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

MODELOS MATEMÁTICOS PARA PROGRAMAÇÃO DE *JOB SHOP* COM TEMPOS DE *SETUP* INDEPENDENTES DA SEQUÊNCIA

MATHEMATICAL MODELS FOR JOB SHOP SCHEDULING PROBLEM WITH SEQUENCE-INDEPENDENT SETUP TIMES

Hélio Yochihiro Fuchigami* E-mail: heliofuchigami@ufg.br
Mirella Augusta Sousa Moura** E-mail: mirella.asm14@hotmail.com
Fábio José Ceron Branco*** E-mail: fbranco@hotmail.com
* Universidade Federal de Goiás (UFG), Aparecida de Goiânia, GO
** Universidade Federal de Goiás (UFG), Catalão, GO
***Universidade Tecnológica Federal do Paraná (UTFPR), Ponta Grossa, PR

Resumo: Este trabalho aborda o problema de programação da produção em *job shop* com tempos de *setup* explícitos e independentes da sequência de processamento das tarefas visando minimizar a duração total da programação (*makespan*). Na conhecida notação de três campos, este problema é representado por $Jm|s_{jk}|C_{max}$. O ambiente *job shop* é aquele em que há um conjunto de máquinas ou recursos diferentes que devem processar um conjunto de tarefas, sendo que cada tarefa possui uma rota específica e pré-determinada para processamento nas máquinas. Para problemas com duas máquinas, nesta pesquisa propõe-se uma adaptação no clássico Algoritmo de Jackson, que fornece a solução ótima para o caso com *setup* incluído nos tempos de processamento das tarefas. E para problemas genéricos com *m* máquinas, foram elaborados e implementados dois modelos matemáticos de programação linear inteira mista para os casos de *setup* antecipado (aquele que pode ser iniciado antes da liberação da tarefa) e não antecipado, que fornecem soluções em tempo computacional viável para problemas-testes com até 20 tarefas e 7 máquinas.

Palavras-chave: Programação da produção. *Job shop*. *Setup* independente. Programação linear inteira mista.

Abstract: This paper addresses the job shop scheduling problem with explicit and sequence-independent setup times to minimize the total time to complete the schedule (makespan). According to the well-known three-field notation, this problem is denoted by $Jm|s_{jk}|C_{max}$. In job shop production environment, a set of different machines or resources must process a set of jobs, where each job has a specific and predetermined route through the machines. For problems with two machines, this research proposes an adaptation of the classical Jackson's Algorithm, which provides the optimal solution for the case with setup times included in the processing times of jobs. And for general problems with *m* machines, two mathematical models of mixed integer linear programming were designed and implemented to solve the two cases with anticipatory and non-anticipatory setup times. Anticipatory setup times can be started before the release of the job. The two models provide solutions to the problems in acceptable computational time for instances with up to 20 jobs and 7 machines.

Keywords: Scheduling. Job shop. Sequence-independent setup times. Mixed Integer Linear Programming.

1 INTRODUÇÃO

A programação da produção (*scheduling*) consiste na alocação de recursos para execução de tarefas em uma base de tempo. Pode ser definida como a determinação de *quando* e *onde* cada operação deve ser realizada para a fabricação de um produto (PINEDO, 2015).

A atividade de programação da produção é uma das mais importantes de uma empresa no nível operacional. Está relacionada à otimização de diversas medidas de desempenho que visam o bom funcionamento do sistema e a satisfação dos clientes como, por exemplo, a utilização eficiente dos recursos, a entrega dos produtos nos prazos estipulados e a redução de custos de produção e estoques.

Neste sentido, um dos ambientes de programação da produção mais típicos e complexos é o *job shop*, um problema de otimização combinatória classificado como *NP-hard*, presente em muitas situações práticas como indústrias aeronáuticas, automotivas e de semicondutores (SHARMA; JAIN, 2015).

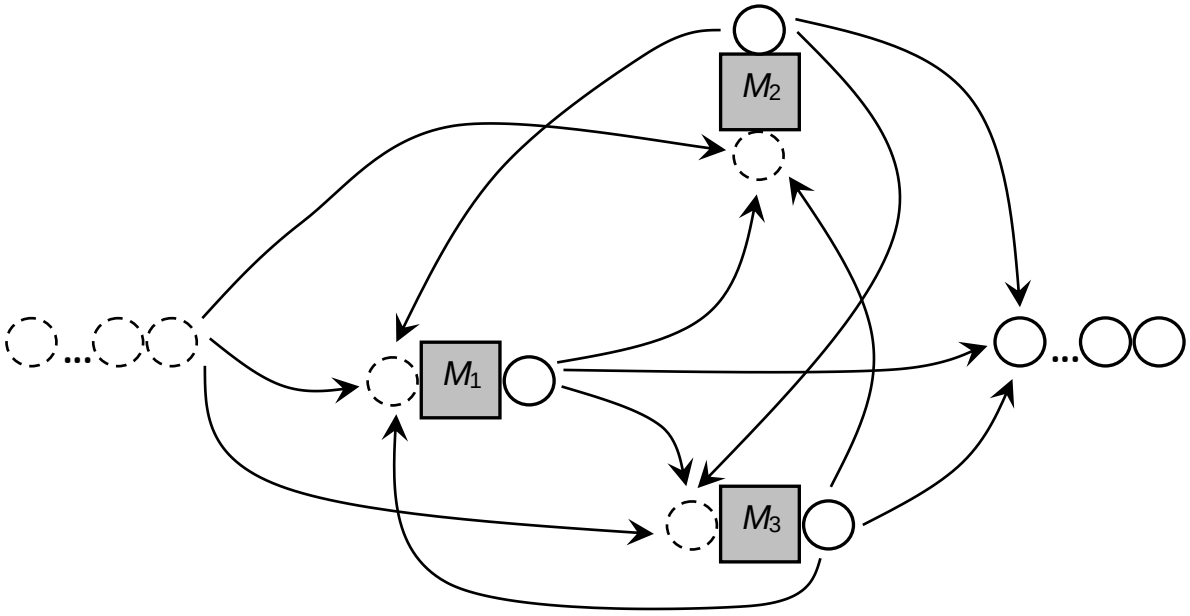
O problema clássico de *job shop* consiste de um conjunto de *m* máquinas diferentes, como torno, fresa e furadeira, que devem processar operações de *n* tarefas. Cada tarefa possui uma ordem específica para a execução das suas operações, ou seja, cada produto tem sua rota própria nas máquinas, que é considerada fixa e conhecida previamente. Além disso, cada operação de cada tarefa não pode ser interrompida e deve ser processada em uma única máquina de cada vez, e cada máquina pode processar apenas uma operação de cada vez.

