SOBRAPO

PODes
Revista Eletrônica Pesquisa Operacional para o Desenvolvimento

Rio de Janeiro, v.6, n.2, p. 318-331, maio a agosto de 2014

INFLUÊNCIA DA RELAÇÃO ENTRE OS TEMPOS DE PROCESSAMENTO E DE *SETUP* EM *FLOW SHOP* HÍBRIDOS

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

[a]*Universidade Federal de Goiás - UFG, Catalão – GO, Brasil*
[b]*Universidade Federal do Ceará - UFC, Fortaleza - CE, Brasil*

Resumo

O foco deste trabalho é o estudo da influência da relação entre as ordens de grandeza dos tempos de processamento das tarefas e os tempos de preparação das máquinas (*setup*) nos ambientes industriais denominados *flow shop* híbridos. No problema considerado, os tempos de preparação são assimétricos e dependentes da sequência de execução das tarefas. A função-objetivo é a minimização da duração total da programação (*makespan*). Este estudo pode ser classificado como "pesquisa aplicada" quanto à natureza, "pesquisa exploratória" quanto aos objetivos e "pesquisa experimental" quanto aos procedimentos, além da abordagem "quantitativa". Os resultados dos métodos heurísticos construtivos propostos são comparados com seis relações de tempos de processamento e de setup. As análises demonstraram que existe uma variabilidade relevante no resultado dos métodos dependendo da relação entre os tempos de processamento e setup, sobretudo em problemas de grande porte.

Palavras-Chave: Programação da produção. *Flow shop* híbrido. Setup dependente. Métodos heurísticos.

Abstract

This paper deals with the Hybrid Flow Shop Scheduling problem with the objective of minimizing the total time to complete the schedule, that is, the makespan. The paper main feature is to evaluate the influence of the ratio between job processing times and set up times on the makespan. Having this in mind it is proposed a heuristic method to solve the scheduling problem which compares the performance of six ratios concerning job processing times and set up times. Experimental results from a computational experience have shown a significant variability of the makespan depending on the used ratio, especially for large size problems.

Keywords: Scheduling. Hybrid flow shop. Sequence-dependent setup time. Heuristics.

***Autor para correspondência:** e-mail: heliofuchigami@yahoo.com.br

Versão inicial submetida em 25/07/2013. Versão final recebida em 13/05/2014.

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. Apenas recentemente alguns trabalhos têm tratado dos ambientes com tempos de *setup* explícitos.

Os *flow shop* híbridos são caracterizados por um fluxo unidirecional de tarefas que passam 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. Em essência, este sistema consiste em dois subproblemas: associar operações nas máquinas, ou seja, um problema de alocação, e sequenciar as operações em cada máquina, isto é, um problema de sequenciamento.

Este tipo de ambiente é relativamente comum e pode ser encontrado em indústrias de eletrônicos, condutores 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).

Muitas pesquisas em programação da produção desconsideram os tempos de preparação das máquinas ou então os incluem nos tempos de processamento das tarefas. Isto simplifica a análise das aplicações, porém afeta a qualidade da solução quando tais tempos têm uma variabilidade relevante em função da ordenação das tarefas nas máquinas, como é o caso de indústrias químicas, gráficas, têxteis e de papel, caracterizadas por sistemas com amplo *mix* de produtos.

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. E no segundo caso, o *setup* depende tanto da tarefa a ser processada como também da que foi executada imediatamente antes na mesma máquina, como é o caso desta pesquisa, sendo chamado dependente da sequência, (ALLAHVERDI, GUPTA e ALDOWAISAN, 1999).

O objetivo básico deste trabalho é avaliar a influência da relação entre as ordens de grandeza dos tempos de processamento das tarefas e os tempos *setup* das máquinas na programação da produção em ambientes *flow shop* híbridos. Os tempos de *setup* considerados

nesta pesquisa são assimétricos e dependentes da sequência de execução das tarefas. O critério de desempenho é a minimização do *makespan* (duração total da programação).

