra ERD. Enquanto SPT2 e LPT2 forneceram os piores resultados, os métodos SPT2_ERD e LPT2_ERD apresentaram resultados melhores inclusive que a regra aleatória RAND.

Com o aumento do número de tarefas e de estágios, os métodos não apresentaram grande variação no resultado, mantendo praticamente a mesma ordem de superioridade. O método SPT1_ERD piora o seu desempenho com o aumento do número de estágios, mas melhora com o aumento do número de tarefas. Em geral os métodos têm uma leve queda no desempenho com o aumento do porte do problema. A maior variação verificou-se na disputa pelo terceiro lugar com o aumento do número de estágios. Em problemas com 3 estágios, os resultados do método LPT3_ERD superam o SPT2_ERD. Nos casos em que há 5 e 7 estágios, o inverso acontece.

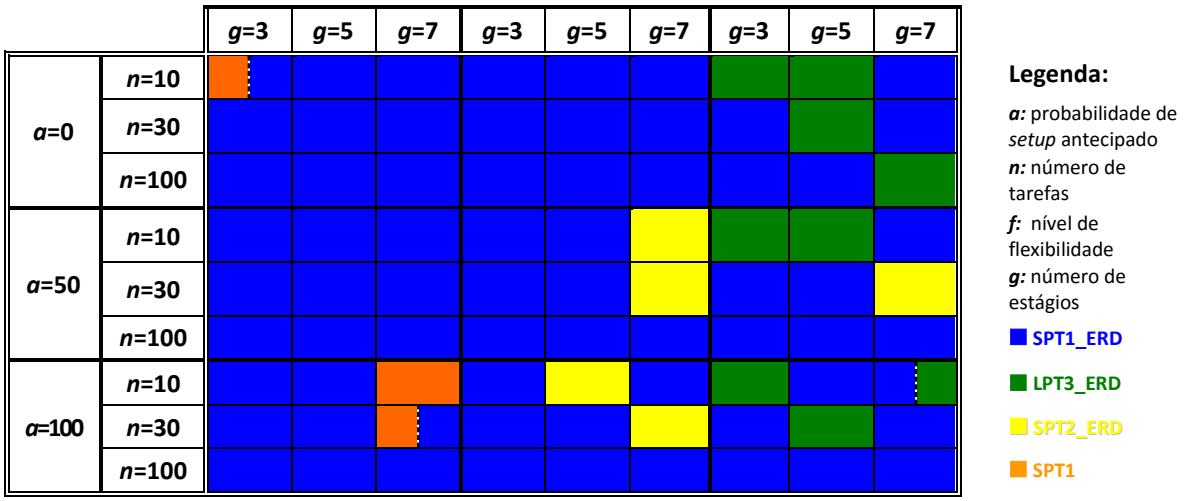
Em termos de flexibilidade do ambiente de produção, ou seja, o número de máquinas por estágio, nas opções baixa e média, manteve-se a ordem de superioridade conforme a Tabela 1. No alto nível de flexibilidade, em que todos os estágios possuem diferentes quantidades de máquinas paralelas idênticas, os métodos SPT1_ERD e LPT3_ERD disputam o primeiro e o segundo lugares, e os métodos SPT1 e SPT2_ERD disputam o terceiro e o quarto.

Houve uma pequena variação no resultado dos métodos (cerca de 2%) em relação às três opções de probabilidade do *setup* ser antecipado, 0%, 50% e 100%. Isso pode indicar que neste ambiente o fato do *setup* ser antecipado (100%) ou ter 50% de probabilidade de ser antecipado não necessariamente melhora duração total da programação em relação às tarefas cujos *setups* não podem ser antecipados (0%). Os dois intervalos de tempos de *setup* considerados tampouco tiveram diferenças significativas no desempenho dos métodos. Desta forma, verificou-se que tarefas com diferentes características de *setup* (tempos e opção antecipação) não afetaram a eficiência dos métodos.

A Tabela 3 apresenta uma comparação geral de desempenho mais detalhada com a variação do número de tarefas (*n*) e de estágios (*g*), a probabilidade do *setup* ser antecipado (*a*) e o nível de flexibilidade (*f*), indicando o melhor método para cada conjunto de opções. Pode-se observar claramente a predominância do método SPT1_ERD e a disputa com o método LPT3_ERD no caso de alta flexibilidade.

Tabela 3 – Resumo dos métodos com melhor desempenho em porcentagem de sucesso

***f* = B**	***f* = M**	***f* = A**



Os valores dos desvios relativos e desvio-padrão confirmam a análise feita para a porcentagem de sucesso dos métodos. O método SPT2, que forneceu o pior desempenho em termos de porcentagem de sucesso, teve também os mais altos valores e grande instabilidade no desvio relativo médio. A amplitude dos valores foi de mais de 300%, ou seja, a variação do *makespan* obtido por este método atingiu até quatro vezes mais do que o do melhor método. Os outros métodos variaram o desvio relativo médio tem torno de 10%.

