UM MODELO DE PROGRAMAÇÃO INTEIRA MISTA PARA A PROGRAMAÇÃO DA PRODUÇÃO EM *FLOWSHOP* HÍBRIDO COM *BUFFERS* LIMITADOS.

Pedro Luis Miranda Lugo [a,*]
Karim Pérez Martínez [a]
Rodolfo Florence Teixeira Jr [a]
[a] Universidade Federal de São Carlos *campus* Sorocaba
Departamento de Engenharia de Produção
Rodovia João Leme dos Santos (SP-264), Km 110 - *Sorocaba* - SP - Brasil - CEP 18052-780.
pmiranda@ufscar.br, karim@ufscar.com.br, rodolfo.florence@ufscar.br
[*] Autor Correspondente. Tel.: +55 15 81353807

RESUMO

Um modelo de programação inteira mista (MIP, *Mixed Integer Programming*) é apresentado para resolver o problema de programação da produção em *Flowshop* Híbrido (HFS, *Hybrid Flowshop*) com máquinas paralelas não relacionadas, tempos de preparação antecipatórios e não antecipatórios dependentes da sequência, tempos de transporte, *buffers* limitados, tempos de liberação e elegibilidade de máquinas. O critério de otimização do modelo é a minimização do *makespan*, isto é, a duração total do programa de produção. Um exemplo numérico é apresentado para ilustrar a aplicabilidade do modelo proposto, assim como algumas características relevantes. Os resultados da avaliação global do modelo, sobre um conjunto de 1.728 instâncias, indicam que o modelo MIP é computacionalmente viável para a resolução de instâncias menores.

Palavras-chave: Programação inteira mista, Programação da produção, *Flowshop* Híbrido, *buffers* limitados.

Área Principal: Administração e Gestão da Produção, Programação Matemática, Otimização Combinatória.

ABSTRACT

A mixed integer programming model is presented in order to solve the Hybrid Flowshop (HFS) scheduling problem with unrelated parallel machines, anticipatory and non-anticipatory sequence-dependent setup times, transportation times, limited buffers, release time for machines and machine eligibility. The objective criterion is the minimization of the makespan. A numerical example is presented to illustrate the applicability of the proposed model, as well as some relevant characteristics. The results of the model evaluation, on a set of 1.728 instances, indicate that the MIP model is computationally viable just to solve small instances.

Keywords: Mixed Integer Programming, scheduling, Hybrid Flowshop, Limited buffers.

Main area: Administration & Production Management, Mathematical Programming, Combinatorial Optimization.

1. Introdução

Em um *Flowshop*, um conjunto de tarefas deve ser processado através de múltiplas estações na mesma ordem, desde a primeira até a última estação, e cada estação tem unicamente uma máquina. Porém, como salientado por Ribas, Leisten e Framiñan (2010), em algumas indústrias a necessidade de incrementar ou equilibrar a capacidade das estações tem levado à duplicação de máquinas. Em outras, a crescente demanda de produtos customizados tem desencadeado a necessidade de comprar máquinas adicionais para algumas estações com o objetivo de dedica-las à produção destes produtos. Em qualquer caso, a aquisição das novas máquinas não implica a remoção do equipamento existente, o que leva à coexistência de várias máquinas em várias estações. Esta nova configuração de produção é conhecida como *flowshop* híbrido (HFS, pelas siglas em inglês) e é encontrada em diferentes ambientes de manufatura, incluindo produção de revestimentos cerâmicos (Ruiz & Maroto, 2006), inserção de componentes eletrônicos em linhas de montagem de televisão (Yaurima, Burtseva, & Tchernykh, 2009), fabricação de produtos eletrônicos, indústria de embalagens, setor farmacêutico, fabricação de recipientes de vidro, montagem de automóveis, montagem de placas de circuito impresso (*PCB*), conformação de metais, entre outros (Chen & Chen, 2009).

O HFS pode ser considerado como a combinação de dois problemas particulares de programação da produção: O problema de máquinas paralelas e o problema do *Flowshop*. No problema de máquinas paralelas, a decisão-chave é a alocação de tarefas a máquinas, enquanto no problema do *Flowshop* a decisão chave é o sequenciamento das tarefas através do chão de fábrica. Portanto, em HFS as decisões chaves são designar e programar as tarefas às máquinas em cada estação, isto é, determinar a ordem na qual as tarefas serão processadas nas diferentes máquinas de cada estação segundo um ou vários critérios de otimização requeridos (Ribas et al., 2010).

Urlings, Ruiz e Stützle (2010) indicam que grande parte da literatura considera problemas de programação da produção básicos como uma única máquina, máquinas paralelas ou *flowshop*, os quais representam uma versão simplificada da maioria dos ambientes de produção. Por tanto, nesta pesquisa é estudado um HFS que considera diversas restrições comuns em sistemas reais de produção, mas que na literatura não são frequentemente estudadas de forma conjunta.