2 Programação da produção em *flow shop* híbrido

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

Muitos trabalhos publicados já abordaram o problema de programação em *flow shop* híbrido, desde que foi introduzido por Salvador (1973). Vignier, Billaut e Proust (1999) apresentaram o estado da arte para aquele momento para *flow hops* híbridos. Linn e Zhang (1999) propuseram uma classificação das pesquisas em três categorias: problemas com dois estágios, três estágios e 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 (2007a) 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 e HANSON, 2005). Como observou Quadt e Kuhn (2007a), 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.

Entretanto, embora tenham sido feitos muitos trabalhos nesta 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.

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 (2007b), Jungwattanakit *et al.* (2008), Ruiz, Şerifoğlu e Urlings (2008), Naderi, Ruiz e Zandieh (2010).

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

3 Heurísticas construtivas propostas

O ambiente de produção *flow shop* híbrido é ilustrado na Figura 1.

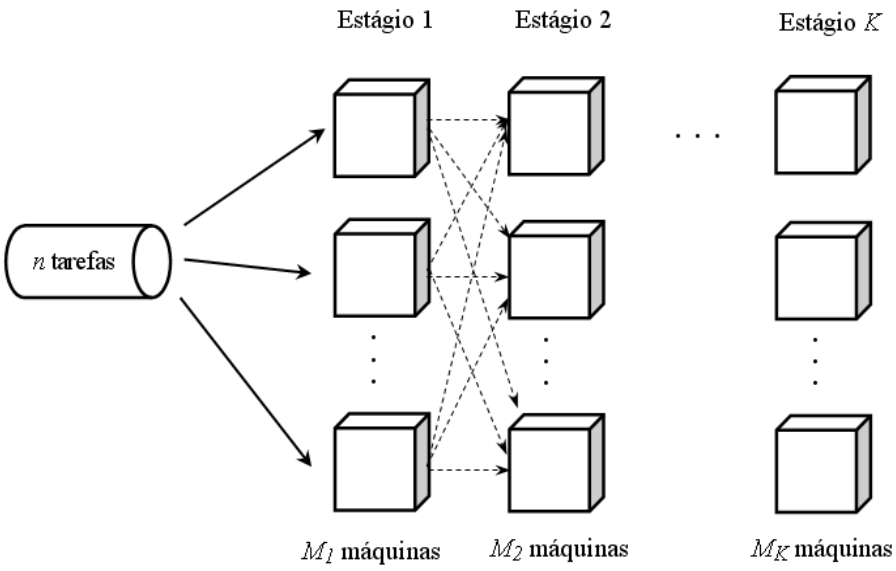

Figura 1: Ilustração do ambiente de produção

O problema consiste em programar um conjunto de n tarefas, definido como $J = \{J_1, ..., J_n\}$, onde cada tarefa possui necessariamente uma única operação em cada estágio de produção.

As operações das tarefas devem ser realizadas sequencialmente e passam por todos os estágios. O objetivo do problema é minimizar o *makespan* como medida de desempenho.

Para obter a solução desse problema de programação da produção, foram desenvolvidos e avaliados quatro métodos heurísticos construtivos com base em algoritmos reportados na literatura para solução de problemas *flow shop* permutacional (SIMONS JR., 1992) e máquinas paralelas (WENG, LU e REN, 2001) no ambiente cujo tempo de *setup* é dependente da sequência de execução das tarefas. Dois procedimentos, cada um executado separadamente com duas respectivas regras, perfazem os quatro métodos de solução.

O mecanismo usado na resolução do problema foi a definição de estratégias para programação das tarefas estágio a estágio, como solução iterativa de K problemas relacionados. O primeiro estágio (k = 1) é programado como se fosse um problema tradicional de M_1 máquinas paralelas idênticas com a mesma data de liberação. Os estágios seguintes consistem de K – 1 problemas consecutivos de M_k máquinas paralelas idênticas com diferentes datas de liberação das tarefas, correspondentes às datas de término das operações das respectivas tarefas no estágio imediatamente anterior.

