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

# MÉTODOS HEURÍSTICOS PARA O PROBLEMA DE PROGRAMAÇÃO *FLOW SHOP* COM TEMPOS DE *SETUP* SEPARADOS

# HEURISTICS METHODS FOR THE FLOW SHOP SCHEDULING PROBLEM WITH SEPARATED SETUP TIMES

Marcelo Seido Nagano* E-mail: drnagano@usp.br
Marco Stabilito Mesquita* E-mail: marco.stabilito@gmail.com
* Universidade de São Paulo, USP, São Carlos, SP

**Resumo:** Neste artigo apresentam-se métodos heurísticos para o problema de programação da produção flow shop permutacional com tempos de setup das máquinas separados dos tempos de processamento das tarefas. A partir de investigações das características do problema, quatro métodos heurísticos foram propostos com procedimentos de construção da seqüência de solução em analogia com o problema assimétrico do caixeiro-viajante, tendo como objetivo a minimização da duração total da programação da produção. Os resultados da experimentação computacional mostraram que um dos novos métodos heurísticos propostos obtém soluções de alta qualidade em comparação com os métodos avaliados considerados na literatura.

**Palavras-chave:** Programação da produção. Flow shop permutacional. Tempos de setup. Método heurístico. Duração total da programação.

**Abstract:** This paper deals with the permutation flow shop scheduling problem with separated machine setup times. As a result of an investigation on the problem characteristics, four heuristics methods are proposed with procedures of the construction sequencing solution by an analogy with the asymmetric traveling salesman problem with the objective of minimizing makespan. Experimental results show that one of the new heuristics methods proposed provide high quality solutions in comparisons with the evaluated methods considered in the literature.

**Keywords**: Production scheduling. Permutation flow shop. Setup times. Heuristic method. Makespan.

## 1 INTRODUÇÃO

O problema de programação da produção *flow shop* é constituído de *n* tarefas a ser processadas, na mesma seqüência, em um conjunto de *m* máquinas distintas. Um caso específico de programação *flow shop*, denominado permutacional, é quando em cada máquina mantém-se a mesma ordem de processamento das tarefas.

A solução do problema consiste em determinar, dentre as (*n*!) seqüências possíveis de tarefas, aquela que otimiza alguma medida de desempenho da programação, sendo que as usuais consistem na minimização da Duração Total da

Programação (*makespan*) ou minimização do Tempo Médio de Fluxo. A primeira relaciona-se a uma utilização eficiente dos recursos (máquinas), enquanto que a segunda busca minimizar o estoque em processamento.

Esse problema de programação da produção tem sido intensamente estudado na literatura desde os resultados reportados por Johnson (1954) para o problema com somente duas máquinas. Hejazi & Saghafian (2005) efetuaram uma ampla revisão da literatura referente ao ambiente de produção *flow shop*, mostrando que a maioria das pesquisas efetuadas considera os tempos de preparação (*setup*) das máquinas não significativos ou os incluem nos tempos de processamento das tarefas. Isto simplifica a análise das aplicações, mas afeta a qualidade da programação quando tais tempos têm uma variabilidade relevante em função da ordenação das tarefas nas máquinas ou quando as mesmas podem ser preparadas antecipadamente para execução das tarefas, não necessitando aguardar o término destas nas máquinas precedentes.

Recentemente, o *European Journal of Operational Research* editou um número especial em comemoração aos 50 anos de publicação do primeiro artigo sobre programação da produção *flow shop* (JOHNSON, 1954). No *Editorial paper* dessa edição, Gupta & Stafford Jr. (2006) procuram mostrar a evolução dos problemas e métodos de solução em ambiente *flow shop* nas últimas 5 décadas. Nesse contexto, pode-se observar que a consideração de problemas com um tratamento explícito dos tempos de preparação das máquinas praticamente teve início a partir da terceira década (1975-1984), com um aumento progressivo de interesse por parte dos pesquisadores a partir da quarta década. Essa tendência de intensificação de esforços em pesquisas abordando tais problemas encontra-se detalhada na revisão da literatura efetuada por Cheng, Gupta & Wang (2000). No artigo, os autores apresentam uma classificação dos problemas de programação da produção *flow shop* com tempos de *setup* e um conjunto de trabalhos relevantes realizados. Na conclusão do mesmo são identificados alguns temas com potencial para futuras pesquisas, entre os quais o problema com múltiplos estágios (número de máquinas m > 2), sistemas de produção *no-wait* e modelos com medidas de desempenho multi-critério.

A consideração explícita dos tempos de preparação das máquinas é necessária em sistemas de produção como o que ocorre na produção de tintas em

indústrias químicas, onde o processo de limpeza é diferenciado dependendo da cor que estava sendo produzida e da que será fabricada em seguida (SIMONS JR., 1992). Tal ambiente de produção também é comum em indústrias gráficas, têxteis e de produtos plásticos (DAS, GUPTA & KHUMAWALA, 1995). Existem basicamente dois tipos de tempos de preparação de máquinas tratados separadamente dos tempos de processamento das tarefas. O primeiro depende somente da tarefa a ser executada (independente da seqüência). O segundo depende tanto da tarefa a ser executada quanto daquela que foi processada imediatamente antes na mesma máquina (dependente da seqüência).

Este trabalho trata do problema *flow shop* permutacional com tempos de preparação das máquinas separados dos tempos de processamento das tarefas, sendo dependentes da seqüência de execução das tarefas. Será denotado por FSP-TPS (*flow shop problem – time processing separated setup times*) e tem como objetivo apresentar e avaliar novos métodos heurísticos para problemas de programação da produção. Na literatura, o problema é classificado como fortemente *NP-hard* (*non-deterministic polynomial-time hard*).

Para o problema clássico de programação da produção *flow shop* permutacional, com os tempos de *setup* incluídos nos tempos de processamento das tarefas, Garey, Johnson & Sethi (1976) provaram que, quando a medida de desempenho for o tempo médio de fluxo, o problema é fortemente *NP-hard* para um ambiente com somente duas máquinas. A mesma complexidade ocorre na minimização da duração total da programação (*makespan*) para o ambiente de produção com três máquinas. Conclui-se, portanto, que para o caso geral com um número de máquinas superior a dois, tais problemas também são fortemente *NP-hard*. Pinedo (1995) apresenta uma estrutura hierárquica de complexidade envolvendo as denominadas Medidas de Desempenho Regulares, duas das quais mencionadas acima (tempo médio de fluxo e *makespan*). Na referida estrutura, essas duas medidas de desempenho ocupam os níveis de menor complexidade, o que permite concluir que a complexidade fortemente *NP-hard* é mantida para qualquer medida de desempenho regular. Para finalizar, a consideração explícita dos tempos de *setup* separados dos tempos de processamento das tarefas, eleva o problema a um nível superior de complexidade (PINEDO, 1995). Pode-se concluir, portanto, que o problema FSP-TPS aqui tratado é fortemente *NP-hard*.

Neste trabalho, reportam-se sucintamente os resultados finais de uma pesquisa, sendo o mesmo divido nas seguintes seções: duas subseções iniciais apresentando o problema de *flow shop* com tempos de preparação independentes e dependentes da seqüência das tarefas; uma seção apresentando os métodos heurísticos para o problema; uma seção referente ao planejamento da experimentação computacional seguida de outra de análise dos resultados e finalizando com uma seção de considerações finais. Os métodos propostos buscarão verificar as características essenciais de um método heurístico: adequado equilíbrio entre a qualidade da solução e esforço computacional, simplicidade e facilidade de implementação.