A Figura 1 ilustra um ambiente *job shop* e exemplifica as diferentes rotas para as tarefas que serão processadas em cada uma das três máquinas.

As primeiras pesquisas na área de programação da produção foram publicadas em meados da década de 1950. Desde então, inúmeros artigos com diferentes problemas de programação têm surgido na literatura. A maioria desses trabalhos assume que o tempo e/ou custo de *setup* é insignificante ou parte do tempo de processamento. Enquanto esse pressuposto simplifica a análise e reflete certas aplicações (quando os *setup*s são pequenos, independentes da sequência e inseparáveis dos tempos de processamento), adversamente afeta a qualidade da solução de várias aplicações de programação da produção que necessitam do

tratamento separado dos tempos de *setup* (ALLAHVERDI, 2015; ALLAHVERDI *et al.*, 2008; ALLAHVERDI *et al.*, 1999).

Figura 1 – Ilustração de um *job shop* com três máquinas



Fonte: autores

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 para execução de uma tarefa. 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 realizada imediatamente antes na mesma máquina. Por serem tratados separados dos tempos de processamento das tarefas, os tempos de *setup* são também referenciados como "*setup* explícitos".

Além disso, os tempos de *setup* também podem ser definidos como antecipados ou não antecipados. O *setup* antecipado é aquele que pode ser realizado antes da liberação da tarefa, ou seja, antes do término da operação no estágio anterior. Uma importante implicação dos tempos de *setup* antecipados é que eles podem ser iniciados na máquina subsequente enquanto a tarefa ainda está sendo executada. E como é comum haver tempo ocioso na segunda máquina em diante, antecipar o *setup* é uma vantagem para medidas de desempenho regulares, como é o caso do *makespan* e do tempo de fluxo (FUCHIGAMI, 2015, p.39).

Já o *setup* não antecipado pode ser realizado somente após a liberação da tarefa. São os casos que requerem que a peça ou o material a ser processado esteja presente na máquina para que o setup seja realizado, como, por exemplo, nas operações de ajuste e posicionamento (FUCHIGAMI, 2015, p.40).

Nesta pesquisa será tratado um problema de *job shop n* tarefas e *m* máquinas, tempos de *setup* independentes da sequência e minimização da duração total da programação (*makespan)*. Os tempos de processamento e de *setup* das máquinas são fixos e conhecidos previamente. Foram consideradas separadamente duas variantes do problema: com *setup* antecipado e não antecipado. O objetivo deste estudo é propor e comparar modelos matemáticos para estes problemas de forma a mensurar a influência da antecipação do *setup* no *makespan* em ambientes *job shop*.

Na conhecida notação de três campos, este problema é representado por $Jm|s_{jk}|C_{max}$, com "*Jm*" indicando o ambiente de produção (*job shop* com *m*-máquinas), "s_{jk}" a restrição (*setup* independente, ou seja, relacionado apenas à operação da tarefa J_j que será executada na máquina M_k) e "C_{max}" a função objetivo (*makespan*).

O trabalho está estruturado da seguinte forma: após a contextualização da introdução (seção 1), é apresentada a revisão da literatura (seção 2), com as principais publicações envolvendo *job shop* com *setup* independente, desde o artigo pioneiro até os mais relacionados à presente pesquisa. Em seguida, são propostos um algoritmo para o problema específico com duas máquinas e *setup* independente (seção 3) e os modelos matemáticos para os problemas com *m* máquinas e *setup* antecipado e não antecipado (seção 4). Ao final, são feitas as considerações finais (seção 5).

2 REVISÃO DA LITERATURA

Muitas publicações já abordaram o problema da programação em *job shop*, desde o pioneiro Jackson (1956), que propôs um algoritmo de solução ótima para o problema com duas máquinas e minimização do *makespan*. O método de Jackson (1956) baseia-se no algoritmo de Johnson (1954), que fornece a solução ótima para o problema de *flow shop* com duas máquinas e é considerado a primeira publicação de programação da produção.

Blazewicz *et al.* (1996) fizeram uma revisão dos problemas de *job shop*, descreveu os principais trabalhos das primeiras décadas da área e classificou as pesquisas por técnicas de solução. Conforme os autores afirmaram, durante muitos anos as regras de prioridade foram utilizadas como os métodos heurísticos mais frequentes na prática, por causa da facilidade de implementação e baixo tempo computacional. Além disso, enfatizaram que a heurística *Shifting Bottleneck*, de Adams *et al.* (1988), era à época um dos procedimentos mais poderosos entre as heurísticas para *job shop*.

Uma classificação hierarquizada das técnicas utilizadas na programação do *job shop* foi proposta por Pacheco e Santoro (1999), que evidenciou as diferentes abordagens do problema e suas soluções. Parveen e Ullah (2010) fizeram uma revisão da literatura sobre *job shop* e *flow shop* multicritério, com técnicas exatas e heurísticas.

Especificamente para problemas de *scheduling* com tempos e custos de *setup*, pode-se encontrar as revisões bastante abrangentes de Allahverdi *et al.* (1999), Allahverdi *et al.* (2008), Allahverdi e Soroush (2008) e Allahverdi *et al.* (2015).

Recentemente, Sharma e Jain (2015) publicaram uma extensa revisão da literatura sobre *job shop* com tempos de *setup*, e categorizou os trabalhos em problemas que consideram ou não lotes de produção (*batch* e *non-batch*, respectivamente). Além disso, a pesquisa classifica os problemas com *setup* dependente e independente da sequência e pelo método de solução – exato, heurístico e híbrido.

Nesse levantamento de Sharma e Jain (2015), foi descrito apenas um trabalho conjunto para *job shop*, *setup* independente e *non-batch* (publicado por

Wilbrecht e Prescott, 1969), tal como o escopo da presente pesquisa, o que indica o potencial de estudo deste problema. A *survey* ainda sugere que Wilbrecht e Prescott (1969) foram os primeiros a estudar a influência dos tempos de *setup* no desempenho do ambiente *job shop*.

O exame da literatura mostra que, diferentemente da frequência dos problemas com tempos de *setup* dependentes, os trabalhos que abordam *job shop* com *setup* independente são escassos, e a maioria considera o *setup* entre lotes de produção. As pesquisas encontradas com esta abordagem são descritas a seguir.

Sotskov *et al.* (1999) consideraram *job shop* com tempos de *setup* independentes da sequência entre lotes de produção, com técnicas de inserção de tarefas combinadas com o algoritmo *beam search*, que foram comparadas com heurísticas construtivas baseadas em regras de prioridade. Um problema com famílias compostas por tarefas idênticas e *setup* independente da sequência (de famílias) foi abordado por Low *et al.* (2004).