- Máquinas paralelas não relacionadas: O tempo de processamento das tarefas varia para cada máquina em cada estação;
- Tempos de preparação dependentes da sequência: Há uma preparação em cada máquina antes de processar a seguinte tarefa, cuja duração depende tanto da tarefa atualmente sendo processada em tal máquina, quanto da tarefa a ser programada;
- Elegibilidade de máquinas: Nem todas as máquinas em cada estação são capazes de processar todas as tarefas (Ruiz & Maroto, 2006);
- Tempos de liberação das máquinas: Nenhuma tarefa pode iniciar seu processamento em uma máquina até que tal máquina tenha sido liberada pelo programa de produção anterior (Urlings, 2010);
- Preparações antecipatórias e não antecipatórias: Na maioria das vezes, a preparação da máquina pode ser realizada no mesmo momento no qual fica disponível. No entanto, às vezes a preparação pode ser realizada somente quando tanto a tarefa quanto a máquina estão disponíveis. O primeiro caso é denominado preparação antecipatória, enquanto o segundo é denominado preparação não antecipatória (Naderi, Zandieh, & Shirazi, 2009);

- *Buffers* limitados: Se não há espaço disponível no *buffer* para uma tarefa finalizada em uma máquina de uma estação dada, então a tarefa deverá permanecer nessa máquina até que um espaço no *buffer*, ou alguma máquina da seguinte estação, encontre-se disponível (Wang & Tang, 2009);
- Tempo de transporte entre estações: O tempo de carga e descarga está incluído no tempo de transporte, o qual é assumido ser independente da tarefa, isto é, somente depende da distância a ser percorrida entre duas estações consecutivas (Naderi et al., 2009).

O critério a ser otimizado é o *makespan*, o qual é de amplo interesse prático dado que sua minimização é equivalente à maximização taxa de utilização das máquinas (Pinedo, 2008).

2. Descrição do problema

Formalmente, um conjunto N de tarefas, $N=\{1,...,n\}$, deve ser processado em um conjunto M de estações, $M=\{1,...,m\}$. Em cada estação i, $i\in M$, há um conjunto $M_i=\{1,...,m_i\}$ de processadores paralelos não relacionados. Os processadores podem ser máquinas ou *buffers*, sendo os últimos considerados máquinas especiais com tempos de processamento e de preparação iguais a zero. E_{ij} denota o conjunto de processadores capazes de processar a tarefa j, $j\in N$, na estação i. O parâmetro rm_{il} indica o tempo de liberação do processador l, $l\in M_i$, na estação i. Cada tarefa deve ser processada por exatamente um dos processadores paralelos em cada estação i. O tempo de processamento da tarefa j, no processador l da estação i é denotado como p_{ilj}. O parâmetro S_{iljk} denota o tempo de preparação entre as tarefas j e k, $k\in N$, no processador l da estação i. Adicionalmente, o parâmetro binário A_{iljk} indica se a preparação entre as tarefas j e k, no processador l da estação i é antecipatória ou não antecipatória. Operações de transporte entre estações devem ser feitas antes de iniciar o processamento de qualquer tarefa, assim t_{iq} indica o tempo de transporte entre as estações i e q, $q\in M$. O critério de otimização é o $C_{\max}$, tal que $C_{\max}=\max_{j\in N}\{C_j\}$ e C_j indica o tempo de finalização da tarefa j na última estação.

A abordagem unificada de considerar conjuntamente máquinas e *buffers* como processadores foi utilizada anteriormente por Sawik (2000, 2002). Como resultado, o problema de programação da produção com *buffers* limitados é transformado em um problema sem *buffers*, mas com bloqueio. Neste caso, se uma tarefa é finalizada em um processador de uma estação dada e não há um processador disponível na seguinte estação, a tarefa deve permanecer na estação atual, evitando que o processador possa processar outra tarefa, até que algum processador da seguinte estação esteja disponível. O tempo de bloqueio de uma máquina com tempo de processamento zero denota o tempo de espera da correspondente tarefa no *buffer* representado por essa máquina. Se esse tempo for zero, a tarefa não precisa esperar no *buffer*.

Um caso especial do HFS é o *Flowshop* com múltiplos processadores (FSMP), no qual as máquinas disponíveis em cada estação são idênticas. Gupta (1988) mostrou que o FSMP com somente duas estações é NP-Hard, incluso quando uma das estações contém uma única máquina. Por tanto, conclui-se que o problema aqui estudado é também NP-Hard. Em decorrência do anterior, a maioria das pesquisas relacionadas com programação da produção em HFS tem se focado no desenvolvimento de métodos heurísticos e meta-heurísticos para encontrar soluções de

boa qualidade em tempos computacionais acetáveis. Mesmo assim, alguns estudos utilizando modelos de programação inteira podem ser mencionados.

Sawik (2000) apresenta diversos modelos de programação matemática para o problema de programação da produção em *Flowline* flexíveis com bloqueio, considerando máquinas paralelas idênticas e visando a minimização do *makespan.* Exemplos numéricos ilustrando a viabilidade dos diferentes modelos são resolvidos utilizando o CPLEX 6.5.2 com configuração padrão e tempo limite de 3.600 segundos. Os resultados encontrados indicam o solver nem sempre foi capaz de provar a otimalidade da solução encontrada dentro do limite de tempo considerado.

