A Computational Study of the Influence of the Parameter Big-M in a Time-Indexed MILP for Assembly Flowshop

José R. Santana, Universidade Federal de Goiás, Hélio Y. Fuchigami, Universidade Federal de São Carlos

***Abstract*—In this paper, a mixed integer linear programming model was proposed to solve the *assembly flowshop* scheduling problem. The objective of the study was to evaluate the impact on computational time (CPU) from the variation of the value of the big-M parameter considering three different dimensions. The performance of the mathematical model was evaluated from the application of a commercial solver that has different strategies in search of the optimal solution. 240 test-problems were solved on the optimality in an acceptable computational time, with the number of jobs ranging from 4 to 8, the number of products from 2 to 4 and the number of machines from 1 to 4. The performance profiles of Dolan and Moré (2002) were employed for a more accurate statistical analysis of the results, which indicated the most efficient value for the big-M parameter in the proposed mathematical formulation.**

***Index Terms*—Mixed Integer Linear Programming, Scheduling, Assembly Flow shop, big-M influence.**

I. Introdução

As mudanças ocorridas na economia mundial nos últimos 30 anos trouxeram novos desafios para os sistemas de produção em todos os segmentos. Novos produtos com alto nível de personalização fizeram com que o *design* de processos fosse alterado para o atendimento dessas demandas. Essas mudanças nos sistemas de produção começaram a ser apontadas por volta de 1975 [1], [2]. Os autores abordaram um sistema flexível de produção que consiste em uma produção com alto nível de automação e capacidade de adaptação aos processos. Importante ressaltar que esses processos estão diretamente ligados à redução de tempo e custos, sobretudo por conta das metodologias trazidas pela filosofia *just-in-time* [3], [4].

O ambiente de produção apresentado neste estudo é definido como *assembly flowshop*, consistindo em duas etapas de produção: fabricação e montagem. No primeiro estágio existem m_1 máquinas para a produção das partes de um produto. As máquinas não são idênticas, ou seja, os recursos são específicos para produzir determinadas partes de cada produto. O segundo estágio é de montagem, onde há m_2 máquinas disponíveis. Além disso, a utilização de todas as máquinas nesta fase não é obrigatória, pois cada produto requer diferentes números de componentes para ser montado.

Kusiak [5] apresentou pela primeira vez um estudo sobre o a*ssembly flowshop,* propondo duas heurísticas baseadas nos níveis de profundidade dos dígrafos, ou seja, na representação visual em árvore das partes e processos de sub-montagem que formavam um produto, e no tempo de espera.

A partir daquele momento, outros trabalhos foram elaborados, apresentando heurísticas construtivas e metaheurísticas para solucionar o problema de produção. He, Babayan e Kusiak [6] apresentaram pela primeira vez um modelo de programação linear inteira mista para solucionar o problema. Essa evolução está ligada diretamente ao desenvolvimento de novos *softwares* e *hardwares* que facilitaram a implementação de modelos matemáticos que pudessem encontrar soluções ótimas em um intervalo de tempo aceitável.

Desde o início das pesquisas na área de sequenciamento de produção, diferentes técnicas e abordagens foram criadas e vêm sendo utilizadas por diversos autores nos modelos matemáticos, como uma forma de agilizar o processo de busca da solução ótima. Wagner [7] apresentou um modelo matemático baseado na posição das tarefas na sequência. Manne [8] abordou uma formulação baseada em precedência de tarefas. Outra abordagem que também contribuiu para a criação de outros modelos, considerou a indexação pelo tempo como forma de sequenciar tarefas por meio da discretização, ou seja, um horizonte de tempo é definido previamente e as tarefas são alocadas nas unidades [9].

No presente artigo, foi formulado um modelo indexado pelo tempo para sequenciamento de tarefas em *assembly flowshop*. O objetivo da pesquisa consiste em analisar estatisticamente a eficiência do modelo matemático, aplicando diferentes valores ao parâmetro *big-M*, bem conhecido por influenciar o desempenho das formulações. O modelo é inédito e a experimentação computacional foi executada de forma a abranger diversas configurações de número de tarefas e de máquinas no primeiro e no segundo estágios, além do número de produtos.

O restante do artigo está organizado da seguinte forma: a seção 2 fornece a revisão da literatura sobre modelos indexados pelo tempo. A seção 3 apresenta o delineamento das etapas da pesquisa. A seção 4 formaliza a descrição do problema estudado. Já a seção 5 apresenta o modelo de programação linear mista proposto. A seção 6 fornece uma avaliação computacional do modelo com a variação do *big-M* seguida da análise estatística. Finalmente, a Seção 7 conclui o artigo e aponta algumas direções de pesquisas futuras.

