Universidade Tecnológica Federal do Paraná - UTFPR
Campus Ponta Grossa - Paraná - Brasil
ISSN 1808-0448 / v. 05, n. 04: p. 152-167, 2009

Revista Gestão Industrial

UM EFICIENTE MÉTODO HEURÍSTICO CONSTRUTIVO PARA O PROBLEMA *NO-WAIT FLOWSHOP* COM TEMPOS DE SETUP DEPENDENTES DA SEQÜÊNCIA

AN EFFICIENT CONSTRUCTIVE HEURISTIC FOR THE SEQUENCE DEPENDENT SETUP TIMES NO-WAIT FLOWSHOP PROBLEM

Daniella Castro Araújo[1]; Marcelo Seido Nagano[2]
[1]Universidade de São Paulo – USP – São Carlos – Brasil
daniella.araujo@usp.br
[2]Universidade de São Paulo – USP – São Carlos – Brasil
drnagano@sc.usp.br

Resumo

Este artigo trata do problema de programação de operações em um ambiente de produção no-wait flowshop com tempos de setup separáveis e dependentes da sequência. Uma breve revisão bibliográfica é feita acerca do tema e um novo método heurístico construtivo é proposto, com o objetivo de minimização da duração total da programação (makespan). O desempenho do método é então analisado através da experimentação computacional e comparado com o melhor método heurístico reportado na literatura para o problema, mostrando-se superior em relação à eficiência computacional para o conjunto de problemas avaliados.

Palavras-chave: métodos heurísticos, *no-wait flowshop*, *setups* dependentes, *makespan*.

1. Introdução

A primeira abordagem sistemática para problemas de programação da produção foi realizada nos anos 1950. Desde então, milhares de artigos com diferentes problemas de programação têm surgido na literatura. A maioria desses artigos assume que o tempo/custo de *setup* é insignificante ou parte do tempo de processamento. Enquanto esse pressuposto simplifica a análise e reflete certas aplicações (quando os *setups* são pequenos, independentes da seqüência e inseparáveis dos tempos de processamento), ele adversamente afeta a qualidade da solução de várias aplicações de programação da produção que necessitam do tratamento separado dos tempos de *setup* (ALLAHVERDI *et al*, 2008 e ALDOWAISAN, 2001).

Por definição, tempo de *setup* é o tempo necessário na preparação de determinado recurso (máquinas, pessoas) para executar uma atividade (operação, tarefa). Custo de *setup* é o custo para

iniciar um recurso utilizado anteriormente na execução de uma atividade. As atividades de *setup* incluem a obtenção de ferramentas, posicionamento do material a ser trabalhado, devolução de ferramentas, limpeza, ajuste de peças e acessórios, ajuste de ferramentas e inspeção de material em um sistema de manufatura, adequação do ambiente à execução de atividades em uma organização de serviços, entre outros (ALLAHVERDI; SOROUSH, 2008).

Os tempos de *setup* são classificados em dependentes da sequência e independentes da sequência. Se o tempo de *setup* é função somente da tarefa a ser processada, independentemente da tarefa anterior, é chamado de independente da sequência. Já o *setup* dependente da sequência depende tanto da tarefa a ser processada quanto da tarefa anterior (ALLAHVERDI; SOROUSH, 2008).

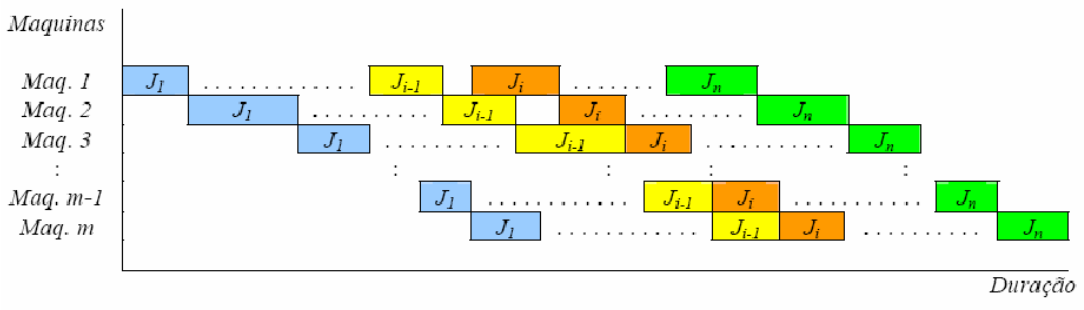
Há imensas economias na produção quando os tempos de *setup* são explicitamente incorporados nas decisões de programação da produção. Tratar os tempos de *setup* separados dos tempos de processamento permite que algumas operações sejam executadas simultaneamente, aumentando assim a utilização de recursos, o que é muito importante para os sistemas modernos de gestão da produção, como o *just-in-time* (JIT), a tecnologia de grupo, a tecnologia de produção otimizada (OPT - O*ptimized Production Technology*) e a manufatura celular.

A importância e as aplicações de modelos de sequenciamento com considerações explícitas de tempos de *setup* foram amplamente discutidas em vários estudos a partir de meados de 1960. Recentemente, Allahverdi e Soroush (2008) apresentaram um estudo com várias aplicações e benefícios envolvendo tempos de *setup* separáveis dos tempos de processamento. Allahverdi *et al* (2008) realizaram uma ampla revisão da literatura sobre problemas de programação com tempos de *setup* separáveis para todos os ambientes *shop*, continuando as pesquisas realizadas por Allahverdi *et al* (1999) e Potts e Kovalyov (2000), que cobriram mais de 300 artigos de 1999 a 2006.