O problema de *job shop* com lotes de produção e *setup* independente foi ainda enfocado por Jeong *et al.* (1999), Aldakhilallah e Ramesh (2001) e Rahime e Arslan (2009), todos com a aplicação de métodos heurísticos a ambientes de diversos tipos de restrições e especificidades, como possibilidade de reentrada de tarefas, divisão de lotes e filas de transporte.

Müller e Gómez (2007) propuseram estratégias de escalonamento por meio de um modelo aplicado ao *job shop*, que considerou o *makespan* e o tempo total de atraso. O modelo era composto por uma função objetivo que refletiu as estratégias de otimização, e de uma arquitetura que era dividida em cinco fases. A primeira fase era responsável por obter a demanda de produção; a segunda fase era realizada por meio da teoria dos grafos, a geração da matriz “partes versus máquinas”, em que aplicaram o algoritmo de identificação de agrupamento; na terceira fase, geraram as famílias de partes para o escalonamento inicial; na quarta fase aplicaram o algoritmo busca tabu para obter o escalonamento final; e por último, a quinta fase que era a gravação no plano de produção, que possibilitou observar seus históricos e compará-los como o escalonamento efetivo. Em muitos casos, obtiveram soluções ótimas comparadas a outros modelos e em outros obtiveram soluções aproximadas.

Freitas e Vieira (2010) analisaram as principais tendências do método de otimização por colônia de formigas em programação de *job shop*, com a

consideração dos tempos de *setup* de forma explícita. Outro importante estudo do tempo padrão do *setup* independente da sequência e dependente da habilidade dos operadores foi realizado por Song *et al.* (2016).

Chan *et al.* (2011) desenvolveram um algoritmo genético flexível em que as sequências e os roteiros podem ser variáveis. O trabalho foi realizado para uma fábrica de chapas metálicas paralelas, para os tempos de *setup* explícitos. Os autores afirmaram que os resultados da programação do seu algoritmo foram considerados melhores que os obtidos por regras de sequenciamento simples.

O problema de *flexible job shop*, em que há múltiplas máquinas em cada etapa pelas quais as tarefas devem passar, com tempos de *setup* que podem ser independentes, dependentes ou nulos, foi abordado por Chin (1995). As medidas de desempenho consideradas foram o tempo médio de fluxo, atraso médio e tempo médio ocioso de máquina.

Como pode ser visto, não foram encontrados outros trabalhos abordando o mesmo problema tratado na presente pesquisa, evidenciando a contribuição deste estudo.

3 *JOB SHOP* COM DUAS MÁQUINAS E *SETUP* INDEPENDENTE

Como mencionado, o Algoritmo de Jackson (1956) fornece a solução ótima para o problema denotado por $J2||C_{max}$, ou seja, *job shop* com duas máquinas e minimização do *makespan*. Este algoritmo pode ser facilmente adaptado para o problema com *setup* independente, conforme será apresentado nesta seção.

Antes, é preciso revisar o Algoritmo de Jackson (1956) original, conforme descrito a seguir. Este método separa as tarefas em dois grupos: as que visitam primeiro a máquina M1 e depois a máquina M2 e, opostamente, as que visitam primeiro a máquina M2 e em seguida a máquina M1.

> *ALGORITMO DE JACKSON:*
>
> *PASSO 1* – Ordene as tarefas que visitam primeiro a máquina M1 e depois a M2, de acordo com o Algoritmo de Johnson, formando a sequência S12.

PASSO 2 – Ordene as tarefas que visitam primeiro a máquina M2 e depois a M1, de acordo com o Algoritmo de Johnson, formando a sequência S21.

PASSO 3 – A programação ótima é obtida combinando as sequências da seguinte forma: M1: S12 + S21 e M2: S21 + S12.

Pode ainda haver tarefas que visitam apenas uma máquina, M1 ou M2. Estas tarefas farão parte dos conjuntos S1 e S2, respectivamente, em qualquer ordem. E neste caso, a programação ótima de cada máquina será dada por: M1: S12 + S1 + S21 e M2: S21 + S2 + S12.

É necessário ainda revisar o Algoritmo de Johnson (1954), originalmente usado para encontrar a solução ótima do problema de *flow shop* com duas máquinas e empregado no procedimento acima. Seja p_{jk} o tempo de processamento da tarefa J_j na máquina M_k.

ALGORITMO DE JOHNSON:

PASSO 1 – Determine min{p_{jk}}.

PASSO 2 – Se min{p_{jk}} pertencer à primeira máquina, coloque a tarefa na primeira posição disponível da sequência, senão coloque-a na última posição disponível.

PASSO 3 – Desconsiderando a tarefa alocada, repita os passos 1 e 2 até que todas as tarefas sejam programadas.

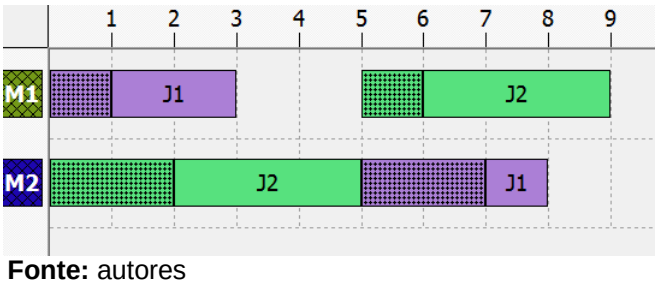
A principal diferença estrutural entre o problema com *setup* independente e o problema sem *setup* explícito que influenciaria na programação e em medidas de desempenho regulares seria a possibilidade de se antecipar o *setup*. Conforme já definido, o *setup* antecipado é aquele que pode ser iniciado mesmo antes da operação estar liberada, ou seja, enquanto ela ainda está sendo executada na máquina anterior. Embora algumas atividades de *setup*, como inspeção da máquina e limpeza, possam ser feitas antecipadamente, existem aquelas que só podem ser

realizadas quando o produto estiver presente na máquina, por exemplo, operações de posicionamento e ajustes, ou seja, de forma não antecipada.

Nesta parte da pesquisa optou-se por considerar os tempos de *setup* não antecipados, portanto aqui não se pode fazer uso da "vantagem" da antecipação. O *setup* de cada tarefa só pode iniciar após a liberação da própria tarefa (ou término do processamento na máquina anterior). Isto torna as programações análogas para o problema com duas máquinas: pode-se aplicar o Algoritmo de Jackson à situação com *setup* explícito com a utilização das somas dos tempos de processamento e de *setup* (como se o *setup* fosse implícito).

As Figuras 2 e 3 mostram exemplos de programação em *job shop* com duas máquinas e tempos de *setup* explícitos e implícitos, respectivamente.