II. Fundamentação teórica

Esta seção apresenta a revisão das pesquisas abordando formulações indexadas pelo tempo em problemas de sequenciamento. Esses trabalhos apresentam aspectos práticos

J. R. Santana, Universidade Federal de Goiás (UFG). Faculdade de Ciência e Tecnologia, Aparecida de Goiás, Goiás, Brasil, jose.renatho@gmail.com

H. Y. Fuchigami, Universidade Federal de São Carlos (UFSCar). São Carlos, São Paulo, Brasil, helio@dep.ufscar.br

e teóricos tanto na formulação dos modelos matemáticos, quanto na experimentação para avaliação da eficiência computacional. Logo, foram consideradas diversas aplicações em diferentes ambientes de produção para análise das principais contribuições apresentadas pelos autores.

Pesquisas com formulações indexadas pelo tempo são mais recentes que outras estratégias [8]. Diversas pesquisas já abordaram esta estratégia de formulação para o problema de sequenciamento em máquina única com diferentes restrições [9], [10], [11], [12].

D'Auria e Avella [13] apresentaram uma formulação para o problema de máquina única com tempos de liberação. Os autores propuseram uma relaxação langrageana para o problema. Savard *et al* [14] desenvolveram uma abordagem de solução baseada na técnica de geração de colunas para o problema de atraso total ponderado. Como forma de otimização do tempo computacional, uma estratégia de aceleração baseada na decomposição do horizonte de tempo em pequenos períodos, onde cada parte está associada a uma parte do problema, é usada para resolver a relaxação linear do problema. Estratégias de ramificação e regras de dominância também foram utilizadas na busca da solução ótima.

Alguns trabalhos apresentaram comparações de desempenho das formulações para problemas de máquina única [15], [16]. Mason e Unlu [17] apresentaram quatro diferentes formulações para problemas de programação em máquinas paralelas. T'Kindt *et al* [18], que também analisaram o ambiente com máquinas paralelas, construíram uma formulação indexada pelo tempo com técnicas de pré-processamento baseadas no custo reduzido. Os experimentos também mostraram que o algoritmo de planos de corte proposto com esse pré-processamento possibilitou resolver problemas-testes de forma mais eficiente.

Diversos autores, a partir da pesquisa realizada em 1989 por Kusiak, vêm apresentando métodos heurísticos e com modelos de programação linear mista para o problema de a*ssembly flowshop*. A maioria dos autores mantiveram seu foco em heurísticas e metaheurísticas [19], [20]. Algumas pesquisas focaram em apresentar uma solução por meio de modelos de programação matemática que utilizaram diferentes abordagens [21], [22], [23]. Portanto, destacam-se a seguir algumas dessas pesquisas por estudarem o *assembly flowshop* com formulação indexada pelo tempo.

Beck *et al* [24] apresentaram três formulações com abordagens diferentes para resolver um problema de fabricação e montagem de componentes. Os resultados mostraram que quando não há componentes compartilhados entre dois fabricantes, o modelo indexado pelo tempo é o melhor considerando tempos de processamento curtos. Além disso, apesar de outros modelos apresentarem resultados satisfatórios com abordagens diferentes, quando componentes compartilhados são inseridos na programação, o desempenho de todos os modelos se deteriora, e mesmo assim, o modelo indexado pelo tempo se destaca. É válido salientar que o calendário de disponibilidade dos componentes para a programação da produção é fundamental para a programação de estoques e disponibilidade de produtos.

Isvandi e Yavari [25] desenvolveram um modelo integrado, considerando pedidos de peças necessárias em cadeias de abastecimento de três níveis, incluindo os fornecedores de peças, fabricantes de componentes e montadores. O modelo matemático foi proposto para minimizar a soma do tempo total de conclusão, a quantidade de pedidos de peças e o custo de manutenção. Akbari *et al* [26] propuseram um modelo de programação linear inteira mista para firmar pedidos e decisões de programação em um problema de montagem de dois estágios.

Como pode ser visto no exame da literatura, modelos matemáticos para o problema de *assembly flowshop* foram relativamente pouco explorados, sobretudo com a estratégia indexada pelo tempo, evidenciando a importância e contribuição desta pesquisa.

III. Delineamento de Pesquisa

A Fig. 1 apresenta o delineamento dessa pesquisa que contou com as seguintes etapas: revisão bibliográfica, definição do problema a ser estudado, criação do modelo de programação linear mista, levantamento de dados, experimentação e análise de dados:

Fig. 1. Delineamento de pesquisa.

Inicialmente, foi realizado um levantamento de pesquisas que utilizaram como método de solução para o problema de sequenciamento, modelos de programação linear mista baseados na indexação do tempo. Após isso, o ambiente de produção foi definido considerando os parâmetros em um sistema *assembly flow shop*: máquinas, quantidade de estágios e complexidade de produtos. Assim, um modelo matemático foi desenvolvido para sequenciar as atividades com o objetivo de redução de *makespan* conforme as características levantadas no problema.