As suposições comumente consideradas em problema de programação *flowshop* (BAKER (1974) e PINEDO (2002)) impedem várias aplicações práticas importantes (DUDEK *et al*, 1992). No *flowshop* clássico, considera-se que não há limite de estocagem entre as máquinas. Porém, essa situação pode não ser possível em situações reais. Em um problema de programação da produção no ambiente *no-wait flowshop*, as tarefas devem ser processadas do início ao final sem interrupções (HALL; SRISKANDARAJAH, 1996). Consequentemente, o processamento de uma tarefa na primeira máquina pode necessitar de um atraso para garantir que não ocorra interrupção em uma máquina subsequente.

A principal característica do *no-wait flowshop* é que a operação $i+1$ de uma tarefa tem que ser processada logo após o término da operação i, sendo $1 \leq i \leq m - 1$. Assim, não pode haver tempo de espera no processamento de uma tarefa de uma máquina para a próxima. Um exemplo de sequenciamento de tarefas ($1 \leq i \leq n$) para o problema *no-wait flowshop* é apresentado na Figura 1.

Figura 1 – *No-wait flowshop* com *m* máquinas e *n* tarefas



A Figura 1 apresenta o sequenciamento de um problema de *n* tarefas em *m* máquinas e considera que todas as tarefas estão disponíveis ao mesmo tempo no chão de fábrica. Uma sequência (solução) pode ser representada por $\sigma = (J_{[1]}, J_{[2]},..., J_{[i]}, J_{[i+1]},..., J_{[n]})$, onde $[i]$ representa a tarefa que aparece na *i*-ésima posição da sequência.

O ambiente *no-wait* é comum na produção de aço, plástico, químicas, produtos de alumínio e alimentos (BROWN *et al*, 2004). Por exemplo, no caso da produção de aço, o metal aquecido deve atravessar a sequência de operações continuamente, antes que esfrie, para prevenir defeitos na composição do material (ALDOWAISAN, 2001). Este ambiente é também motivado por conceitos modernos de gestão da produção, como o *just-in-time* e a produção enxuta (BIANCO *et al*, 1999).

Neste trabalho, aborda-se o problema de programação em ambiente *flowshop* com *n* tarefas e *m* máquinas com duas condições: os tempos de *setup* são considerados assimétricos e dependentes da sequência, e não ocorre interrupção entre operações da mesma tarefa quando programada nas *m* máquinas. Este problema é conhecido também como "*no-wait flowshop* com tempos de *setup* dependentes da sequência" ($Fm/ST_{sd}, no-wait/C_{\max}$). O trabalho está organizado da seguinte maneira: primeiramente, uma revisão bibliográfica acerca do problema é realizada, seguida pela apresentação da notação e definição para o caso. A seguir, é apresentado um novo método heurístico, que é comparado ao melhor método heurístico existente na literatura (Allahverdi *et al* (2008)) para o problema, o BIH – *Best Insertion Heuristic*, proposto por Bianco *et al* (1999).

2. O problema *no-wait flowshop* com tempos de *setup*

O problema *no-wait flowshop* com tempos de *setup* independentes/dependentes é mais complexo do que o problema tradicional de *flowshop*, pois quando ambas as características de *no-wait* e *setups* separados são consideradas simultaneamente, os *setups* devem ser realizados nas máquinas antes de a tarefa chegar, para satisfazer a primeira condição. Isso significa que o *setup* na máquina subseqüente ($m = i+1$) deve ser feito enquanto a tarefa está em operação na máquina precedente ($m = i$). Adicionalmente, a condição *no-wait* pode forçar atrasos na primeira máquina para garantir que a tarefa seja processada sem interrupções.

Gilmore e Gomory (1964), juntamente com Garfinkel (1986) mostraram que o problema $F2/ST_{si}, no-wait, r_j / C_{\max}$ pode ser reduzido análogo ao problema do Caixeiro Viajante (TSP – *Traveling Salesman problem*), que pode ser resolvido em tempo polinomial

Gupta (1986) mostrou que o problema $Fm/ST_{sd}/C_{\max}$ é NP-completo para os casos de ambiente *no-wait*, de limite de estocagem e de estocagem infinita entre as máquinas. Ele também formulou o problema em ambiente *no-wait* como o problema do caixeiro viajante (TSP), e usou essa formulação para encontrar soluções aproximadas para o caso.

Gupta *et al* (1997) estudaram o problema de programação da produção com uma única máquina, considerando que as tarefas pertencem a várias classes e que os pedidos dos clientes possuem pelo menos uma tarefa de cada classe. A cada mudança de classe na máquina há um tempo de *setup* e para cada pedido dos clientes há um custo de carregamento, que representa o custo de estocagem das tarefas finalizadas em um pedido ainda incompleto. Eles consideraram dois objetivos simultaneamente: minimizar o *makespan* e minimizar o custo de carregamento. Dois algoritmos em tempo polinomial foram propostos, considerando uma hierarquia no sequenciamento: um objetivo era otimizado enquanto o outro era fixado no seu valor ótimo.