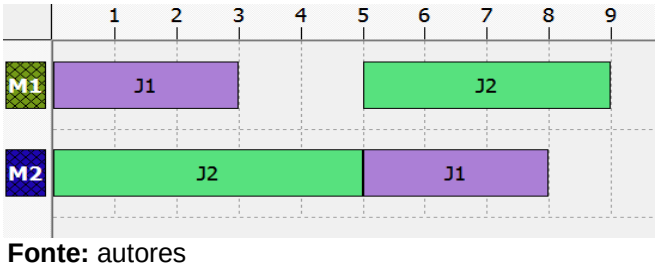
Figura 2 – Programação em *job shop* com duas máquinas e *setup* explícito (independente da sequência)



Fonte: autores

Note na Figura 2 que o *setup* é também não antecipado, ou seja, só é iniciado quando a tarefa estiver liberada (finalizada na máquina anterior). Isto pode ser visto na execução do *setup* da tarefa J2 na máquina M1, que só é iniciado no instante 5, mesmo com a máquina M1 já disponível a partir do instante 3. Para respeitar a sua rota, a tarefa J2 deve ser completamente processada na máquina M2 para só então iniciar o seu *setup* na máquina M1.

Figura 3 – Programação em *job shop* com duas máquinas e *setup* implícito (independente da sequência), incluído no tempo de processamento da tarefa



Fonte: autores

Como pode ser observado nas Figuras 2 e 3, embora na prática os problemas sejam diferentes, os problemas podem ser considerados análogos, e indica que o Algoritmo de Jackson é perfeitamente aplicável ao caso com *setup* explícito não antecipado. Esta transformação do problema com *setup* explícito não antecipado no problema com *setup* implícito não requer modificações no problema nem interfere na sua estrutura, já que a solução do Algoritmo de Jackson fornece a solução ótima para ambos.

A seguir será apresentada a adaptação do Algoritmo de Jackson para o caso com *setup* explícito não antecipado.

> *ADAPTAÇÃO DO ALGORITMO DE JACKSON:*
>
> *PASSO 1* – Considerando a soma dos tempos de processamento e *setup*, ordene as tarefas que visitam primeiro a máquina M1 e depois a M2, de acordo com o Algoritmo de Johnson, formando a sequência S12.
>
> *PASSO 2* – Considerando a soma dos tempos de processamento e *setup*, ordene as tarefas que visitam primeiro a máquina M2 e depois a M1, de acordo com o Algoritmo de Johnson, formando a sequência S21.
>
> *PASSO 3* – A programação ótima é obtida combinando as sequências da seguinte forma: M1: S12 + S21 e M2: S21 + S12.

O clássico Algoritmo de Johnson (1954) é bem conhecido por fornecer a solução ótima para o problema de minimização do *makespan* em *flow shop* com duas máquinas ($F2||C_{max}$) e, ao ser aplicado no presente estudo, também precisa ser adaptado para o problema com *setup* explícito, conforme descrito a seguir. Sejam p_{jk} e s_{jk}, respectivamente, o tempo de processamento e de *setup* da tarefa J_j na máquina M_k.

> *ADAPTAÇÃO DO ALGORITMO DE JOHNSON:*

PASSO 1 – Determine min{p_{jk} + s_{jk}}.

PASSO 2 – Se min{p_{jk} + s_{jk}} pertencer à primeira máquina visitada (independente de ser M1 ou M2), coloque a tarefa na primeira posição disponível da sequência, senão coloque-a na última posição disponível.

PASSO 3 – Desconsiderando a tarefa alocada, repita os passos 1 e 2 até que todas as tarefas sejam programadas.

Além de fornecer a solução ótima, a grande vantagem desta adaptação do Algoritmo de Jackson para *job shop* com duas máquinas e *setup* independente é que ela pode ser aplicada em problemas com qualquer número de tarefas, diferentemente de outros métodos de solução exata, como os modelos de programação linear inteira mista, que possuem a restrição de viabilidade do tempo computacional a depender do tamanho do problema-teste.

Para maior clareza do método proposto, será apresentado a seguir um exemplo numérico de um *job shop* com seis tarefas e duas máquinas, em que será aplicada a adaptação do Algoritmo de Jackson descrita nesta seção. Os tempos de processamento (p_{jk}) e de *setup* (s_{jk}) de cada tarefa J_j em cada máquina M_k são apresentados nas Tabelas 1 e 2. Já as rotas das tarefas nas máquinas, ou seja, a ordem em que as operações op_{jk} das tarefas J_j passam nas máquinas M_k são descritas na Tabela 3.

Tabela 1 – Tempos de processamento das tarefas nas máquinas

Tempo de processamento	**J1**	**J2**	**J3**	**J4**	**J5**	**J6**
p_{j1}	2	3	4	6	8	7
p_{j2}	5	8	1	3	1	2

Fonte: autores

Tabela 2 – Tempos de *setup* das tarefas nas máquinas

Tempo de *setup*	**J1**	**J2**	**J3**	**J4**	**J5**	**J6**
s_{j1}	3	1	2	4	1	3
s_{j2}	2	4	2	3	2	2

Fonte: autores

Tabela 3 – Rotas das tarefas nas máquinas

Rota	J1	J2	J3	J4	J5	J6
op_{j1}	M1	M2	M2	M1	M1	M2
op_{j2}	M2	M1	M1	M2	M2	M1

Fonte: autores

Como parte da resolução, a Tabela 4 apresenta a soma dos tempos de processamento e de *setup* de cada uma das tarefas.

Tabela 4 – Soma dos tempos de processamento e de *setup* das tarefas

Soma dos tempos	J1	J2	J3	J4	J5	J6
$p_{j1}+s_{j1}$	5	4	6	10	9	10
$p_{j1}+s_{j2}$	7	12	3	6	3	4

Fonte: autores

Separando em dois conjuntos, das tarefas com rota M1–M2 e daquelas com que percorrem M2–M1, tem-se a Tabela 5.

Tabela 5 – Conjuntos das tarefas com diferentes rotas

Rota	M1–M2			M2–M1		
Máquinas	**J1**	**J4**	**J5**	**J2**	**J3**	**J6**
M1	5	10	9	4	6	10
M2	7	6	3	12	3	4

Fonte: autores

Com a aplicação do Algoritmo de Johnson separadamente para cada rota da Tabela 5, as sequências parciais são:

Sequência S12 (rota M1–M2): J1 – J4 – J5.

Sequência S21 (rota M2–M1): J3 – J6 – J2.