## 1.1 Problema FSP-TPS com tempos de preparação independentes da sequência das tarefas

O exame da literatura indica que o primeiro trabalho para o caso do problema FSP-TPS foi desenvolvido por Yoshida & Hitomi (1979) para um *flow shop* com apenas duas máquinas, com o objetivo de minimizar o *makespan*. Os autores mostraram que o problema poderia ser resolvido com solução ótima, aplicando-se a tradicional regra de Johnson (1954). Diversos outros trabalhos, com minimização do *makespan* e solução ótima, surgiram posteriormente para o mesmo caso de duas máquinas, porém com restrições adicionais como *due dates* para as tarefas (KHURANA & BAGGA, 1985), tempos de remoção das tarefas (*removal times*) e condição de processamento contínuo (GUPTA, STRUSEVICH & ZWANEVELD, 1997). Outros trabalhos, também buscando a solução ótima, utilizaram medidas de desempenho da programação tais como a minimização do tempo médio de fluxo (BAGGA & KHURANA, 1986) e a minimização do *lateness* máximo (ALLAHVERDI & ALDOWAISAN, 1998).

Para *flow shops* com múltiplas máquinas, no trabalho já mencionado de Yoshida & Hitomi (1979) foi mostrado que mesmo para três máquinas, a minimização do *makespan* não necessariamente é obtida por uma programação permutacional. Mesmo assim, para o caso de diversas máquinas, as pesquisas relatadas na literatura consideram somente soluções permutacionais obtidas por

métodos heurísticos. Nessas pesquisas, busca-se a minimização do *makespan* sob restrições como, por exemplo, *buffer* limitado (PARK & STEUDEL, 1991), tempos de transferência das tarefas entre as máquinas (CAO & BEDWORTH, 1992) e tempos de remoção das tarefas não nulos (RAJENDRAN & ZIEGLER, 1997).

## 1.2 Problema FSP-TPS com tempos de preparação dependentes da seqüência das tarefas

Para o caso de tempos de preparação das máquinas dependentes da seqüência das tarefas, o procedimento pioneiro de solução do problema foi desenvolvido por Gupta (1969, 1975), fundamentado na técnica de busca lexicográfica.

Para *flow shops* com duas máquinas, Corwin & Esogbue (1974) propuseram um procedimento de Programação Dinâmica para obtenção da solução ótima quanto ao *makespan*, porém com tempos de preparação dependentes da seqüência das tarefas em somente uma das máquinas. Os primeiros métodos heurísticos para o ambiente com duas máquinas foram propostos por Gupta & Darrow (1985, 1986). Os métodos foram avaliados entre si considerando diferentes relações na ordem de grandeza dos tempos de processamento das tarefas e dos tempos de preparação das máquinas. Os resultados experimentais concluíram que as heurísticas propostas apresentaram boas soluções em relação à solução ótima para problemas onde os tempos de *setup* representam 1/10 dos tempos de processamento (em média).

A grande maioria dos trabalhos reportados na literatura para o caso de múltiplas máquinas tem como medida de desempenho o *makespan*. Mesmo com aplicação restrita a problemas de pequeno porte, diversos métodos de solução ótima foram desenvolvidos, utilizando técnicas de Programação Linear Inteira Mista (SRIKAR & GHOSH, 1986; STAFFORD & TSENG, 1990; RIOS-MERCADO & BARD, 1996) e métodos de enumeração *Branch & Bound* (RIOS-MERCADO & BARD, 1999).

Para problemas de médio e grande porte, salientam-se os métodos heurísticos. Os mais conhecidos são os métodos denominados *SETUP* e *TOTAL*

propostos por Simons Jr. (1992) e a heurística *NEHT-RB* e um procedimento do tipo *GRASP* (*Greedy Randomized Adaptive Search Procedure*) desenvolvido por Rios-Mercado & Bard (1998). Os métodos *SETUP* e *TOTAL* fundamentam-se em uma analogia do problema FSP-TPS com o problema assimétrico do caixeiro-viajante (*ATSP - Asymmetric Traveling Salesman Problem*). No caso do *TOTAL*, a matriz de "distâncias" entre pares de tarefas é obtida considerando-se a somatória dos tempos de processamento das tarefas com os tempos de preparação em todas as máquinas, enquanto que no *SETUP* somente os tempos de preparação das máquinas são considerados. No trabalho de Rios-Mercado & Bard (1998) foi efetuada uma comparação de desempenho entre o método *GRASP* e o *SETUP*. De maneira geral, o *GRASP* superou o *SETUP* quando os tempos de preparação das máquinas foram gerados a partir de uma distribuição uniforme discreta no intervalo [1, 10] e os tempos de processamento das tarefas utilizando a distribuição [1, 99].

Entretanto, quando esta última distribuição foi utilizada para tempos de preparação e processamento, o método *SETUP* teve um desempenho melhor que o *GRASP*. Rios-Mercado & Bard (1999b) propuseram uma extensão do método *SETUP* (SIMONS JR., 1992) que foi denominado *HYBRID*. Essencialmente, esse método em uma primeira fase incorpora ao *SETUP* alguns aspectos característicos do problema FSP-TPS, e na segunda fase procura melhorar a solução inicial por meio de um procedimento de busca local. Os resultados advindos de uma experimentação computacional mostraram que, em geral, o *HYBRID* pode ser considerado superior ao método *GRASP* quando os tempos de preparação das máquinas apresentam uma maior variabilidade, por exemplo, se forem gerados a partir de uma distribuição uniforme discreta no intervalo [1, 50] em comparação com os valores de uma distribuição [1, 10].

Allahverdi & Aldowaisan (2001) resolveram um tipo específico de problema *flow shop* com tempos de preparação dependentes da seqüência das tarefas onde as tarefas não podem esperar entre máquinas, e o critério de avaliação considerado foi a soma das datas de entrega das tarefas. Os autores discutiram uma série de métodos que proporcionam a solução ótima para o caso de duas máquinas quando uma série de condições especiais era cumprida. Também propuseram cinco heurísticas diferentes para o problema de *m* máquinas. Neste caso, os tempos de *setup* considerados foram do tipo aditivo.

Maddux III & Gupta (2003) estudaram o problema de *flow shop* com duas máquinas com critério de minimização do *makespan* sem estoques entre as máquinas (*buffers* de capacidade zero), e onde algumas tarefas são processadas apenas pela primeira máquina e algumas tarefas devem ser processadas por ambas as máquinas. Eles desenvolveram limitantes inferiores e uma heurística para o problema.

Rajendran & Ziegler (2003) estudaram o problema *flow shop* com a combinação de dois critérios de avaliação considerados por Rajendran & Ziegler (1997b) e Sonmez & Baykasoglu (1998), ou seja, soma ponderada dos atrasos e soma ponderada dos tempos de fluxo. Eles propuseram métodos heurísticos e as compararam com a heurística de Gelders & Sambandan (1978), com um procedimento de busca aleatória e com um procedimento de busca local gulosa. Os resultados experimentais mostram que a heurística proposta superava as demais heurísticas comparadas.

Moccellin & Nagano (2007) apresentaram uma propriedade estrutural para o problema de programação da produção com tempos de *setup* separados dos tempos de processamento, sejam dependentes ou independentes da seqüência de programação. Essa propriedade fornece um limitante superior para o tempo em que a máquina fica ociosa entre o fim de sua preparação (*setup*) e o início do processamento. Esta propriedade foi utilizada para a solução do problema de *flow shop* com tempos de setup separados com a minimização do *makespan* através de uma analogia com o problema do caixeiro assimétrico. Os resultados da experimentação computacional mostram que a analogia foi adequada para problemas com 10 ou mais tarefas.

Dong, Huang &Chen (2009) estudaram a heurística *NEHT-RB* e propuseram três novas regras de prioridade para a primeira etapa da heurística construtiva, em substituição à regra original *LPT*. Todas as novas regras de prioridade são baseadas nas relações entre a média e o desvio padrão dos tempos de processamento e uma média adaptada para os tempos de *setup*. A experimentação computacional mostrou que as heurísticas adaptadas com as novas regras de prioridade superam a heurística *NEHT-RB* quando a distribuição dos tempos de *setup* foi no máximo igual à distribuição dos tempos de processamento. No mesmo

trabalho eles propuseram uma estratégia de desempates para a segunda etapa da heurística *NEHT-RB*, porém sem conseguir melhores resultados.

