Gest. Prod., São Carlos, v. 17, n. 2, p. 367-375, 2010

Métodos heurísticos construtivos para redução do estoque em processo em ambientes de produção *flow shop* híbridos com tempos de *setup* dependentes da sequência

Constructive heuristics methods to minimizing work in process in environment production hybrid flow shop with asymmetric sequence dependent setup times

Márcia de Fátima Morais[1]
João Vitor Moccellin[1]

Resumo: Este artigo apresenta uma investigação sobre o problema de programação da produção em ambientes *flow shop* com múltiplas máquinas (híbridos) e tempos de preparação (*setup*) das máquinas assimétricos e dependentes da sequência, e propõe métodos heurísticos construtivos para a minimização do tempo médio de fluxo (*Mean Flow Time*), que objetiva uma resposta rápida à demanda e à redução do estoque em processamento. Os algoritmos propostos foram comparados entre si, uma vez que nenhum método de solução para o problema investigado foi encontrado na literatura. Um estudo da influência da relação entre as ordens de grandeza dos tempos de processamento das tarefas e de preparação das máquinas em cada método de solução, bem como a influência do procedimento de programação adotado, foi efetuado com o intuito de avaliar o desempenho dos métodos.

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

Abstract: *This paper presents an investigation about the hybrid flow shop problem with asymmetric sequence dependent setup times and proposes constructive heuristic methods to minimize the mean flow time aiming at fast demand response and work in process reduction. The proposed heuristic methods were compared among themselves since no constructive heuristic method was found in the literature on the scheduling problem considered in this work. A study on the influence of the relation between the orders of magnitude of processing and setup time for each method was carried out. The influence of the scheduling procedure adopted was investigated to assess the performance of the methods used.*

Keywords: *Production scheduling. Hybrid flow shop. Sequence dependent setup time. Heuristics.*

1 Introdução

A produção desempenha um importante papel no auxílio ao cumprimento dos objetivos estratégicos da empresa e sua gestão da produção, na maioria das situações é conduzida de forma sistemática. A responsabilidade pelo planejamento, execução e controle das atividades realizadas no sistema de produção, ou seja, pela operacionalização do sistema, é do Planejamento e Controle da Produção (PCP).

A atividade de programação da produção constitui uma das várias funções executadas pelo PCP e caracteriza uma das atividades mais complexas no gerenciamento dos sistemas produtivos, uma vez que lida com diversos tipos diferentes de recursos e tarefas simultaneamente. As decisões envolvidas no nível de programação são: designação de tarefas às máquinas e programação das tarefas em cada máquina, isto é, a sequência de processamento das tarefas e o instante de início e término do processamento de cada tarefa.

A programação da produção refere-se à ordenação de tarefas a serem executadas, em uma ou diversas máquinas, considerando-se uma base de tempo, ou seja, determinando-se principalmente, as datas de início e fim de cada tarefa. Em outras palavras, a programação da produção pode ser definida como a determinação de quanto e onde cada operação necessária para a fabricação de um produto deve ser realizada. As tarefas são conhecidas, determinadas e devem ser executadas nas máquinas. Cada tarefa corresponde a um dado conjunto de operações que

[1] Departamento de Engenharia de Produção, Escola de Engenharia de São Carlos – EESC, Universidade de São Paulo – USP, Av. Trabalhador São carlense, 400, CEP 13566-590, São Carlos, SP, Brasil, E-mails: marciafmorais@yahoo.com.br; jvmoccel@sc.usp.br

Recebido em 27/2/2009 — Aceito em 17/2/2010

Suporte financeiro: CAPES.

tem uma sequência a ser seguida para a execução completa.

A programação das operações ao longo do tempo e sua atribuição aos recursos adequados apresentam estreitas ligações com o desempenho da empresa no âmbito de dimensões estratégicas como rapidez, confiabilidade, flexibilidade, qualidade e custos. As consequências da programação da produção para as dimensões estratégicas anteriormente mencionadas são extremamente relevantes e, portanto influenciam profundamente o desempenho global das empresas manufatureiras.

A teoria de programação da produção preocupa-se em fornecer diretrizes e métodos eficientes para a utilização dos recursos em suas atividades. Os problemas de programação da produção têm sido alvo de incontáveis trabalhos de pesquisa operacional há cerca de cinco décadas.

A atividade de programação é uma das mais complexas tarefas no gerenciamento de produção. Primeiro, os programadores podem lidar com diversos tipos diferentes de recursos simultaneamente. As máquinas podem ter diferentes capacidades e o pessoal terá diferentes habilidades. De maneira mais importante, geralmente, o número de programações possíveis cresce rapidamente à medida que o número de atividades e processos aumenta, conforme afirma (SLACK et al., 1999).