Aldowaisan e Allahverdi (1998) estudaram o problema $F2/ST_{si}, no-wait/\sum F_i$ e apresentaram um critério de eliminação, soluções ótimas para dois casos especiais e uma heurística para o caso geral Allahverdi e Aldowaisan (2000) estenderam o trabalho anterior para três máquinas com uma regra de dominância melhorada, e apresentam cinco heurísticas de alto desempenho. Posteriormente, Aldowaisan (2001) apresentou uma relação de dominância global juntamente com uma heurística e um método *branch* e *bound* para o mesmo problema abordado por Aldowaisan e Allahverdi (1998). A experimentação computacional mostrou a superioridade de sua heurística e da relação de dominância local.

Bianco *et al* (1999) abordaram o problema $Fm/ST_{sd}, no-wait, r_j / C_{\max}$ e provaram que ele é equivalente ao problema assimétrico do caixeiro viajante com datas de liberação – ATSP-RT (*Asymmetric Travelling Salesman Problem with Ready Times*). Também mostraram que, mesmo quando as datas de liberação são iguais a zero, o problema é fortemente NP-*hard*. Eles apresentaram dois limitantes inferiores, LB1, que dualiza as restrições das datas de liberação, e LB2, baseado na relaxação da matriz de custo do ATSP, que representa o tempo gasto para o caixeiro viajante ir de um vértice a outro. Além disso, apresentaram dois métodos heurísticos, o BAH – *Best Adding Heuristic*, em que a seqüência solução é construída através da adição de tarefas no fim de uma seqüência parcial, e o BIH – *Best Insertion Heuristic*, em que a seqüência solução é construída através da inserção de tarefas em uma seqüência parcial Na experimentação computacional, mostraram que o método BIH apresenta resultados melhores do que o BAH, apesar

de possuir alto tempo computacional O comportamento dos limitantes inferiores depende da faixa de valores em que estão as datas de liberação e do tamanho da instância dos problemas. Em geral, nos casos mais congestionados, LB1 dá valores melhores do que LB2, e vice-versa.

Sidney *et al* (2000) estudaram o problema $F2/ST_{si}, no-wait/C_{\max}$, onde os tempos de *setup* na segunda máquina são antecipatórios e possuem duas partes. Durante a primeira fase do *setup*, a tarefa não pode estar presente na máquina, enquanto que a segunda fase dele pode ser realizada com ou sem a tarefa presente. Eles propuseram um algoritmo heurístico e mostraram que seu desempenho no pior caso equivalia à fração 4/3.

Allahverdi e Aldowaisan (2001) desenvolveram soluções ótimas para certos casos do problema $F2/ST_{sd}, no-wait/\sum C_j$. Além disso, desenvolveram uma relação de dominância e apresentaram cinco heurísticas com a complexidade computacional de $O(n^2)$ e $O(n^3)$. As heurísticas são constituídas de duas fases; na primeira fase, uma sequência inicial é desenvolvida, e na segunda uma técnica de inserção repetida é aplicada para chegar a uma solução. Também desenvolveram um algoritmo *branch-and-bound*, a fins de comparação. Os experimentos computacionais demonstraram que o conceito de inserções repetidas é bastante útil, e que as soluções para todas as sequências iniciais convergem para aproximadamente o mesmo valor após algumas iterações.

Brown *et al* (2004) apresentaram dois métodos de solução em tempo não-polinomial e uma heurística em tempo polinomial para o problema $Fm/ST_{si}, no-wait/\sum F_j$, também considerando $C_{\max}$ como critério de minimização. A heurística criada (TRIPS) tenta quebrar o problema grande em problemas menores que podem ser enumerados facilmente, examinando todas as combinações de três tarefas possíveis, escolhendo-se a seqüência (x_i, x_j, x_k) que minimiza o critério adotado. Para testá-la com o critério do *makespan*, a heurística LKH de Helsgaun (2000) foi utilizada, e para o critério do tempo total de fluxo, foi implementado um algoritmo de *simulated annealing*. Os resultados mostraram que a heurística TRIPS é recomendada para o critério de minimização da soma total dos fluxos das tarefas (*total flowtime*), mas não para o *makespan*.

Shyu *et al* (2004) estudaram o problema $F2/ST_{si}, no-wait/\sum C_j$ e transformaram-no em um modelo baseado em gráfico. Então, apresentaram duas versões de um algoritmo meta-heurístico que encontra soluções aproximadas do problema através da emulação do comportamento natural das formigas, conhecido por Otimização da Colônia de Formigas. As versões do algoritmo se diferenciavam pela presença de otimizadores locais.

França *et al* (2006) estudaram o problema em ambiente *no-wait flowshop* com *setups* dependentes da sequência e datas de liberação com o objetivo de minimizar o *makespan.* Eles desenvolveram um algoritmo genético híbrido que endereça uma nova árvore ternária completa organizada hierarquicamente para representar a população que, agregada a um operador de

recombinação, assemelha-se ao modelo de processamento paralelo que resolve problemas de otimização combinatorial Ainda apresentaram um novo modelo recursivo de busca local, que é crucial ao bom desempenho do algoritmo proposto, já que representa aproximadamente 90% do tempo de processamento computacional dele. Na experimentação computacional, mostram que o algoritmo obtém melhores resultados do que a heurística BIH, de Bianco *et al* (1999), em menor tempo computacional