O levantamento de dados foi realizado levando em consideração a influência do número de atividades, máquinas no primeiro e segundo estágio, além da complexidade de produtos. Por isso, problemas-testes com diferentes tamanhos foram gerados para que essa avaliação se tornasse mais completa. Por fim, a experimentação computacional com os dados gerados, resultou em uma análise completa que avaliou a influência do parâmetro *big-M* no tempo computacional (CPU) do modelo matemático desenvolvido.

IV. Problema de Sequenciamento no Ambiente *Assembly Flowshop*

Para o problema apresentado neste artigo, consideram-se

m_1 máquinas não idênticas projetadas para fabricação das partes de um produto no primeiro estágio. No segundo estágio, existem m_2 máquinas para a montagem dos produtos. O objetivo é designar as partes a serem fabricadas na primeira etapa e montadas na segunda, considerando a minimização do *makespan*. As Fig. 2 e 3 apresentam a representação do ambiente de produção e da montagem, bem como a estrutura dos produtos, respectivamente:

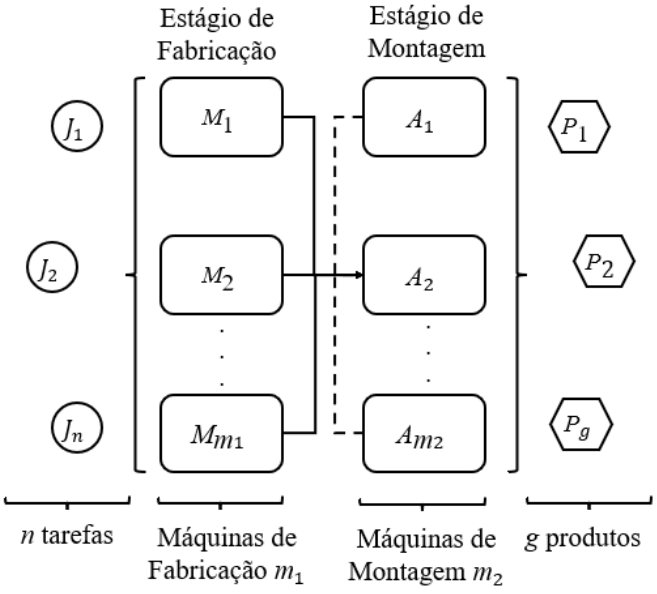

Fig. 2. Representação visual do ambiente de produção *assembly flowshop*.

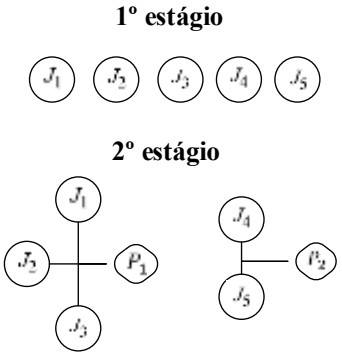

Fig. 3. Representação visual da formação dos produtos.

Neste problema, denotado pela conhecida notação de três campos por $AF(m_1, m_2)||C_{max}$, considera-se que há n tarefas J_j, j = 1,..., n, a serem realizadas, disponíveis no instante zero da programação e com um tempo de processamento p_{jk}, dependendo da máquina M_k, k = 1,...,m_1, do primeiro estágio em que será executada. No segundo estágio, considera-se que há g produtos P_u, u = 1,..,g, a serem montados após a fabricação de todas suas respectivas partes, demandando um tempo a_{ul} de cada produto P_u em cada máquina A_l, l = 1,..., m_2. Tanto o processamento das tarefas, como a montagem dos produtos, não podem ser interrompidos e cada operação pode ser feita por somente uma máquina.

Como pode ser visto na Fig. 2, todas as tarefas devem ser processadas obrigatoriamente em todas as máquinas disponíveis no primeiro estágio. No segundo estágio, um produto só pode ser montado em uma única máquina disponível.

V. Proposta: formulação indexada pelo tempo

Nesta estratégia de modelagem, ocorre a discretização do tempo, ou seja, o tempo é particionado em unidades para que cada tarefa seja designada e alocada em um intervalo definido dentro de um horizonte total denotado por H. Existem cinco variáveis de decisão: a primeira para alocação das tarefas no primeiro estágio de fabricação, a segunda para alocar os produtos nas máquinas de montagem, seguido das duas variáveis que definem os instantes de término das tarefas no primeiro e segundo estágios, e, por fim, a variável do *makespan*.