Em problemas de programação de operações em máquinas, as restrições tecnológicas das tarefas e os objetivos da programação devem ser especificados. As restrições tecnológicas são determinadas principalmente pelo padrão do fluxo das tarefas nas máquinas, levando a uma classificação conforme segue:

- *Job shop*: cada tarefa tem sua própria ordem de processamento nas máquinas;
- *Flow shop*: todas as tarefas têm o mesmo fluxo de processamento nas máquinas;
- *Open shop*: não há fluxo definido (específico) para as tarefas serem processadas nas máquinas;
- *Flow shop* permutacional: trata-se de *flow shop* no qual a ordem de processamento das tarefas deve ser a mesma em todas as máquinas;
- Máquina única: existe apenas uma máquina a ser utilizada;
- Máquinas paralelas: são disponíveis mais de uma máquina, geralmente idênticas, para as mesmas operações;
- *Job shop* com múltiplas máquinas: *job shop* no qual em cada estágio de produção existe um conjunto de máquinas paralelas; e
- *Flow shop* com múltiplas máquinas: *flow shop* no qual em cada estágio de produção existe um conjunto de máquinas paralelas.

Esses diversos tipos de problemas são ilustrados na Figura 1.

Desde o trabalho pioneiro de Johnson (1954) apud Morais (2008), que aborda *flow shop* com duas máquinas, muitas pesquisas têm sido conduzidas na busca de métodos exatos e heurísticos para o problema *flow shop*. Todavia, alguns ambientes ainda foram pouco explorados, como é o caso do ambiente *flow shop* híbrido. Ressalta-se que, para o problema

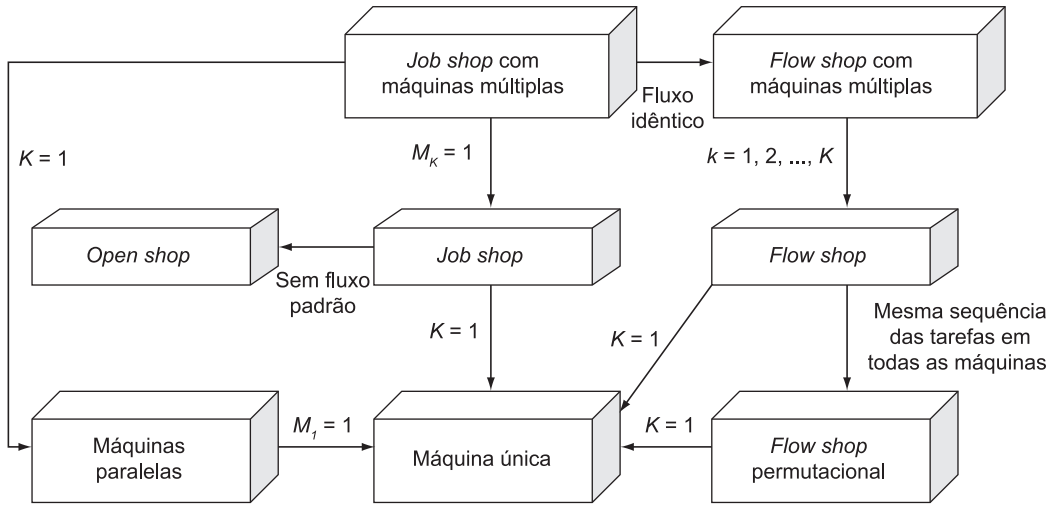

Figura 1. Relação entre as classes de problemas de programação de operações em máquinas (adaptado de MACCARTHY; LIU, 1993).

tratado aqui, até o momento, nenhum trabalho que abordasse os tempos de preparação separados dos tempos de processamento e critério de minimização do tempo médio de fluxo, foi encontrado na literatura examinada.

Diante do exposto, este trabalho tem como objetivo desenvolver métodos heurísticos de solução para o problema de programação definido como:

1) *Flow shop* híbrido composto por múltiplos estágios de produção, ou seja, $K \geq 2$;
2) Em cada estágio k, existem M_k máquinas paralelas idênticas, em que $M_k \geq 2$; e
3) Os tempos de preparação das máquinas são assimétricos e dependentes da sequência de execução das tarefas.

O ambiente descrito é ilustrado na Figura 2 a seguir.

As principais hipóteses consideradas no problema são as seguintes:

1) Os tempos de processamento das tarefas nas diversas máquinas são determinados e fixos;
2) As tarefas têm a mesma data de liberação, a partir da qual qualquer uma pode ser programada e executada. Esta data de liberação pode ser considerada igual a zero, no primeiro estágio, sem perda de generalidade;
3) Uma vez iniciadas, as operações de cada tarefa não podem ser interrompidas nem subdivididas em suboperações simultâneas;
4) Uma tarefa só pode começar a ser executada em uma máquina após a execução completa da sua operação no estágio anterior e desde que a máquina já esteja preparada; e
5) O *setup* de uma máquina, para determinada tarefa, pode ser executado antes da operação dessa tarefa estar concluída no estágio anterior e considera-se que o *setup* da primeira operação em cada máquina já esteja realizado (no problema, o tempo de setup dessas operações é considerado igual a zero).

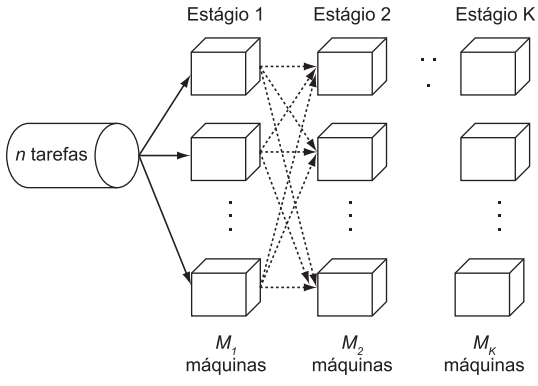

Figura 2. Ambiente de produção – *flow shop* híbrido.