Eren (2010) apresentou três heurísticas para problema *flow shop* com tempos de preparação dependentes da seqüência das tarefas com *m* máquinas, considerando o critério de minimização de *makespan* e a soma ponderada das datas de término. Todas as heurísticas foram baseadas na heurística *NEH*. As soluções das heurísticas foram comparadas com as soluções ótimas para problemas pequenos (6 máquinas e 18 tarefas) e comparadas entre si para problemas maiores.

## 2 MÉTODOS HEURÍSTICOS PARA O PROBLEMA FSP-TPS

Este trabalho se restringiu à proposição de novos métodos heurísticos construtivos, onde foram desenvolvidos quatro métodos para fins de comparação com os melhores métodos reportados na literatura, denominados *SETUP* e *TOTAL*. Com esse objetivo, foram desenvolvidas duas classes de métodos:

- Métodos heurísticos rápidos (*High Speed*), que foram compostos pelos métodos *TOT-HS* e *SET-HS*;
- Métodos heurísticos lentos (*Low-Speed*), que foram compostos pelos métodos *TOT-LS* e *SET-LS*.

As duas classes de métodos foram estabelecidas com base nos esforços computacionais apresentados para atingir a solução do problema. Ambas foram fundamentadas pelo processo de busca da seqüência solução do problema. Para os métodos heurísticos rápidos, o procedimento de busca é menos intensivo, enquanto que para os métodos heurísticos lentos, este é mais intensivo.

Os métodos heurísticos propostos serão apresentados a seguir.

### 2.1 Métodos heurísticos rápidos (*High Speed*)

- **Método *TOT-HS***

Para esse método, inicialmente deve-se construir uma matriz, denominada matriz *TOTAL* (SIMONS, 1992), em que cada elemento da matriz é carregado com

$p_{q\,j}$, correspondente ao tempo de processamento da tarefa $J_j$ na máquina $M_q$, e com $s_{q\,uv}$, que é o tempo de preparação da máquina $M_q$ para execução da tarefa $J_v$, que é diretamente precedida pela tarefa $J_u$ na seqüência de execução das *n* tarefas.

A nova heurística proposta é composta pelos seguintes passos:

- **Passo 1:** Encontre as tarefas $J_x$ e $J_y$ tal que

  $TOTAL^m_{S_{x\,y}\,y} = Min(TOTAL^m_{S_{i\,j}\,j} \;\; para\, i, j = 1, 2, ..., n)$.

- **Passo 2:** Faça $TOTAL^m_{S_{i\,x}\,x} = TOTAL^m_{S_{i\,y}\,y} = \infty \;\; para\, todo\, i = 1, 2, ..., n$.

- **Passo 3:** Encontre a tarefa $J_w$ tal que

  $TOTAL^m_{S_{y\,w}\,w} = Min(TOTAL^m_{S_{y\,j}\,j} \;\; para\; j = 1, 2, ..., n)$.

- **Passo 4:** Faça $TOTAL^m_{S_{i\,w}\,w} = \infty \;\; para\, todo\, i = 1, 2, ..., n$.

- **Passo 5:** Faça $k = 3$ e deixe $\sigma$ ser a seqüência parcial das *k*-tarefas, dada por $\sigma = (J_x, J_y, J_w)$.

- **Passo 6:** Deixe a tarefa $J_L$ na última posição da seqüência $\sigma$.

  Encontre a tarefa $J_z$ tal que $TOTAL^m_{S_{L\,z}\,z} = Min(TOTAL^m_{S_{L\,j}\,j} \;\; para\; j = 1, 2, ..., n)$.

- **Passo 7:** Faça $k = k + 1$ e $TOTAL^m_{S_{i\,z}\,z} = \infty \;\; para\, todo\, i = 1, 2, ..., n$.

- **Passo 8:** Faça que $\sigma'$ seja uma nova seqüência pela adição da tarefa $J_z$ na seqüência $\sigma$, e colocando-a na última posição da seqüência $\sigma'$. Atualize $\sigma$ com $\sigma'$ e retorne para o Passo 6. Se $k = n$, vá para o Passo 9.

- **Passo 9:** Encontre a melhor seqüência gerada com a vizinhança de permutação (*Interchange Neighborhood*) com a seqüência $\sigma$. Se o *makespan* da melhor vizinhança gerada for melhor que o da seqüência atual, então atualize $\sigma$ com a nova seqüência.

- **Passo 10:** Encontre a melhor seqüência gerada com a vizinhança de inserção (*Shift Neighborhood*) com a seqüência $\sigma$. Se o *makespan* da melhor vizinhança gerada for melhor que o da seqüência atual, então atualize $\sigma$ com a nova seqüência.

- **Passo 11:** Considere a seqüência $\sigma$ como um ciclo de tarefas e calcule o *makespan* de cada seqüência obtida através de $\sigma$. Recarregue $\sigma$ com a seqüência com menor *makespan*. A nova seqüência $\sigma$ é a melhor solução.

- **Método *SET-HS***

Esse método inicialmente deve construir uma matriz, denominada matriz *SETUP* (SIMONS, 1992), em que cada elemento da matriz é carregado com $s_{q\,uv}$, que é o tempo de preparação da máquina $M_q$ para execução da tarefa $J_v$, que é diretamente precedida pela tarefa $J_u$ na seqüência de execução das *n* tarefas.

A nova heurística proposta é composta pelos seguintes passos:

- **Passo 1:** Encontre as tarefas $J_x$ e $J_y$ tal que

  $SETUP^m_{S_x\,y\,y} = Min(SETUP^m_{S_i\,j\,j} \;\; para\, i, j = 1, 2, ..., n)$.

- **Passo 2:** Faça $SETUP^m_{S_i\,x\,x} = SETUP^m_{S_i\,y\,y} = \infty \;\; para\, todo\, i = 1, 2, ..., n$.

- **Passo 3:** Encontre a tarefa $J_w$ tal que

  $SETUP^m_{S_y\,w\,w} = Min(SETUP^m_{S_y\,j\,j} \;\; para\; j = 1, 2, ..., n)$.

- **Passo 4:** Faça $SETUP^m_{S_i\,w\,w} = \infty \;\; para\, todo\, i = 1, 2, ..., n$.

- **Passo 5:** Faça $k = 3$ e deixe $\sigma$ ser a seqüência parcial das *k*-tarefas, dada por $\sigma = (J_x, J_y, J_w)$.

- **Passo 6:** Deixe a tarefa $J_L$ na última posição da seqüência $\sigma$.

  Encontre a tarefa $J_z$ tal que $SETUP^m_{S_L\,z\,z} = Min(SETUP^m_{S_L\,j\,j} \;\; para\; j = 1, 2, ..., n)$.

- **Passo 7:** Faça $k = k + 1$ e $SETUP^m_{S_i\,z\,z} = \infty \;\; para\, todo\, i = 1, 2, ..., n$.

- **Passo 8:** Faça que $\sigma'$ seja uma nova seqüência pela adição da tarefa $J_z$ na seqüência $\sigma$, e colocando-a na última posição da seqüência $\sigma'$. Atualize $\sigma$ com $\sigma'$, e retorne para o Passo 6. Se $k = n$, vá para o Passo 9.

- **Passo 9:** Encontre a melhor seqüência gerada com a vizinhança de permutação (*Interchange Neighborhood*) com a seqüência $\sigma$. Se o *makespan*

da melhor vizinhança gerada for melhor que o da seqüência atual, então atualize $\sigma$ com a nova seqüência.