Sawik (2002) apresenta um modelo MIP para a minimização do *makespan* em *Flowlines* flexíveis com buffers intermediários finitos, máquinas paralelas idênticas e produção em batelada, isto é, produtos idênticos (do mesmo tipo ou família) são programados consecutivamente. O modelo foi implementado no AMPL e resolvido pelo CPLEX 7.1. Alguns testes com as configurações do CPLEX, visando acelerar o processo de solução, indicaram que melhores resultados foram atingidos com a estratégia de seleção de nós de busca em profundidade. Finalmente, o autor destaca que instâncias de tamanho real precisam de muito tempo computacional, fazendo inviável sua utilização em problemas práticos.

Um *Flowshop* híbrido e flexível com máquinas paralelas não relacionadas, tempos de preparação dependentes da sequência, elegibilidade de máquinas, restrições de precedência, tempos de liberação de máquinas e tempos mínimos de transferência foi estudado em (Ruiz, Serifoglu, & Urlings, 2008). O critério de otimização é o *makespan.* Um modelo MIP é presentado e resolvido para instâncias de pequeno porte, utilizando o *solver* CPLEX 9.1. Para a resolução de instâncias de grande porte, tradicionais regras de liberação e a heurística construtiva NEH (Nawaz, Enscore, & Ham, 1983) são consideradas.

Jungwattanakit, Reodecha, Chaovalitwongse e Werner (2009) estudam um *Flowshop* híbrido com máquinas paralelas não relacionadas e tempos de preparação dependentes da sequência, visando minimizar a soma convexa do *makespan* e o número de tarefas atrasadas. Um modelo MIP é proposto e algumas instâncias de pequeno porte são consideradas. Para instâncias de médio e grande porte, métodos heurísticos e meta-heurísticos são testados. O modelo MIP foi implementado no AMPL e resolvido com o solver CPLEX 8.0.0. Os resultados do MIP indicam que somente instâncias de até 7 tarefas e 4 estações podem ser resolvidas em tempos computacionais acetáveis. Entre os métodos heurísticos, as heurísticas construtivas foram superiores às simples regras de liberação. Finalmente, entre as meta-heurísticas o *Simulated Annealing* apresenta os melhores resultados.

3. Modelo proposto de programação inteira mista

Nesta seção apresentamos um modelo de programação inteira mista para o HFS estudado nesta pesquisa. Este modelo está baseado no modelo proposto por Ruiz, Şerifoğlu e Urlings (2008), o qual considera máquinas paralelas não relacionadas, tempos de preparação dependentes da sequência, elegibilidade de máquinas, restrições de precedência, tempos de liberação das máquinas e tempos de transferência entre estações.

O modelo apresentado nesta seção não considera flexibilidade, restrições de precedência nem tempos de transferência, mas inclui bloqueio devido ao espaço limitado de armazenagem intermediária e tempo de transporte entre estações. Portanto, o modelo proposto não é uma extensão, mas uma variação do modelo original.

O modelo é testado em um *benchmark* de 1.728 instâncias e os resultados são utilizados para determinar a viabilidade e desempenho do modelo proposto como ferramenta de solução.

Índices e Conjuntos:

$j,k,h =$ tarefas;

$i,q =$ estações;

$l =$ processadores;

$N =$ conjunto de tarefas;

$M =$ conjunto de estações;

$M_i =$ conjunto de processadores paralelos na estação i;

$E_{ij} =$ conjunto de processadores elegíveis para a tarefa j na estação i;

$G_{il} =$ conjunto de tarefas que podem ser processadas no processador l da estação i:

$$G_{il} = \{ j \mid l \in E_{ij} \}$$

Parâmetros:

$r_{il} =$ data de liberação do processador l da estação i;

$p_{ilj} =$ tempo de processamento da tarefa j no processador l da estação i;

$S_{iljk} =$ tempo de preparação entre as tarefas j e k no processador l da estação i;

$t_{iq} =$ tempo de transporte entre as estações i e q;

$$A_{iljk} = \begin{cases} 1, \text{ se a preparação entre as tarefas } j \text{ e } k \text{ no processador } l \text{ da estação } i \text{ é anticipatória} \\ 0, \text{ caso contrário} \end{cases}$$

$FS(LS) =$ Primeira (última) estação do sistema.