O problema consiste em programar um conjunto de n tarefas, definido como $J = \{J_1, \ldots, J_n\}$, em que cada tarefa possui necessariamente uma única operação em cada estágio de produção. O objetivo do problema é minimizar o tempo médio de fluxo como medida de desempenho.

2 Métodos heurísticos para programação de operações em máquinas em ambientes *flow shop* híbridos

2.1 Métodos heurísticos propostos

Para a resolução do problema investigado foram propostos quatro métodos de solução, denominados MM-*FlowTime*$_X$ (MM-FT$_X$, com x = 1, ... ,4) nos quais os dois primeiros (MM-FT$_1$ e MM-FT$_2$) enfocam a programação estágio a estágio e os dois últimos (MM-FT$_3$ e MM-FT$_4$) enfocam a programação tarefa a tarefa.

A programação das tarefas, estágio a estágio, é feita como solução iterativa de K problemas relacionados. O primeiro estágio ($k = 1$) é programado como se fosse um problema tradicional de M_k máquinas paralelas idênticas com a mesma data de liberação, convencionada igual a zero ($r_i = r = 0$). Os estágios seguintes ($k \geq 2$) consistem de $K - 1$ problemas consecutivos de M_k máquinas paralelas idênticas.

A programação das tarefas, tarefa a tarefa, é feita como solução iterativa de n problemas relacionados. A primeira tarefa é programada como se fosse um problema de *flow shop* tradicional, constituído de K máquinas. As tarefas seguintes consistem de $n - 1$ problemas relacionados de M_k máquinas paralelas idênticas.

Os métodos de solução MM-FT$_1$ e MM-FT$_3$ estabelecem uma ordenação inicial com base na regra de prioridade TSPT (*Total Shortest Processing Time*), enquanto os métodos MM-FT$_2$ e MM-FT$_4$ definem a ordenação inicial utilizando a regra de prioridade TLPT (*Total Longest Processing Time*). Os métodos MM-FT$_1$ e MM-FT$_2$, além das regras de prioridade, anteriormente mencionadas, para o estabelecimento da ordenação inicial, utilizam a regra de prioridade SRD (*Shortest Release Date*) para a programação sucessiva das tarefas em cada estágio $k \geq 2$. Os quatro métodos propostos utilizam a regra de alocação SCT (*Shortest Completion Time*), baseada no algoritmo de Weng, Lu e Ren (2001).

A regra de prioridade TSPT considera a soma dos tempos de processamento de cada tarefa em todos os estágios de produção, sendo o sequenciamento das tarefas feito em ordem não decrescente destas somas.

A regra de prioridade TLPT considera a soma dos tempos de processamento de cada tarefa em todos os estágios de produção e o sequenciamento das tarefas é feito em ordem não crescente destas somas.

As regras de prioridade TSPT e TLPT são utilizadas apenas no primeiro estágio de produção, enquanto a regra de prioridade SRD é utilizada nos demais estágios de produção. Na regra de prioridade SRD a ordenação das tarefas é feita em ordem não decrescente dos valores das datas de liberação das tarefas (r_{ik}, k = 2, 3, ..., K), devendo ser observado que a data de liberação de uma tarefa em um estágio $k \geq 2$ corresponde à sua data de término no estágio anterior.

A regra de ordenação SCT considera um determinado estágio de produção k (k = 1, 2, ... ,K) e uma vez definida a tarefa a ser programada, em função de uma das regras de prioridade anteriormente definidas, a alocação dessa tarefa é efetuada à máquina do respectivo estágio que leva à menor data de término dessa tarefa.

Para a definição dos algoritmos propostos neste trabalho consideraramos:

- J conjunto de n tarefas a serem programadas;
- J' conjunto das tarefas ainda não programadas;
- $J'_{[j]}$ tarefa que ocupa a j-ésima posição de J';
- M número total de máquinas, considerando todos os estágios de produção ($M = \sum_{k=1}^{K} M_k$);
- M_k é o número de máquinas no estágio k;
- M_k' conjunto de k estágio de produção ainda não programados;
- K conjunto de k estágios de produção;
- K' conjunto de k estágio de produção ainda não programados; e
- σ_m conjunto de tarefas já alocadas à máquina m (m = 1, 2, ..., M).

2.2.1 Algoritmo 1 – MM – *flowtime* 1 (TSPT/SRD/SCT)

Passo 0 – Inicialização de conjuntos.

$k = \phi$ para k = 1, 2, ..., K, ou seja, o conjunto de estágios a serem programados inicia vazio e o conjunto de estágios ainda não programadas recebe o conjunto de estágios a serem programadas, ou seja, $K' \leftarrow K$.

$Mk = \phi$ para m = 1, 2, ..., Mk, o conjunto de máquinas em cada estágio de produção inicia vazio e recebe o conjunto de máquinas a serem programadas em cada estágio de produção, ou seja, $Mk' \leftarrow Mk$.

$\sigma_m = \phi$ para m = 1, 2, ..., M, ou seja, o subconjunto de tarefas alocadas à máquina m inicia a programação vazio, para todas as máquinas, e o conjunto de tarefas ainda não programadas recebe o conjunto de tarefas a serem programadas, ou seja, $J' \leftarrow J$.