- **Passo 10:** Encontre a melhor seqüência gerada com a vizinhança de inserção (*Shift Neighborhood*) com a seqüência $\sigma$. Se o *makespan* da melhor vizinhança gerada for melhor que o da seqüência atual, então atualize $\sigma$ com a nova seqüência.
- **Passo 11:** Considere a seqüência $\sigma$ como um ciclo de tarefas e calcule o *makespan* de cada seqüência obtida através de $\sigma$. Recarregue $\sigma$ com a seqüência com menor *makespan*. A nova seqüência $\sigma$ é a melhor solução.

## 2.2 Métodos heurísticos lentos (*Low-Speed*)

- **Método *TOT-LS***

Inicialmente, deve-se construir uma matriz, denominada matriz *TOTAL* (SIMONS, 1992), em que cada elemento da matriz seja carregado com $p_{q\,j}$, correspondente ao tempo de processamento da tarefa $J_j$ na máquina $M_q$, e com $s_{q\,uv}$, que é o tempo de preparação da máquina $M_q$ para execução da tarefa $J_v$, que é diretamente precedida pela tarefa $J_u$ na seqüência de execução das *n* tarefas.

A nova heurística proposta é composta pelos seguintes passos:

- **Passo 1:** Encontre as tarefas $J_x$ e $J_y$ tal que

  $TOTAL^m_{S_{x\,y}\,y} = Min(TOTAL^m_{S_{i\,j}\,j} \;\; para\, i, j = 1, 2, ..., n)$.

- **Passo 2:** Faça $TOTAL^m_{S_i\,x\,x} = TOTAL^m_{S_i\,y\,y} = \infty \;\; para\, todo\, i = 1, 2, ..., n$.

- **Passo 3:** Encontre a tarefa $J_w$ tal que

  $TOTAL^m_{S_y\,w\,w} = Min(TOTAL^m_{S_y\,j\,j} \;\; para\; j = 1, 2, ..., n)$.

- **Passo 4:** Faça $TOTAL^m_{S_i\,w\,w} = \infty \;\; para\, todo\, i = 1, 2, ..., n$.

- **Passo 5:** Faça $k = 3$ e deixe $\sigma$ ser a seqüência parcial das *k*-tarefas, dada por $\sigma = (J_x, J_y, J_w)$.

- **Passo 6:** Encontre a melhor seqüência gerada com a vizinhança de permutação (*Interchange Neighborhood*) com a seqüência $\sigma$. Se o *makespan* da melhor vizinhança gerada for melhor que o da seqüência atual, então atualize $\sigma$ com a nova seqüência.
- **Passo 7:** Encontre a melhor seqüência gerada com a vizinhança de inserção (*Shift Neighborhood*) com a seqüência $\sigma$. Se o *makespan* da melhor vizinhança gerada for melhor que o da seqüência atual, então atualize $\sigma$ com a nova seqüência. Se $k = n$, então pare. A seqüência $\sigma$ é a solução. Caso contrário, faça $k = k + 1$ e vá para o Passo 8.
- **Passo 8:** Deixe a tarefa $J_L$ na última posição da seqüência $\sigma$.

  Encontre a tarefa $J_z$ tal que

$TOTAL^m_{S_L\ z\ z} = Min(TOTAL^m_{S_L\ j\ j}\ \ para\ j = 1, 2, ..., n)$.

- **Passo 9:** Faça $TOTAL^m_{S_i\ z\ z} = \infty\ \ para\ todo\ i = 1, 2, ..., n$.
- **Passo 10:** Faça que $\sigma'$ seja uma nova seqüência pela adição da tarefa $J_z$ na seqüência $\sigma$, colocando-a na última posição de $\sigma'$. Atualize $\sigma$ com $\sigma'$ e retorne para o Passo 6.

- **Método *SET-LS***

Inicialmente, deve-se construir uma matriz, denominada matriz *SETUP* (SIMONS, 1992), em que cada elemento da matriz seja carregado com $s_{q\ uv}$, que é o tempo de preparação da máquina $M_q$ para execução da tarefa $J_v$, que é diretamente precedida pela tarefa $J_u$ na seqüência de execução das *n* tarefas.

A nova heurística proposta é composta pelos seguintes passos:

- **Passo 1:** Encontre as tarefas $J_x$ e $J_y$ tal que

$SETUP^m_{S_x\ y\ y} = Min(SETUP^m_{S_i\ j\ j}\ \ para\ i, j = 1, 2, ..., n)$.

- **Passo 2:** Faça $SETUP^m_{S_i\ x\ x} = SETUP^m_{S_i\ y\ y} = \infty\ \ para\ todo\ i = 1, 2, ..., n$.
- **Passo 3:** Encontre a tarefa $J_w$ tal que

$SETUP^m_{S_y\ w\ w} = Min(SETUP^m_{S_y\ j\ j}\ \ para\ j = 1, 2, ..., n)$.

- **Passo 4:** Faça $SETUP^{m}_{S_i\ w\ w} = \infty \ \ para\ todo\ i = 1, 2, ..., n$.
- **Passo 5:** Faça $k = 3$ e deixe $\sigma$ ser a seqüência parcial das *k*-tarefas, dada por $\sigma = (J_x, J_y, J_w)$.
- **Passo 6:** Encontre a melhor seqüência gerada com a vizinhança de permutação (*Interchange Neighborhood*) com a seqüência $\sigma$. Se o *makespan* da melhor vizinhança gerada for melhor que o da seqüência atual, então atualize $\sigma$ com a nova seqüência.
- **Passo 7:** Encontre a melhor seqüência gerada com a vizinhança de inserção (*Shift Neighborhood*) com a seqüência $\sigma$. Se o *makespan* da melhor vizinhança gerada for melhor que o da seqüência atual, então atualize $\sigma$ com a nova seqüência. Se $k = n$, então pare. A seqüência $\sigma$ é a solução. Caso contrário, faça $k = k + 1$ e vá para Passo 8.
- **Passo 8:** Deixe a tarefa $J_L$ na última posição da seqüência $\sigma$.

  Encontre a tarefa $J_z$ tal que $SETUP^{m}_{S_L\ z\ z} = Min(SETUP^{m}_{S_L\ j\ j} \ \ para\ j = 1, 2, ..., n)$.
- **Passo 9:** Faça $SETUP^{m}_{S_i\ z\ z} = \infty \ \ para\ todo\ i = 1, 2, ..., n$.
- **Passo 10:** Faça que $\sigma'$ seja uma nova seqüência pela adição da tarefa $J_z$ na seqüência $\sigma$, colocando-a na última posição de $\sigma'$. Atualize $\sigma$ com $\sigma'$ e retorne para o Passo 6.

Para avaliar o desempenho dos métodos heurísticos classificados nas duas categorias (*High Speed* e *Low-Speed*), foram também programados os métodos originais *TOTAL* e *SETUP* (SIMONS, 1992).

A seguir serão apresentados os procedimentos do planejamento da experimentação computacional para fins de avaliação dos métodos propostos.

## 3 PLANEJAMENTO DA EXPERIMENTAÇÃO COMPUTACIONAL

Para a experimentação computacional, foi utilizado um banco de dados adaptado com os problemas de Taillard para o problema FSP-TPS, conforme detalhado no site *http://soa.iti.es/instancias-de-problemas/*. O mesmo banco de

dados foi utilizado nostrabalhos de Ruiz et al. (2005) e Ruiz e Stützle (2008). A experimentação foi composta de quatro grupos de problemas (SSD-10, SSD-50, SSD-100 e SSD-125). Para os problemas do grupo SSD-10, por exemplo, o tempo de *setup* é 10% do tempo de processamento da tarefa, isto é, se o tempo de processamento for gerado conforme uma distribuição uniforme no intervalo de [1, 99], então os tempos de *setup* são uniformemente distribuídos no intervalo de [1, 9].