Posteriormente, Ruiz e Allahverdi (2007a) abordaram o problema $Fm/ST_{si},no-wait/\sum F_j$ e apresentaram uma relação de dominância para o caso de quatro máquinas, que também pode ser adaptado para o caso de *m* máquinas. Além disso, desenvolveram cinco heurísticas e dois métodos de busca local estocástica, sendo um deles baseado em busca local iterativa (ILS - *Iterated Local Search*). Na experimentação computacional, testaram as cinco heurísticas propostas e os dois métodos de busca local juntamente com o algoritmo de otimização da colônia de formigas de Shyu *et al* (2004) adaptado para o caso de *m* máquinas e com a heurística TRIPS de Brown *et al* (2004). Como resultado, observaram que o método de busca local iterativa proposto obtém os melhores resultados, dentre todas as heurísticas comparadas. Três das cinco heurísticas propostas se mostraram superiores aos métodos de Shyu *et al* (2004) e Brown *et al* (2004), sendo que uma delas é melhor do que as demais, e se torna ainda melhor quando utilizada conjuntamente ao método de busca local iterativa.

Ruiz e Allahverdi (2007b) estudaram o problema $Fm/ST_{si},no-wait/L_{\max}$ e apresentaram sete heurísticas e quatro algoritmos genéticos, além de uma relação de dominância para o caso especial de três máquinas. Na experimentação computacional, comparam as sete heurísticas propostas com a melhor heurística de Ruiz e Allahverdi (2007a) adaptada para o caso. Os resultados mostraram que as heurísticas propostas eram inferiores à heurística de Ruiz e Allahverdi (2007a), apesar de exigirem menor esforço computacional Os algoritmos genéticos propostos se mostraram superiores aos de Ruiz e Allahverdi (2007a).

Como pode ser verificado na análise prévia da revisão da literatura, há apenas dois trabalhos científicos sobre o assunto a ser pesquisado, os trabalhos de Bianco *et al* (1999) e França *et al* (2006).

A seguir serão apresentadas as notações e expressões a serem utilizadas para a formulação do novo método heurístico para o problema tratado neste trabalho.

3. Notação e definição

Um problema de programação *no-wait flowshop* consiste em um conjunto $J = \{j_1, j_2, j_3, ..., j_n\}$ de *n* tarefas a serem processadas em um conjunto $M = \{m_1, m_2, m_3, ..., m_m\}$ de *m* máquinas, sendo que cada máquina processa apenas uma tarefa por vez. Uma determinada tarefa j_i é composta por *m* operações ($op_{ki}, op_{k+1i}, ..., op_{mi}$), a serem executadas em sequência e sem interrupção, para satisfazer a condição de *no-wait*. A operação op_{ki} deve ser executada na máquina *k*, com tempo de processamento p_{ki}.

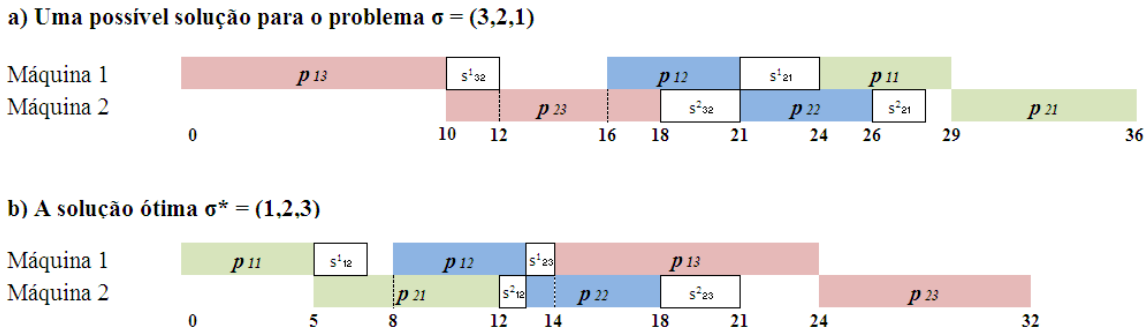
Para executar a operação op_{kj} na máquina *k*, existe um tempo de *setup* dependente da sequência (s^k_{ij}), se a operação op_{ki} for processada imediatamente antes da operação op_{kj}.

A Figura 2 apresenta um exemplo do problema, com 2 máquinas e 3 tarefas.

Figura 2 - Um exemplo do problema de programação $F2/ST_{sd}, no-wait/C_{\max}$

s^1_{ij}	j_1	j_2	j_3
j_1	-	2	4
j_2	3	-	1
j_3	3	2	-
p_{1i}	5	5	10

s^2_{ij}	j_1	j_1	j_1
j_1	-	1	2
j_2	2	-	3
j_3	4	3	-
p_{2i}	7	5	8



3.1. Formulação do método heurístico proposto

Seja σ uma sequência de J, $j_{[i]}$ é a tarefa de J que ocupa a *i*-ésima posição em σ. O intervalo de tempo entre o início da tarefa $j_{[i+1]}$ e o início da tarefa $j_{[i]}$ na máquina *k* é $\Delta t^k_{[i][i+1]}$, calculado conforme as expressões (1) e (2) abaixo:

$$\Delta t^1_{[i][i+1]} = \max_{1 \le k \le m} \left[s^k_{[i][i+1]} + \sum_{h=1}^{k} \left(p_{h[i]} - p_{h[i+1]} \right) + p_{k[i+1]} \right] \qquad (1)$$

$$\Delta t^{k}_{[i][i+1]} = \Delta t^{1}_{[i][i+1]} + \sum_{h=1}^{k-1}\left(p_{h[i+1]} - p_{h[i]}\right) \quad (2)$$

Definindo $GAP^{k}_{[i][i+1]}$ como o intervalo de tempo entre o término da tarefa $j_{[i]}$ e o início da tarefa $j_{[i+1]}$ na máquina *k*, o mesmo pode ser obtido pela expressão (3) abaixo.