Passo 1 – Ordene todas as tarefas do conjunto J' de acordo com a regra de prioridade TSPT. Selecione a tarefa que ocupa a primeira posição em J' ($J'_{[1]}$). Vá para o Passo 3.

Passo 2 – Ordene todas as tarefas de J' de acordo com a regra de prioridade SRD, considerando as datas de término das tarefas no estágio anterior como as datas de liberação do estágio atual. Selecione a tarefa que ocupa a primeira posição em J' ($J'_{[1]}$). Vá para o Passo 3.

Passo 3 – Em cada uma das máquinas do estágio, analise todas as possibilidades de alocação da tarefa $J'_{[1]}$ (primeira posição de J'), e escolha a máquina m com a data de término mais cedo, ou seja, aplique a regra de alocação SCT.

Passo 4 – Atualização de conjuntos.

O conjunto de tarefas alocadas à máquina m recebe a tarefa a tarefa que ocupa a primeira posição do conjunto de tarefas a serem programadas, ou seja, $\sigma_m \leftarrow \sigma_m \cup \{J'_{[1]}\}$ e a tarefa alocada à máquina m é excluída do conjunto de tarefas a serem programadas, ou seja, $J' \leftarrow J' - \{J'_{[1]}\}$.

Se o conjunto de tarefas a serem programadas ainda não estiver vazio e a programação no estágio 1 ainda não foi concluída, $J' \neq \phi$ e k = 1 (primeiro estágio), vá para o Passo 1.

Se o conjunto de tarefas a serem programadas ainda não estiver vazio e a programação no estágio 1 foi concluída, ou seja, $J' \neq \phi$ e k = 1, vá para o Passo 2.

Caso contrário, a programação do estágio está concluída.

Faça o conjunto J' passar a conter novamente as n tarefas ($J' \leftarrow J$) para programação no próximo estágio.

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

Caso contrário pare, a programação está concluída.

Saída: σm para m = 1, 2, ..., M.

2.2.2 Algoritmo 2 – MM – *flowtime* 2 (TLPT/SRD/SCT)

Passo 0 – Inicialização de conjuntos.

$k = \phi$ para k = 1, 2. ... , K, ou seja, o conjunto de estágios a serem programados inicia vazio e o conjunto de estágios ainda não programadas recebe o conjunto de estágios a serem programadas, ou seja, $K' \leftarrow K$.

$M_k = \phi$ para m = 1, 2, ..., Mk, o conjunto de máquinas em cada estágio de produção inicia vazio e recebe o conjunto de máquinas a serem programadas em cada estágio de produção, ou seja, $Mk' \leftarrow Mk$.

$\sigma_m = \phi$ para m = 1,2, ..., M, ou seja, o subconjunto de tarefas alocadas à máquina m inicia a programação vazio, para todas as máquinas, e o conjunto de tarefas

ainda não programadas recebe o conjunto de tarefas a serem programadas, ou seja, $J' \leftarrow J$.

Passo 1 – Ordene todas as tarefas do conjunto J' de acordo com a regra de prioridade TLPT. Selecione a tarefa que ocupa a primeira posição em J' ($J'_{[1]}$). Vá para o Passo 3.

Passo 2 – Ordene todas as tarefas de J' de acordo com a regra de prioridade SRD, considerando as datas de término das tarefas no estágio anterior como as datas de liberação do estágio atual. Selecione a tarefa que ocupa a primeira posição em J' ($J'_{[1]}$). Vá para o Passo 3.

Passo 3 – Em cada uma das máquinas do estágio, analise todas as possibilidades de alocação da tarefa $J'_{[1]}$ (primeira posição de J'), e escolha a máquina m com a data de término mais cedo, ou seja, aplique a regra de alocação SCT.

Passo 4 – Atualização de conjuntos.

O conjunto de tarefas alocadas à máquina m recebe a tarefa a tarefa que ocupa a primeira posição do conjunto de tarefas a serem programadas, ou seja, $\sigma_m \leftarrow \sigma_m \cup \{J'_{[1]}\}$ e a tarefa alocada à máquina m é excluída do conjunto de tarefas a serem programadas, ou seja, $J' \leftarrow J' - \{J'_{[1]}\}$.

Se o conjunto de tarefas a serem programadas ainda não estiver vazio e a programação no estágio 1 ainda não foi concluída, $J' \neq \phi$ e $k = 1$ (primeiro estágio), vá para o Passo1.

Se o conjunto de tarefas a serem programadas ainda não estiver vazio e a programação no estágio 1 foi concluída, ou seja, $J' \neq \phi$ e $k = 1$, vá para o Passo 2.

Caso contrário, a programação do estágio está concluída.

Faça o conjunto J' passar a conter novamente as n tarefas ($J' \leftarrow J$) para programação no próximo estágio.

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

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

2.2.3 Algoritmo 3 – MM – *flowtime* 3 (TSPT/SCT)

Passo 0 – Inicialização de conjuntos.

$k = \phi$ para k = 1, 2, ..., K, ou seja, o conjunto de estágios a serem programados inicia vazio e o conjunto de estágios ainda não programadas recebe o conjunto de estágios a serem programadas, ou seja, $K' \leftarrow K$.