A. Índices:

j: índice de tarefas, onde j = 1,…, n
k: índice de máquinas no 1º estágio, onde k = 1,…, m_1
l: índice de máquinas no 2º estágio, onde l = 1,…, m_2
u, e: índices de produtos, onde u, e = 1,..., g
t, t': índices do tempo, onde t, t' = 1,..., H

B. Parâmetros:

m_1: número de máquinas no 1º estágio
m_2: número de máquinas no 2º estágio
n: número de tarefas
g: número de produtos
p_{jk}: tempo de processamento (fabricação) da tarefa J_j na máquina M_k no 1º estágio
a_{ul} : tempo de montagem (a*ssembly*) do produto P_u na máquina A_l no 2º estágio
M: número suficientemente grande (*big M*)
H: horizonte de tempo, onde H= $\sum p_{jk} + \sum a_{ul}$
G_{ju}: matriz que define as tarefas J_j pertencentes a um produto P_u

C. Variáveis de Decisão:

C_{jk}: instante de término da tarefa J_j na máquina M_k (1º estágio)
CA_{ul}: instante de término do produto P_u na máquina A_l (2º estágio)

$$x_{jkt} = \begin{cases} 1, \text{ se a tarefa } J_j \text{ inicia no instante } t \text{ na máquina } M_k \\ 0, \text{ caso contrário} \end{cases}$$

$$y_{ult} = \begin{cases} 1, \text{se o produto } P_u \text{ inicia no instante no tempo } t \text{ na máquina } A_l \\ 0, \text{ caso contrário} \end{cases}$$

C_{max}: duração total de programação (*makespan*)

D. Modelo matemático:

$$\text{Min } C_{max} \qquad (1)$$

Sujeito a:

$$\sum_{t=1}^{H-p_{jk}+1} x_{jkt} = 1 \qquad j=1,\ldots,n;\ k=1,\ldots,m_1 \qquad (2)$$

$$\sum_{j=1}^{n} \sum_{t'=max(1,\, t-p_{jk}+1)}^{t} x_{jkt'} \leq 1 \quad k=1,\ldots,m_1,\ t=1,\ldots,H \qquad (3)$$

$$C_{jk} = \sum_{t=1}^{H} (t + p_{jk} - 1)\, x_{jkt} \qquad j=1,\ldots,n;\ k=1,\ldots,m_1 \qquad (4)$$

$$\sum_{l=1}^{m_2} \sum_{t=1}^{H-a_{ul}+1} y_{ult} = 1 \qquad u=1,\ldots,g \qquad (5)$$

$$\sum_{u=1}^{g} \sum_{t'=max(1,\, t-a_{ul}+1)}^{t} y_{ult'} \leq 1 \quad l=1,\ldots,m_2,\ t=1,\ldots,H \qquad (6)$$

$$CA_{ul} = \sum_{t=1}^{H} (t + a_{ul} - 1)\, y_{ult} \qquad u=1,\ldots,g;\ l=1,\ldots,m_2 \qquad (7)$$

$$\sum_{t=1}^{H} (t+p_{jk})\, x_{jkt} \leq \sum_{t=1}^{H} t\, y_{ult} + M\,(2 - \sum_{t=1}^{H} y_{ult} - G_{ju})$$

$$j=1,\ldots,n;\ k=1,\ldots,m_1;\ u=1,\ldots,g;\ l=1,\ldots,m_2 \qquad (8)$$

$$C_{max} \geq CA_{ul} \qquad u=1,\ldots,g;\ l=1,\ldots,m_2 \qquad (9)$$

$$x_{jkt} \in \{0,1\}, y_{ult} \in \{0,1\}, C_{jk} \geq 0, CA_{ul} \geq 0, C_{max} \geq 0. \qquad (10)$$

A equação (1) representa a função objetivo a ser minimizada: o *makespan*. A restrição (2) garante que uma atividade seja processada obrigatoriamente em todas as máquinas disponíveis no primeiro estágio. As expressões (3) e (6) garantem que cada máquina não recebe mais de uma tarefa e mais de um produto, respectivamente, em nenhum momento, ou seja, não pode haver sobrecarga nos recursos. A expressão (4) define o instante de término de cada tarefa em cada máquina do primeiro estágio. A restrição (5) define a alocação dos produtos a serem montados nas máquinas disponíveis do segundo estágio e cada produto só pode ser alocado em apenas uma máquina. Analogamente, na expressão (7) calcula-se o instante de término de cada produto no segundo estágio. A expressão (8) refere-se à restrição de precedência, garantindo que cada produto deve iniciar no segundo estágio somente após o processamento de todas as suas partes fabricadas no primeiro estágio. A restrição (9) garante que o *makespan* seja pelo menos o maior instante de término da última operação de montagem dentre todos os produtos no segundo estágio. A restrição (10) refere-se ao domínio das variáveis.