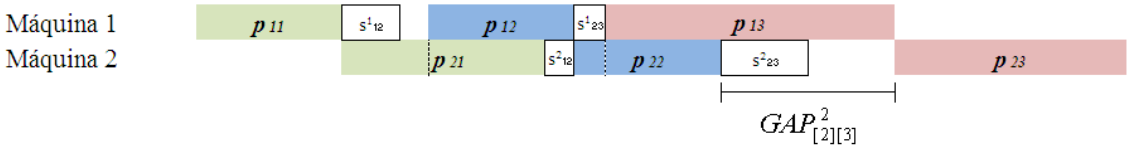
$$GAP^{k}_{[i][i+1]} = \Delta t^{k}_{[i][i+1]} - p_{k[i]} \quad (3)$$

O *GAP* da primeira tarefa da sequência na máquina *k* é definido pela expressão (4):

$$GAP^{k}_{[0][1]} = \sum_{h=1}^{k-1} p_{h[1]} \quad (4)$$

A Figura 3 ilustra o $GAP^{2}_{[2][3]}$, ou seja, o intervalo de tempo entre o término da tarefa j_2 e o início da tarefa j_3 na máquina 2.

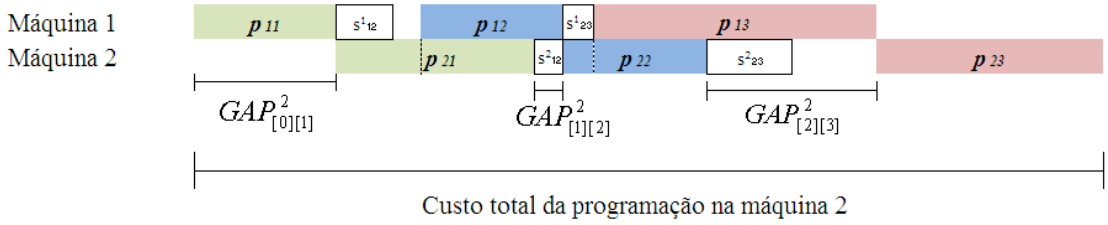
Figura 3 - Exemplo do cálculo do *GAP*



3.2 Método heurístico proposto

Define-se a duração da programação em uma máquina *k* como o "custo total da programação na máquina *k*". Assim, o custo total da programação compreende o somatório dos *GAP*s da máquina *k* e dos tempos de processamento das operações sequênciadas na máquina *k*. Nota-se que o custo total da programação na última máquina é equivalente ao *makespan.* Na Figura 4, ilustra-se o cálculo do custo total na máquina 2.

Figura 4 - Exemplo do cálculo do custo total da programação



O método heurístico proposto obtem uma sequência de *n* tarefas por meio de um processo de *n* iterações. A cada iteração, o método busca a sequência de menor custo total da programação na última máquina, inserindo todas as tarefas não-sequênciadas em todas possíveis posições da sequência.

Para o presente trabalho o método proposto será chamado de GAP, sendo composto pelos seguintes passos:

Dado um conjunto $J = \{j_1, j_2, j_3, ..., j_n\}$ de n tarefas, σ o conjunto das tarefas programadas e U o conjunto das tarefas não-programadas.

Passo 1: $U \leftarrow j_i$; $\sigma \leftarrow \emptyset$

Passo 2: Enquanto $U \neq \emptyset$, fazer:

Passo 2.1: Calcule o custo total na última máquina para todas as possíveis inserções em σ de cada tarefa $j_i \in U$, sendo h a posição relativa de inserção.

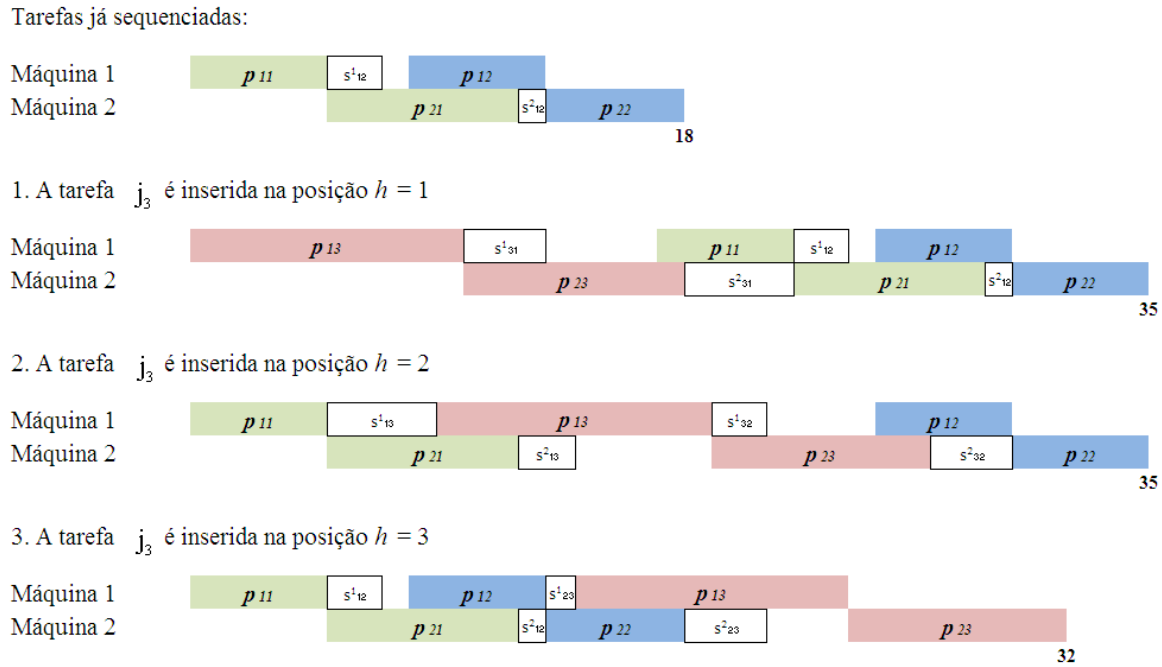
Passo 2.2: Escolha a tarefa $j_i \in U$ que forneça o menor custo total na última máquina.