Todos os métodos foram programados em linguagem Pascal para fins de comparação, inclusive o método *SETUP* e *TOTAL* de Simons (1992), que é considerado o melhor método heurístico construtivo para o problema. O sistema operacional utilizado foi o Windows Vista Ultimate. A configuração do micro-computador foi a seguinte: processador Intel Core 2 Duo com 2.20GHz, 2 GB de memória RAM e disco rígido de 150 GB.

Os resultados obtidos pelos métodos heurísticos foram avaliados de acordo com as estatísticas de Porcentagem de Sucesso, Desvio Relativo Médio e Tempo Médio de Computação.

A primeira foi definida pelo quociente entre o número de problemas para os quais o método obteve a melhor solução (*makespan*) e o número total de problemas resolvidos. Obviamente, quando os métodos obtêm a melhor solução para um mesmo problema, suas Porcentagens de Sucesso são simultaneamente melhoradas.

O Desvio Relativo ($DR_h$) quantifica o desvio que o método $h$ obtém em relação ao melhor *makespan* obtido para o mesmo problema, sendo calculado conforme segue:

$$DR_h(\%) = \frac{D_h - D*}{D*} \times 100$$

Onde:

$D_h$ é a duração total da programação (*makespan*) obtida pelo método $h$;

$D^*$ é o melhor *makespan* obtido pelos métodos para um determinado problema.

O Tempo Médio de Computação de um método foi calculado pela soma dos tempos de computação de cada problema dividida pelo número total de problemas resolvidos (média aritmética dos tempos de computação). Na experimentação, o tempo médio de computação por problema foi medido em segundos (*s*).

## 4 ANÁLISE DOS RESULTADOS OBTIDOS

A análise dos resultados foi divida em duas etapas: a primeira teve como objetivo avaliar todos os métodos heurísticos propostos para os problemas de pequeno e médio porte, e a segunda para os problemas de grande porte. As Tabelas 1 e 2 apresentam os resultados da experimentação computacional para a classe de problemas avaliados.

**Tabela 1 -** Porcentagem de sucesso, Porcentagem de desvio relativo e Tempo de computação para problemas de pequeno e médio porte

(continua)

| | | | | SSD-10 | | | |
|---|---|---|---|---|---|---|---|
| *n* | *m* | *TOTAL* | *SETUP* | *TOT-HS* | *SET-HS* | *TOT-LS* | *SET-LS* |
| 20 | 5 | 0* | 0 | 0 | 0 | 50 | 50 |
| | | 14,63** | 8,49 | 9,04 | 8,02 | 0,98 | 0,56 |
| | | 0,00*** | 0,00 | 0,00 | 0,00 | 0,02 | 0,01 |
| 20 | 10 | 0 | 0 | 0 | 0 | 40 | 60 |
| | | 14,56 | 12,93 | 10,01 | 9,04 | 0,72 | 0,29 |
| | | 0,00 | 0,00 | 0,00 | 0,00 | 0,02 | 0,02 |
| 20 | 20 | 0 | 0 | 0 | 0 | 50 | 50 |
| | | 10,22 | 11,31 | 8,82 | 8,30 | 0,65 | 0,54 |
| | | 0,00 | 0,00 | 0,01 | 0,01 | 0,03 | 0,03 |
| 50 | 5 | 0 | 0 | 0 | 0 | 20 | 80 |
| | | 8,96 | 9,78 | 8,27 | 6,45 | 0,59 | 0,20 |
| | | 0,00 | 0,00 | 0,02 | 0,03 | 0,33 | 0,33 |
| 50 | 10 | 0 | 0 | 0 | 0 | 10 | 90 |
| | | 13,74 | 11,90 | 12,41 | 10,56 | 0,83 | 0,15 |
| | | 0,00 | 0,00 | 0,05 | 0,05 | 0,60 | 0,60 |
| 50 | 20 | 0 | 0 | 0 | 0 | 30 | 70 |
| | | 13,55 | 14,75 | 13,21 | 12,43 | 0,70 | 0,38 |
| | | 0,00 | 0,00 | 0,09 | 0,08 | 1,12 | 1,11 |
| Média | | 0 | 0 | 0 | 0 | 33 | 67 |
| | | 12,61 | 11,53 | 10,29 | 9,13 | 0,75 | 0,35 |
| | | 0,00 | 0,00 | 0,03 | 0,03 | 0,35 | 0,35 |
| | | | | SSD-50 | | | |
| *n* | *m* | *TOTAL* | *SETUP* | *TOT-HS* | *SET-HS* | *TOT-LS* | *SET-LS* |
| 20 | 5 | 0 | 0 | 0 | 0 | 40 | 70 |
| | | 8,37 | 9,77 | 7,61 | 5,20 | 1,19 | 0,46 |
| | | 0,00 | 0,00 | 0,00 | 0,00 | 0,01 | 0,01 |
| 20 | 10 | 0 | 0 | 0 | 0 | 70 | 30 |
| | | 9,87 | 7,95 | 7,87 | 6,93 | 0,40 | 0,96 |
| | | 0,00 | 0,00 | 0,00 | 0,01 | 0,02 | 0,02 |
| 20 | 20 | 0 | 0 | 0 | 0 | 30 | 70 |
| | | 7,41 | 8,35 | 7,52 | 6,66 | 0,94 | 0,11 |
| | | 0,00 | 0,00 | 0,01 | 0,00 | 0,03 | 0,03 |
| 50 | 5 | 0 | 0 | 0 | 0 | 30 | 70 |
| | | 7,95 | 4,31 | 7,70 | 3,10 | 0,86 | 0,12 |
| | | 0,00 | 0,00 | 0,03 | 0,03 | 0,32 | 0,32 |
| 50 | 10 | 0 | 0 | 0 | 0 | 30 | 70 |
| | | 9,68 | 8,32 | 10,46 | 6,85 | 0,83 | 0,18 |
| | | 0,00 | 0,00 | 0,04 | 0,05 | 0,59 | 0,58 |

**Tabela 1 -** Porcentagem de sucesso, Porcentagem de desvio relativo e Tempo de computação para problemas de pequeno e médio porte (conclusão)