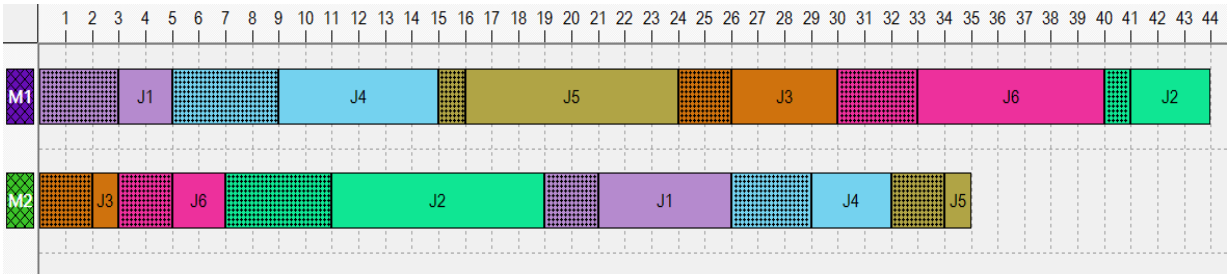
Assim, a programação ótima completa é a seguinte:

Máquina M1 (S12 + S21): J1 – J4 – J5 – J3 – J6 – J2.

Máquina M2 (S21 + S12): J3 – J6 – J2 – J1 – J4 – J5.

Esta programação é apresentada na Figura 4.

Figura 4 – Programação dada pelo algoritmo proposto



Fonte: autores

Especificamente neste caso, a análise gráfica também confirma que a programação obtida pela adaptação proposta é ótima, pois não há tempos de espera entre as operações, ou seja, não há como reduzir a duração total da programação. O *makespan* ótimo é, portanto, $C^*_{máx}$ = 44.

Nesta seção, foi discutida a adaptação do Algoritmo de Jackson para o problema com *setup* não antecipado. É importante salientar que é perfeitamente possível aplicar a mesma adaptação no caso do *setup* antecipado, porém sem garantir a solução ótima, pois a programação não mais seria análoga ao problema com *setup* implícito. Isto ocorreria porque a antecipação do *setup* poderia acarretar adiantamento nos instantes de término das tarefas e afetar o *makespan*, o que não ocorre quando o *setup* não é antecipado.

4 MODELOS MATEMÁTICOS PROPOSTOS

4.1 *Job shop* com *m* máquinas e *setup* não antecipado

Uma das formas de se obter a solução ótima de problemas *job shop* com mais do que duas máquinas é a modelagem de programação linear inteira mista. A seguir é proposto um modelo para o problema com *n* tarefas e *m* máquinas em que o *setup* é não antecipado.

Variáveis de decisão: $X_{ijk} = \begin{cases} 1, se\ a\ tarefa\ J_i\ precede\ imediatamente \\ \quad a\ tarefa\ J_j\ na\ máquina\ M_k \\ 0, caso\ contrário \end{cases}$

Variáveis auxiliares: C_{jk} = data de término da tarefa J_j na máquina M_k

C_{max} = *makespan*

Modelo:

Min C_{max} (1)

Sujeito a

$$\sum_{j=0}^{n} X_{ijk} = 1, \quad i = 0, \dots, n, k = 1, \dots, m, \tag{2}$$

$$\sum_{i=0}^{n} X_{ijk} = 1, \quad j = 0, \dots, n, k = 1, \dots, m, \tag{3}$$

$$C_{j[1]} \geq p_{j[1]} + s_{j[1]}, \quad j = 1, \dots, n, \tag{4}$$

$$C_{j[k+1]} \geq C_{j[k]} + p_{j[k+1]} + s_{j[k+1]}, \quad j = 1, \dots, n, \tag{5}$$

$$C_{jk} - C_{ik} + B\left(1 - X_{ijk}\right) \geq p_{jk} + s_{jk}, \quad i = 0, \dots, n, j = 1, \dots, n, k = 1, \dots, m \tag{6}$$

$$C_{ik} - C_{jk} + B\, X_{ijk} \geq p_{ik} + s_{ik}, \quad i = 0, \dots, n, j = 1, \dots, n, k = 1, \dots, m \tag{7}$$

$$C_{max} - C_{jk} \geq 0, \quad j = 1, \dots, n, k = 1, \dots, m \tag{8}$$

$$C_{jk} \geq 0, \qquad X_{ijk} \in \{0,1\} \quad i = 0, \dots, n, j = 1, \dots, n, k = 1, \dots, m \tag{9}$$

A função objetivo (1) expressa a minimização da duração total da programação. As restrições (2) e (3) garantem respectivamente que cada tarefa tenha uma única sucessora e uma única antecessora em cada máquina. O conjunto de restrições (4) assegura que a primeira operação de cada tarefa será completada após o respectivo tempo de processamento e de *setup*. A notação [*k*] indica a *k*-ésima máquina visitada pela tarefa. Já as restrições (5) impõem que cada operação *k*+1 da tarefa J_j será concluída após o término da operação *k* e dos tempos de processamento e de *setup* da operação *k*+1.

As expressões (6) e (7) são restrições disjuntivas que garantem a consistência das datas de término se na máquina M_k a tarefa J_i precede a tarefa J_j, ou quando a tarefa J_j precede a tarefa J_i. Se X_{ijk} = 0, a restrição (6) é desativada e se X_{ijk} = 1, então a restrição (7) é desativada. A restrição (8) assegura que o valor do *makespan* seja a maior data de término. Por fim, as restrições (9) estabelecem o domínio das variáveis, sendo que as datas de término C_{jk} devem ser positivas e as variáveis de decisão X_{ijk} binárias.

4.2 *Job shop* com *m* máquinas e *setup* antecipado

A seguir é proposto um modelo para o problema com *n* tarefas e *m* máquinas em que o *setup* pode ser antecipado.