Passo 2.3: Insira esta tarefa na posição h da sequência σ.

Passo 2.4: $U \leftarrow U - j_i$

A Figura 5 ilustra o funcionamento do método. No exemplo, as tarefas j_1 e j_2 já foram programadas ($\sigma = \{ j_1, j_2 \}$) e a tarefa j_3 é programada nas possíveis posições da sequência (h). Observa-se, através da Figura, que a posição de menor custo total é a de $h = 3$. Assim, a tarefa j_3 será inserida na posição $h = 3$ da sequência ($\sigma = \{ j_1, j_2, j_3 \}$).

Figura 5 - Exemplo numérico do método proposto



4. Experimentação computacional e análise dos resultados

Para a avaliação dos métodos BIH e GAP utilizou-se um banco de dados constituído de 720 problemas-teste. O banco de dados foi composto por 18 classes (*n* x *m*), $n \in \{10, 20, 30, 40, 50, 100\}$ e $m \in \{5, 10, 20\}$, com 10 problemas cada. As classes foram divididas em quatro instâncias, SSD-10, SSD-50, SSD-100 e SSD-125, de acordo com a porcentagem do tempo máximo de *setup* em relação ao tempo máximo de processamento. Por exemplo, na instância SSD-10 o tempo máximo de setup representa 10% do tempo máximo de processamento. As instâncias SSD-50, SSD-100 e SSD-125 possuem tempos máximos de *setup* correspondentes a 50%, 100% e 125% do tempo máximo de processamento, respectivamente.

Os métodos heurísticos foram codificados em linguagem *Python* e processados conjuntamente em um microcomputador *Pentium IV 3.00GHz, 512MB de RAM*.

Os tempos de processamento das tarefas e os tempos de *setup* das foram gerados aleatoriamente no intervalo conforme uma distribuição uniforme. O intervalo de variação dos tempos de processamento foi de [1, 99], e dos tempos de *setup* foram de [1, 9], [1, 49], [1, 99] e [1, 124] para as classes de problemas SSD-10, SSD-50, SSD-100 e SSD-125, respectivamente.

As estatísticas utilizadas para avaliar o desempenho dos métodos foram a Porcentagem de Sucesso, o Desvio Relativo Médio e o Tempo Médio de Computação.

A Porcentagem de Sucesso é definida pelo quociente entre o número de problemas para os quais um determinado método obteve a melhor solução (menor *makespan*) e o número total de programas resolvidos. Logo, 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 médio que o método *h* obtém em relação ao melhor *makespan* obtido para o mesmo problema, sendo calculado conforme segue:

$$DR_h(\%) = \frac{D_h - D*}{D*} \times 100 \qquad (3)$$

onde:

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

$D*$ é o melhor *makespan* obtido pelo(s) método(s) para um determinado problema.

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

A Tabela 1 apresenta os resultados obtidos dos métodos BIH, de Bianco *et al* (1999) e GAP para as instâncias SSD-10, SSD-50, SSD-100 e SSD-125, respectivamente.

Tabela 1 - Resultados para as classes de problemas SSD-10, SSD-50, SSD-100 e SSD-125

$n \times m$	SSD-10		SSD-50		SSD-100		SSD-125	
	BIH	**GAP**	**BIH**	**GAP**	**BIH**	**GAP**	**BIH**	**GAP**
10 x 5	100*	100	100	100	100	100	100	100
	0**	0	0	0	0	0	0	0
	0,09***	0,01	0,09	0,01	0,09	0,01	0,09	0,01
10 x 10	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	0,22	0,02	0,22	0,02	0,22	0,02	0,22	0,02
10 x 20	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	0,65	0,06	0,65	0,06	0,64	0,05	0,63	0,05
Média	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	0,32	0,03	0,32	0,03	0,31	0,03	0,31	0,03
20 x 5	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	1,26	0,11	1,26	0,11	1,30	0,11	1,27	0,11
20 x 10	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	2,78	0,15	2,80	0,15	2,81	0,14	2,82	0,15
20 x 20	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	7,79	0,26	7,72	0,26	7,75	0,26	7,75	0,26
Média	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	3,94	0,17	3,93	0,17	3,95	0,17	3,95	0,17
30 x 5	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	5,04	0,41	5,57	0,42	5,09	0,42	5,10	0,42
30 x 10	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	12,51	0,51	12,65	0,50	12,69	0,51	12,59	0,50
30 x 20	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	37,39	0,78	37,49	0,79	37,34	0,77	37,60	0,79
Média	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	18,31	0,57	18,57	0,57	18,37	0,57	18,43	0,57
40 x 5	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	21,58	1,22	15,13	1,16	15,10	1,16	15,17	1,16
40 x 10	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	37,29	1,30	37,44	1,31	37,80	1,32	37,98	1,33
40 x 20	100	100	100	100	100	100	100	100