| SSD-10 | | | | | | | |
|---|---|---|---|---|---|---|---|
| *n* | *m* | *TOTAL* | *SETUP* | *TOT-HS* | *SET-HS* | *TOT-LS* | *SET-LS* |
| 50 | 20 | 0 | 0 | 0 | 0 | 30 | 70 |
| | | 9,24 | 8,34 | 9,44 | 7,94 | 0,78 | 0,11 |
| | | 0,00 | 0,00 | 0,09 | 0,09 | 1,11 | 1,12 |
| Média | | 0 | 0 | 0 | 0 | 38 | 63 |
| | | 8,75 | 7,84 | 8,43 | 6,11 | 0,83 | 0,32 |
| | | 0,00 | 0,00 | 0,03 | 0,03 | 0,35 | 0,35 |
| SSD-100 | | | | | | | |
| *n* | *m* | *TOTAL* | *SETUP* | *TOT-HS* | *SET-HS* | *TOT-LS* | *SET-LS* |
| 20 | 5 | 0 | 10 | 0 | 0 | 60 | 30 |
| | | 7,72 | 5,48 | 6,63 | 4,47 | 1,51 | 0,61 |
| | | 0,00 | 0,00 | 0,00 | 0,00 | 0,06 | 0,01 |
| 20 | 10 | 0 | 0 | 0 | 0 | 50 | 50 |
| | | 8,48 | 7,84 | 7,17 | 6,94 | 0,52 | 0,66 |
| | | 0,00 | 0,00 | 0,00 | 0,00 | 0,07 | 0,02 |
| 20 | 20 | 0 | 0 | 0 | 0 | 30 | 80 |
| | | 6,78 | 7,12 | 6,37 | 5,54 | 0,92 | 0,07 |
| | | 0,00 | 0,00 | 0,01 | 0,01 | 0,08 | 0,03 |
| 50 | 5 | 0 | 0 | 0 | 10 | 20 | 70 |
| | | 8,67 | 3,74 | 7,83 | 2,59 | 1,45 | 0,24 |
| | | 0,00 | 0,00 | 0,02 | 0,03 | 1,01 | 0,33 |
| 50 | 10 | 0 | 0 | 0 | 0 | 20 | 80 |
| | | 8,61 | 5,12 | 7,81 | 4,20 | 1,07 | 0,12 |
| | | 0,00 | 0,00 | 0,05 | 0,04 | 1,22 | 0,59 |
| 50 | 20 | 0 | 0 | 0 | 0 | 50 | 50 |
| | | 7,37 | 6,49 | 7,12 | 6,55 | 0,59 | 0,32 |
| | | 0,01 | 0,00 | 0,09 | 0,08 | 1,77 | 1,11 |
| Média | | 0 | 2 | 0 | 2 | 38 | 60 |
| | | 7,94 | 5,97 | 7,15 | 5,05 | 1,01 | 0,34 |
| | | 0,00 | 0,00 | 0,03 | 0,03 | 0,70 | 0,35 |
| SSD-125 | | | | | | | |
| *n* | *m* | *TOTAL* | *SETUP* | *TOT-HS* | *SET-HS* | *TOT-LS* | *SET-LS* |
| 20 | 5 | 10 | 0 | 0 | 0 | 60 | 30 |
| | | 3,75 | 7,19 | 7,24 | 4,94 | 1,14 | 1,98 |
| | | 0,00 | 0,00 | 0,00 | 0,00 | 0,01 | 0,02 |
| 20 | 10 | 0 | 10 | 0 | 0 | 40 | 50 |
| | | 7,05 | 6,26 | 5,80 | 5,34 | 0,90 | 0,68 |
| | | 0,00 | 0,00 | 0,00 | 0,00 | 0,02 | 0,02 |
| 20 | 20 | 10 | 0 | 0 | 0 | 20 | 70 |
| | | 7,69 | 7,55 | 6,84 | 6,27 | 1,33 | 0,35 |
| | | 0,00 | 0,00 | 0,01 | 0,00 | 0,03 | 0,03 |
| 50 | 5 | 40 | 30 | 0 | 20 | 0 | 10 |
| | | 5,35 | 2,76 | 6,58 | 2,57 | 2,90 | 1,55 |
| | | 0,00 | 0,00 | 0,03 | 0,02 | 0,34 | 0,32 |
| 50 | 10 | 0 | 0 | 0 | 0 | 20 | 80 |
| | | 5,91 | 3,96 | 6,72 | 4,26 | 1,04 | 0,16 |
| | | 0,00 | 0,00 | 0,04 | 0,05 | 0,58 | 0,59 |
| 50 | 20 | 0 | 0 | 0 | 0 | 30 | 70 |
| | | 6,18 | 8,89 | 7,36 | 5,66 | 0,60 | 0,23 |
| | | 0,01 | 0,01 | 0,09 | 0,09 | 1,11 | 1,11 |
| Média | | 10 | 7 | 0 | 3 | 28 | 52 |
| | | 5,99 | 6,10 | 6,76 | 4,84 | 1,32 | 0,83 |
| | | 0,00 | 0,00 | 0,03 | 0,03 | 0,35 | 0,35 |

*Porcentagem de sucesso;
**Porcentagem de desvio relativo;
***Tempo de computação.

**Tabela 2** - Porcentagem de sucesso, Porcentagem de desvio relativo e Tempo de computação para problemas de grande porte (continua)

| SSD-10 | | | | | | | |
|---|---|---|---|---|---|---|---|
| *n* | *M* | *TOTAL* | *SETUP* | *TOT-HS* | *SET-HS* | *TOT-LS* | *SET-LS* |
| 100 | 5 | 10* | 0 | 0 | 0 | 20 | 70 |
| | | 6,70** | 5,99 | 7,80 | 4,86 | 0,70 | 0,26 |
| | | 0,01*** | 0,01 | 0,18 | 0,18 | 4,65 | 4,54 |
| 100 | 10 | 0 | 0 | 0 | 0 | 20 | 80 |
| | | 11,91 | 12,71 | 11,03 | 10,21 | 0,40 | 0,09 |
| | | 0,02 | 0,01 | 0,34 | 0,35 | 8,84 | 8,69 |
| 100 | 20 | 0 | 0 | 0 | 0 | 30 | 70 |
| | | 13,46 | 13,36 | 12,74 | 11,68 | 0,30 | 0,17 |
| | | 0,02 | 0,02 | 0,68 | 0,70 | 17,92 | 17,27 |
| 200 | 10 | 0 | 0 | 0 | 0 | 30 | 70 |
| | | 9,66 | 9,27 | 9,60 | 8,26 | 0,29 | 0,14 |
| | | 0,11 | 0,10 | 3,25 | 3,35 | 149,55 | 148,86 |
| 200 | 20 | 0 | 0 | 0 | 0 | 20 | 80 |
| | | 11,56 | 11,82 | 11,41 | 11,42 | 0,38 | 0,05 |
| | | 0,13 | 0,12 | 6,50 | 6,53 | 316,11 | 302,02 |
| Média | | 2 | 0 | 0 | 0 | 24 | 74 |
| | | 10,66 | 10,63 | 10,51 | 9,28 | 0,42 | 0,14 |
| | | 0,06 | 0,05 | 2,19 | 2,22 | 99,41 | 96,27 |
| SSD-50 | | | | | | | |
| *n* | *M* | *TOTAL* | *SETUP* | *TOT-HS* | *SET-HS* | *TOT-LS* | *SET-LS* |
| 100 | 5 | 10 | 30 | 0 | 10 | 0 | 50 |
| | | 3,55 | 3,95 | 7,65 | 1,92 | 1,02 | 0,52 |
| | | 0,01 | 0,01 | 0,18 | 0,18 | 4,61 | 4,64 |
| 100 | 10 | 10 | 0 | 0 | 0 | 30 | 60 |
| | | 6,58 | 9,04 | 8,27 | 4,98 | 0,73 | 0,26 |
| | | 0,01 | 0,02 | 0,35 | 0,35 | 8,87 | 8,64 |
| 100 | 20 | 0 | 0 | 0 | 0 | 10 | 90 |
| | | 7,53 | 7,86 | 8,88 | 7,04 | 0,39 | 0,06 |
| | | 0,02 | 0,01 | 0,68 | 0,68 | 17,33 | 17,28 |
| 200 | 10 | 0 | 0 | 0 | 0 | 30 | 70 |
| | | 3,86 | 3,30 | 5,79 | 2,92 | 0,41 | 0,06 |
| | | 0,11 | 0,11 | 3,22 | 3,21 | 147,57 | 147,45 |
| 200 | 20 | 0 | 0 | 0 | 0 | 40 | 70 |
| | | 5,93 | 5,36 | 7,21 | 5,35 | 0,37 | 0,03 |
| | | 0,13 | 0,13 | 6,51 | 6,43 | 302,75 | 302,24 |
| Média | | 4 | 6 | 0 | 2 | 22 | 68 |
| | | 5,49 | 5,90 | 7,56 | 4,44 | 0,59 | 0,19 |
| | | 0,06 | 0,05 | 2,19 | 2,17 | 96,22 | 96,05 |
| SSD-100 | | | | | | | |
| *n* | *M* | *TOTAL* | *SETUP* | *TOT-HS* | *SET-HS* | *TOT-LS* | *SET-LS* |
| 100 | 5 | 20 | 60 | 0 | 10 | 0 | 10 |
| | | 2,61 | 3,24 | 7,00 | 1,55 | 3,39 | 2,06 |
| | | 0,02 | 0,01 | 0,18 | 0,18 | 9,27 | 4,52 |
| 100 | 10 | 0 | 0 | 0 | 0 | 30 | 70 |
| | | 4,05 | 2,42 | 5,45 | 2,57 | 0,59 | 0,07 |
| | | 0,01 | 0,01 | 0,35 | 0,34 | 13,61 | 8,61 |
| 100 | 20 | 0 | 0 | 0 | 0 | 50 | 50 |
| | | 4,85 | 3,97 | 6,76 | 5,05 | 0,26 | 0,26 |
| | | 0,02 | 0,01 | 0,68 | 0,67 | 22,57 | 17,17 |
| 200 | 10 | 50 | 30 | 0 | 20 | 0 | 0 |
| | | 1,14 | 1,00 | 5,40 | 1,75 | 2,57 | 1,62 |