VI. Experimentação Computacional

A. Levantamento de Dados

Na experimentação computacional, foram avaliados 240 problemas-testes divididos em três classes de 80 cada, considerando o número de tarefas (n), número de produtos (g), número de máquinas no primeiro estágio (m_1) e número de máquinas no segundo estágio (m_2).

O número de tarefas e de produtos foram respectivamente $n \in \{4, 6, 8\}$ e $g \in \{2, 3, 4\}$, sendo consideradas as configurações: 4x2, 6x3 e 8x4. Além disso, o número de máquinas no primeiro e segundo estágios foram de $m_1 \in \{1, 2, 3, 4\}$ e $m_2 \in \{1, 2, 3, 4\}$, sendo que para cada combinação m_1x m_2 foram geradas 5 réplicas, totalizando 80 problemas-teste para cada configuração.

Os tempos de processamento das tarefas executadas no primeiro estágio foram gerados no intervalo uniforme U [1,99] e os tempos de montagem foram gerados variando conforme a quantidade de tarefas, sendo U [1*n, 99*n]. O experimento consistiu na avaliação de três estratégias, denominadas de I, II e III, com diferentes valores do parâmetro M, conforme a Tabela I, que mostra o detalhamento do delineamento da experimentação computacional:

TABELA I
PARÂMETROS PARA EXPERIMENTAÇÃO COMPUTACIONAL

Parâmetros	Valores
Número de tarefas (n)	4, 6, 8
Número de produtos (g)	2, 3, 4
Número de máquinas no 1º estágio (m_1)	1, 2, 3, 4
Número de máquinas no 2º estágio (m_2)	1, 2, 3, 4
Configurações - n x g	4x2, 6x3, 8x4
Número de problemas-testes por classe	80
Número total de problemas-testes	240
Variação dos valores do parâmetro M	I) $10 \sum p_{jk}$ II) $100 \sum p_{jk}$ III) $1000 \sum p_{jk}$

O modelo foi implementado por meio da linguagem Julia/JuMP e resolvido utilizando o *solver* Gurobi 9.0.3 usando os parâmetros *default*. Todos os testes foram executados em um processador Intel Core i7 com 2.3 GHz, 8.0 GB de memória RAM e sistema operacional Windows. Todas os 240 problemas-testes foram resolvidos na otimalidade pelo algoritmo *branch-and-cut* incluído no Gurobi.

B. Análise de resultados

A Tabela II a seguir apresenta os tempos de CPU (em segundos) de cada estratégia na resolução dos problemas-testes, para as diferentes classes de número de tarefas e de produtos:

TABELA II
MÉDIAS DOS TEMPOS DE CPU (EM SEGUNDOS) DAS ESTRATÉGIAS AVALIADAS

Configurações (n x g)	Estratégias		
	I	II	III
4 x 2	10,82	10,88	10,37
6 x 3	54,10	54,15	55,37
8 x 4	242,83	242,46	244,46
Média Geral	102,59	102,49	103,40

Como pode ser visto na Tabela II, pela média geral dos tempos de CPU, a estratégia II demonstrou-se levemente mais eficiente para a formulação proposta, embora os valores tenham ficado bastante próximos, requerendo uma análise estatística mais acurada. Assim, os resultados foram avaliados por meio dos perfis de desempenho [27].

Para simplificar a explicação das análises por meios dos perfis de desempenho, os resultados dos diferentes valores do parâmetro M serão denominados genericamente como "métodos de solução". Dado um conjunto de problemas-teste T e um conjunto S de métodos de solução, os perfis de desempenho são baseados na função de distribuição cumulativa $P_i(f)$ que indica a probabilidade para o método i, que o desempenho r_{ij} para cada exemplar j dentro do fator f seja o melhor possível. Destaca-se que o r_{ij} é descrito como o índice de desempenho sendo que quando o seu valor for igual a 1, o

método i é o mais eficiente. Quanto maior for o valor de r_{ij}, menor o desempenho do método i. A função $P_i(f)$ e o índice de desempenho r_{ij} são definidos a seguir:

$$P_i(f)=\frac{1}{n_T}\ card\left\{j \in T \mid r_{ij} \le f\right\}, \text{ para } f \ge 1 \quad (11)$$

$$r_{ij}=\frac{s_{ij}}{\min\{s_{ij} \mid i \in j\}} \quad (12)$$

onde $s_{ij} \ge 0$ é a estatística do método $i \in S$ ao resolver o exemplar $j \in T$, e $card\left\{j \in T \mid r_{ij} \le f\right\}$ é o número de exemplares que satisfazem $r_{ij} \le f$.

Valores de $P_i(f)$ quando $f = 0$ indicam a fração de problemas-testes para as quais o método i alcançou a melhor solução no menor tempo. Para $f > 0$, $P_i(1)$ consiste na fração de problemas-teste cuja estratégia i obteve soluções com qualidade dentro de um fator f das melhores soluções.

