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

UM ESTUDO COMPUTACIONAL DE NOVAS REGRAS DE PRIORIDADE PARA *FLOW SHOP* COM TEMPOS DE *SETUP* E DIFERENTES DATAS DE LIBERAÇÃO

A COMPUTATIONAL STUDY OF NEW PRIORITY RULES FOR FLOW SHOP PROBLEM WITH SETUP TIMES AND DIFFERENT RELEASE DATES

Caio Soares de Araújo* E-mail: caio.ufg@gmail.com
Hélio Yochihiro Fuchigami** E-mail: heliofuchigami@ufg.br
*Universidade Federal de Goiás (UFG), Catalão, GO
**Universidade Federal de Goiás (UFG), Aparecida de Goiânia, GO

Resumo: Este trabalho aborda o sistema de produção *flow shop* com tempos de *setup* independentes da sequência, tarefas com diferentes instantes de liberação e dois problemas distintos: minimização da duração total da programação (*makespan*) e do tempo médio de fluxo (*flowtime*). Foi empreendido um amplo estudo computacional do desempenho de novas regras de prioridade, muito utilizadas na prática pelas empresas, aqui definidas com base na estrutura dos problemas analisados. Conforme é bem conhecido, a eficiência computacional das regras de prioridade como métodos de solução já é presumida. Portanto, nesta pesquisa focou-se na análise da eficácia, ou seja, da qualidade da solução atingida pela aplicação de tais regras nos dois problemas considerados. Os resultados provaram a aplicabilidade dos métodos propostos e identificaram as melhores regras para cada caso, com seus respectivos critérios de ordenação.

Palavras-chave: Programação da produção. *Flow shop*. Regras de prioridade. *Setup* independente. Datas de liberação.

Abstract: This research deals with the flow shop production system with sequence-independent setup times, different job release dates and two distinct problems: makespan and total flowtime minimization. It was undertaken a large computational study of the performance of new priority rules, widely used in practice by companies and defined here based on the structure of the problems analyzed. As is well known, the computational efficiency of the priority rules as solution methods is already presumed. Therefore, this work focused on the analysis of the effectiveness, i.e., the quality of the solution reached by the application of such rules in the two problems considered. The results proved the applicability of the proposed methods and identified the best rules for each case, with their respective ordering criteria.

Keywords: Scheduling. Flow shop. Priority rules. Sequence-independent setup times. Release dates.

1 INTRODUÇÃO

O problema de programação da produção em *flow shop* é bastante conhecido e consiste em sequenciar um conjunto de *n* tarefas que deverão ser processadas por *m* máquinas em série. O "*flow shop* permutacional" requer que em cada máquina a ordem de processamento das tarefas seja a mesma. O problema consiste em determinar a ordem das tarefas que otimiza uma medida de desempenho.

A minimização do *makespan*, ou seja, da duração total da programação, é relevante em situações onde um conjunto todo de pedidos (tarefas) precisa ser executado o mais cedo possível e, como critério de otimização, equivale à maximização da utilização dos recursos. Em outras situações reais, cada tarefa precisa ser processada o mais rápido possível; assim, é preferível a minimização do *flowtime* ao invés do *makepan*. O *flowtime*, ou tempo de fluxo das tarefas, é um objetivo particularmente importante em casos práticos em que se deseja reduzir os estoques ou o custo de manutenção de matéria-prima, por exemplo. Por estas razões, os dois problemas (com o *makespan* e o *flowtime* como medidas) foram considerados neste trabalho.

Muitas pesquisas em programação da produção desconsideram os tempos de *setup* (preparação das máquinas) ou então os incluem nos tempos de processamento das tarefas. Isto simplifica a análise dos resultados, porém afeta a qualidade da solução, principalmente quando tais tempos têm uma variabilidade relevante em função da ordenação das tarefas nas máquinas. Seu tratamento explícito influencia a solução, sobretudo quando puder ser antecipado, como é o caso do problema abordado neste trabalho (FUCHIGAMI, MOCCELLIN; 2016).

É mais realista considerar que algumas atividades de *setup* podem ser realizadas antecipadamente, ou seja, antes da liberação da tarefa ou do seu processamento na máquina anterior. Isto acarreta uma antecipação da conclusão final de cada tarefa (*flowtime*) e consequentemente da programação toda (*makespan*).

Além disso, a maioria dos trabalhos publicados na área de programação da produção utiliza como premissa a chegada simultânea de um conjunto de tarefas a serem processadas, ou seja, não considera a possibilidade de diferentes instantes de liberação (*release dates*), o que também é muito mais realista. Por este motivo, neste trabalho foi utilizado o tratamento de chegada dinâmica de tarefas.