$M_k = \phi$ para $m = 1, 2, ..., Mk$, o conjunto de máquinas em cada estágio de produção inicia vazio e recebe o conjunto de máquinas a serem programadas em cada estágio de produção, ou seja, $Mk' \leftarrow Mk$.

$\sigma_m = \phi$ para $m = 1, 2, ... M$, ou seja, o subconjunto de tarefas alocadas à máquina m inicia a programação vazio, para todas as máquinas, e o conjunto de tarefas ainda não programadas recebe o conjunto de tarefas a serem programadas, ou seja, $J' \leftarrow J$.

Passo 1 – Ordene todas as tarefas do conjunto J' de acordo com a regra de prioridade TSPT. Selecione a tarefa que ocupa a primeira posição em J' ($J'_{[1]}$).

Faça $k = 1$. Vá para o Passo 2.

Passo 2 – No estágio k, aloque a primeira tarefa de J' ($J'_{[1]}$) à máquina x, cuja programação possua a data mais cedo de término, ou seja, aplique a regra de alocação SCT.

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

Faça $k \leftarrow k + 1$. Vá para o Passo 3.

Passo 3 – Se $k > K$, vá para o Passo 4.

Caso contrário, vá para o Passo 2.

Passo 4 – Atualização de conjuntos.

O conjunto de tarefas alocadas à máquina m recebe a tarefa a tarefa que ocupa a primeira posição do conjunto de tarefas a serem programadas, ou seja, $\sigma_m \leftarrow \sigma_m \cup \{J'_{[1]}\}$ e a tarefa alocada à máquina m é excluída do conjunto de tarefas a serem programadas, ou seja, $J' \leftarrow J' - \{J'_{[1]}\}$.

Se $J' \neq \phi$, ou seja, se o conjunto de tarefas ainda não programadas não estiver vazio, volte para o estágio, isto é, faça $k = 1$ e vá para o Passo 2.

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

2.2.4 Algoritmo 4 – MM – *flowtime* 4 (TLPT/SCT)

Passo 0 – Inicialização de conjuntos.

$k = \phi$ para $k = 1, 2, ..., K$, ou seja, o conjunto de estágios a serem programados inicia vazio e o conjunto de estágios ainda não programadas recebe o conjunto de estágios a serem programadas, ou seja, $K' \leftarrow K$.

$Mk = \phi$ para $m = 1, 2, ..., Mk$, o conjunto de máquinas em cada estágio de produção inicia vazio e recebe o conjunto de máquinas a serem programadas em cada estágio de produção, ou seja, $Mk' \leftarrow Mk$.

$\sigma m = \phi$ para $m = 1, 2, ... M$, ou seja, o subconjunto de tarefas alocadas à máquina m inicia a programação vazio, para todas as máquinas, e o conjunto de tarefas ainda não programadas recebe o conjunto de tarefas a serem programadas, ou seja, $J' \leftarrow J$.

Passo 1 – Ordene todas as tarefas do conjunto J' de acordo com a regra de prioridade TLPT. Selecione a tarefa que ocupa a primeira posição em J' ($J'_{[1]}$).

Faça $k = 1$. Vá para o Passo 2.

Passo 2 – No estágio k, aloque a primeira tarefa de J' ($J'_{[1]}$) à máquina m, cuja programação possua a data mais cedo de término, ou seja, aplique a regra de alocação SCT.

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

Faça $k \leftarrow k + 1$. Vá para o Passo 3.

Passo 3 – Se $k > K$, vá para o Passo 4.

Caso contrário, vá para o Passo 2.

Passo 4 – Atualização de conjuntos.

O conjunto de tarefas alocadas à máquina *m* recebe a tarefa a tarefa que ocupa a primeira posição do conjunto de tarefas a serem programadas, ou seja, $\sigma_m \leftarrow \sigma_m \cup \{J'_{[1]}\}$ e a tarefa alocada à máquina *m* é excluída do conjunto de tarefas a serem programadas, ou seja, $J' \leftarrow J' - \{J'_{[1]}\}$.

Se $J' \neq \phi$, ou seja, se o conjunto de tarefas ainda não programadas não estiver vazio, volte para o estágio, isto é, faça $k = 1$ e vá para o Passo 2.

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

3 Experimentação computacional

3.1 Delineamento do experimento

Como não foram encontrados na literatura, algoritmos para solução do problema proposto, os algoritmos propostos foram comparados entre si e o foco da experimentação computacional foi a análise da influência da relação entre as ordens de grandeza dos tempos de processamento e de preparação das máquinas em cada método de solução, bem como a influência das ordenações iniciais no resultado da programação e a influência do procedimento da programação, uma vez que os quatro métodos de programação aqui desenvolvidos trabalham com programação estágio a estágio (métodos MM-FT_1 e MM-FT_2) e com programação tarefa a tarefa (métodos MM-FT_3 e MM-FT_4).

Na experimentação computacional foram testados 14,4 mil problemas, divididos em 144 classes definidas pelo número de tarefas (*n*), número de estágios de produção (*K*) e pelas relações entre as ordens de grandeza dos tempos de processamento e de preparação das máquinas ($O(p_i)/O(s_{ij})$).