A Fig. 4 mostra o gráfico da função $P_i(f)$ para os 240 problemas-testes que foram executados com diferentes valores do parâmetro M, denominados de estratégias I, II e III, conforme definido na Tabela I. A estatística analisada nessa experimentação computacional foi o tempo de execução (CPU) na busca da solução ótima, uma vez que os *gaps* de otimalidade são zero.

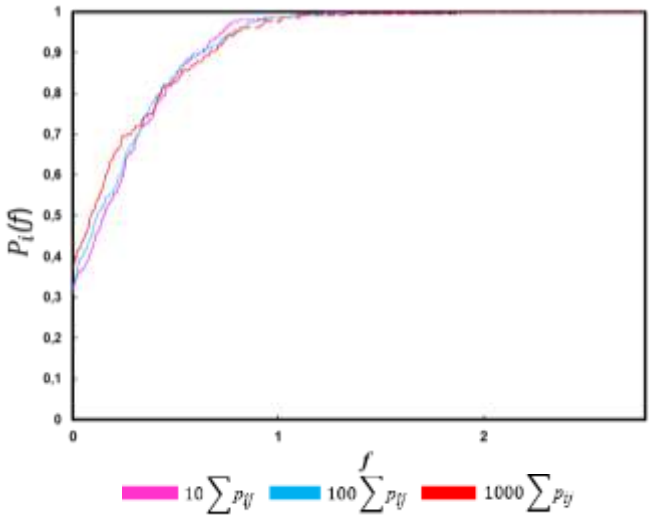

Fig. 4. Perfis de desempenho das estratégias avaliadas.

Conforme já salientado anteriormente, a Fig. 4 confirma que todos os problemas-testes foram resolvidos na otimalidade para as três estratégias I, II e III, pois todas as curvas atingem os valores de $P_i(f) = 1$. Ainda de acordo com a Fig. 4, a estratégia III, que considera o maior valor do parâmetro M dentre os avaliados, foi mais eficiente em 37,09% dos problemas-testes, ou seja, atingiu a solução mais rapidamente do que as demais estratégias neste percentual de exemplares. Já com a estratégia II, o modelo foi mais eficiente em 32,08% dos problemas-testes e com a estratégia I, em 30,83% dos exemplares. Por esta análise, quanto maior o valor do parâmetro M, mais eficiente torna-se a formulação matemática. A Tabela III apresenta os valores extremos dos perfis de desempenho do modelo com os diferentes valores para o parâmetro M, com $P_i(f)$ quando $f = 0$ e f quando $P_i(f) = 1$:

TABELA III
VALORES EXTREMOS DOS PERFIS DE DESEMPENHO PARA AS ESTRATÉGIAS AVALIADAS

Valor do Parâmetro M	$P_i(f)$	f
$10\sum p_{ij}$	0,31	1,23
$100\sum p_{ij}$	0,32	1,87
$1000\sum p_{ij}$	0,37	1,89

Conforme mostrado na Tabela III e no gráfico da Fig. 4 apresentado, o valor do parâmetro M influencia diretamente no tempo de execução do modelo. Essa verificação foi feita de forma experimental, conforme a alocação do parâmetro M na restrição (8), e a experimentação computacional evidenciou que maiores valores reduzirem o tempo de solução do problema.

No entanto, como pode ser observado, a estratégia III apresentou um fator f maior em relação às demais estratégias. Isso significa que alguns problemas-testes demandaram um tempo de execução maior quando comparado ao menor tempo para o mesmo problema. Para evidenciar essa afirmação, a Tabela IV mostra a média de tempo de execução (CPU) em segundos, conforme a variação do número de máquinas para o primeiro e segundo estágio:

TABELA IV
TEMPO MÉDIO DE EXECUÇÃO (CPU) EM SEGUNDOS PARA AS COMBINAÇÕES DE MÁQUINAS DO PRIMEIRO E SEGUNDO ESTÁGIOS

	Estratégias		
	I	II	III
m_1 x m_2	Tempo de execução - CPU (s)	Tempo de execução - CPU (s)	Tempo de execução - CPU (s)
1x1	26,70	29,23	34,84
1x2	50,42	43,27	45,76
1x3	52,81	50,70	55,53
1x4	86,36	86,43	82,35
2x1	58,44	69,86	63,11
2x2	51,31	48,78	47,93
2x3	76,45	66,08	67,84
2x4	130,82	115,04	117,38
3x1	120,10	126,27	121,92
3x2	87,38	78,63	91,64
3x3	102,21	97,93	114,56
3x4	193,52	208,55	203,44
4x1	153,19	172,37	151,26
4x2	123,61	120,77	134,41
4x3	130,08	153,89	138,96
4x4	197,98	172,11	183,44

Conforme mostrado na Tabela IV, algumas configurações apresentaram um tempo de execução mais elevado em relação às outras estratégias e isso influencia diretamente no tempo médio geral de cada combinação entre as máquinas nos estágios. No entanto, essa análise é suficiente para mostrar que mesmo com essa disparidade em algumas configurações

específicas e conforme a quantidade de problemas-testes solucionados com o melhor tempo de execução, sugere-se a utilização no modelo do valor $M = 1000\sum p_{jk}$ correspondente à estratégia III, no modelo matemático indexado pelo tempo para o problema de *assembly flowshop*.

