Production, v. 25, n. 4, p. 779-790, out./dez. 2015
http://dx.doi.org/10.1590/0103-6513.089212

Novas regras de prioridade para programação em *flexible flow line* com tempos de *setup* explícitos

Hélio Yochihiro Fuchigami[a*], João Vitor Moccellin[b], Rubén Ruiz[c]

[a*]Universidade Federal de Goiás, Catalão, GO, Brasil, heliofuchigami@ufg.br
[b]Universidade Federal do Ceará, Fortaleza, CE, Brasil
[c]Universidade Politécnica de Valência, Valência, Espanha

Resumo

Neste artigo são propostos e avaliados 12 métodos para minimização da duração total da programação (*makespan*) em sistemas *flexible flow line* com tempos de *setup* independentes da sequência de execução das tarefas. Esse ambiente é caracterizado pela possibilidade de as tarefas saltarem um ou mais estágios de produção. Além disso, os tempos de *setup* podem ou não ser antecipados. Os desempenhos relativos dos métodos de solução foram avaliados por meio de experimentação computacional com base na porcentagem de sucesso, desvio relativo, desvio-padrão do desvio relativo e tempo médio de computação. Para avaliar-se a qualidade da solução dos métodos propostos, foi desenvolvido um limitante inferior (*lower bound*) para a função objetivo. Testes computacionais mostraram a maior eficácia do método que sequencia as tarefas no primeiro estágio, pela ordem decrescente da soma dos tempos de processamento e de *setup* de todos os estágios, e nos estágios seguintes, pela ordem em que as tarefas são liberadas para o processamento.

Palavras-chave

Programação da produção. *Flexible flow line*. *Setup* independente.

1. Introdução

Neste trabalho são apresentadas e comparadas 12 regras de sequenciamento, com a respectiva regra de alocação, para o problema de programação da produção em ambientes denominados *flexible flow line*. Esses sistemas são caracterizados por um fluxo unidirecional de tarefas por sucessivos estágios de produção compostos por uma ou mais máquinas paralelas idênticas. Em cada estágio, as tarefas devem ser processadas necessariamente por uma única máquina. O que diferencia os *flexible flow line* dos conhecidos *flow shop* híbridos é a possibilidade de as tarefas saltarem estágios.

Essa situação pode ocorrer em indústrias em que algumas tarefas não necessitem uma operação, como no caso da produção de placas de circuito impresso ou na indústria automotiva (Kurz & Askin, 2004). O 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 explicitamente (Garey et al., 1976; Kis & Pesch, 2005; Quadt & Kuhn, 2007a).

Neste estudo foram considerados explicitamente os tempos de *setup* ou preparação das máquinas, ou seja, separados dos tempos de processamento das tarefas. E o critério de desempenho é a minimização da duração total da programação (*makespan*).

O tempo de *setup* inclui todo o 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**. E, no segundo caso, o *setup* depende tanto da tarefa a ser processada como da que foi executada imediatamente antes na mesma máquina, sendo chamado **dependente da**

Recebido: Ago. 16, 2012; Aceito: Fev. 19, 2014

sequência (Allahverdi et al., 1999). Neste trabalho foi considerado o ambiente em que os tempos de *setup* são independentes da sequência de execução das tarefas.

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 & Chang, 2000). Entretanto, é mais realista assumir que em cada estágio de produção certo número de máquinas idênticas disponíveis possa operar de forma paralela, situação tradicionalmente conhecida como *flow shop* híbrido ou *flow shop* com múltiplas máquinas (Low, 2005).

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 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* nessa máquina pode ser realizado antes do término do processamento da tarefa na primeira máquina. Isso significa que uma medida de desempenho regular, caso do *makespan*, pode ser melhorada considerando os tempos de *setup* separados dos tempos de processamento (Allahverdi, 2000).

Assim, o problema tratado nesta pesquisa tem sua complexidade por três principais motivos:

- Ainda que o ambiente *flexible flow line* tenha sido abordado por alguns pesquisadores anteriormente, poucos consideraram os tempos de *setup* separados dos tempos de processamento das tarefas. Além disso, até o momento não foi encontrado nenhum trabalho na literatura com as mesmas características do ambiente em questão;
- Foram consideradas no mesmo problema ambas as opções de *setup* independente: antecipado e não antecipado. Isso abrange um maior número de situações práticas e aumenta a flexibilidade do ambiente estudado; e
- O sistema com máquinas paralelas nos estágios torna a análise mais realista, pois na prática os processos de produção podem ter diferentes números de máquinas em cada estágio. Isso não ocorria em alguns trabalhos publicados como, por exemplo, em Ruiz & Maroto (2006) e Ruiz et al. (2008), nos quais o número de máquinas em todos os estágios é sempre o mesmo.

2. Programação em sistemas *flexible flow line*

Muitas publicações já abordaram o problema da programação em *flow shops* híbridos. Linn & Zhang (1999) propuseram uma classificação das pesquisas nesse ambiente em três categorias: problemas com dois estágios, três estágios e mais de três estágios. Vignier et al. (1999) apresentaram para esse momento o estado da arte de *flow shops* híbridos.

Kis & 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 os métodos como de solução ótima, heurística e de inteligência artificial. Quadt & Khun (2007a) publicaram uma taxonomia para *flow shop* híbridos, porém denominando-os de *flexible flow line*.

Ruiz & 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 apresentarem as oportunidades de pesquisa na área. E uma extensa revisão dos trabalhos publicados recentemente (desde 1995) foi elaborada por Ribas et al. (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 et al., 2005). Como observaram Quadt & Kuhn (2007a), o sistema *flow shop* híbrido pode ser encontrado em um vasto número de indústrias, como química, eletrônica, de empacotamento, farmacêutica, automotiva, de embalagens de vidro, madeireira, têxtil, de herbicidas, alimentícia, de cosméticos e de semicondutores.

Várias pesquisas desse ambiente já foram reportadas, como as de Brah & Hunsucker (1991), Ding & Kittichartphayak (1994), Lee & Vairaktarakis (1994), Leon & Ramamoorthy (1997), Wittrock (1998) e Brah & Loo (1999).