A Tabela 1, a seguir, ilustra as relações ($O(p_i)/O(s_{ij})$) que foram definidas com base nos trabalhos de Simons Jr. (1992), Das, Gupta e Khumawala (1995), Rajendran e Ziegler (1997), Rios-Mercado e Bard (1998) e Weng, Lu e Ren (2001), reportados em Fuchigami (2005).

Tabela 1. Relação entre ordens de grandeza dos tempos de processamento e de preparação das máquinas.

Relação	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: $O(p_i)/O(s_{ij}) \geq 1$	1-99	1-120
Relação VI: $O(p_i)/O(s_{ij}) \geq 1$	1-99	1-20

Para cada classe, foram gerados aleatoriamente 100 problemas com $n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120\}$, $K \in \{4, 7\}$ e uma das seis relações de grandeza entre os tempos. Assim, 12 (alternativas do número de tarefas) × 2 (alternativas do número de estágios) × 6 (relações $O(p_i)/O(s_{ij})$) = 144 classes. A quantidade de problemas gerados em cada classe objetiva reduzir o erro amostral.

Em cada estágio, o número de máquinas paralelas idênticas varia de 2 a 5, ou seja, $Mk \in \{2, 3, 4, 5\}$. Como *Mk* foi gerado aleatoriamente com distribuição uniforme dentro deste conjunto, sua variação não altera a quantidade de classes. O número de estágios e o número de máquinas paralelas idênticas em cada estágio definem o *layout* do sistema de produção.

Para a realização dos testes, foram utilizados tempos de processamento e de preparação das máquinas gerados aleatoriamente, seguindo as relações anteriormente estabelecidas, por meio de um *software* denominado "Gerador de Arquivos" construído em linguagem Delphi 7. Para a resolução dos problemas gerados, foi construído um *software* em linguagem Delphi 7, denominado MM-*flowtime*.

3.2 Análise dos resultados

As estatísticas para comparar o desempenho dos algoritmos, foram a porcentagem de sucesso, o desvio relativo, o desvio padrão do desvio relativo e o tempo médio de computação, conforme descrito a seguir.

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 resolvidos.

O desvio relativo mede a variação correspondente à melhor solução obtida pelos métodos. Quando o desvio relativo é igual a zero para um determinado método, significa que o tempo médio de fluxo da programação é o menor, ou seja, o algoritmo apresentou a melhor programação. Entretanto, mais de um método pode fornecer a melhor programação (empates). O melhor algoritmo é aquele que apresenta o menor valor de desvio relativo médio (a média aritmética dos desvios relativos) para uma determinada classe de problemas. O desvio relativo (DR_h) de um método *h* para um determinado problema, em que D_h é o tempo de fluxo obtido pelo método *h* e D^* é o melhor tempo de fluxo obtido pelos quatro métodos, é calculado utilizando a Equação 1:

$$DR_h = \frac{D_h - D^*}{D^*} \quad (1)$$

O desvio padrão do desvio relativo é o valor da variação dos desvios relativos de uma classe de problemas em torno do desvio relativo médio.

Quanto menor for o valor do desvio padrão, melhor é o método de solução quando comparado com um outro, no caso em que ambos apresentarem desvios relativos médios com diferença não significativa. O desvio padrão (S_h) do desvio relativo de um método h, em que DR_{hi} é o desvio relativo da solução do problema i, DRM_h é o desvio relativo médio da classe de problema e L é o número de problemas da classe, é calculado da seguinte forma (Equação 2):

$$S_h = \sqrt{\frac{\sum_{i=1}^{L} (DR_{hi} - DRM_h)^2}{L-1}} \qquad (2)$$

O tempo médio de computação é 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 computacional, o tempo médio de computação foi medido em milissegundos (ms).

Conforme mencionado anteriormente, a experimentação computacional teve como foco a análise da influência das ordenações iniciais no resultado da programação, assim como a influência do procedimento de programação, e também da relação entre as ordens de grandeza dos tempos de processamento e de preparação das máquinas em cada método de solução.

Além da comparação geral entre os quatro métodos, foram efetuadas comparações entre pares de métodos, conforme segue:

- Comparação dos métodos MM-FT_1 e MM-FT_2, MM-FT_3 e MM-FT_4 para investigar a influência da ordenação inicial no resultado da programação; e
- Comparação dos métodos MM-FT_1 e MM-FT_3, MM-FT_2 e MM-FT_4 para investigar a influência da programação estágio a estágio e da programação tarefa a tarefa.

Em função dos resultados obtidos nas comparações anteriormente descritas, duas comparações adicionais foram realizadas:

- Comparação dos métodos MM-FT_1 e MM-FT_4 com o objetivo de consubstanciar os resultados obtidos nas comparações anteriores; e
- Comparação dos métodos MM-FT_2 e MM-FT_3 com o objetivo de avaliar as influências relativas da programação estágio a estágio e da ordenação inicial TSPT.

Na comparação geral dos quatro métodos MM-FT_1, MM-FT_2, MM-FT_3 e MM-FT_4, os resultados obtidos na experimentação computacional mostraram que, em geral, o método MM-FT_1 forneceu os melhores resultados em termos de porcentagem de sucesso, para todas as relações. A Tabela 1 apresenta o resultado da porcentagem de sucesso para 4 e 7 estágios com relações ($O(p_i)/O(s_{ij})$) agregadas.

A Tabela 2 contém o total geral das porcentagens de sucesso, agregando as relações ($O(p_i)/O(s_{ij})$) e o número de tarefas.