De acordo com Graham et al. (1979) e T'Kindt e Billaut (2006), os problemas tratados nesta pesquisa podem ser representados na conhecida notação de três campos por $Fm|\text{prmu}, s_{jk}, r_j|C_{\max}$ e $Fm|\text{prmu}, s_{jk}, r_j|\overline{F}$, onde "*Fm*" e "prmu" indicam o ambiente *flow shop* permutacional com *m* máquinas, "s_{jk}" representa os tempos de *setup* (independentes da sequência) da tarefa J_j na máquina M_k, "r_j" é o instante de liberação da tarefa J_j, "$C_{\max}$" o *makespan* e $\overline{F}$ o tempo médio de fluxo das tarefas.

Os objetivos desta pesquisa são apresentar uma extensa revisão e atualização do estado-da-arte dos problemas de programação da produção com diferentes datas de liberação e avaliar computacionalmente o desempenho de regras de prioridade como método de programação nos ambientes *flow shop* com minimização do *makespan* e do *flowtime*, separadamente.

Este texto foi estruturado da seguinte forma: após esta introdução (seção 1), é apresentada a revisão da literatura (seção 2), as regras de prioridade propostas (seção 3), a experimentação computacional e a análise dos resultados (seção 4) e, finalmente, as considerações finais (seção 5).

2 REVISÃO DA LITERATURA

2.1 Problemas de programação com diferentes datas de liberação

É interessante notar que o trabalho pioneiro da área de programação da produção (*scheduling*), publicado por Johnson (1954), abordou o sistema *flow shop*, e não ambientes mais simples como o problema de máquina única. Ou seja, a história da programação da produção iniciou com o *flow shop*.

Nestas seis décadas desta área, centenas de pesquisas foram publicadas abordando o problema de *flow shop*, como pode ser observado nos trabalhos de revisão de MacCarthy e Liu (1993), Framinan, Gupta e Leisten (2004), Ruiz e Maroto (2005), Framinan, Leisten e Ruiz-Usano (2005), Reza-Hejazi e Saghafian (2005), Gupta e Stafford Jr. (2006), Potts e Strusevich (2009) e Tyagi, Varshney e Chandramouli (2013).

Neste artigo, optou-se pela **revisão da literatura com abordagem bibliográfica**, baseada principalmente em periódicos científicos internacionais, considerando especificamente as pesquisas em *flow shop* com diferentes datas de liberação. Revisões de trabalhos abordando as várias categorias de tempos e custos de *setup* podem ser encontradas em Allahverdi, Gupta e Aldowaisan (1999), Cheng, Gupta e Wang (2000), Allahverdi et al. (2008) e Allahverdi (2015).

Logo em 1977, Lenstra, Rinnooy Kan e Brucker (1977) provaram que mesmo o problema com duas máquinas $F2|r_j|C_{\max}$ é classificado como fortemente *NP-hard*. Desde então, muitos pesquisadores investigaram os algoritmos de aproximação de tempo polinomial (*polynomial time approximation algorithms*).

Como pode ser visto logo à frente, não foi encontrada nenhuma publicação abordando exatamente o mesmo problema desta pesquisa. Em muitos artigos, foi considerada a situação específica de duas máquinas. E é mais frequente encontrar a análise teórica do pior caso do problema (*worst-case performance*) por meio de algoritmos de aproximação, a chamada “otimalidade assintótica”. Isto evidencia a relevante contribuição deste trabalho.

Todos os artigos encontrados que abordam os vários ambientes de produção e consideram diferentes datas de liberação são descritos a seguir.

O problema de máquina única com minimização do tempo total de fluxo foi verificado em três trabalhos. Chu (1992a) propôs algumas heurísticas eficientes e um algoritmo *branch-and-bound* testado em exemplares com até 100 tarefas. Chu (1992b) aplicou uma regra de prioridade comprovada ótima local e apresentou análises do pior caso de algumas heurísticas. E Kaminsky e Simchi-Levi (2001) provaram a otimalidade assintótica da regra *shortest processing time among available jobs* (SPTA).

A otimalidade assintótica da regra SPTA para o problema de *flow shop* foi investigada por Ren et al. (2014) para minimização do tempo total de fluxo e por Bai (2015) para o instante de término quadrático total, incluindo alguns esquemas de melhoria.