Embora tenham sido feitos muitos trabalhos nessa área de programação da produção, muitas pesquisas se restringiram a casos especiais de dois estágios ou de configurações específicas de máquinas

nos estágios, ou então sem a presença de tempos de *setup* separados dos tempos de processamento.

Andrés et al. (2005) aplicaram procedimentos heurísticos para o problema de agrupamento de tarefas em indústrias de telha e obtiveram resultados positivos. O problema de dimensionamento de lote em um *flexible flow line* com custos de *setup* foi considerado por Quadt & Kuhn (2007b). O objetivo era minimizar os custos de *setup* e o tempo médio de fluxo.

Diversos trabalhos enfocaram ambientes multiestágio híbridos com tempos de *setup* dependentes da sequência, considerando diferentes técnicas de solução, como os de Kurz & Askin (2003, 2004), Tang & Zhang (2005), Ruiz & Maroto (2006), Zandieh et al. (2006), Tavakkoli-Moghaddam & Safaei (2007), Jenabi et al. (2007), Jungwattanakit et al. (2008), Ruiz et al. (2008) e Naderi et al. (2010).

Os trabalhos reportados na literatura envolvendo especificamente tempos de *setup* independentes da sequência estão relacionados a seguir.

O ambiente *flow shop* híbrido com dois estágios, sendo uma única máquina no primeiro e várias máquinas paralelas idênticas no segundo, com o critério de minimização do *makespan*, foi estudado por Gupta & Tunc (1994), Li (1997) e Huang & Li (1998).

Botta-Genoulaz (2000) estudou o problema de minimização do atraso máximo das tarefas sujeito ao mínimo tempo de transporte. Há também a presença de tempos de remoção das tarefas. O autor propôs e avaliou o desempenho de seis heurísticas. Allaoui & Artiba (2004) analisaram o problema *flexible flow shop* com tempos de transporte e várias medidas de desempenho, entre elas a minimização do *makespan* e do atraso máximo.

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

Low (2005) enfocou a minimização do tempo total de fluxo em problemas em que os estágios possuíam várias máquinas paralelas não relacionadas e tempos de remoção. Foi proposta uma heurística para gerar uma solução inicial e um algoritmo *simulated annealing* para melhorá-la. Heurísticas construtivas para minimização do *makespan* foram apresentadas por Logendran et al. (2005). As tarefas foram agrupadas em famílias. Foram considerados os tempos de *setup* dependentes das máquinas e independentes das famílias de tarefas.

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

3. Regras de prioridade propostas

Regras de prioridade, também conhecidas como **regras de sequenciamento** ou **regras de despacho**, são procedimentos de extrema importância na prática. São tecnicamente simples, fáceis de compreender e requerem pouco esforço para serem aplicadas. 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.

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 predeterminados para emitir ordens de compra, fabricação e montagem dos itens. Já nos sistemas puxados, normalmente são implementados *kanbans* para gerenciar a produção.

Por essas 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.

O problema consiste em programar um conjunto de n tarefas, definido como $J = \{1, ..., n\}$, que será processado em um conjunto de g estágios. Em cada estágio k, com $k = 1, ..., g$, 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. Pelo menos um dos estágios possui mais de uma máquina ($M_k > 1$). Todas as tarefas têm o mesmo fluxo, passando pelos estágios na mesma ordem e sendo processadas por apenas uma máquina em cada estágio. As tarefas têm uma probabilidade de saltar estágios. O tempo de processamento da tarefa j, $j \in J$, no estágio k, com $k = 1, ..., g$, 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 um intervalo de probabilidade. O *setup*

antecipado é aquele que pode ser realizado antes da liberação da operação no estágio anterior. A medida de desempenho utilizada é a minimização do *makespan*.

Para obter a solução desse problema de programação da produção foram definidas 12 regras de sequenciamento para as tarefas e políticas de alocação nas máquinas. As propostas baseiam-se nas conhecidas regras SPT (Shortest Processing Time) e LPT (Longest Processing Time) e algumas são adaptações de heurísticas apresentadas por Gupta & Tunc (1994) e Li (1997).

Do segundo estágio de produção em diante, quatro regras respeitam a mesma ordenação definida para o primeiro estágio; duas regras utilizam o mesmo critério de ordenação, porém podem gerar ordenações diferentes das tarefas a partir do segundo estágio, e as outras seis utilizam a regra ERD (Earliest Release Date) para ordenação das tarefas a partir do segundo estágio. A regra de prioridade ERD é equivalente à bem conhecida ordenação FIFO (First In First Out), em que as tarefas são programadas na ordem em que chegam ao estágio, ou seja, na ordem em que são concluídas no estágio anterior e, portanto, liberadas em fila para o processamento no estágio atual.

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.

É importante observar que **a regra de prioridade aplicada em um estágio k não altera no estágio k+1 a data de liberação das tarefas que não visitam esse estágio k**. Portanto, qualquer que seja a regra de prioridade aplicada no estágio 1, a data de liberação das tarefas que não o visitam será sempre zero no estágio 2.

Para cada tarefa *j* foi definido um valor utilizado na construção da sequência.

- **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 essa 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* no primeiro estágio ($p_{j1} + s_{j1}$) e, nos demais estágios, ordena pela regra ERD, ou seja, aloca as tarefas na sequência em que elas são liberadas no estágio anterior.
- **SPT2:** Em cada estágio *k*, 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 ($\Sigma^{g}_{k=1}[p_{jk} + s_{jk}]$).
- **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 essa 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 *k*, 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 ($\Sigma^{g}_{k=1}[p_{jk} + s_{jk}]$).
- **LPT3_ERD:** No primeiro estágio é análoga à regra LPT3 e, nos demais, considera a regra ERD.

Após o estabelecimento da ordenação em cada estágio, **cada tarefa é alocada sequencialmente à máquina de menor carga, que levará à menor data de término.** Isso 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.

Como essas 12 regras já incorporam tanto o sequenciamento como a alocação, doravante serão referenciadas como métodos de solução.

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

3.1. *Algoritmo para flexible flow line 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.