	0	0	0	0	0	0	0	0
	114,81	1,46	115,00	1,85	115,06	1,87	114,99	1,86
Média	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	57,89	1,46	55,86	1,44	55,99	1,45	56,05	1,45
50 x 5	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	55,00	2,87	55,46	2,89	55,73	2,88	55,86	2,88
50 x 10	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	105,07	3,09	105,84	3,14	105,90	3,11	106,32	3,11
50 x 20	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	272,30	3,84	272,95	3,88	273,43	3,85	274,04	3,85
Média	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	144,12	3,27	144,75	3,30	145,02	3,28	145,41	3,28
100 x 5	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	1217,10	40,70	1233,57	41,15	1233,57	41,15	1236,28	40,93
100 x 10	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	1983,20	39,97	1997,64	40,67	1997,64	40,67	2006,29	40,36
100 x 20	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	4508,41	43,73	4549,57	43,86	4549,57	43,86	4565,64	44,09
Média	100	100	100	100	100	100	100	100
	0	0	0	0	0	0	0	0
	2569,57	41,47	2581,15	41,58	2593,59	41,90	2602,73	41,80

* Porcentagem de Sucesso (%)
** Desvio Médio Relativo (%)
*** Tempo Médio Computacional (segundo)

A Tabela 2 apresenta a relação da razão entre os tempos de computação dos métodos BIH e GAP (BIH/GAP). Essa relação permite verificar o desempenho computacional para a obtenção da melhor solução.

Tabela 2 - Relação BIH/GAP dos tempos de computação

$n \times m$	SSD-10	SSD-50	SSD-100	SSD-125
10 x 5	9.00	9.00	9.00	9.00
10 x 10	11.00	11.00	11.00	11.00
10 x 20	10.83	10.83	12.80	12.60
Média	10.67	10.67	10.33	10.33
20 x 5	11.45	11.45	11.82	11.55
20 x 10	18.53	18.67	20.07	18.80
20 x 20	29.96	29.69	29.81	29.81
Média	23.18	23.12	23.24	23.24
30 x 5	12.29	13.26	12.12	12.14
30 x 10	24.53	25.30	24.88	25.18
30 x 20	47.94	47.46	48.49	47.59
Média	32.12	32.58	32.23	32.33
40 x 5	17.69	13.04	13.02	13.08
40 x 10	28.68	28.58	28.64	28.56
40 x 20	78.64	62.16	61.53	61.82
Média	39.65	38.79	38.61	38.66
50 x 5	19.16	19.19	19.35	19.40
50 x 10	34.00	33.71	34.05	34.19
50 x 20	70.91	70.35	71.02	71.18
Média	44.07	43.86	44.21	44.33
100 x 5	29.90	29.98	29.98	30.20
100 x 10	49.62	49.12	49.12	49.71
100 x 20	103.10	103.73	103.73	103.55
Média	61.96	62.08	61.90	62.27

Analisando a Tabela 1, verifica-se que os dois métodos heurísticos apresentam os mesmos desempenhos em relação à porcentagem de sucesso e à porcentagem de desvio relativo médio, uma vez que ambos obtêm soluções idênticas. O método BIH proposto por Bianco *et al* (1999) apresenta elevando tempo computacional: como pode ser observado no maior problema (100 x 20), o tempo médio de computação foi de aproximadamente de 2600 segundos, enquanto que o método GAP resolve o mesmo problema com o tempo aproximado de 42 segundos.

A Tabela 2 apresenta de forma resumida a relação de eficiência computacional observada na experimentação computacional entre os dois métodos avaliados.

6. Considerações Finais

Neste trabalho foi apresentado e avaliado um novo método heurístico, denominado GAP, para o problema de programação da produção em sistemas *no-wait flowshop* com tempos de *setup* dependentes da sequência, com o objetivo de minimizar a duração total da programação. O método

GAP foi comparado ao melhor método heurístico construtivo existente na literatura, o BIH - *Best Insertion Heuristic*, de Bianco *et al* (1999).

O principal aspecto a ser destacado neste artigo refere-se ao tempo computacional Os resultados experimentais mostraram que o novo método heurístico obteve tempo computacional menor do que o BIH na ordem aproximada de 1/100 para os maiores problemas.

O método GAP, além de obter soluções viáveis e de boa qualidade, idênticas ao BIH, apresenta maior eficiência computacional e simplicidade na sua implementação. Assim, o método GAP pode ser aplicado em ambientes práticos, visto que o excessivo tempo computacional é um dos fatores que impossibilitam a aplicação dos métodos já existentes.

Abstract

This paper addresses the *m*-machine no-wait flowshop problem where the setup time of a job is separated from its processing time and sequence dependent. A briefly literature review is made and a new constructive heuristic method is proposed, considering the makespan objective. The proposed method is evaluated through computacional experiments and compared with the best constructive heuristic repported in the literature to the problem and found to be more efficient to the set of problems assessed.