Os algoritmos dos quatro métodos de solução são apresentados a seguir.

3.1 Procedimento 1 – estabelecimento de ordenação inicial

O Procedimento 1 define uma ordenação inicial de tarefas e utiliza duas regras de prioridade: a *Longest Processing Time* (LPT) e um método baseado no algoritmo TOTAL de Simons Jr. (1992). Desta forma, as regras de prioridade deste procedimento são denominadas LPT e TOTAL, respectivamente.

A regra LPT considera a soma dos tempos de processamento de cada tarefa em todos os estágios e faz o sequenciamento das tarefas pela ordem não-crescente destas somas.

A regra de prioridade TOTAL baseia-se na heurística de Simons Jr. (1992) para programação de *flow shop* permutacional com tempos de *setup* dependentes da sequência.

O cálculo das datas de término das tarefas J_i em um determinado estágio k é feito utilizando a seguinte expressão:

$$C_{E_{ik}} = \min_{m=1}^{M_k} \left\{ \max \left\{ C_{u_m k} + s_{u_m ik}; C_{i,k-1} \right\} + p_{ik} \right\} \qquad (3.1)$$

com i = 1, ..., n e k = 1, ..., K.

$C_{E_{ik}}$ representa a menor data de término da tarefa J_i no estágio k, M_k é o número de máquinas no estágio k, u_m é o índice da última tarefa alocada na máquina m (J_{u_m}), $C_{u_m k}$ denota a data de término da tarefa J_{u_m} no estágio k, $s_{u_m ik}$ é o tempo de *setup* da tarefa J_i no estágio k após o processamento de J_{u_m} , $C_{i,k\text{-}1}$ denota a data de término da tarefa J_i no estágio k-1 e p_{ik} é o tempo de processamento da tarefa J_i no estágio k.

3.1.1 Algoritmos do Procedimento 1 – Regras LPT e TOTAL

Seja J um conjunto de n tarefas a serem programadas, J' o conjunto das tarefas ainda não programadas, $J_{[j]}$ a tarefa que ocupa a j-ésima posição de J', M o número total de máquinas considerando todos os estágios de produção ($M = \sum_{k=1}^{K} M_k$, onde M_k é o número de máquinas no estágio k) e σ_m o subconjunto de tarefas alocadas à máquina m. Seja r_i a data de liberação da tarefa J_i e SRD (*Shortest Release Date*) a ordenação das tarefas segundo os valores não-decrescentes de r_i.

PASSO 1 (INICIALIZAÇÃO)

$\sigma_m = \varnothing$, para $m = 1, 2, ..., M$

$J' \leftarrow J$

Ordene as tarefas do conjunto J' de acordo com a regra de prioridade (LPT ou TOTAL).

Vá para o PASSO 3.

PASSO 2

$J' \leftarrow J$

Ordene as tarefas do conjunto J' de acordo com a ordenação SRD, considerando as datas de término das tarefas no estágio anterior como as datas de liberação do estágio atual.

PASSO 3

A primeira tarefa de J' será alocada à máquina x, cuja programação possuirá a data mais cedo de término.

PASSO 4 (ATUALIZAÇÃO DOS CONJUNTOS E TESTE DE PARADA)

$\sigma_x \leftarrow \sigma_x \cup \{ J'_{[1]} \}$

$J' \leftarrow J' - \{ J'_{[1]} \}$

Se $J' \neq \varnothing$, vá para o PASSO 3.

Caso contrário, programação do estágio concluída.

Se não for o último estágio, vá para o PASSO 2.

Caso contrário PARE, programação concluída. Saída: σ_m para $m = 1, 2... M$.

3.2 Procedimento 2 – sem estabelecimento de ordenação inicial

O Procedimento 2 não utiliza ordenação inicial para as tarefas e uma das regras de alocação emprega uma adaptação do melhor entre sete algoritmos para máquinas paralelas com *setup* dependente apresentados por Weng, Lu e Ren (2001). A outra regra de alocação deste procedimento também utiliza a adaptação do algoritmo citado, mas quando todas as tarefas estiverem liberadas respeita a ordenação pela regra LPST (*Longest Processing-Setup Time*), envolvendo a soma do tempo de processamento com o tempo de *setup*. As regras de alocação deste procedimento são designadas SCT (*Shortest Completion Time*) e SCT/LPST, respectivamente.