4. Experimentação computacional e resultados

4.1. *Delineamento do experimento*

Na experimentação computacional foram testados e avaliados 21.600 problemas, divididos em 216 classes definidas pelo número de tarefas (n), número de estágios de produção (g), níveis de flexibilidade (f), intervalos de tempos de *setup* (s), probabilidade de antecipação do *setup* (a) e probabilidade de a tarefa saltar um estágio (l).

Para cada classe, foram gerados aleatoriamente 100 problemas visando reduzir o erro amostral. Com base em trabalhos reportados na literatura, foram definidos os seguintes parâmetros de dados dos problemas: 10, 30 e 100 tarefas (Kurz & Askin, 2004); três, cinco e sete estágios (Logendran et al., 2005); intervalo fixo de tempos de processamento, U[1, 99]; dois intervalos de tempos de *setup*, U[25, 74] e U[75, 125]; dois intervalos de probabilidade do *setup* ser antecipado, U[0, 50]% e U[50, 100]%; duas opções de probabilidades da tarefa saltar o estágio, 0% e 50% (Ruiz et al., 2008).

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. 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-se, tem-se três estágios com máquinas paralelas e dois estágios com uma máquina. Para definir quais estágios terão as máquinas paralelas, foram utilizados valores inteiros uniformemente distribuídos. Os parâmetros e valores da experimentação computacional são resumidos na Tabela 1.

Dessa forma, o número de classes de problemas é 3 (n) * 3 (g) * 3 (f) * 2 (s) * 2 (a) * 2 (l) = 216. Sendo 100 problemas por classe, a quantidade total de problemas é 21.600.

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

Foi utilizado o sistema operacional Windows e a linguagem de programação Delphi. As configurações da máquina são as seguintes: processador Pentium 4 da Intel com 3 GHz de frequência e 512 MB de memória RAM.

Tabela 1. Parâmetros e valores da experimentação computacional.

Parâmetro	Símbolo	Número de níveis	Valores
Número de tarefas	n	3	10, 30, 100
Número de estágios	g	3	3, 5, 7
Flexibilidade	f	3	Baixa, Média, Alta
Distribuição dos tempos de *setup*	s	2	U[25, 74], U[75, 125]
Probabilidade de antecipação do *setup*	a	2	U[0, 50]%, U[50, 100]%
Probabilidade de uma tarefa saltar um estágio	l	2	0%, 50%

4.2. *Limitante inferior proposto*

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 12 métodos desenvolvidos. Além disso, para se averiguar a qualidade da solução, foi proposto um limitante inferior (*lower bound*) para o *makespan*.

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 de cada 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 da média aritmética dos desvios relativos para uma determinada classe de problemas.

O **desvio padrão** do desvio relativo mede o grau de dispersão ou a estabilidade dos métodos em torno do desvio relativo médio. E o **tempo médio de computação** de um método é a média aritmética do custo de CPU durante a execução, medido em milissegundos (ms).

O limitante inferior (LB) foi proposto com base nos trabalhos de Kurz & Askin (2004) e de Tavakkoli-Moghaddam & Safaei (2007), com as devidas adaptações para o problema tratado, uma vez que esses autores consideraram o problema com *setup* dependente e não antecipado. O limitante LB é composto por outros três limitantes, descritos a seguir.

O limitante LB_1 é baseado em tarefas (*job-based*) e calculado da seguinte forma:

$$LB_1 = \max_{j=1,\ldots,n} \left\{ s_{j1} + \sum_{k=1}^{g} p_{jk} \right\} \quad (1)$$

Os limitantes LB_2 e LB_3 são baseados em estágios (*stage-based*), considerando respectivamente, o primeiro estágio e os seguintes. Seu cálculo é apresentado de forma separada devido à peculiaridade de não se poder antecipar o *setup* do primeiro estágio:

$$LB_2 = \frac{\sum_{j=1}^{n} (s_{j1} + p_{j1})}{m_1} + \min_{j=1,\ldots,n} \sum_{k=2}^{g} p_{jk} \quad (2)$$

$$LB_3 = \max_{k=2,\ldots,g} \left\{ \frac{\sum_{j=1}^{n} (s_{jk} + p_{jk}) - \sum_{q=1}^{m_k} \max_{[q]\, j=1,\ldots,n} s_{jk}}{m_k} + \min_{j=1,\ldots,n} \left[s_{j1} + \sum_{h=1}^{k-1} p_{jh} \right] + \min_{j=1,\ldots,n} \sum_{h=k+1}^{g} p_{jh} \right\} \quad (3)$$

onde m_k é o número de máquinas do estágio k e $\max_{[q]}$ é o q-ésimo maior valor.

O LB_1 considera a tarefa com a maior soma dos tempos de processamento em todos os estágios e o tempo de *setup* no primeiro estágio, que é o único que não pode ser antecipado, ou seja, o tempo mínimo requerido para se processar a tarefa com a maior carga de trabalho.

Já o LB_2 avalia a carga média de trabalho do primeiro estágio (soma de todos os tempos de processamento e de *setup* dividida pelo número de máquinas do estágio) mais o menor tempo requerido para a menor tarefa concluir o seu processamento até o último estágio (apenas tempos de processamento, considerando a hipótese otimista de se ter todos os *setups* antecipados nos estágios $k \geq 2$).

E o LB_3 calcula, para os estágios k posteriores ao primeiro, a carga média de trabalho do estágio, considerando totalmente antecipados os m_k maiores tempos de *setup* do estágio, e somando-se o menor tempo requerido para a menor tarefa atingir o estágio k (soma dos tempos de processamento dos estágios 1 a $k-1$ mais o tempo de *setup* da tarefa no primeiro estágio) e o menor tempo necessário para a menor tarefa terminar o seu processamento até o último estágio (soma dos tempos de processamento dos estágios $k+1$ a g). Esse cálculo é feito para todos os estágios $k \geq 2$, considerando-se o maior valor dentre os calculados.