Variáveis de Decisão:

$$X_{iljk} = \begin{cases} 1, \text{ se a tarefa } j \text{ precede a tarefa } k \text{ no processador } l \text{ da estação } i \\ 0, \text{ caso contrário} \end{cases}$$

$C_{ij} =$ tempo de finalização da tarefa j na estação i;

$d_{ij} =$ tempo de saída da tarefa j da estação i;

$C_{\max} =$ *Makespan.*

$$\min C_{\max} \tag{3.1}$$

Sujeito a:

$$\sum_{\substack{j \in \{N,0\} \\ j \neq k}} \sum_{l \in E_{ij} \cap E_{ik}} X_{iljk} = 1, \qquad k \in N, i \in M \tag{3.2}$$

$$\sum_{\substack{j \in N \\ j \neq k}} \sum_{l \in E_{ij} \cap E_{ik}} X_{ilkj} \leq 1, \qquad k \in N, i \in M \tag{3.3}$$

$$\sum_{\substack{h \in \{G_{il},0\} \\ h \neq j, h \neq k}} X_{ilhj} \geq X_{iljk}, \qquad j,k \in N, j \neq k, i \in M, l \in E_{ij} \cap E_{ik} \tag{3.4}$$

$$\sum_{l \in E_{ij} \cap E_{ik}} (X_{iljk} + X_{ilkj}) \leq 1, \qquad j \in N, k = j+1, \ldots, n, i \in M \tag{3.5}$$

$$\sum_{k \in G_{il}} X_{il0k} \leq 1, \qquad i \in M, l \in M_i \tag{3.6}$$

$$C_{i0} = 0, \qquad i \in M \tag{3.7}$$

$$C_{ik} + B(1 - X_{iljk}) \geq rm_{il} + (1 - A_{iljk})S_{iljk} + p_{ilk}, \quad k \in N, i \in M, l \in E_{ik}, \; j \in \{G_{il}, 0\}, j \neq k \tag{3.8}$$

$$C_{ik} + B(1 - X_{iljk}) \geq d_{ij} + S_{iljk} + p_{ilk}, \quad k \in N, i \in M, l \in E_{ik}, \; j \in \{G_{il}, 0\}, j \neq k \tag{3.9}$$

$$C_{ik} + B(1 - X_{iljk}) \geq C_{i-1,k} + t_{i-1,i} + (1 - A_{iljk})S_{iljk} + p_{ilk}, \quad k \in N, i \in \{M \setminus FS\}, \; l \in E_{ik}, j \in \{G_{il}, 0\}, j \neq k \tag{3.10}$$

$$d_{i-1,k} = C_{ik} - \sum_{\substack{j \in \{N,0\} \\ j \neq k}} \sum_{l \in E_{ij} \cap E_{ik}} X_{iljk} \cdot p_{ilk} - \sum_{\substack{j \in \{N,0\} \\ j \neq k}} \sum_{l \in E_{ij} \cap E_{ik}} X_{iljk}(1 - A_{iljk})S_{iljk} - t_{i-1,i}, \quad k \in N, i \in \{M \setminus FS\} \tag{3.11}$$

$$d_{ik} \geq C_{ik}, \qquad k \in N, i \in \{M \setminus LS\} \tag{3.12}$$

$$d_{ik} = C_{ik}, \qquad k \in N, i = LS \tag{3.13}$$

$$C_{\max} \geq C_{ik}, \qquad k \in N, i = LS \tag{3.14}$$

$$X_{iljk} \in \{0,1\}, \qquad j \in \{N,0\}, k \in N, j \neq k, i \in M, l \in E_{ij} \cap E_{ik} \tag{3.15}$$

$$C_{ij}, d_{ij} \geq 0, \qquad j \in N, i \in M \tag{3.16}$$

A função objetivo, representada por (3.1), visa à minimização do *makespan*. O conjunto de restrições (3.2) assegura que cada tarefa deve ser precedida somente por uma tarefa em apenas um processador em cada estação. As restrições (3.3) indicam que cada tarefa deve ter, no máximo, um sucessor. O conjunto (3.4) indica que se uma tarefa é processada em uma máquina dada em uma estação, então ela deve ter uma predecessora na mesma máquina. O conjunto de restrições (3.5) evita a ocorrência de precedências cruzadas. O conjunto de restrições (3.6) impõe que a tarefa fictícia zero pode ser predecessora de, no máximo, uma tarefa em cada máquina em cada estação. O conjunto de restrições (3.7) simplesmente garante que a tarefa fictícia zero seja finalizada no tempo zero em todas as estações. O conjunto de restrições (3.8) indica que nenhuma tarefa pode iniciar seu processamento na máquina onde seja designada antes que esta seja liberada do programa de produção anterior. O conjunto de restrições (3.9) calcula o tempo de finalização de cada tarefa em cada estação, considerando o tempo de saída da tarefa predecessora na mesma estação. O conjunto de restrições (3.10) garante que cada tarefa seja processada subsequentemente em todas as estações. O valor B, utilizado em (3.8)-(3.10), representa um numero muito grande de modo a tornar redundantes as restrições se a variável de designação for zero. O conjunto de restrições (3.11) calcula o tempo de saída de cada tarefa de cada estação. O conjunto de restrições (3.12) garante que nenhuma tarefa possa sair de uma estação até que seu processamento nesta estação seja finalizado. O conjunto de restrições (3.13) indica que cada tarefa sai do sistema assim que seu processamento na última estação seja finalizado. O conjunto de restrições (3.14) define o *makespan*. Finalmente, os conjuntos (3.15) e (3.16) simplesmente definem as variáveis de decisão.