Variáveis de decisão: $X_{ijk} = \begin{cases} 1, se\ a\ tarefa\ J_i\ precede\ imediatamente \\ \quad a\ tarefa\ J_j\ na\ máquina\ M_k \\ 0, caso\ contrário \end{cases}$

Variáveis auxiliares: C_{jk} = data de término da tarefa J_j na máquina M_k

C_{max} = *makespan*

Modelo:

Min C_{max} (10)

Sujeito a

$$\sum_{j=0}^{n} X_{ijk} = 1, \qquad i = 0, \dots, n, k = 1, \dots, m, \tag{11}$$

$$\sum_{i=0}^{n} X_{ijk} = 1, \qquad j = 0, \dots, n, k = 1, \dots, m, \tag{12}$$

$$C_{j[1]} \geq p_{j[1]} + s_{j[1]}, \qquad j = 1, \dots, n \tag{13}$$

$$C_{j[k+1]} \geq C_{i[k+1]} + s_{j[k+1]} + p_{j[k+1]}, \qquad j = 1, \dots, n, k = 1, \dots, m-1, \tag{14}$$

$$C_{j[k+1]} \geq C_{j[k+1]} + p_{j[k]}, \qquad j = 1, \dots, n, k = 1, \dots, m-1, \tag{15}$$

$$C_{jk} - C_{ik} + B(1 - X_{ijk}) \geq p_{jk} + s_{jk}, \qquad i = 0, \dots, n, j = 1, \dots, n, k = 1, \dots, m \tag{16}$$

$$C_{ik} - C_{jk} + B\,X_{ijk} \geq p_{ik} + s_{ik}, \qquad i = 0, \dots, n, j = 1, \dots, n, k = 1, \dots, m \tag{17}$$

$$C_{max} - C_{jk} \geq 0, \qquad j = 1, \dots, n, k = 1, \dots, m \tag{18}$$

$$C_{jk} \geq 0, \quad X_{ijk} \in \{0,1\} \qquad i = 0, \dots, n, j = 1, \dots, n, k = 1, \dots, m \tag{19}$$

A função objetivo (10) expressa a minimização da duração total da programação (*makespan*). As restrições (11) e (12) garantem respectivamente que cada tarefa tenha uma única sucessora e uma única antecessora em cada máquina. O conjunto de restrições (13) assegura que a primeira operação de cada tarefa será completada após o respectivo tempo de processamento e de *setup*. A notação [*k*] indica a *k*-ésima máquina visitada pela tarefa. Já as restrições (14) e (15) são restrições disjuntivas que garantem a consistência das datas de término se na

máquina M_k a tarefa J_i precede a tarefa J_j nos dois casos: quando a data de inicio da tarefa J_j é definida pelo término da tarefa J_i mais o *setup* s_{jk} ; ou quando é definida pelo término da tarefa J_j máquina anterior.

As expressões (16) e (17) são restrições disjuntivas que garantem a consistência das datas de término se na máquina M_k a tarefa J_i precede a tarefa J_j, ou quando a tarefa J_j precede a tarefa J_i. Se X_{ijk} = 0, a restrição (16) é desativada e se X_{ijk} = 1, então a restrição (17) é desativada. A restrição (18) assegura que o valor do *makespan* seja a maior data de término. Por fim, as restrições (19) estabelecem o domínio das variáveis, sendo que as datas de término C_{jk} devem ser positivas e as variáveis de decisão X_{ijk} binárias.

5 EXPERIMENTAÇÃO COMPUTACIONAL E RESULTADOS

Para a implementação do modelo foi utilizado o software modelador MPL (*Mathematical Programming Language*) e o otimizador Gurobi 1.0.4.

O MPL é um *software* de modelagem de sistemas avançados que permite desenvolver modelos complexos de otimização de forma clara, concisa e eficiente. O principal objetivo de uma linguagem de modelagem é recuperar dados de uma fonte de dados estruturados como um banco de dados e gerar uma matriz que o *software* de otimização pode manipular. O modelador possui uma característica que conecta com os resolvedores diretamente por meio de sua memória. Fornece opções de caixas de diálogo fáceis de compreender.

O MPL trabalha com alguns resolvedores (*solvers*) de otimização rápidos e avançados, que oferecem uma solução completa que contendo praticamente todos os recursos que o desenvolvedor do modelo precisaria para um *software* de otimização.

Para realizar os testes computacionais utilizou-se um computador pessoal com processador Intel® Dual Core™ 2.53 GHz, 4 GB de memória RAM e sistema operacional de 32 Bits, sob plataforma Windows.

Os parâmetros da experimentação computacional foram definidos com base em trabalhos publicados na literatura. Tal como Al-Salem (2004), foram gerados problemas com 3, 5, 7, 9, 10, 20, 50 e 100 tarefas. O número de máquinas em cada

job shop foi 3, 5 e 7. Os tempos de processamento no *job shop* seguiram a distribuição uniforme no intervalo *U*[1,99] e os tempos de *setup* em *U*[0,99].

Com combinações do número de tarefas e de máquinas, foram gerados 11 problemas para o caso com *setup* não antecipado, mostrados na Tabela 6, junto com os respectivos tempos de execução, medidos em segundos, o número de iterações e a solução obtida.

Tabela 6 – Resultados dos problemas com *setup* não antecipado

Nº de tarefas	Nº de máquinas	Tempo de execução (s)	Nº de iterações	Solução obtida
3	3	0,10	219	56
5	3	17,57	345869	322
7	5	2255	9295140	822
9	3	4906	71375633	1374
10	3	7200 *	106621507	1951**
10	5	7200 *	63225403	3737**
20	3	7200 *	47438752	4228**
20	7	7200 *	11447558	10900**
50	5	***	***	***
50	7	***	***	***
100	3	***	***	***

*Tempo de execução limitado por 7200 s.
**Melhor solução encontrada durante a execução.
***Algoritmo finalizado por memória 'insuficiente'.

Fonte: autores

Os mesmos problemas foram usados na resolução do caso em que o *setup* pode ser antecipado, cujos resultados são apresentados na Tabela 7 a seguir.

Tabela 7 – Resultados dos problemas com *setup* antecipado

Nº de tarefas	Nº de máquinas	Tempo de execução (s)	Nº de iterações	Solução obtida
3	3	0,80	192	46
5	3	12,56	310834	234
7	5	1854	9134915	645
9	3	4283	81342132	974
10	3	7200 *	109851707	1251**
10	5	7200 *	72226940	2932**
20	3	7200 *	51103282	3289**
20	7	7200 *	11917212	9962**
50	5	***	***	***
50	7	***	***	***
100	3	***	***	***

*Tempo de execução limitado por 7200 s.
**Melhor solução encontrada durante a execução.
***Algoritmo finalizado por memória 'insuficiente'.

Fonte: autores

Pela análise dos resultados obtidos na execução, nota-se que os dois modelos fornecem a solução ótima para problemas com até nove tarefas em um tempo computacional aceitável. Já para problemas com 10 e 20 tarefas, o tempo de CPU passa a ser uma limitação para se encontrar a solução ótima do problema. Nestes casos, optou-se por limitar a execução em 7200 segundos (2 horas), obtendo-se uma solução viável. Em problemas com 50 e 100 tarefas, o algoritmo foi interrompido por falta de memória.

Como esperado, à medida que a quantidade de tarefas aumenta, as iterações gastam mais tempo para serem executadas. Isso acarreta uma grande diferença no número de iterações em problemas com 10 e 20 tarefas, já que o tempo de execução em ambas as opções foi o mesmo. Como pode ser visto nas Tabelas 6 e 7, problemas com 20 tarefas executaram um número muito menor de iterações do que aqueles com 10 tarefas.