Finalmente, o LB é definido da seguinte forma:

$$LB = \max\left\{LB_1, LB_2, LB_3\right\} \quad (4)$$

Para se avaliar a qualidade da solução das heurísticas, o *makespan* obtido em cada um dos 12 métodos foi comparado com o LB descrito. Assim, o **índice de desvio percentual** (IDP) mede a variação percentual da solução em relação ao LB.

4.3. *Análise dos resultados*

O gráfico da Figura 1 mostra a comparação da porcentagem de sucesso entre os 12 métodos de solução por porte do problema **n** x **g** (número de tarefas e número de estágios), agregando os demais parâmetros.

Como pode ser observado na Figura 1, o método LPT3_ERD, além de apresentar o desempenho superior em todas as opções de porte de problemas, também melhora seu resultado com o aumento do número de tarefas, demonstrando sua eficácia para problemas de grande porte. Em geral, todos os outros métodos pioram o desempenho relativo com o aumento do porte do problema.

Claramente, em segundo lugar ficou o método SPT1_ERD, porém com desempenho descendente em relação ao aumento do porte do problema. E o terceiro lugar foi disputado pelos métodos LPT2_ERD, para problemas com três estágios, e SPT2_ERD, nas demais opções.

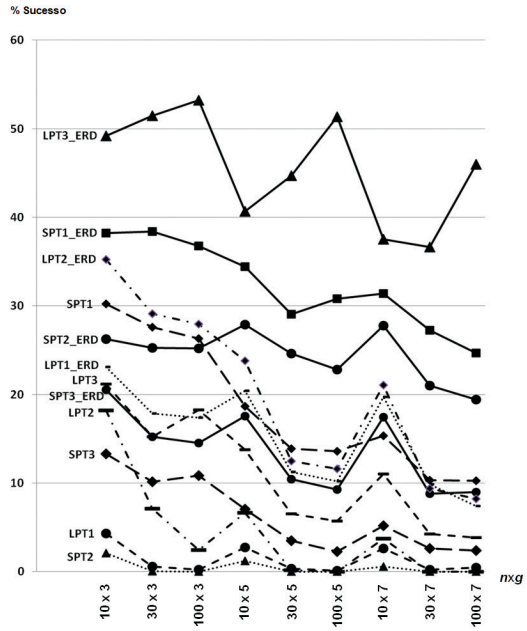

Figura 1. Comparação da porcentagem de sucesso dos métodos por porte do problema (número de tarefas e número de estágios).

Na totalidade dos problemas resolvidos, **o método LPT3_ERD obteve melhor desempenho relativo, atingindo 45,7% de sucesso**, conforme pode ser observado na Tabela 2. Em seguida, o método SPT1_ERD alcançou 32,3% de sucesso e o método SPT2_ERD apresentou 24,5% de sucesso. Os piores desempenhos relativos foram apresentados pelos métodos SPT2 (0,4%), LPT1 (1,3%) e LPT2 (4,3%).

A Tabela 2 também apresenta os índices de desvio percentual (IDP) em relação à qualidade da solução dos métodos. A regra LPT3_ERD comprovou a sua superioridade, tendo apenas 16,9% de desvio em relação ao limitante inferior.

Por definição, os limitantes inferiores são valores determinados apenas com base nos dados do problema (sem o sequenciamento e alocação), garantindo-se que sejam no máximo iguais ao *makespan* ótimo (C^*_{max}), ou seja, $LB \leq C^*_{max}$. Isso significa que em média a solução do método LPT3_ERD fica distante da solução ótima do problema em no máximo 16,9% do valor, um resultado expressivo tratando-se de um ambiente de produção com tamanha complexidade e dada a simplicidade da heurística baseada em regra de prioridade.

Com algumas poucas exceções, a ordem de superioridade da porcentagem de sucesso dos métodos se manteve também para o índice de desvio percentual em relação ao limitante inferior. A maior inversão ocorreu entre as regras LPT1 e LPT2 – embora a primeira tenha obtido pior desempenho na

Tabela 2. Porcentagens de sucesso e Índice de Desvio Percentual em relação ao limitante inferior.

Método	% Sucesso	IDP (%)
LPT3_ERD	45,7	16,9
SPT1_ERD	32,3	18,5
SPT2_ERD	24,5	20,7
LPT2_ERD	19,9	20,4
SPT1	18,5	22,2
LPT1_ERD	15,2	22,1
SPT3_ERD	13,6	22,3
LPT3	11,1	25,4
SPT3	6,4	26,7
LPT2	4,3	98,1
LPT1	1,3	38,4
SPT2	0,4	135,6

porcentagem de sucesso (1,3% contra 4,3% da LPT2), por outro lado atingiu o menor desvio em relação ao limitante inferior (38,4% contra 98,1% da LPT2).

Ao se comparar para cada problema o resultado do melhor método com o limitante inferior, o desvio percentual médio considerando todos os 21.600 problemas foi de 14,0%. Esses resultados endossam as análises das medidas consideradas e comprovam a aplicabilidade prática dos métodos propostos.

O método que obteve o pior índice relativo percentual coincidiu com o pior método na porcentagem de sucesso: a regra SPT2 teve apenas 0,4% de sucesso e altíssimos 135,5% de desvio em relação ao limitante inferior.

Pode-se verificar claramente que aplicar a regra ERD do segundo estágio em diante é mais viável do que manter a mesma regra de prioridade aplicada no primeiro estágio. Comparando-se os pares de regras análogas (a regra simples e a que utiliza a ERD), em todos os casos o desempenho da aplicação da ERD predomina.

No estudo da influência dos parâmetros do problema, o número de estágios mostrou-se relevante, o que também pode ser visto na Figura 1, pois com o aumento do número de estágios houve certa variabilidade não linear no desempenho relativo dos métodos.

Já os intervalos de tempos de *setup* e os intervalos das probabilidades de antecipação do *setup* não afetaram de forma significativa o desempenho relativo dos métodos. Como pode ser visto na Tabela 3, as diferenças das porcentagens de sucesso de cada método para os intervalos de *setup* considerados, $U[25,74]$ e $U[75,125]$, chegaram a no máximo 5,2 pontos percentuais (na regra LPT2_ERD em problemas com três estágios e 10 tarefas). De forma geral, a diferença média em valores absolutos foi de apenas 1,1 ponto percentual.