3.1. Exemplo Ilustrativo

De forma ilustrativa, apresentamos uma instância exemplo com cinco 5 tarefas e 3 estações. As estações 1 e 3 são estações de processamento, enquanto a estação 2 é uma estação de armazenagem intermediária. O número de processadores paralelos não relacionados é 2, 1 e 2, para a primeira, segunda e terceira estação respectivamente. Os tempos de liberação dos processadores na estação 1 são 163 e 182, respectivamente; 26 para o processador na estação 2 e, finalmente, 183 e 127 para os processadores na estação 3. O tempo de transporte entre as estações 1 e 2 é 7, enquanto o tempo entre as estações 2 e 3 é 8. Todas as preparações são consideradas não antecipatórias. As demais informações da instância exemplo são apresentadas na Tabela 1 e Tabela 2. O símbolo "-" indica que o processador não é capaz de processar a correspondente tarefa.

Estação	1		2	3	
Processador	1	2	3	4	5
Tarefa					
1	10	16	0	-	76
2	-	97	0	-	74
3	-	95	0	85	39
4	-	49	0	93	-
5	96	-	0	68	-

Tabela 1: Tempos de processamento de cada tarefa em cada máquina elegível.

	Processador 1					Processador 2				
Tarefa	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
1	-	-	-	-	39	-	48	51	62	-
2	-	-	-	-	-	51	-	33	47	-
3	-	-	-	-	-	63	41	-	29	-
4	-	-	-	-	-	71	33	38	-	-
5	72	-	-	-	-	-	-	-	-	-
0	52	-	-	-	28	53	40	59	32	-
	Processador 4					Processador 5				
1	-	-	-	-	-	-	70	45	-	-
2	-	-	-	-	-	70	-	29	-	-
3	-	-	-	56	31	72	30	-	-	-
4	-	-	68	-	34	-	-	-	-	-
5	-	-	53	50	-	-	-	-	-	-
0	-	-	52	45	45	41	37	53	-	-

Tabela 2: Tempos de preparação entre pares de tarefas em cada processador. Para o processador 3 todos os tempos são zero, pois representa um *buffer*.

O *makespan* ótimo desta instância é 620. A Figura 1 mostra um dos programas de produção ótimo para esta instância. Note que depois de terminar seu processamento no processador 1, a tarefa cinco permanece na estação 1 bloqueando o processador, mesmo quando o *buffer* está disponível. No entanto, transferir a tarefa cinco para o *buffer* assim que terminar seu processamento no processador 1 não terá efeito sobre o valor do *makespan*, dado que o *buffer* está ocioso desde o momento da sua liberação. O resultado desta mudança no programa de produção será a liberação mais cedo do processador 1 e o aumento do tempo de espera da tarefa 5

no *buffer*. Note também que as tarefas 1, 2 e 4 passaram diretamente desde a estação 1 até a estação 3, pois os processadores elegíveis desta estação estavam disponíveis quando as tarefas finalizaram seu processamento na estação 1, fazendo desnecessária a armazenagem intermediária das tarefas. Finalmente, note que a preparação em cada processador somente começa depois do transporte da tarefa designada desde a estação anterior.

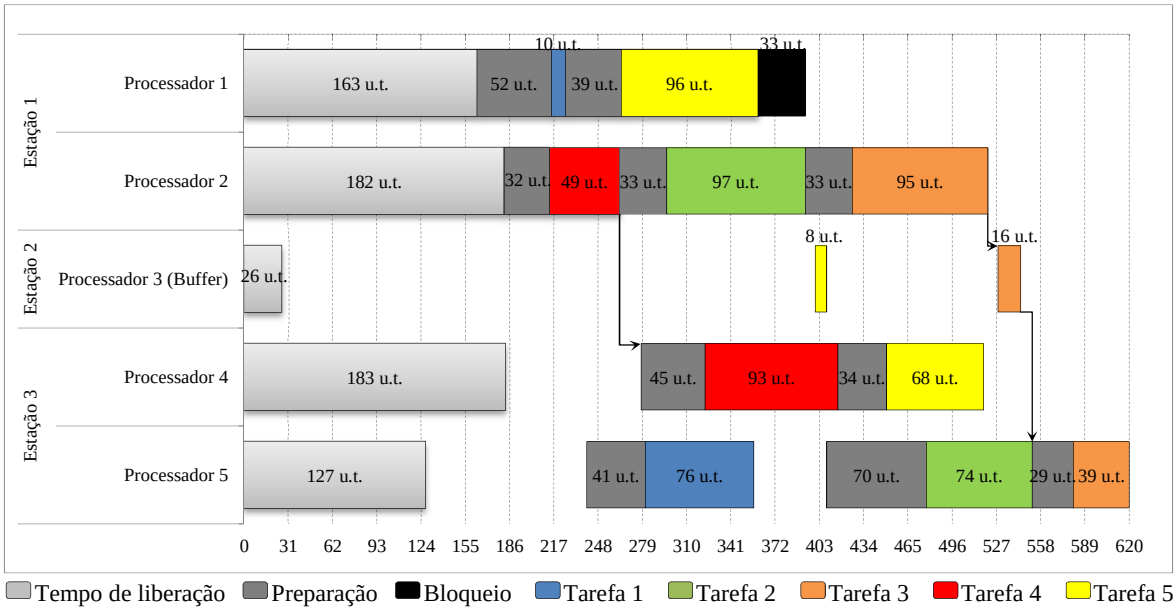

Figura 1: Gráfico de Gantt do exemplo ilustrativo.