VII. CONCLUSÃO

Esse artigo apresentou uma experimentação computacional avaliando a influência do parâmetro *M* em um novo modelo de programação linear inteira mista para o problema de sequenciamento de tarefas em um ambiente *assembly flowshop*. A abordagem utilizada na formulação foi a indexação pelo tempo e para avaliar o comportamento do modelo estudado, foram consideradas diferentes configurações de layout e estrutura do produto.

O parâmetro *M* é conhecido na literatura por influenciar o desempenho computacional dos modelos de programação linear inteira mista, embora não haja muitas publicações circunstanciando este grau de influência. Nesse estudo, foram avaliados três valores: $10\sum p_{jk}$, $100\sum p_{jk}$ e $1000\sum p_{jk}$. A experimentação computacional demonstrou que o maior valor do parâmetro *M* conduz o modelo a menores tempos de execução e isso pode ser observado por meio da ferramenta estatística de perfis de desempenho, que aponta que dentre os 240 problemas-testes resolvidos, 37% apresentaram maior eficiência quando o parâmetro *M* é maior.

Os modelos de programação linear mista são desenvolvidos com o principal objetivo de encontrar respostas ótimas, ou pelo menos satisfatórias, dentro de um intervalo de tempo de execução aceitável. Logo, esse estudo comprova que a utilização otimizada de um determinado parâmetro pode influenciar diretamente na performance do *solver* Gurobi 9.0.3 para o modelo e consequentemente, no tempo computacional para a busca de soluções.

Finalmente, é importante salientar também, que este estudo considerou um problema realístico de *assembly flowshop*, com a estrutura de montagem dos produtos composta por variados números de tarefas, diferentemente da maioria dos trabalhos publicados, que não agrupam as tarefas dos respectivos produtos montados, conforme pode ser visto na revisão da literatura. Ainda como principal contribuição desta pesquisa está a proposição de um novo modelo matemático indexado pelo tempo e a análise experimental pormenorizada da influência do parâmetro *M* na formulação apresentada. Como sugestões para trabalhos futuros, sugere-se a inclusão de novas restrições no problema, como prazos de entrega, tempos de *setup*, *no-wait*, entre outras, e a utilização de algum método heurístico como solução inicial do modelo.

REFERÊNCIAS

[1] Morton, Thomas E.; Smunt, Timothy L. A planning and scheduling system for flexible manufacturing. In: Flexible Manufacturing Systems: Methods and Studies. North-Holland, 1986. pp. 151-164.

[2] A.S. Kiran and S. Alptekin, Scheduling algorithms for flexible manufacturing system. Dept. of Industrial and Systems Engineering, University of Southern California, Los Angeles. no. 85, 1985.

[3] Ríos-Mercado, Roger Z.; Ríos-Sólis, Yasmín A. Just-in-Time Systems. New York: Springer Optimization And Its Application, 2012.

[4] Józefowska, J. Just-in-Time Scheduling: Models and Algorithms for Computer and Manufacturing Systems. *Springer, International Series in Operations Research & Management Science*, 2007.

[5] Kusiak A. Aggregate scheduling of a flexible machining and Assembly system. *IEEE Trans Robotics Automat*, 1989.

[6] He, D.; Babayan and A; Kusiak A. Scheduling manufacturing systems in an agile Environment. Robotics and Computer Integrated Manufacturing, vol 17. pp. 2461-2481, 2001.

[7] Wagner HM. An integer linear-programming model for machine scheduling. Naval Research Logistics Quarterly, vol. 6, pp. 131–140, 1959.

[8] Manne As. On the job shop scheduling problem. Cowles Foundation Discussion. Cowles Foundation for Research in Economics, Yale University, vol. 73, 1959.

[9] J.P. Sousa, Time indexed formulations of non-preemptive single-machine scheduling problems. Ph.D. Thesis, Faculté des Sciences Appliquées, Université Catholique de Louvain (Louvain-la-Neuve, Belgium), 1989.

[10] Dyer Me and Wolsey LA. Formulating the single machine sequencing problem with release dates as a mixed integer program. Discrete Appl. Math., vol. 26, pp. 255–270, 1990.