A Tabela 4 apresenta as diferenças da porcentagem de sucesso de cada método para a probabilidade de antecipação do *setup* em *U*[0, 50]% e *U*[50, 100]%. Novamente, os resultados mostraram-se bastante próximos em ambas as opções desse parâmetro, tendo diferenças de no máximo 4,0 pontos percentuais (na regra SPT2_ERD com sete estágios e 30 tarefas). No geral, a diferença média em valores absolutos foi de 0,9 ponto percentual. Isso pode indicar que nesse ambiente uma maior probabilidade de antecipação do *setup* não necessariamente melhoraria o resultado dos métodos.

Os diferentes níveis de flexibilidade do sistema, ou seja, a variação na quantidade de máquinas por estágio, geram variabilidade no desempenho relativo dos métodos de acordo com o porte do problema, como pode ser visto nas Tabelas 5a, b e c. As diferenças de um valor para outro foram bem maiores do que as apresentadas nas tabelas dos parâmetros anteriores, chegando a 28,0 pontos percentuais (na Tabela 5b, com a regra LPT2_ERD em problemas com três estágios e 100 tarefas).

Analisando-se conjuntamente as Tabelas 5a, b e c, nota-se que os métodos que tiveram maior variabilidade foram SPT1_ERD e LPT2_ERD, com médias em valores absolutos próximos de 8,5 pontos percentuais. Para esses métodos, as grandes diferenças são mais visíveis na Tabela 5b, entre as flexibilidades média e alta (em ambos, as médias ficaram em quase 13%).

Os métodos SPT2 e LPT1 foram os que obtiveram as menores diferenças, respectivamente, com 0,4 e 0,9 ponto percentual (em valores absolutos).

A probabilidade de as tarefas saltarem estágio foi o parâmetro que mais afetou o resultado dos métodos, como pode ser observado na Tabela 6, com diferenças que chegam a 77,3 pontos percentuais, no método LPT3_ERD com cinco estágios e 100 tarefas. A média geral das diferenças para as duas opções de probabilidade de salto das tarefas ficou em 13,8 pontos percentuais.

O método LPT3_ERD foi o que mais sofreu variabilidade, ao passar de 0% de probabilidade de salto para 50%, tendo 56,7 pontos percentuais de diferença em valores absolutos médios. Por outro lado, novamente os métodos SPT2 e LPT1 tiveram as menores diferenças, respectivamente com 0,8 e 1,3 ponto percentual.

Este estudo demonstra que tanto a configuração do ambiente de produção, representada pelo número de estágios e o nível de flexibilidade, como as características das tarefas, especialmente a possibilidade

Tabela 3. Diferenças (em pontos percentuais) da porcentagem de sucesso de cada método para os **intervalos de setup** de *U*[25,74] e *U*[75,125] e médias em valores absolutos.

g	*n*	SPT1	SPT1_ERD	SPT2	SPT2_ERD	SPT3	SPT3_ERD	LPT1	LPT1_ERD	LPT2	LPT2_ERD	LPT3	LPT3_ERD
	10	−0,2	0,6	−0,8	2,2	−3,4	−0,8	−1,2	−0,3	−0,6	−5,2	−1,8	0,4
3	30	2,8	1,2	0,1	2,0	0,3	1,5	−0,7	−3,3	−0,1	0,9	0,4	2,6
	100	3,3	3,9	0,0	−1,6	1,2	2,6	−0,4	−1,1	0,8	−0,4	−1,3	1,7
	10	1,0	−0,9	−0,4	3,6	0,4	0,8	0,3	−1,5	1,7	−1,3	0,6	2,2
5	30	−1,1	0,7	0,0	1,6	0,5	−0,3	−0,8	0,8	0,2	−0,9	−2,3	1,6
	100	−1,0	2,7	0,0	−1,9	0,1	0,7	0,0	0,4	0,0	−1,8	−0,1	1,9
	10	0,5	0,4	−0,5	0,8	−1,1	0,1	−0,3	−0,6	0,1	−2,2	1,8	1,6
7	30	0,5	1,5	0,0	1,2	0,6	0,6	−0,4	−0,1	0,0	−1,7	−0,5	0,2
	100	−0,5	−2,4	0,0	1,4	−0,5	0,7	0,1	−0,8	0,0	0,6	−0,7	1,7
Média		**1,2**	**1,6**	**0,2**	**1,8**	**0,9**	**0,9**	**0,5**	**1,0**	**0,4**	**1,7**	**1,1**	**1,5**

Tabela 4. Diferenças (em pontos percentuais) da porcentagem de sucesso de cada método para **intervalos de probabilidade de antecipação do setup** em *U*[0, 50]% e *U*[50, 100]% e médias em valores absolutos.

g	*n*	SPT1	SPT1_ERD	SPT2	SPT2_ERD	SPT3	SPT3_ERD	LPT1	LPT1_ERD	LPT2	LPT2_ERD	LPT3	LPT3_ERD
	10	−1,5	−2,1	0,6	0,2	−1,9	−1,6	1,2	−0,5	1,8	−1,2	1,7	1,1
3	30	2,0	0,8	0,1	−2,2	−0,3	−0,8	−0,2	−0,1	1,3	1,3	−0,6	−1,9
	100	2,2	−0,8	0,0	0,9	−0,3	−2,4	−0,3	−0,3	−0,8	−0,6	0,5	0,3
	10	−2,2	1,1	0,3	3,3	−0,6	1,7	0,8	0,5	−0,3	−0,8	0,9	−0,7
5	30	−2,3	−1,0	0,0	0,8	0,8	0,3	−0,4	0,0	−0,2	3,6	−1,3	1,3
	100	−0,7	−0,5	0,0	−0,8	−0,4	−1,7	0,2	−1,4	0,0	0,6	0,3	−1,6
	10	1,2	0,3	0,0	−2,0	0,3	−0,8	0,1	−2,9	0,1	1,8	1,5	2,1
7	30	−1,3	−1,3	0,0	4,0	−0,1	−1,6	0,3	0,3	0,0	0,7	0,0	−2,0
	100	1,2	−0,8	0,0	1,8	−0,5	0,0	0,4	0,7	0,0	−0,3	0,7	−1,5
Média		**1,6**	**1,0**	**0,1**	**1,8**	**0,6**	**1,2**	**0,4**	**0,7**	**0,5**	**1,2**	**0,8**	**1,4**