Como não há estabelecimento de uma ordenação inicial de tarefas, a regra SCT estabelece que a alocação é feita por meio de uma expressão que analisa todas as possibilidades de combinação tarefa-máquina e escolhe o par com a data de término mais cedo. Em cada estágio *k*, a seguinte expressão é calculada:

$$C_{E_{ik}} = \min\left\{C_{u_m k} + s_{u_m ik} + p_{ik} \mid m \in \mu_k\right\} \qquad (3.2)$$

com $i = 1,..., n$ e $k = 1, ..., K$, onde μ_k é o conjunto dos índices das máquinas do estágio *k* e as variáveis são as mesmas da expressão 3.1.

A regra de alocação SCT/LPST segue o mesmo procedimento da regra SCT, porém quando todas as tarefas estiverem liberadas, respeita a ordenação pela regra LPST. Considerando a máquina de menor carga, a tarefa alocada é aquela com o maior valor da soma dos tempos de *setup* e de processamento $(s_{u_m ik} + p_{ik})$.

3.2.1 Algoritmo do Procedimento 2 – Regra SCT

PASSO 1 (INICIALIZAÇÃO)

$\sigma_m = \varnothing$, para $m = 1, 2, ..., M$

PASSO 2

$J' \leftarrow J$

PASSO 3

A partir da primeira data com tarefas liberadas, calcule a data de término de cada uma das tarefas de *J'* já liberadas, em cada uma das máquinas do estágio, ou seja, analise todas as possibilidades de associação tarefa-máquina, e escolha o par tarefa J_i e máquina *x* com a data de término mais cedo.

PASSO 4 (ATUALIZAÇÃO DOS CONJUNTOS E TESTE DE PARADA)

$\sigma_x \leftarrow \sigma_x \cup \{J_i\}$

$J' \leftarrow J' - \{J_i\}$

Se $J' \neq \varnothing$, vá para o PASSO 3.

Caso contrário, programação do estágio concluída.

Se não for o último estágio, vá para o PASSO 2.

Caso contrário PARE, programação concluída. Saída: σ_m para $m = 1, 2... M$.

3.2.2 Algoritmo do Procedimento 2 – Regra SCT/LPST

PASSO 1 (INICIALIZAÇÃO)

$\sigma_m = \varnothing$, para $m = 1, 2, ..., M$

PASSO 2

$J' \leftarrow J$

PASSO 3

Se todas as tarefas estiverem liberadas, vá para o PASSO 4.

Caso contrário, a partir da primeira data com tarefas liberadas, calcule a data de término de cada uma das tarefas de J' já liberadas, em cada uma das máquinas do estágio, ou seja, analise todas as possibilidades de associação tarefa-máquina, e escolha o par tarefa J_i e máquina x com a data de término mais cedo. Vá para o PASSO 5.

PASSO 4

Na máquina de menor carga, associe a tarefa J_i com o maior valor de $(s_{u_m ik} + p_{ik})$, ou seja, observe a regra LPST.

PASSO 5 (ATUALIZAÇÃO DOS CONJUNTOS E TESTE DE PARADA)

$\sigma_x \leftarrow \sigma_x \cup \{J_i\}$

$J' \leftarrow J' - \{J_i\}$

Se $J' \neq \varnothing$, vá para o PASSO 3.

Caso contrário, programação do estágio concluída.

Se não é o último estágio, vá para o PASSO 2.

Caso, contrário PARE, programação concluída. Saída: σ_m para $m = 1, 2... M$.

Observação: nos algoritmos dos dois procedimentos, considera-se que uma máquina qualquer, ao receber a primeira tarefa, já esteja preparada para sua execução.