**Tabela 2** - Porcentagem de sucesso, Porcentagem de desvio relativo e Tempo de computação para problemas de grande porte (conclusão)

| SSD-10 | | | | | | | |
|---|---|---|---|---|---|---|---|
| *n* | *M* | *TOTAL* | *SETUP* | *TOT-HS* | *SET-HS* | *TOT-LS* | *SET-LS* |
| | | 0,11 | 0,11 | 3,25 | 3,22 | 189,26 | 147,49 |
| 200 | 20 | 10 | 0 | 0 | 0 | 30 | 60 |
| | | 3,57 | 4,64 | 4,78 | 2,90 | 0,42 | 0,12 |
| | | 0,13 | 0,13 | 6,45 | 6,45 | 333,94 | 301,25 |
| Média | | 16 | 18 | 0 | 6 | 22 | 38 |
| | | 3,24 | 3,05 | 5,88 | 2,76 | 1,45 | 0,82 |
| | | 0,06 | 0,06 | 2,18 | 2,17 | 113,73 | 95,81 |
| SSD-125 | | | | | | | |
| *n* | *M* | *TOTAL* | *SETUP* | *TOT-HS* | *SET-HS* | *TOT-LS* | *SET-LS* |
| 100 | 5 | 20 | 60 | 0 | 20 | 0 | 0 |
| | | 4,29 | 1,53 | 7,42 | 3,22 | 5,05 | 4,05 |
| | | 0,01 | 0,02 | 0,18 | 0,18 | 4,65 | 4,52 |
| 100 | 10 | 0 | 0 | 0 | 0 | 50 | 50 |
| | | 5,30 | 2,50 | 5,92 | 2,66 | 0,87 | 0,45 |
| | | 0,02 | 0,01 | 0,35 | 0,34 | 8,68 | 8,62 |
| 100 | 20 | 0 | 0 | 0 | 0 | 20 | 80 |
| | | 4,76 | 4,36 | 6,88 | 4,71 | 0,67 | 0,17 |
| | | 0,02 | 0,02 | 0,68 | 0,67 | 17,15 | 17,15 |
| 200 | 10 | 10 | 60 | 0 | 20 | 0 | 10 |
| | | 1,48 | 1,54 | 4,00 | 1,17 | 2,47 | 1,51 |
| | | 0,11 | 0,10 | 3,19 | 3,21 | 147,56 | 147,07 |
| 200 | 20 | 0 | 20 | 0 | 10 | 30 | 40 |
| | | 2,33 | 1,85 | 4,16 | 2,69 | 0,63 | 0,42 |
| | | 0,14 | 0,13 | 6,46 | 6,40 | 301,52 | 300,24 |
| Média | | 6 | 28 | 0 | 10 | 20 | 36 |
| | | 3,63 | 2,36 | 5,68 | 2,89 | 1,94 | 1,32 |
| | | 0,06 | 0,06 | 2,17 | 2,16 | 95,91 | 95,52 |

*Porcentagem de sucesso;
**Porcentagem de desvio relativo;
***Tempo de computação.

## 4.1 Análise da porcentagem de sucesso

Analisando inicialmente o critério de desempenho porcentagem de sucesso para os métodos com problemas de pequeno e médio porte, verificou-se que as heurísticas que utilizaram a matriz *SETUP* (SIMONS, 1992) obtiveram desempenhos superiores em comparação aos métodos que utilizaram a matriz *TOTAL* (SIMONS, 1992). Entre os seis métodos avaliados para os problemas de pequeno e médio porte, evidencia-se que o melhor método foi o *SET-LS,* que obteve maior porcentagem de sucesso média de 67% para os problemas SSD-10 e menor porcentagem de sucesso média de 52% para os problemas SSD-125. O segundo melhor método foi o *TOT-LS,* que obteve sua maior porcentagem de sucesso média

de 38% para os problemas SSD-50 e SSD-100 e a menor de 28% para os problemas SSD-125.

Para os problemas de grande porte, as melhores porcentagens de sucesso médias foi obtida pelo método heurístico *SET-LS*, sendo que a maior porcentagem de sucesso média foi de 74% para problemas SSD-10 e a menor foi de 36% para problemas SSD-125. O Segundo melhor método foi o *TOT-LS,* com a maior porcentagem de sucesso média de 24% para problemas SSD-10 e menor de 20% para problemas SSD-125. Tais resultados foram decorrentes do procedimento de construção da seqüência solução que foi mais intensiva, levando a um tempo computacional maior.

Uma importante observação obtida pela experimentação computacional foi a diminuição da porcentagem de sucesso dos métodos propostos à medida que os tempos de *setups* aumentam em proporção relativa ao tempo de processamento. Os resultados comprovam que o método S*ETUP* apresenta desempenho melhor quando os tempos de *setup* são superiores que os tempos de processamento.

### 4.2 Análise da porcentagem de desvio relativo

Em relação à qualidade da solução obtida pela análise da porcentagem de desvio relativo, o melhor método foi o *SET-LS,* com o menor desvio de 0,32% para problemas de pequeno e médio porte e 0,14% para problemas de grande porte. O segundo melhor método, neste quesito, foi o *TOT-LS,* com desvios de 0,75% e 0,42% para os problemas de pequeno, médio e grande porte respectivamente.
Os métodos *TOTAL*, *SETUP*, *TOT-HS* e *SET-HS* apresentaram qualidade de solução bem inferiores comparados aos métodos *SET-LS* e *TOT-LS*.

### 4.3 Tempo médio de computação

Analisando o tempo de computação dos métodos, verificou-se que nenhum deles ultrapassou o tempo médio de 2 minutos. Uma importante constatação a ser reportada foi o tempo médio de computação do método *SETUP,* que para problemas de grande porte não foi superior a 0,13 segundos.

Analisando a Tabela 2, verificou-se que o tempo de computação para os problemas de grande porte foi extremamente alto para os métodos *TOT-LS* e *SET-LS,* em comparação a outros métodos heurísticos. Entretanto, o caso em que o método obteve maior tempo computacional não ultrapassou 6 minutos. As heurísticas *TOTAL* e *SETUP* obtiveram os menores tempos computacionais em todas as categorias de problemas, entretanto, o desempenho em relação à qualidade das soluções foi inferior quando comparado aos melhores métodos desenvolvidos.

## 5 CONSIDERAÇÕES FINAIS

Neste artigo foi apresentada uma avaliação de novos métodos heurísticos para o problema de programação da produção *flow shop* permutacional com tempos de preparação das máquinas separados dos tempos de processamento das tarefas, no qual o melhor método heurístico proposto apresentou desempenho superior comparado aos melhores métodos reportados na literatura.