A comparação das Tabelas 6 e 7 mostra que o número de iterações na resolução de problemas em que o tempo de execução foi limitado a 7200 segundos foi maior nos casos em que o *setup* pode ser antecipado. Já quando não há limitação de tempo, o número de iterações foi menor, assim como o tempo de execução para encontrar a solução ótima. Conforme previsto, a solução encontrada diminui consideravelmente quando o *setup* pode ser antecipado, mostrando que é possível reduzir substancialmente o tempo de produção quando se permite antecipar o *setup*.

A título de ilustração, é apresentada a solução do menor problema resolvido na experimentação (com 3 tarefas e 3 máquinas), para os casos com *setup* antecipado e não antecipado. Os dados do problema são apresentados nas Tabelas 8 a 10 a seguir.

Tabela 8 – Tempos de processamento das tarefas nas máquinas

Tempo de processamento	**J1**	**J2**	**J3**
p_{j1}	3	6	5
p_{j2}	1	9	10
p_{j3}	8	7	8

Fonte: autores

Tabela 9 – Tempos de *setup* das tarefas nas máquinas

Tempo de *setup*	J1	J2	J3
s_{j1}	2	7	7
s_{j2}	4	2	9
s_{j3}	8	6	1

Fonte: autores

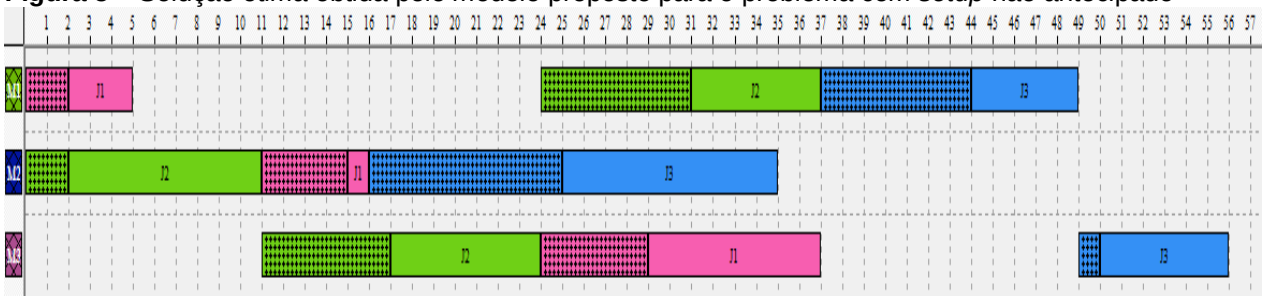
Tabela 10 – Rotas das tarefas nas máquinas

Rota	J1	J2	J3
op_{j1}	M1	M2	M2
op_{j2}	M2	M3	M1
op_{j3}	M3	M1	M3

Fonte: autores

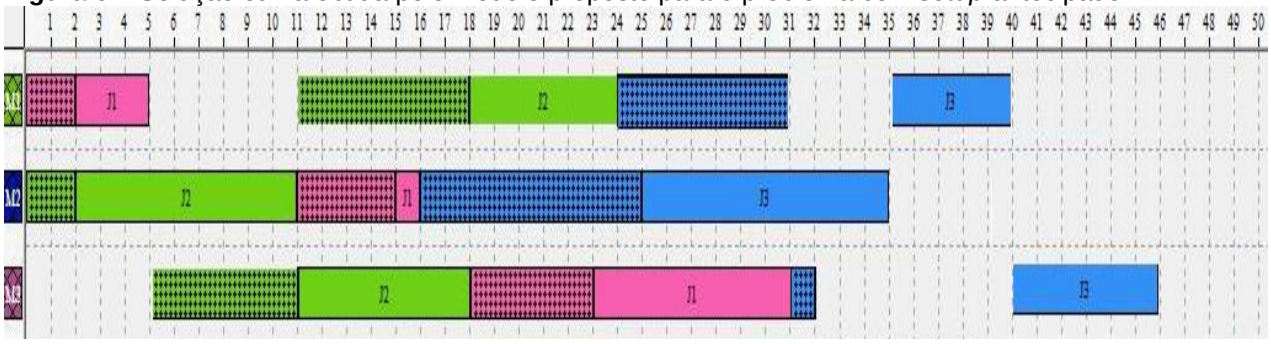
Os gráficos das Figuras 5 e 6 apresentam as programações ótimas obtidas por meio do modelo matemático para os casos com *setup* não antecipado e antecipado, respectivamente.

Figura 5 – Solução ótima obtida pelo modelo proposto para o problema com *setup* não antecipado



Fonte: autores

Figura 6 – Solução ótima obtida pelo modelo proposto para o problema com *setup* antecipado



Fonte: autores

A Figura 5 mostra a programação das tarefas nas máquinas e o *makespan* ótimo de 56 unidades de tempo ($C^*_{máx}$ = 56) da solução fornecida pelo modelo proposto em que o tempo de *setup* não antecipado. Já a Figura 6 apresenta o *makespan* ótimo de 46 unidades de tempo ($C^*_{máx}$ = 46) da solução fornecida pelo

modelo para o caso em que o *setup* é antecipado. Conforme salientado, observa-se neste exemplo que o *makespan* diminui consideravelmente em casos que o *setup* é antecipado.

6 CONSIDERAÇÕES FINAIS

Nesta pesquisa foram propostos métodos de solução exata para o problema de minimização do *makespan* em *job shop* com tempos de *setup* independentes da sequência, denotado por $Jm|s_{jk}|C_{max}$.

Para problemas com duas máquinas, pode-se utilizar a adaptação do Algoritmo de Jackson que considera tempos de *setup* explícitos e que também fornece a solução ótima. O novo algoritmo resolve problemas com qualquer número de tarefas. Para ambientes com três ou mais máquinas, foram apresentados modelos de programação linear inteira mista tanto para problemas com tempos de *setup* antecipados como também não antecipados.

Os modelos matemáticos propostos foram implementados usando o *software* modelador MPL e o resolvedor Gurobi. Como característico de modelos de programação linear inteira mista, é possível encontrar a solução ótima em tempo computacional aceitável apenas para problemas de pequeno porte. No caso desta pesquisa, obteve-se a solução ótima para problemas com até 9 tarefas e 3 máquinas. Limitando-se o tempo computacional a 7200 segundos, foram encontradas soluções viáveis para problemas com 10 e 20 tarefas.