4 Experimentação computacional

4.1 Método da pesquisa

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

4.2 Delineamento do experimento

O foco da experimentação computacional foi a análise da influência da relação entre as ordens de grandeza dos tempos de processamento e de *setup* das tarefas, denotada por $\boldsymbol{O(p_i)/O(s_{ij})}$, no desempenho dos métodos desenvolvidos. Por este motivo, foram definidas seis relações, como consta na Tabela 1.

Tabela 1: Relações entre as ordens de grandeza dos tempos de processamento e de *setup*

Relação	Notação	Intervalo p_i	Intervalo s_{ij}
Relação I	$O(p_i)/O(s_{ij}) = 1$	1-99	1-99
Relação II	$O(p_i)/O(s_{ij}) < 1$	1-99	100-120
Relação III	$O(p_i)/O(s_{ij}) > 1$	10-99	1-9
Relação IV	$O(p_i)/O(s_{ij}) > 1$	50-99	1-49
Relação V	$O(p_i)/O(s_{ij}) \leq 1$	1-99	1-120
Relação VI	$O(p_i)/O(s_{ij}) \geq 1$	1-99	1-20

A geração dos tempos de processamento e de *setup* dentro de cada intervalo foi feita de forma aleatória com distribuição uniforme. Os limites dos intervalos das relações foram definidos e padronizados com base em trabalhos reportados na literatura (SIMONS JR., 1992;

RAJENDRAN e ZIEGLER, 1997; RÍOS-MERCADO e BARD, 1998 e 1999; DAS, GUPTA e KHUMAWALA, 1995; WENG, LU e REN, 2001).

Na experimentação computacional foram testados 14.400 problemas, divididos em 144 classes definidas pelo número de tarefas $n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120\}$, número de estágios de produção $K \in \{4, 7\}$ e pelas seis relações entre as ordens de grandeza dos tempos de processamento e de *setup* ($O(p_i)/O(s_{ij})$). Para cada classe, foram gerados aleatoriamente 100 problemas, com o objetivo de reduzir o erro amostral.

Em cada estágio, o número de máquinas paralelas idênticas varia de 2 a 5, ou seja, $M_k \in \{2, 3, 4, 5\}$. Como M_k é gerado aleatoriamente com distribuição uniforme dentro deste conjunto, sua variação não altera a quantidade de classes.

Para simplificação da notação, o Procedimento 1 com a regra de prioridade LPT foi denotado como "11" e com a regra de prioridade TOTAL como "12". O Procedimento 2 com a regra de alocação SCT foi denominado "21" e com a regra SCT/LPST foi indicado como "22".

4.3 Análise dos resultados

Tendo em vista o objetivo básico deste trabalho, os resultados obtidos na experimentação computacional foram analisados por meio da porcentagem de sucesso dos quatro métodos desenvolvidos para cada classe de problemas. 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.

Como não foram encontrados na literatura métodos heurísticos construtivos para programação do ambiente tratado neste trabalho, os algoritmos desenvolvidos foram comparados entre si.

Na totalidade dos problemas resolvidos, o método 21 apresentou a melhor solução 7.130 vezes num total de 14.400 problemas, correspondendo a 49,5% de sucesso. Em segundo lugar, o método 12 obteve a melhor solução em 4.531 problemas, equivalente a 31,5% de sucesso. Em seguida, o método 11 forneceu a melhor solução 2581 vezes, ou seja, atingiu 17,9% de sucesso. E, por último, 232 problemas tiveram a melhor solução pelo método 22, com 1,6% de sucesso.

A Tabela 2 apresenta uma comparação geral do desempenho dos métodos em todas as relações $O(p_i)/O(s_{ij})$ e para 4 e 7 estágios com base na porcentagem de sucesso. Essa tabela

recomenda um determinado método em função do número de tarefas, número de estágios e relação $O(p_i)/O(s_{ij})$.