Uma comparação geral do desempenho dos métodos em todas as relações e para 4 e 7 estágios com base na porcentagem de sucesso confirmou que o método MM-FT_1 apresentou desempenho superior para todas as 144 classes de problemas testados.

A análise dos desvios relativos e dos desvios padrão dos desvios relativos confirmam as conclusões dos resultados de porcentagem de sucesso. O método MM-FT_1 apresentou em todas as relações o menor desvio relativo médio e uma boa estabilidade de desempenho.

Também foi observada uma relativa estabilidade e tendência à diminuição dos valores dos desvios

Tabela 2. Total geral de porcentagens de sucesso dos métodos.

n	*k* = 4				*k* = 7			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
10	65,33	34	0,66	0	59,16	40,16	0,66	0
20	66,33	32,83	0,83	0	66,16	33,66	0,16	0
30	68	31,83	0,16	0	66,66	32,66	0,66	0
40	67,5	31,66	0,83	0	71,66	28,16	0,16	0
50	68,5	31,16	0,33	0	71	29	0	0
60	70	29,5	0,5	0	74	25,66	0,33	0
70	75,66	24	0,16	0,16	70,83	28,66	0,5	0
80	75,33	24,66	0	0	72,83	26,83	0,33	0
90	77	22,5	0,5	0	74,5	25	0,5	0
100	78,83	21	0,16	0	75,66	23,83	0,5	0
110	78,33	21	0,66	0	71,83	27,5	0,66	0
120	76,66	22,66	0,66	0	76,33	23,66	0	0
% média	72,29	27,23	0,45	0,01	70,88	28,73	0,37	0

Tabela 3. Total geral de porcentagens de sucesso dos métodos.

n	MM-FT$_1$	MM-FT$_2$	MM-FT$_3$	MM-FT$_4$
Total de problemas	10309	4030	60	1
% total média	71,59	27,98	0,41	0,0069

relativos dos métodos MM-FT$_1$ e MM-FT$_2$ com o aumento do número de tarefas, significando que, à medida que o porte do problema aumenta, a diferença no desempenho dos métodos tende a diminuir. Por outro lado, verificou-se uma grande instabilidade dos valores dos desvios relativos médios dos métodos MM-FT$_3$ e MM-FT$_4$. Em geral, as curvas de desempenho dos problemas com 4 e 7 estágios mantêm o mesmo comportamento, indicando que o número de estágios não deve afetar o desempenho dos métodos.

A análise da porcentagem de sucesso para os métodos MM-FT$_1$ e MM-FT$_2$, que investiga a influência da ordenação inicial, mostrou que o método MM-FT$_1$, que usa a ordenação inicial TSPT, foi superior. De modo análogo, na comparação dos métodos MM-FT$_3$ e MM-FT$_4$, o método MM-FT$_3$, que também usa a ordenação inicial TSPT, forneceu os melhores resultados. Logo, pode-se concluir que para o problema em questão, a regra de ordenação inicial TSPT apresenta melhor desempenho que a TLPT.

Na análise de porcentagem de sucesso para os métodos MM-FT$_1$ e MM-FT$_3$, juntamente com MM-FT$_2$ e MM-FT$_4$, que investiga a influência do foco da programação no desempenho dos métodos, pode-se observar que o método MM-FT$_1$, que utiliza a programação das tarefas estágio a estágio, obteve desempenho superior ao método MM-FT$_3$, e que o método MM-FT$_2$, que também realiza a programação das tarefas estágio a estágio, obteve desempenho superior ao do método MM-FT$_4$. Com base nesses resultados pode-se afirmar que a programação das tarefas realizada estágio a estágio apresenta-se como melhor procedimento de programação.

Como os métodos MM-FT$_3$ e MM-FT$_4$ apresentaram um desempenho muito inferior aos métodos MM-FT$_1$ e MM-FT$_2$, foi realizada uma avaliação numérica a partir de um problema-exemplo (MORAIS, 2008, p. 167-182) com o objetivo de eventualmente detectar a causa de tal desempenho. Pode-se então verificar nos métodos MM-FT$_3$ e MM-FT$_4$, em que a programação é realizada tarefa a tarefa, que as tarefas necessitam aguardar mais tempo para terem suas operações iniciadas nos estágios subsequentes, ou seja, o tempo de espera entre operações sucessivas aumenta, fato este que ocasiona um aumento no tempo médio de fluxo.

A superioridade da programação estágio a estágio combinada com a ordenação inicial TSPT foi confirmada na comparação entre os métodos MM-FT$_1$ e MM-FT$_4$.

A comparação entre os métodos MM-FT$_2$ e MM-FT$_3$ mostrou a força da programação estágio a estágio quando comparada com a programação tarefa a tarefa, mesmo utilizando uma ordenação inicial de menor qualidade, ou seja, a TLPT relativamente à TSPT.

Quanto aos tempos de computação, pode-se observar que os métodos propostos neste trabalho apresentam uma boa eficiência computacional, uma vez que para os problemas de maior porte (120 tarefas e 7 estágios de produção) o tempo médio de computação foi de 60 ms, com um valor máximo de 92 ms (relação II).