Evidentemente, no problema em que o *setup* é antecipado, o *makepan* é menor do que o caso em que o *setup* é não antecipado. Entretanto, esta característica depende do ambiente de produção e não do método de solução. Problemas com *setup* não antecipado são encontrados em sistemas produtivos em que a peça ou o produto precisam estar presentes na máquina para que as operações de preparação sejam realizadas.

Como esperado, para problemas de grande porte é aconselhável utilizar métodos heurísticos na busca de boas soluções. Para futuros trabalhos, sugere-se a criação de métodos heurísticos para resolução problemas de grande porte.

AGRADECIMENTOS

A pesquisa relatada neste artigo teve o apoio da FAPEG, da CAPES e do CNPq. Os autores também agradecem as contribuições dos revisores anônimos.

REFERÊNCIAS

ADAMS, J.; BALAS, E.; ZAWACK, D. The shifting bottleneck procedure for job shop scheduling. **Management Science**, v. 34, p. 391-401, 1988. http://dx.doi.org/10.1287/mnsc.34.3.391

ALDAKHILALLAH, K.A.; RAMESH, R. Cyclic scheduling heuristic for a re-entrant job shop manufacturing environment. **International Journal of Production Research**, v. 39, p. 2635-2657, 2001. http://dx.doi.org/10.1080/00207540110047711

ALLAHVERDI, A. Third comprehensive survey on scheduling problems with setup times/costs. **European Journal of Operational Research**, v. 246, n. 2, p. 345-378, 2015. http://dx.doi.org/10.1016/j.ejor.2015.04.004

ALLAHVERDI, A.; 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. http://dx.doi.org/10.1016/j.ejor.2006.06.060

ALLAHVERDI, A.; GUPTA, J.N.D.; ALDOWAISAN, T. A review of scheduling research involving setup considerations. **Omega – International Journal of Management Science**, v. 27, p. 219-239, 1999. http://dx.doi.org/10.1016/S0305-0483(98)00042-5

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. http://dx.doi.org/10.1016/j.ejor.2006.09.010

AL-SALEM, A. A heuristic to minimize makespan in proportional parallel flow shops. **International Journal of Computing & Information Sciences**, v. 2, p. 98-107, 2004. http://www.ijcis.info/Vol2N2/98-107OKS.pdf

BLAZEWICZ, J.; DOMSCHKE, W., PESCH, E. The job shop scheduling problem: conventional and new solution techniques. **European Journal of Operational Research**, v. 93, p. 1-33, 1996. http://dx.doi.org/10.1016/0377-2217(95)00362-2

CHAN, F. T. S.; CHOY K.L.; BIBHUSHAN. A genetic algorithm-based scheduler for multiproduct parallel machine sheet metal job shop. **Expert Systems with Applications**, v. 38, p. 8703-8715, 2011. http://dx.doi.org/10.1016/j.eswa.2011.01.078

CHIN, Y.L. Job shop scheduling heuristics for sequence dependent setup. **Computers & Industrial Engineering**, v. 29, p. 279-283, 1995. http://dx.doi.org/10.1016/0360-8352(95)00085-F

FREITAS, F.F.T.; VIEIRA, G.E. Tendências de aplicações da otimização por colônia de formigas na programação de *job-shops*. **Revista Produção Online**, v. 10, n. 1, p. 95-123, 2010. http://dx.doi.org/10.14488/1676-1901.v10i1.441

FUCHIGAMI, H. Y. **Sequenciamento da produção em sistemas *flow shop***. UFG, Goiânia – GO, 2015.

JACKSON, J. R. An extension of Johnson's results on job lot scheduling. **Naval Research Logistics Quarterly**, v. 3, p. 201-203, 1956. http://dx.doi.org/10.1002/nav.3800030307

JEONG, H.; PARK, J. LEACHMAN, R.C. A batch splitting method for a job shop scheduling problem in an MRP environment. **International Journal of Production Research**, v. 37, p. 3583-3598, 1999. http://dx.doi.org/10.1080/002075499190194

JOHNSON, S. M. Optimal two- and three- stage production schedules with setup times included. **Naval Research Logistic Quarterly**, v. 1, p. 61-68, 1954. http://dx.doi.org/10.1002/nav.3800010110

LOW, C.; HSU, C.-M., HUANG, K.-I. Benefits of lot splitting in job-shop scheduling. **International Journal of Advanced Manufacturing Technology**, v. 24, p. 773-780, 2004. http://dx.doi.org/10.1007/s00170-003-1785-9

MÜLLER, G. I.; GÓMEZ, A. T. Um estudo do comportamento dos tempos de *makespan*, atraso e *setup* no problema de escalonamento de *job shop*. **Hifen***,* v. 31, n. 59-60, p. 148-154, 2007.

PACHECO, R.F.; SANTORO, M.C. Proposta de classificação hierarquizada dos modelos de solução para o problema de *job shop scheduling*. **Gestão & Produção**, v. 6, n. 1, p. 1-15, 1999. http://dx.doi.org/10.1590/S0104-530X1999000100001

PARVEEN, S.; ULLAH, H. Review on job-shop and flow-shop scheduling using multi criteria decision making. **Journal of Mechanical Engineering**, v. 41, n. 2, p. 130-146, 2010.

PINEDO, M. **Scheduling***:* theory, algorithms, and systems. 5. ed. New Jersey: Prentice-Hall, 2015.

RAHIME, S.E.; ARSLAN, O. Simulation analysis of lot streaming in job shops with transportation queue disciplines. **Simulation Modelling Practice and Theory**, v. 17, p. 442-453, 2009. http://dx.doi.org/10.1016/j.simpat.2008.10.002

SONG, H.; YI, S.; SHEN, H. A study of job shop standard setup time quota. **Production Management**, v. 10, n. 2, p. 185-196, 2016. http://dx.doi.org/10.1007/s11740-015-0649-0

SOTSKOV, Y.N.; TAUTENHAHN, T.; WERNER, F. On the application of insertion techniques for job shop problems with setup times. **RAIRO – Operations Research**, v. 33, n. 2, p. 209-245, 2010. http://dx.doi.org/10.1051/ro:1999110

TARANTILIS, C. D.; KIRANOUDIS, C. T. A list-based threshold accepting method for job shop scheduling problems**. International Journal of Production Economics**, v. 77, p. 159–171, 2002. http://dx.doi.org/10.1016/S0925-5273(01)00231-6

WILBRECHT, J.K.; PRESCOTT, W.B. The influence of setup time on job shop performance. **Management Science**, v. 16, p. B274-B280, 1969. http://dx.doi.org/10.1287/mnsc.16.4.B274

Artigo recebido em 20/06/2016 e aceito para publicação em 25/08/2016
DOI: http://dx.doi.org/10.14488/1676-1901.v17i1.2504