Fatores	Símbolo	Valores
Número de tarefas	n	5, 7, 9, 11
Número de estações	m	3, 5, 7
Número de processadores paralelos em estações de processamento	m_i	2, 3
Número de processadores paralelos em estações *buffers*	b_i	$m_i/2, m_i$
Probabilidade de um processador ser elegível	PE_{ij}	0.5, 1
Distribuição dos tempos de preparação	DS_{iljk}	U[25,74], U[75,125]
Probabilidade da preparação ser antecipatória	PA_{iljk}	0, 0.50, 1

Tabela 3: Fatores considerados no conjunto de instâncias de pequeno porte

4. Avaliação Computacional

O modelo de programação inteira mista foi implementado na linguagem *GAMS* e resolvido utilizando o *solver CPLEX 12* em um *notebook* com processador Intel Core i5 2.67 GHz com 4 Gigabytes de memória RAM. O tempo máximo de execução foi configurado em 3.600 segundos para cada instância. Os fatores considerados para a geração das instâncias de teste são mostrados na Tabela 3.

O número total de combinações é $4 \cdot 3^2 \cdot 2^4 = 576$. Foram feitas 3 réplicas por combinação, por tanto há um total de 1.728 instâncias. A distribuição dos tempos de processamento é fixada em U[1,99], a distribuição dos tempos de liberação dos processadores é fixada em U[1,200] e, finalmente, a distribuição dos tempos de transporte é fixada em U[1,10]. É importante destacar que quando as instâncias foram geradas, foi garantido que em cada estação

de processamento cada tarefa tivesse, no mínimo, um processador elegível, isto é, $|E_{ij}| \geq 1$. Já no caso de estações *buffer*, considerou-se que todos eles são elegíveis para todas as tarefas, assim $|E_{ij}| = b_i$.

Na Tabela 4 são apresentados os resultados consolidados do modelo. Cada célula representa a média para cada tamanho de instância. Adicionalmente, é mostrada a percentagem de instâncias para as quais uma solução ótima foi encontrada dentro do limite de tempo (%Ótimo); a percentagem de instâncias para as quais uma solução inteira factível foi encontrada (%IntSol); a percentagem de instâncias para as quais nenhuma solução inteira factível foi encontrada dentro do tempo limite (%NãoSol).

		m		
n		3	5	7
5	%Ótimo	100,00	100,00	100,00
	%IntSol	0,00	0,00	0,00
	%NãoSol	0,00	0,00	0,00
7	%Ótimo	100,00	97,92	86,81
	%IntSol	0,00	2,08	9,72
	%NãoSol	0,00	0,00	3,47
9	%Ótimo	94,44	52,08	22,92
	%IntSol	5,56	46,53	70,83
	%NãoSol	0,00	1,39	6,25
11	%Ótimo	40,97	13,19	1,39
	%IntSol	59,03	81,94	79,17
	%NãoSol	0,00	4,86	19,44

Tabela 4: Resultados consolidados Modelo *MIP*.

Como observado, todas as instâncias de cinco tarefas são resolvidas até otimalidade, independentemente do número de estações. Como esperado, a dificuldade na resolução das instâncias aumenta na medida em que o número de tarefas e estações aumenta. Note também que, aumentar o número de tarefas reduz em maior proporção a percentagem de instâncias resolvidas até otimalidade em comparação com a redução gerada quando aumentarmos o número de estações. Este comportamento é um indício da maior relevância do número de tarefas na dificuldade das instâncias e, por tanto, na dificuldade para encontrar uma solução ótima. A Figura 2 ilustra claramente esta situação.

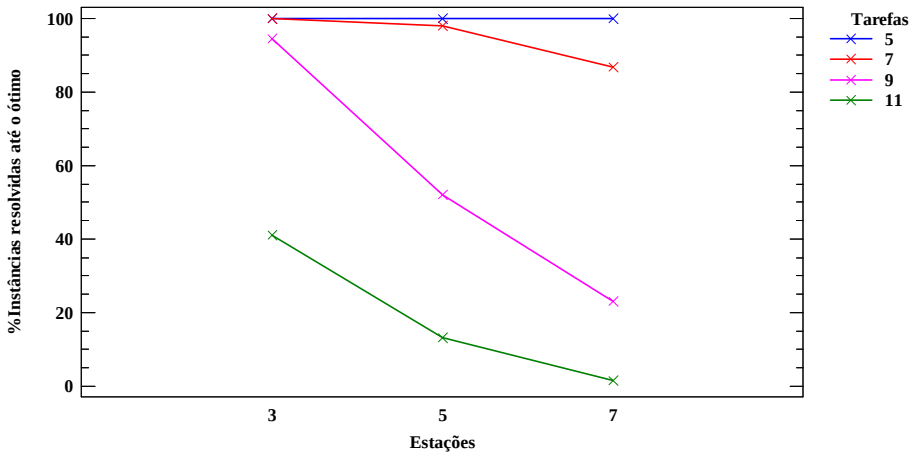

Figura 2: Percentagem de instâncias resolvidas até o ótimo: Tarefas vs. Estações.