Tabela 2: Resumo do desempenho dos métodos em termos de porcentagem de sucesso

	K = 4						K = 7					
n	RI	RII	RIII	RIV	RV	RVI	RI	RII	RIII	RIV	RV	RVI
10	21	21	21	21	21	11	21	11	21	21	21	21
20	21	21	21	21	21	21	21	21	21	21	21	21
30	21	21	21	21	21	21	21	21	21	21	21	21
40	21	21	21	21	21	21	21	21	21	21	21	21
50	12 21	21	21	21	12	21	21	21	21	21	12	21
60	21	21	21	12	12	21	21	21	21	12	12	21
70	12	21	21	12	12	21	12	21	21	12	12	21
80	12	21	21	12	12	21	12	21	21	12	12	21
90	12	21	21	12	12	21	12	21	21	12	12	21
100	12	21	21	12	12	21	12	21	21	12	12	21
110	12	21	21	12	12	21	12	12	21	12	12	21
120	12	21	21	12	12	21	12	21	21	12	12	21

Em geral, o desempenho dos problemas com 4 e 7 estágios é semelhante, podendo indicar que o número de estágios não afeta o desempenho de um método.

O método 21, em que o procedimento não utiliza ordenação inicial e observa a regra de alocação SCT, de maneira geral forneceu os melhores resultados, ou seja:

- ✓ Para todas as relações $O(p_i)/O(s_{ij})$ nos problemas de 10 a 40 tarefas, com exceção dos problemas de 10 tarefas para as relações II e VI;
- ✓ Para as relações II, III e VI, nos demais problemas de 50 a 120 tarefas, com exceção dos problemas de 110 tarefas para a relação II;

O gráfico da Figura 2 mostra que, com o aumento do número de tarefas, o método 21 melhora o desempenho nas relações III e VI, piora nas relações I, IV e V, e apresenta certa variação na amplitude do desempenho na relação II (de 1 a 25%). Quanto maior o número de tarefas, maior é variação do desempenho do método para as relações $O(_{pi})/O(s_{ij})$.

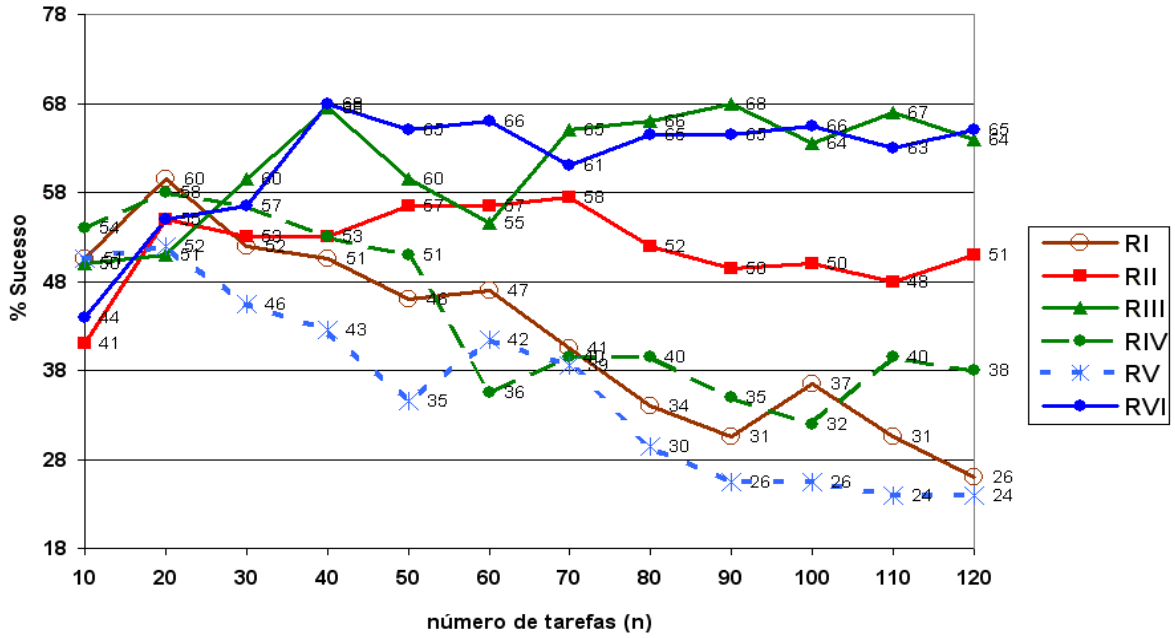

Figura 2: Comparação da porcentagem de sucesso do método 21 entre as relações $O(p_i)/O(s_{ij})$ juntando resultados de problemas com 4 e 7 estágios