Tabela 5a. Diferenças (em pontos percentuais) da porcentagem de sucesso de cada método entre **flexibilidades baixa e média** e médias em valores absolutos.

g	*n*	SPT1	SPT1_ERD	SPT2	SPT2_ERD	SPT3	SPT3_ERD	LPT1	LPT1_ERD	LPT2	LPT2_ERD	LPT3	LPT3_ERD
	10	2,1	−1,1	−2,4	−1,9	−7,0	−10,0	−3,4	−6,6	−8,0	−3,0	−9,1	−4,9
3	30	4,8	3,9	−0,1	0,0	−6,6	−9,9	−1,3	−3,8	−4,8	−2,5	−13,4	−5,6
	100	3,6	5,9	0,0	1,9	−11,5	−9,3	−0,5	−4,0	−6,0	−2,1	−16,6	−9,6
	10	4,3	3,9	0,1	−4,9	0,6	−2,3	0,1	−2,8	−3,3	−3,6	−1,1	0,3
5	30	4,3	5,4	0,0	−0,1	−0,9	−2,8	−0,3	−2,5	−0,1	−3,8	−0,3	−1,5
	100	3,6	4,0	0,0	−0,6	−2,3	−2,9	0,3	−1,3	0,0	−4,1	−0,6	0,8
	10	3,9	1,1	−0,3	1,1	−2,0	−6,3	0,0	−3,6	−2,6	−5,0	−1,8	−4,5
7	30	9,3	8,1	0,0	−4,1	0,4	−3,5	0,1	−3,6	0,0	−7,6	−2,1	−6,6
	100	7,0	6,4	0,0	−7,1	0,1	−1,9	−0,3	−7,1	0,0	−9,5	0,0	−7,1
Média		**4,8**	**4,4**	**0,3**	**2,4**	**3,5**	**5,4**	**0,7**	**3,9**	**2,8**	**4,6**	**5,0**	**4,5**

Tabela 5b. Diferenças (em pontos percentuais) da porcentagem de sucesso de cada método entre **flexibilidades média e alta** e médias em valores absolutos.

g	*n*	SPT1	SPT1_ERD	SPT2	SPT2_ERD	SPT3	SPT3_ERD	LPT1	LPT1_ERD	LPT2	LPT2_ERD	LPT3	LPT3_ERD
	10	14,5	19,3	0,6	13,5	13,5	16,1	−1,4	8,1	5,6	12,0	1,6	5,0
3	30	18,4	24,4	0,1	23,3	15,4	19,4	1,1	20,6	10,6	27,5	13,8	15,9
	100	15,0	20,8	0,0	23,4	18,6	18,8	0,4	23,4	6,5	**28,0**	11,9	17,8
	10	1,1	7,4	−2,8	7,1	3,5	4,6	−5,1	0,9	−9,4	−2,4	−7,3	−4,0
5	30	6,5	9,6	0,0	16,4	2,5	7,3	0,5	10,3	0,1	11,0	−0,4	3,1
	100	7,9	12,8	0,0	14,6	3,3	6,5	0,0	13,1	0,0	15,0	−2,4	1,9
	10	−0,1	3,8	−1,3	−0,4	2,6	4,4	−1,9	−1,6	−5,5	−0,8	−4,4	−5,0
7	30	3,8	5,9	0,0	8,3	2,3	5,8	0,3	6,9	0,0	7,6	2,0	0,8
	100	5,9	9,1	0,0	9,5	2,3	4,9	0,3	7,8	0,0	9,8	−0,6	2,3
Média		**8,1**	**12,5**	**0,5**	**12,9**	**7,1**	**9,7**	**1,2**	**10,3**	**4,2**	**12,7**	**4,9**	**6,2**

Tabela 5c. Médias gerais em valores absolutos das diferenças (em pontos percentuais) da porcentagem de sucesso de cada método entre as **opções de flexibilidade.**

Média geral	SPT1	SPT1_ERD	SPT2	SPT2_ERD	SPT3	SPT3_ERD	LPT1	LPT1_ERD	LPT2	LPT2_ERD	LPT3	LPT3_ERD
	6,4	**8,3**	**0,4**	**7,4**	**5,2**	**7,5**	**0,9**	**6,9**	**3,4**	**8,4**	**5,0**	**5,3**

Tabela 6. Diferenças (em pontos percentuais) da porcentagem de sucesso de cada método para **probabilidades de salto** de 0% e 50% e médias em valores absolutos.

g	*n*	SPT1	SPT1_ERD	SPT2	SPT2_ERD	SPT3	SPT3_ERD	LPT1	LPT1_ERD	LPT2	LPT2_ERD	LPT3	LPT3_ERD
	10	−2,3	−10,9	−3,9	−18,0	−10,1	−13,9	−3,7	−36,3	−29,9	−33,7	−14,3	−52,1
3	30	8,0	−1,3	−0,1	−10,3	−10,2	−10,5	0,2	−32,4	−13,3	−25,6	−9,4	−67,8
	100	8,2	−5,1	0,0	−14,6	−10,0	−11,8	0,4	−33,3	−4,9	−26,3	−9,2	−70,2
	10	10,2	−6,6	−2,3	−13,6	−2,6	−15,7	−3,2	−32,8	−13,0	−29,5	−3,8	−43,8
5	30	22,8	10,0	0,0	3,1	0,8	−3,5	0,4	−19,5	−0,3	−16,8	3,9	−55,4
	100	25,2	19,0	0,0	5,1	1,8	−3,2	0,2	−17,4	0,0	−16,3	2,8	**−77,3**
	10	6,7	−5,1	−1,2	−9,2	0,6	−13,1	−2,3	−31,1	−7,4	−26,7	−4,2	−35,8
7	30	19,0	15,2	0,0	6,5	2,8	−1,8	0,4	−15,1	0,0	−8,8	3,3	−44,2
	100	20,3	19,8	0,0	7,8	2,7	2,2	0,9	−10,7	0,0	−7,9	5,0	−63,5
Média		**13,6**	**10,3**	**0,8**	**9,8**	**4,6**	**8,4**	**1,3**	**25,4**	**7,6**	**21,3**	**6,2**	**56,7**

de saltar estágios, influenciam significativamente no resultado dos métodos propostos.