Em relação ao tempo computacional, como esperado, aumentar o número de tarefas e estações aumenta consideravelmente o tempo requerido para encontrar uma solução ótima. A análise do tempo médio permite identificar que a partir de certos valores para o número de tarefas e o número de estações, o tempo utilizado torna-se proibitivo. Em particular, para instâncias além de nove tarefas e cinco estações, atingir uma solução ótima é inviável desde o ponto de vista prático, dado a quantidade de tempo requerida. A Figura 3 ilustra melhor o comportamento de tempo médio utilizado pelo *solver* como função do número de tarefas e estações. Nesta figura, é possível observar como o *solver* requer menor tempo para resolver instâncias com maior número de processadores paralelos, o qual indica uma menor dificuldade para resolver tais instâncias.

Em resumo, o *solver* encontrou ao menos uma solução ótima em 1.166 instâncias. Para 511 instâncias, encontrou uma solução inteira factível dentro do limite de tempo de execução de 3.600 segundos. Finalmente, para as restantes 51 instâncias nenhuma solução inteira factível foi encontrada depois de uma hora de execução. Estes resultados são apresentados graficamente na Figura 4.

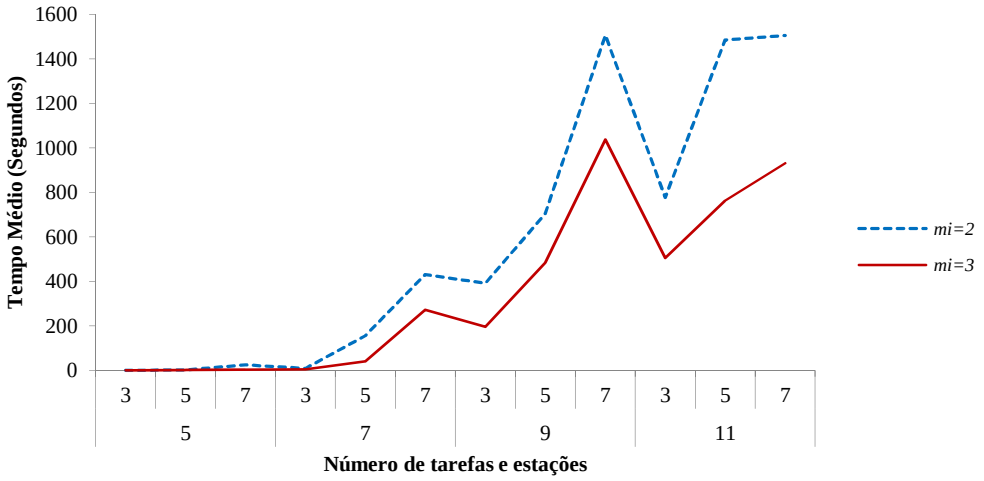

Figura 3: Tempo Médio utilizado pelo *solver* para encontrar uma solução ótima.

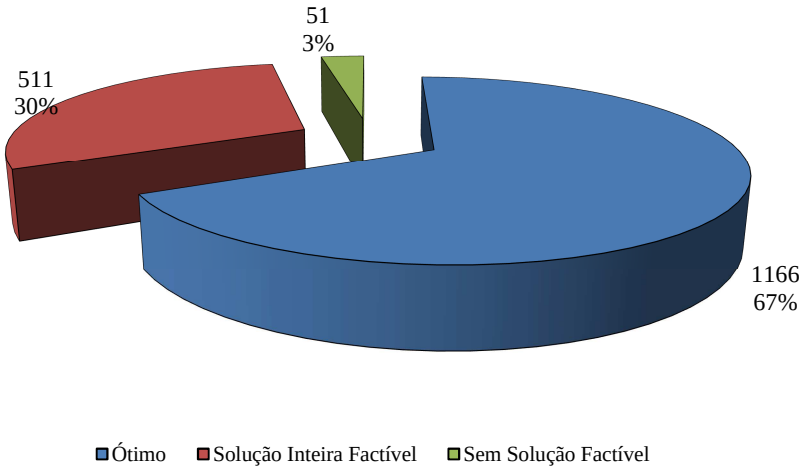

Figura 4: Resumo geral do modelo MIP

A Tabela 5 mostra claramente como o *gap* médio é relativamente alto, oscilando entre 22,25% e 56,67%. Em geral, o *gap* tende a ser maior com o aumento do número de tarefas e

estações. Este comportamento era esperado porque, como mostrado pela Tabela 4, quanto maior número de tarefas e estações, maior a dificuldade teve o CPLEX para resolver as instâncias. Por outro lado, o *gap* para instâncias de três processadores paralelos por estação é menor que o *gap* de instâncias com dois processadores paralelos. Esta última observação, junto com o menor tempo computacional requerido por instâncias com três processadores paralelos (Figura 3), permite concluir que a dificuldade das instâncias incrementa com o aumento do número de tarefas e estações, porém tende a reduzir quando há um maior número de processadores paralelos por estação de processamento.

	m_i	2		3	
$n \times m$	b_i	1	2	2	3
7 × 5		24,28	47,02	-	-
7 × 7		31,85	27,43	31,56	-
9 × 3		24,34	52,11	-	22,25
9 × 5		39,13	26,49	26,49	22,67
9 × 7		41,87	35,48	34,08	29,38
11 × 3		33,99	39,45	33,57	27,60
11 × 5		52,10	46,29	43,17	42,61
11 × 7		56,67	50,27	44,66	42,72

Tabela 5: Gap Médio de instâncias com solução inteira factível sem garantia de otimalidade.