5 Considerações finais

Na literatura, encontram-se poucas pesquisas sobre o desenvolvimento de métodos heurísticos para a programação de *flow shop* híbrido com *setup* dependente da sequência de processamento das tarefas. Neste trabalho, foram propostos quatro métodos heurísticos construtivos para solução de problemas no ambiente citado. Foi apresentado um estudo sobre a influência da relação entre as ordens de grandeza dos tempos de processamento das tarefas e dos de *setup* das máquinas com base no melhor entre os quatro métodos propostos. Existe uma variabilidade relevante no desempenho dos métodos dependendo da relação $O(p_i)/O(s_{ij})$, com o aumento do número de tarefas.

Para o desenvolvimento de futuros trabalhos, sugere-se um estudo "estrutural" do ambiente de produção *flow shop* híbrido com tempos de preparação das máquinas separados dos tempos de processamento das tarefas (dependentes e/ou independentes da sequência de execução), buscando o desenvolvimento de novos métodos heurísticos eventualmente com melhores desempenhos do que os propostos neste trabalho.

Agradecimentos

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.

Referências

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

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

Chen, Y.-Y.; Cheng, C-Y.; Wang, L.-C. & Chen, T.-Z. (2013). 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, 141, 66-78.

Das, S.R.; Gupta, J.N.D. & Khumawala, B.M. (1995). A saving index heuristic algorithm for flowshop scheduling with sequence dependent set-up times. Journal of Operational Research Society, 46, 1365-1373.

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

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. International Journal of Operational Research, 159, 66-82.

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

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

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

Martins, R.A. (2010). 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, 45-61.

Nakano, D. (2010). 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, 63-72.

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

Quadt, D. & Kuhn, H. (2007a). A taxonomy of flexible flow line scheduling procedures. European Journal of Operational Research, 178, 686-698.

Quadt, D. & Kuhn, H. (2007b). Batch scheduling of jobs with identical process times on flexible flow lines. International Journal of Production Economics, 105, 385-401.

Rajendran, C. & Ziegler, H. (1997). A heuristic for scheduling to minimize the sum of weighted flowtime of jobs in a flowshop with sequence-dependent setup times of jobs. Computers & Industrial Engineering, 33, 1-2, 281-284.

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

Ríos-Mercado, R.Z. & Bard, J.F. (1998). Heuristics for the flow line problem with setup costs. European Journal of Operational Research, 110, 76-98.

Ríos-Mercado, R.Z. & Bard, J.F. (1999). An enhanced TSP-based heuristic for makespan minimization in a flow shop with setup times. Journal of Heuristics, 5, 53-70.

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, 3, 781-800.

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

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

Salvador, M.S. (1973). 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, 83-91.

Simons Jr., J.V. (1992). Heuristics in flow shop scheduling with sequence dependent setup times. Omega – The International Journal of Management Scienc, 20, 2, 215-225.

Vairaktarakis, G. (2004). Flexible Hybrid Flowshops. In: Leung, J.Y-T. Handbook of Scheduling: algorithms, models, and performance analysis. San Diego: Chapman & Hall/CRC Press, 5.1-5.33.

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

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

Weng, M.X.; Lu, J. & Ren, H. (2001). Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. International Journal of Production Economics, 70, 3, 215-226.

Weng, W.; Wei, X. & Fujimura, S. (2012). Dynamic routings strategies for JIT production in hybrid flow shops. Computers & Operations Research, 39, 3316-3324.