[11] Sousa JP and Wolsey LA. A time indexed formulation of non-preemptive single machine scheduling problems. Mathematical Programming, vol. 54, pp. 353–367, 1992.

[12] Crama, Y., Spieksma, Scheduling jobs of equal length: complexity, facets and computational result. Mathematical Programming vol. 72, pp. 207–227, 1996.

[13] Avella P, Boccia, M. and D'auria, B. Near-optimal solutions of large-scale single-machine scheduling problems. Informs Journal on Computing, vol. 17, pp. 183–191, 2005.

[14] Bigras L-P, Gamache M and Savard G. The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. Discrete Optimization, vol. 5, pp. 685–699, 2008.

[15] Keha AB, Khowala K and Fowler JW. Mixed integer programming formulations for single machine scheduling problems. *Computers & Industrial Engineering*, vol. 56, pp. 357-367, 2009.

[16] WU, Shuang; REN, Xiaoqiang; DEY, Subhrakanti; SHI, Ling. Optimal Scheduling of Multiple Sensors with Packet Length Constraint, vol. 50, no. 1, pp. 14430-14435, 2017.

[17] Unlu, Y. and Mason SJ. Evaluation of mixed integer programming formulations for nonpreemptive parallel machine scheduling problems. *Computers & Industrial Engineering*, vol. 58, pp. 785–800, 2010.

[18] M. Garraffa, L. Shang, F. Della Croce, V. T'kindt. An exact exponential branch-and-merge algorithm for the single machine total tardiness problem, Theoretical Computer Science, vol. 745, pp. 133-149, 2018.

[19] Potts, C. N., Sevast'Janov, S. V., and Zwaneveld, C. M. The two-stage Assembly scheduling problem: *complexity and approximation. Operations Researc*h, vol. 43, pp. 346-355, 1995.

[20] Koulamas, C., and G. J. Kyparisis. The Three-Stage Assembly Flowshop Scheduling Problem. Computers and Operations Research, vol. 28, pp. 689–704, 2001.

[21] Zhang, Y., Z. Zhou, and J. Liu. The Production Scheduling Problem in a Multi-Page *Invoice Printing System. Computers and Operations Research*, vol. 37, pp. 1814–1821, 2010.

[22] Deng, J., L. Wang, S. Y. Wang, and X. L. Zheng. A Competitive Memetic Algorithm for the Distributed two-Stage Assembly Flow-Shop Scheduling Problem. International Journal of Production Research vol. 54, pp. 3561–3577, 2016.

[23] Sheikh, Shaya; Komaki, G.M.; Kayvanfar, Vahid; Teymourian, Ehsan. Multi-Stage Assembly Flowshop with setup time and release time. Operations Research Perspectives, vol. 6, pp. 100-111, 2019.

[24] Terekhov, D., M. K. Dogru, U. Özen, and J. C. Beck. Solving Two-Machine Assembly Scheduling Problems with Inventory Constraints. *Computers & Industrial Engineering*, vol. 63, pp. 120–134, 2012.

[25] Yavari, Mohammad; Isvandi, Sahar. Integrated decision making for parts ordering and scheduling of jobs on two-stage Assembly problem in three level supply chain. Journal of Manufacturing Systems, vol. 46, pp. 137-151, 2018.

[26] Yavari, Mohammad; Marvi, Mohammad; Akbari, A. Hosein. Semi-permutation-based genetic algorithm for order acceptance and scheduling in two-stage Assembly problem. *Neural Computing and Applications*, vol. 32, pp. 2989–3003, 2019.

[27] Dolan, Elizabeth D.; Moré and Jorge J. Benchmarking optimization software with performance profiles. *Mathematical Programming, Springer Science and Business Media LLC*. vol. 91, no. 2, pp. 201-213, 2002.

José Renatho da Silva Santana. Production Engineer from Pontifical Catholic University of Goiás. Student in Production Engineering at the Faculty of Science and Technology (FCT) at UFG. He works as a researcher in the field of Production Scheduling. He has knowledge in project management with a focus on Agile Methodologies. He currently works as a project manager in a consulting and advisory company for companies with e-commerce. Developed research and extension project in a food industry with a focus on reducing costs and waste, as well as having experience in developing projects focused on Lean Manufacturing and people management in the application of agile routines focused on strategic and operational results in companies of different segments.

Hélio Yochihiro Fuchigami. Professor and researcher at the Federal University of São Carlos (UFSCar). He researches in the Production Scheduling and Operational Research area. Computer Scientist from UNESP, Master and phD in Production Engineering from USP, with part of his doctorate at the Polytechnic University of Valencia, Spain. Post-doctorate at the University of New South Wales (Australian Defense Force Academy) in Australia and at UNESP/São José do Rio Preto. Chronicler and poet, with three published books.