5. Conclusões e perspectivas futuras

Nesta pesquisa um modelo MIP foi apresentado para resolver o problema de programação da produção em *Flowshop* Híbrido com máquinas paralelas não relacionadas, tempos de preparação antecipatórios e não antecipatórios dependentes da sequência, tempos de transporte entre estações, *buffers* limitados e elegibilidade de máquinas, visando minimizar o *makespan*. O modelo foi testado em um conjunto de 1.728 instâncias com tempo limite de 3.600 segundos. Os resultados encontrados indicam que o modelo é computacionalmente viável para instância de até nove tarefas e cinco estações. Em termos gerais, o CPLEX encontrou ao menos uma solução ótima para 1.166 instâncias. Para 511 instâncias, encontrou ao menos uma solução inteira factível e, finalmente, para as restantes 51 instâncias nenhuma solução inteira factível foi encontrada depois de 3.600 segundos.

Como trabalho futuro, pretende-se implementar e utilizar métodos meta-heurísticos para resolver o problema estudado. Estes métodos precisam maior tempo computacional que outros métodos mais simples, como as heurísticas, mas em contrapartida espera-se encontrar soluções de melhor qualidade para instâncias de qualquer tamanho.

6. Agradecimentos

Os autores agradecem ao programa de pós-graduação em Engenharia de Produção da Universidade Federal de São Carlos *campus* Sorocaba e à CAPES pelo apoio financeiro.

7. Referências

Chen, C., & Chen, C. (2009). A bottleneck-based heuristic for minimizing makespan in a flexible flow line with unrelated parallel machines. *Computers & Operations Research*, *36*(11), 3073–3081. doi:10.1016/j.cor.2009.02.004

Gupta, J. N. D. (1988). Two-Stage , Hybrid Flowshop Scheduling Problem. *The Journal of the Operational Research Society*, *39*(4), 359–364.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., & Werner, F. (2009). A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *Computers & Operations Research*, *36*(2), 358–378. doi:10.1016/j.cor.2007.10.004

Naderi, B., Zandieh, M., & Shirazi, A. (2009). Modeling and scheduling a case of flexible flowshops: Total weighted tardiness minimization. *Computers & Industrial Engineering*, *57*, 1258–1267. doi:10.1016/j.cie.2009.06.005

Nawaz, M., Enscore, E., & Ham, I. (1983). A Heuristic Algorithm for the m-Machine , n-Job Flow-shop Sequencing Problem. *International Journal of Management Science*, *11*(1), 91–95.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems*. *American Society of Mechanical Engineers* (3rd ed., p. 671). New York: Springer.

Ribas, I., Leisten, R., & Framiñan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, *37*(8), 1439–1454. doi:10.1016/j.cor.2009.11.001

Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, *169*(3), 781–800. doi:10.1016/j.ejor.2004.06.038

Ruiz, R., Serifoglu, F., & Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, *35*(4), 1151–1175. doi:10.1016/j.cor.2006.07.014

Sawik, T. (2000). Mixed integer programming for scheduling flexible flow lines with limited intermediate buffers. *Mathematical and Computer Modelling*, *31*(13), 39–52. doi:10.1016/S0895-7177(00)00110-2

Sawik, T. (2002). An exact approach for batch scheduling in flexible flow lines with limited intermediate buffers. *Mathematical and Computer Modelling*, *36*(4-5), 461–471. doi:10.1016/S0895-7177(02)00176-0

Urlings, T. (2010). *Heuristics and metaheuristics for heavily constrained hybrid flowshop problems*. Ph.D. Thesis. Universidad Politénica de Valencia.

Urlings, T., Ruiz, R., & Stützle, T. (2010). Shifting representation search for hybrid flexible flowline problems. *European Journal Of Operational Research*, *207*, 1086–1095. doi:10.1016/j.ejor.2010.05.041

Wang, X., & Tang, L. (2009). A tabu search heuristic for the hybrid flowshop scheduling with finite intermediate buffers. *Computers & Operations Research*, *36*(3), 907–918. doi:10.1016/j.cor.2007.11.004

Yaurima, V., Burtseva, L., & Tchernykh, A. (2009). Hybrid flowshop with unrelated machines, sequence-dependent setup time, availability constraints and limited buffers. *Computers & Industrial Engineering*, *56*(4), 1452–1463. doi:10.1016/j.cie.2008.09.004