Para todas as relações, a amplitude de variação no desempenho dos métodos, em função do número de tarefas apresenta certa estabilidade, e a ordem de superioridade se confirma. Em todos os pares de métodos comparados e analisados, a maior amplitude de variação nos desvios relativos médios e nos desvios padrão dos desvios relativos médios foi observado na Relação IV, em que os tempos de processamento são de 50 a 99 e os valores dos tempos de *setup* são de 1 a 49.

4 Considerações finais

A programação da produção em ambientes em que os tempos de preparação das máquinas não devem ser incluídos nos tempos de processamento das tarefas, uma vez que são significativamente dependentes da ordenação das tarefas nas máquinas, tende a ser um processo complexo. Grande parte das pesquisas em programação de operações em máquinas considera os tempos de tempos de preparação das máquinas como não relevantes ou de pequena variação e, geralmente, os incluem nos tempos de processamento das tarefas. Todavia, há casos em que há a necessidade de tratá-los diferenciadamente, uma vez que eles têm relação direta com a disponibilidade de equipamentos e acarretam custos específicos, como a necessidade de pessoal especializado para sua execução.

Na literatura examinada, para o ambiente de produção *flow shop* Híbrido com tempos de preparação dependentes da sequência, caso tratado neste trabalho, até o momento, nenhum trabalho que abordasse o critério de minimização do tempo médio de fluxo, foi encontrado.

Neste trabalho foram apresentados e avaliados quatro métodos heurísticos construtivos para o problema de programação em ambientes de produção *flow shop* híbrido, visando a minimização do tempo médio de fluxo, que está associado à redução do estoque em processo. Os métodos foram avaliados e comparados entre si e, como síntese, pode-se notar que:

- O Método MM-FT_1 obteve o melhor desempenho para todas as classes de problemas testados e apresentou uma relativa estabilidade quanto ao desvio relativo médio;
- O Método MM-FT_1, que apresentou melhor desempenho, utiliza a regra de prioridade TSPT, que é uma variação da regra SPT, confirma resultados de diversos estudos que apresentam esta regra como uma boa regra para minimização dos tempos de fluxos; e
- Com base nesses resultados das avaliações e comparações entre os métodos, constatou-se que a programação das tarefas, realizada estágio a estágio, apresenta-se como melhor procedimento de programação.

De um modo geral, os resultados da pesquisa mostraram que o método heurístico construtivo MM-*FlowTime*1 que utiliza a regra de prioridade TSPT para ordenação inicial das tarefas e a regra de alocação SCT em combinação com a programação estágio a estágio apresenta, com pequeno esforço computacional, uma solução de boa qualidade para o problema de programação de operações em ambiente *flow shop* híbrido com tempos de preparação das máquinas dependentes da sequência das tarefas e com o objetivo de minimização do tempo médio de fluxo.

Sugere-se para pesquisas futuras, a proposição de novos métodos heurísticos para o problema de programação em ambientes *flow shop* híbrido com tempos de preparação das máquinas dependentes da sequência das tarefas e com o objetivo de minimização do tempo médio de fluxo, eventualmente com melhores desempenhos do que os aqui apresentados, além de um estudo estrutural do ambiente objeto deste estudo.

Também como recomendações para trabalhos futuros, outras regras de prioridades poderão ser testadas, uma vez que os métodos heurísticos propostos neste trabalho utilizam apenas duas regras de prioridades, a TSPT e a TLPT.

Referências

DAS, S. R.; GUPTA, J. N. D.; KHUMAWALA, B. M. A saving index heuristic algorithm for flow shop scheduling with sequence dependent set-up times. **Journal of the Operational Research Society**, v. 46, n. 11, p. 1365-1373, 1995.

FUCHIGAMI, H. Y. **Métodos heurísticos construtivos para o problema de programação da produção em sistemas flow shop híbridos com tempos de preparação das máquinas assimétricos e dependentes da seqüência.** 2005**.** f. 135 **–** Dissertação (Mestrado) – Universidade de São Paulo, São Carlos, 2005.

MACCARTHY, B. L.; LIU, J. Y. Adressing a gap in scheduling research: a review of optimization and heuristic methods in production scheduling. **International Journal of Production Research**, v. 31, n. 1, p. 59-79, 1993.

MORAIS, M. F. **Métodos heurísticos construtivos para redução do estoque em processo em ambientes de produção *flow shop* híbridos com tempos de *setup* dependentes da seqüência.** 2008. f. 301. **–** Dissertação (Mestrado) – Universidade de São Paulo, São Carlos, 2008.

RAJENDRAN, C.; ZIEGLER, H. Heuristics for scheduling in a flowshop with setup, processing and removal times separated. **Production Planning & Control**, v. 8, n. 6, p. 568-576, 1997.

RIOS-MERCADO, R. Z.; BARD, J. F. Heuristics for the flow line problem with setup costs. **European Journal of Operational Research,** v. 110, n. 1, p. 76-98, 1998.

SIMONS Jr., J. V. Heuristics in flow shop scheduling with sequence dependent setup times. **Omega:The International Journal of Management Science,** v. 20, n. 2, p. 215-225, 1992.

SLACK, N. et al. **Administração da produção.** São Paulo: Atlas, 1999.

WENG, M. X.; LU, J.; REN, H. Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. **International Journal of Production Economics**, v. 70, n. 3, p. 215-226, 2001.