Como consta na Tabela 7, os valores dos desvios relativos e desvio padrão confirmaram a análise feita para a porcentagem de sucesso dos métodos, ou seja, de forma geral, os métodos com maior porcentagem de sucesso tiveram os menores desvios e os com menor porcentagem de sucesso, os maiores desvios. Isso indica que quanto maior a porcentagem de sucesso do método, melhor a qualidade da solução em relação aos demais e maior a sua estabilidade.

Os melhores métodos, LPT3_ERD e SPT1_ERD, tiveram, respectivamente, 2,4% e 3,8% de desvio relativo médio e desvio padrão de 0,03 e 0,04.

Tabela 7. Comparação do desvio relativo médio (em %) e do desvio padrão dos métodos (em ordem da porcentagem de sucesso).

Método	% Sucesso	DR	DP
LPT3_ERD	45,7	2,4	0,03
SPT1_ERD	32,3	3,8	0,04
SPT2_ERD	24,5	5,7	0,06
LPT2_ERD	19,9	5,5	0,05
SPT1	18,5	7,0	0,05
LPT1_ERD	15,2	7,0	0,06
SPT3_ERD	13,6	7,0	0,06
LPT3	11,1	9,7	0,07
SPT3	6,4	10,7	0,07
LPT2	4,3	75,4	0,21
LPT1	1,3	21,1	0,11
SPT2	0,4	108,8	0,24

Já os métodos SPT2 e LPT2 apresentaram os mais altos valores de desvios, respectivamente, 108,8% e 75,4% de desvio relativo e 0,24 e 0,21 de desvio padrão. Ou seja, são os de maior instabilidade da solução. Isso indica que sequenciar tanto pela ordem crescente como decrescente da soma dos tempos de processamento e de *setup* do estágio seguinte causa uma grande variabilidade no resultado do método, com uma diferença significativa em relação à melhor solução.

Os resultados de LPT3_ERD e SPT1_ERD, os dois melhores métodos, somam juntos 78% de sucesso; descontando-se os empates, eles atingem 66% de sucesso. Isso significa que se na resolução de cada problema for considerado o melhor valor da função objetivo desses dois métodos essa solução será a melhor dentre os 12 métodos em pelo menos 66% dos casos.

E se o mesmo processo for feito com as três melhores regras, incluindo-se aqui a SPT2_ERD, o desempenho conjunto dos problemas avaliados, já descontados os empates, somam 87,6% de sucesso. Ou seja, facilmente e sem grandes custos computacionais adicionais poder-se-ia alterar o algoritmo proposto neste trabalho de forma a que ele testasse o resultado dessas três melhores regras considerando o melhor entre elas e obter-se-ia assim um método com 87,6% de sucesso e com desvio médio em relação ao limitante inferior reduzido para 14,7%.

Por fim, é importante salientar que o tempo médio de CPU foi praticamente zero (0,011 ms). O maior tempo gasto na resolução de um problema foi de 16 ms. Por esses resultados, parece bastante plausível aplicar as regras de prioridade propostas como métodos de solução para o problema tratado, pois mesmo a melhor das soluções foi obtida com um custo de CPU desprezível.

5. Considerações finais

Neste trabalho foi analisado o desempenho de 12 métodos de programação em ambientes *flexible flow line* caracterizados por múltiplos estágios de produção em que cada um contém uma ou mais máquinas em paralelo. Nesse ambiente, as tarefas podem saltar um ou mais estágios de produção e os tempos de *setup* são independentes da sequência e podem ser antecipados ou não.

A medida de minimização do *makespan* foi comparada por meio da porcentagem de sucesso, desvio relativo médio, desvio padrão e tempo médio de computação. Foi também proposto um limitante inferior para o *makespan,* visando avaliar a qualidade da solução dos métodos.

No geral, o método LPT3_ERD forneceu os melhores resultados, principalmente para problemas de grande porte, com 45,7% de sucesso, desvio em relação ao limitante inferior de apenas 16,9% e uma significativa diferença em relação ao desempenho das demais regras. Esse método prioriza no primeiro estágio as tarefas com a maior soma total dos tempos de processamento e *setup* e nos estágios seguintes sequencia as tarefas de acordo com a ordem em que são liberadas. Em segundo lugar, com 32,3% de sucesso, ficou o método SPT1_ERD, que prioriza no primeiro estágio as tarefas com menor soma dos tempos de processamento e *setup* do estágio e nos seguintes ordena-as segundo a sua liberação.

Os métodos com piores desempenhos foram SPT2, LPT1 e LPT2, que juntos não alcançaram 6% de sucesso, indicando que não é vantajoso sequenciar segundo os tempos de processamento e *setup* do estágio seguinte (sendo a ordenação crescente ou decrescente) ou pela ordem decrescente da soma dos tempos de processamento e *setup* do primeiro estágio.

Foi confirmada a eficácia da regra ERD ou FIFO, que ordena do segundo estágio em diante de acordo com a liberação das tarefas. Não se mantendo a mesma ordenação em todos os estágios, a programação torna-se mais flexível.

Os diferentes valores de alguns parâmetros, como os dois intervalos de tempo de *setup* e as duas opções de probabilidades de antecipação do *setup,* não afetaram diretamente o desempenho relativo dos métodos (as porcentagens de sucesso ficaram muito próximas nas duas opções de valor de cada um desses dois parâmetros). Já o número de estágios, os níveis de flexibilidade e a probabilidade de a tarefa saltar estágio influenciaram significativamente. Dessa forma, verificou-se que tanto a configuração do ambiente (*layout*) como a característica das tarefas são relevantes para o método de programação em *flexible flow line*.