Os valores do desvio-padrão ficaram em torno de 0,1%, demonstrando também grande estabilidade. No caso dos piores métodos, SPT2 e LPT2, o desvio-padrão ficou na faixa de 0,1 e 0,4%, proporcional ao porte do problema.

A informação interessante é que com exceção destes dois piores métodos, tanto os valores do desvio relativo como os do desvio-padrão reduzem com o aumento do porte do problema.

Portanto, parece plausível aplicar as regras de prioridade propostas como um método de solução do problema tratado. Estes procedimentos fornecem até a melhor das soluções com um custo de CPU desprezível (o tempo de computação foi no máximo 16 ms).

6 CONCLUSÃO

Neste trabalho, foi analisado o desempenho de quatorze métodos de programação em sistemas *flow shop* com múltiplas máquinas em que os tempos de

setup das máquinas são independentes da sequência de execução das tarefas. A medida de desempenho adotada foi a minimização do *makespan*.

O sequenciamento inicial aleatório (RAND), utilizado como base de comparação do desempenho entre as regras de prioridade propostas, dividiu os métodos em dois grupos: os oito melhores métodos e os cinco piores.

Dos quatorze métodos propostos, sete utilizaram o mesmo critério de sequenciamento em todos os estágios e os outros sete consideraram a regra ERD nos estágios posteriores ao primeiro. Tal regra permite maior flexibilidade na programação, pois considera a informação da ordem em que as tarefas são liberadas no estágio anterior. Em todos os casos, o método que utiliza a regra ERD forneceu desempenho superior ao método equivalente que não a utiliza.

Em geral, os métodos mantêm a ordem de superioridade, sendo os melhores SPT1_ERD, SPT1, SPT2_ERD e LPT3_ERD, e os piores SPT2, LPT2 e LPT1. Dos parâmetros de problemas utilizados, o nível de flexibilidade foi o que mais afetou o desempenho dos métodos, ainda assim somente no caso de alta flexibilidade. Isto pode indicar que diferentes características desse ambiente de produção e das tarefas não influenciam no resultado dos métodos de solução. Como esperado, o custo de CPU foi desprezível.

Para o desenvolvimento de futuros trabalhos, sugere-se a introdução de outras restrições no sistema, por exemplo, prazos de entrega e datas de liberação, tornando-o ainda mais realista, e o estudo das propriedades estruturais do problema com tempos de *setup* dependentes da sequência. Além disso, novas regras de prioridade e alocação podem ser comparadas e testes estatísticos mais aprofundados, utilizando ANOVA ou Kruskall-Wallis, por exemplo, poderiam ser conduzidos.

Nota

A pesquisa relatada neste artigo teve o apoio da CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior e do CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico. 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 flowshops 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 & Operations Research**, v. 27, p. 111-127, 2000. http:// doi:10.1016/S0305-0548(99)00010-6

ALLAHVERDI, A.; GUPTA, J.N.D.; ALDOWAISAN, T. A review of scheduling research involving setup considerations. **Omega – The International Journal of Management Science**, v. 27, p. 219-239, 1999. http://dx.doi.org/10.1016/S0305-0483(98)00042-5

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

BOTTA-GENOULAZ, V. Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. **International Journal of Production Economics**, v. 64, p. 101-111, 2000. http://dx.doi.org/10.1016/S0925-5273(99)00048-1

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.; RUIZ, R. Novas regras de prioridade para programação em *flexible flow line* com tempos de *setup* explícitos. **Production**, *ahead of print*, 2015. http://dx.doi.org/10.1590/0103-6513.089212

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

HUANG, W.; LI, S. A two-stage hybrid flowshop with uniform machines and setup times. **Mathematical and Computer Modeling**, v. 27, p. 27-45, 1998. http:// doi:10.1016/S0895-7177(97)00258-6

JUNG, C.F. **Metodologia para pesquisa & desenvolvimento**: aplicada a novas tecnologias, produtos e processos. Rio de Janeiro: Axcel Books, 2004.

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.; 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, 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**, 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

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

LOW, C. Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. **Computers & 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.

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.

QUADT, D.; KUHN, H. A taxonomy of flexible flow line scheduling procedures. **European Journal of Operational Research**, v. 178, p. 686-698, 2007. http://doi: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.ijpe.2004.04.013

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, n. 3, 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.

SLACK, N.; CHAMBERS, S.; HARLAND, C.; HARRISON, A; JOHNSON, R. **Administração da produção**. São Paulo: Atlas, 1999. 1. ed.

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 routing 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

Artigo recebido em 06/05/2014 e aceito para publicação em 16/09/2015
DOI: http://dx.doi.org/ 10.14488/1676-1901.v15i4.1791