Os métodos heurísticos que se apresentaram mais lentos para a obtenção da solução (maior tempo de computação) obtiverem melhores desempenhos com relação à porcentagem de sucesso e porcentagem de desvio relativo. Esses métodos utilizam as matrizes *TOTAL* e *SETUP* de SIMONS (1992) de forma combinada com procedimentos mais intensivos de busca local na construção da seqüência solução. Uma aplicação eficiente de novos processos de construção da seqüência solução foi proposta a partir de uma analogia do problema de programação da produção com um problema cíclico assimétrico do caixeiro-viajante. Com o objetivo de avaliar a adequação da analogia proposta, foi efetuada uma experimentação computacional sobre uma amostra de problemas.

De maneira geral, os valores dos resultados obtidos por meio da experimentação e suas formas de análise foram suficientes para verificar e identificar o melhor método heurístico.

Obviamente, a solução do problema é aproximada (heurística) para o problema original de programação da produção.

## REFERÊNCIAS


ALLAHVERDI, A., ALDOWAISAN, T. Job lateness in flowshops with setup and removal times separated. **Journal of the Operational Research Society,** v.49, p.1001-1006, 1998.

_____. Minimizing total completion time in a no-wait flowshop with sequence-dependent additive changeover times. **Journal of the Operational Research Society,** v.52, p.449-462, 2001.

BAGGA, P.C., KHURANA, K. Two-machine flowshop with separated sequence-independent setup times: mean completion time criterion. **Indian Journal of Management and Systems**, v.2, p.47-57, 1986.

CAO, J., BEDWORTH, D.C. Flow shop scheduling in serial multi-product processes with transfer and setup times. **International Journal of Production Research,** v.30, p.1819-1830, 1992.

CHENG, T.C.E., GUPTA, J.N.D., WANG, G. A review of flowshop scheduling research with setup times. **Production and Operations Management,** v.9, p.262-282, 2000.

CORWIN, B.D., ESOGBUE, A.O. Two-machine flowshop scheduling problems with sequence dependent setup times: a dynamic programming approach. **Naval Research Logistics Quarterly,** v.21, p.515-524, 1974.

DAS, S.R., GUPTA, J.N.D., KHUMAWALA, B.M. A savings index heuristic algorithm for flowshop scheduling with sequence dependent set-up times. **Journal of the Operational Research Society**, v.46, p.1365-1373, 1995.

DONG, X., HUANG, H., CHEN, P. Study on heuristics for the permutation flowshop with sequence dependent setup times. **Information Reuse & Integration***, IRI '09. IEEE International Conference*, v.10, p.417-421, 2009.

EREN, T. A bicriteria m-machine flowshop scheduling with sequence-dependent setup times. **Applied Mathematical Modelling**, v.34, p.284-293, 2010.

GAREY, M.R., JOHNSON, D.S., SETHI, R. The complexity of flowshop and jobshop scheduling. **Mathematics of Operations Research,** v.1, p.117-129, 1976.

GELDERS, L. F., SAMBANDAM, N. Four simple heuristics for scheduling a flowshop. **International Journal of Production Research,** v.16, p.221–231, 1978.

GUPTA, J.N.D. **Economic aspects of scheduling theory.** Texas Tech University, Lubbock, Texas, 1969.

_____. A search algorithm for the generalized scheduling problem. **Computers & Operations Research,** v.2, p.83-90, 1975.

_____; DARROW, W.P. Approximate schedules for the two-machine flow-shop with sequence dependent setup times. **Indian Journal of Management and Systems,** v.1, p.6-11, 1985.

_____. The two-machine sequence dependent flowshop scheduling problem. **European Journal of Operational Research,** v.24, p.439-446, 1986.
GUPTA, J.N.D., STAFFORD JR., E.F. Flowshop scheduling research after five decades. **European Journal of Operational Research**, v.169, p.699-711, 2006.

GUPTA, J.N.D., STRUSEVICH, V.A., ZWANEVELD, C. Two-stage no-wait scheduling models with setup and removal times. **Computers & Operations Research,** v.24, p.1025-1031, 1997.

HEJAZI, S.R., SAGHAFIAN, S. Flowshop-scheduling problems with makespan criterion: a review. **International Journal of Production Research,** v.43, p.2895-2929, 2005.

JOHNSON, S. Optimal two- and three-stage production schedules with setup times included. **Naval Research Logistics Quarterly,** v.1, p.61-68, 1954.

KHURANA, K., BAGGA, P.C. Scheduling of job-block with deadline in n X 2 flowshop problem with separated setup times. **Indian Journal of Pure Applied Mathematics,** v.16, p.213-224, 1985.

MADDUX III, H. S., GUPTA, J. N. D. Scheduling intermediate and finished products in a two-stage flowshop with sequence dependent setup times. **Annual Meeting of the Decision Sciences Institute**, v.34, p.1579–1585, 2003.

MOCCELLIN, J.V., NAGANO, M.S. Uma propriedade estrutural do problema de programação da produção *flow shop* permutacional com tempos de *setup*. **Pesquisa Operacional,** v.27, p.487-515, 2007.

PARK, T., STEUDEL, H.J. Analysis of heuristics for job sequencing in manufacturing flow line work-cells. **Computers & Industrial Engineering,** v.20, p.129-140, 1991.

PINEDO, M. **Scheduling**: theory, algorithms, and systems. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1995.

RAJENDRAN, C., ZIEGLER, H. Heuristics for scheduling in a flowshop with setup, processing and removal times separated. **Production Planning & Control,** v.8, p.568-576, 1997a.

_____. A heuristic for scheduling to minimize the sum of wighted flowtime of jobs in a flowshop with sequence-dependent setup times of jobs. **Computers and Industrial Engineering**, v.33, p.281-284, 1997b.

_____. Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flowshop with sequence-dependent setup times. **European Journal of Operational Research**, v.149, p. 513–522, 2003.

RIOS-MERCADO, R.Z., BARD, J.F. **The flowshop scheduling polyhedron with setup times**. Technical Report ORP96-07, Graduate Program in Operations Research, University of Texas at Austin, Austin-TX, 1996.

RIOS-MERCADO, R.Z., Bard, J.F. Heuristics for the flow line problem with setup costs. **European Journal of Operational Research,** v.110, p.76-98, 1998.
RIOS-MERCADO, R.Z., BARD, J.F. A branch-and-bound algorithm for flowshop scheduling with setup times. **IEE Transactions,** v.31, p.721-731, 1999a.

_____. An enhanced TSP-based heuristic for makespan minimization in a flowshop with setup times. **Journal of Heuristics**, v.5, p.53-70, 1999b.

RUIZ, R., MAROTO, C., ALCARAZ, J. Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. **European Journal of Operational Research**, v.165, p.34-54, 2005.

RUIZ, R., STÜTZLE, T. An Iterated Greedy heuristic for the sequence dependent setup times flowshop with makespan and weighted tardiness objectives. *European* **Journal of Operational Research***,* v.187, p.1143-1159, 2008.

SIMONS JR., J.V. Heuristics in flow shop scheduling with sequence dependent setup times. **Omega,** v.20, p.215-225, 1992.

SÖNMEZ, A. I., BAYKASOGLU, A. A new dynamic programming formulation of (n*m) flowshop sequencing problems with due dates**. International Journal of Production Research**, v.36, p.2269-2283, 1998.

SRIKAR, B.N., GHOSH, S. A MILP model for the n-job, m-stage flowshop, with sequence dependent setup times. **International Journal of Production Research,** v.24, p.1459-1472, 1986.

STAFFORD, E.F., TSENG, F.T. On the Srikar-Ghosh MILP model for the N X M SDST flowshop problem. **International Journal of Production Research,** v.28, p.1817-1830, 1990.

Yoshida, T., Hitomi, K. Optimal two-stage production scheduling with setup times separated. **AIEE Transactions**, v.11, p.261-263, 1979.

Artigo recebido em 18/02/2011 e aceito para publicação em 22/12/2011.