Key-words: heuristic, no-wait flowshop, dependent setup, makespan.

Referências

ALDOWAISAN, T. A new heuristic and dominance relations for no-wait flowshops with setups. **Computers & Operations Research**, v.28, n.6, p.563-584, 2001.

ALDOWAISAN, T.; ALLAHVERDI, A. Total flowtime in no-wait flowshops with separated setup times. **Computers & Operations Research**, v. 25, n.9, p.757-765, 1998.

ALLAHVERDI, A.; ALDOWAISAN, T. No-wait and separate setup three-machine flowshop with total completion time criterion. **International Transactions in Operational Research**, v.7, n.3, p.245-264, 2000.

ALLAHVERDI, A.; ALDOWAISAN, T. Minimizing total completion time in a no-wait flowshop with sequence-dependent additive changeover times. **Journal of the Operational Research Society**, v.52, n. 4, p.449–462, 2001.

ALLAHVERDI, A.; GUPTA, J. N. D.; ALDOWAISAN, T. A review of scheduling research involving setup considerations. **Omega-International Journal of Management Science**, *v.* 27, n.2, p.219-239, 1999.

ALLAHVERDI, A.; NG, C.T.; CHENG, T.C.E.; KOVALYOV, M.Y. A survey of scheduling problems with setups times or costs. **European Journal of Operational Research**, v.187, n.3, p.985-1032, 2008.

ALLAHVERDI, A.; SOROUSH, H.M. The significance of reducing setup times/setup costs. **European Journal of Operational Research**, v.187, n.3, p.978-984, 2008.

BAKER, K.R. **Introduction to sequencing and scheduling**. New York: John Wiley & Sons, 1974.

BIANCO, L.; DELL'OLMO, P.; GIORDANI, S. Flow shop no-wait scheduling with sequence dependent setup times and release dates. ***INFOR Journal***, v.37, n.1, p.3-19, 1999.

BROWN, S.I.; MCGARVEY, R.; VENTURA, J.A. Total flowtime and makespan for a no-wait m-machine flowshop with set-up times separated. **Journal of the Operational Research Society**, v.55, n.6, p.614-621, 2004.

DUDEK, R.A.; PANWALKAR, S.S.; SMITH, M.L. The lessons of flowshop scheduling research. **Operations Research**, v.40, n.1, p.7-13, 1992.

FRANÇA, P.M.; TIN JR, G.; BURIOL, L.S. Genetic Algorithms for the no-wait flowshop sequencing problem with time restrictions. **International Journal of Production Research**, v.44, n.5, p.939-957, 2006.

GARFINKEL, R.S. **Motivation and modeling**. New York: John Wiley & Sons, 1983.

GILMORE, P.C.; GOMORY, R.E. Sequencing a one-state variable machine: a solvable case of the traveling salesman problem. **Operations Research**, v.12, n.5, p.655-679, 1964.

GUPTA, J. N. D. Flowshop schedules with sequence dependent setup times. **Journal of the Operational Research Society in Japan**, v.29, n.3, p.206–19, 1986.

HALL, N. G.; SRISKANDARAJAH, C. A Survey of Machine Scheduling Problems with Blocking and No-wait in Process. **Operations Research**, v.44, n.3, p.510-525, 1996.

PINEDO, M. **Scheduling: theory, algorithms, and systems**. 2. ed. New Jersey: Prentice Hall, 2002.

POTTS, C.N.; KOVALYOV, M.Y. Scheduling with batching: A review. **European Journal of Operational Research**, v.120, n.2, p.228–349, 2000.

RUIZ, R.; ALLAHVERDI, A. Some effective heuristics for no-wait flowshops with setup times to minimize total completion time. **Annals of Operations Research**, v.156, n.1, p.143-171, 2007a.

RUIZ, R.; ALLAHVERDI, A. No-wait flowshop with separate setup times to minimize maximum lateness. **The International Journal of Advanced Manufacturing Technology**, v. 35, n.5-6, p.551-565, 2007b.

SHYU, S. J.; LIN, B. M. T.; YIN, P. Y. Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time. **Computers and Industrial Engineering**, v.47, n.2-3, p.181–193, 2004.

SIDNEY, J.B.; POTTS, C.N.; SRISKANDARAJAH, C. A heuristic for scheduling two-machine no-wait flow shops with anticipatory setups. **Operations Research Letters**, v.26, n.4, p.165-173, 2000.

DADOS DOS AUTORES

Nome completo: **Daniella Castro Araújo**

Filiação institucional: Escola de Engenharia de São Carlos - EESC - USP

Departamento: Engenharia de Produção

Função ou cargo ocupado: Aluna de Mestrado

Endereço completo para correspondência (bairro, cidade, estado, país e CEP): Av. Trabalhador Sãocarlense, 400, Parque Arnold Schimidt. São Carlos - SP - Brasil - 13566-590

Telefones para contato: (16) 8807-2035

e-mail: daniella.araujo@usp.br

Nome completo: **Marcelo Seido Nagano**

Filiação institucional: Escola de Engenharia de São Carlos - EESC - USP

Departamento: Engenharia de Produção

Função ou cargo ocupado: Professor Doutor

Endereço completo para correspondência (bairro, cidade, estado, país e CEP): Av. Trabalhador Sãocarlense, 400, Parque Arnold Schimidt. São Carlos - SP - Brasil - 13566-590

Telefones para contato: (16) 3373-9428

e-mail: drnagano@sc.usp.br