Os valores dos desvios relativos e do desvio-padrão confirmaram a análise feita para a porcentagem de sucesso, revelando também que os piores métodos apresentam grande instabilidade na solução. E, como esperado, o custo de CPU foi desprezível.

Para o desenvolvimento de futuros trabalhos, sugere-se o estudo das propriedades do ambiente com tempos de *setup* dependentes da sequência e a introdução de outras restrições no sistema, tornando-o ainda mais realista. Também poderão ser comparadas novas regras de prioridade e alocação, com uma análise mais profunda e detalhada da influência da probabilidade de as tarefas saltarem estágios.

Referências

Aldowaisan, T. (2001). A new heuristic and dominance relations for no-wait flowshop with setups. *Computer and Operations Research, 28*, 563-584.

Allahverdi, A. (2000). Minimizing mean flowtime in a two-machine flowshop with sequence-independent setup times. *Computers and Operations Research, 27*, 111-127.

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

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

Andrés, C., Albarracín, J. M., Tormo, G., Vicens, E., & García-Sabater, J. P. (2005). Group technology in a hybrid flowshop environment: A case study. *European Journal of Operational Research, 167*, 272-281.

Botta-Genoulaz, V. (2000). Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. *International Journal of Production Economics*, *64*(1-3), 101-111.

Brah, S. A., & Hunsucker, J. L. (1991). Branch and bound algorithm for the flow shop with multiple processors. *European Journal of Operational Research, 5*(1), 88-99.

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

Ding, F. Y., & Kittichartphayak, D. (1994). Heuristics for scheduling flexible flow lines. *Computers and Industrial Engineering, 26*, 27-34.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research, 1*(2), 117-129.

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

Huang, W., & Li, S. (1998). A two-stage hybrid flowshop with uniform machines and setup times. *Mathematical and Computer Modeling, 27*(2), 27-45.

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

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

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

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

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

Lee, C-Y., & Vairaktarakis, G. L. (1994). Minimizing makespan in hybrid flowshops. *Operations Research Letters, Amsterdam, 16*(3), 149-158.

Leon, V. J., & Ramamoorthy, B. (1997). An adaptable problem-space-based search method for flexible flow line scheduling. *IIE Transactions, 29*, 115-125.

Li, S. (1997). A hybrid two-stage flowshop with part family, batch production, and major and minor set-ups. *European Journal of Operational Research, 102*, 142-156.

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

Liu, C-Y., & Chang, S-C. (2000). Scheduling flexible flow shops with sequence-dependent setup effects. *IEEE Transactions on Robotics and Automation, 16*(4), 408-419.

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

Low, C. (2005). Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. *Computers and Operations Research, 32*, 2013-2025.

Naderi, B., Ruiz, R., & Zandieh, M. (2010). Algorithms for a realistic variant of flowshop scheduling. Computers & Operations Research. *Computers & Operations Research, 37*(2), 236-246.

Quadt, D., & Kuhn, H. (2007a). A taxonomy of flexible flow line scheduling procedures. *European Journal of Operational Research*, 178(3), 686-698. http://dx.doi.org/10.1016/j.ejor.2006.01.042.

Quadt, D., & Kuhn, H. (2007b). Batch scheduling of jobs with identical process times on flexible flow lines. *International Journal of Production Economics*, 105(2), 385-401. http://dx.doi.org/10.1016/j.ijpe.2004.04.013.

Ribas, I., Leisten, R., & Framiñan, J. M. (2010). Review and classifications of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research, 37*, 1439-1454.

Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research, 169*, 781-800.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research, 205*, 1-18.

Ruiz, R., Şerifoğlu, F. S., & Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research, 35*, 1151-1175.

Tang, L. X., & Zhang, Y. Y. (2005). Heuristic combined artificial neural networks to schedule hybrid flow shop with sequence dependent setup times. *Lecture Notes in Computer Science, 3496*, 788-793.

Tavakkoli-Moghaddam, R., & Safaei, N. (2007). *A new mathematical model for flexible flow lines with blocking processor and sequence-dependent setup time. Multiprocessor scheduling: theory and applications* (pp. 255-272). Viena: ARS Publishing, Ed. Vedran Kordic.

Vignier, A., Billaut, J. C., & Proust, C. (1999). Les problèmes d'ordonnancement de type flow-shop hybride: état de l'art. *RAIRO – Recherche Opérationnelle, 33*(2), 117-183.

Wang, H. (2005). Flexible flow shop scheduling: optimum, heuristic and artificial intelligence solutions. *Expert Systems, 22*(2), 78-85.

Wilson, A. D., King, R. E., & Hodgson, T. J. (2004). Scheduling non-similar groups on a flow line: multiple group setups. *Robotics and Computer-Integrated Manufacturing, 20*, 505-51.

Wittrock, R. J. (1988). An adaptable scheduling algorithm for flexible flow lines. *Operations Research, 36*(4), 445-453.

Zandieh, M., Fatemi Ghomi, S. M. T., & Moattar Husseini, S. M. (2006). An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. *Applied Mathematics and Computation, 180*, 111-127.

Agradecimentos

A pesquisa relatada neste artigo teve o apoio da Capes – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior. Os resultados parciais foram apresentados no XXXIX Simpósio Brasileiro de Pesquisa Operacional. Os autores agradecem as preciosas sugestões e contribuições dos revisores.

New priority rules for the flexible flow line scheduling problem with setup times

Abstract

This paper presents twelve methods for makespan minimization for flexible flow line scheduling problems. This environment is characterized by the ability of jobs to skip stages. Sequence-independent setup times, which can be either anticipatory or non-anticipatory, are also considered. The statistics used to evaluate the heuristic performances were the rate of success (in finding the best solution), the relative deviation, the standard deviation of the relative deviation and the average computation time. A lower bound for makespan was developed to evaluate the solution quality of the proposed methods. The computational tests proved the effectiveness of the heuristic based on the initial sequence using the descending order of the sum of the processing and setup times of all stages and the sequencing of subsequent stages by the order of the job release times.

Keywords

Production scheduling. Flexible flow line. Sequence-independent setup times.