Poucos trabalhos abordaram o problema de máquinas paralelas. Kurz e Askin (2001) formularam um modelo de programação inteira mista e várias heurísticas para a minimização do *makespan* com tempos de *setup* dependentes da sequência. Abedi et al. (2014) também apresentaram um modelo de programação linear inteira mista e duas meta-heurísticas (algoritmo genético de ordenação não dominada e algoritmo competitivo imperialista) para o problema de máquinas paralelas idênticas com processamento em lotes, diferentes datas de liberação, capacidades limitadas e biobjetivo de *makespan* e total de antecipação e atraso ponderados.

O problema de *flow shop* com duas máquinas e minimização do *makespan* foi estudado por vários pesquisadores. Potts (1985) fez a análise do pior caso de três heurísticas. Hall (1994) e Kovalyov e Werner (1997) apresentaram esquemas de aproximação polinomial. Tadei et al. (1998) descreveram vários procedimentos *branch-and-bound* e aplicaram diferentes limitantes inferiores e esquemas de ramificação em exemplares com até 200 tarefas. Kashyrskikh, Potts e Sevastianov

(2001) fizeram a análise do pior caso do algoritmo modificado de Potts (1985). Chihaoui et al. (2011) desenvolveram um algoritmo *branch-and bound* eficiente para exemplares com até 20 tarefas para o problema *no-wait* (sem espera entre as operações de uma tarefa) sujeito à restrição de não disponibilidade. Kalczynski e Kamburowski (2012) confrontaram a análise teórica da otimalidade assintótica com resultados empíricos de heurísticas.

A minimização do *makespan* em *flow shop* com duas máquinas e processamento em lotes é também bastante frequente na literatura. Sung e Yoon (1997) evidenciaram algumas propriedades do problema e desenvolveram um algoritmo de programação dinâmica. Sung e Kim (2002) formularam um modelo matemático e métodos heurísticos para as duas situações com e sem interrupção (*preemption*). Tang and Liu (2009) também propuseram um modelo de programação inteira mista e uma heurística baseada em programação dinâmica. Muthuswamy et al. (2012) estudaram o problema *no-wait*, propondo uma formulação matemática e um algoritmo *particle swarm optimization*. Chen et al. (2014) desenvolveram um modelo de programação inteira mista e um algoritmo híbrido de evolução diferencial discreta. Zhou et al. (2016) propuseram um algoritmo de estimação de distribuição para o problema com restrição *no-wait* e um tempo de *setup* constante na segunda máquina.

Um estudo computacional de um algoritmo *branch-and-bound*, combinando uma regra de ramificação adaptativa com uma estratégia de busca nebulosa para o problema de *flow shop* com três máquinas, foi desenvolvido por Cheng, Steiner e Stephenson (2001). O algoritmo resolveu eficientemente exemplares com até 200 tarefas.

O problema de minimização do máximo desvio de pontualidade (*lateness*) em *flow shop* genérico com *m* máquinas foi resolvido por Grabowski, Skubalska e Smutnicki (1983) com um algoritmo *branch-and-bound* para exemplares com até 50 tarefas. Bülbül, Kaminsky e Yano (2004) examinaram o problema de *flow shop* com *m* máquinas e objetivo de minimização da soma ponderada dos adiantamentos, atrasos e custos de manutenção em estoque intermediário. O problema foi formulado por meio de um modelo de programação inteira e duas heurísticas foram propostas com base em reformulações muito similares às de Dantzig-Wolf. A otimalidade assintótica de heurísticas propostas para *flow shop* não permutacional com minimização da soma

ponderada dos instantes de término foi analisada por Liu, Queyranne e Simchi-Levi (2005).

Um esquema de aproximação polinomial foi elaborado por Hall (1998) para *flow shop* com *m* máquinas, diferentes datas de liberação e tempos de entrega. Para o mesmo problema, Ladhari e Haouari (2006) propuseram um algoritmo *branch-and-bound* e mostraram a sua eficiência para exemplares com até 1000 tarefas e 6 máquinas. Uma extensiva experimentação computacional similar foi desenvolvida por Haouari and Ladhari (2007) para o problema de minimização do máximo desvio de pontualidade.

Bianco, Dell'Olmo e Giordani (1999) consideraram o problema de *flow shop* com *m* máquinas, restrição *no-wait*, e tempos de *setup* dependentes da sequência, propondo uma formulação matemática, dois limitantes inferiores e duas heurísticas. Ainda para o *flow shop* com *m* máquinas, Bai, Huo e Tang (2008) apresentaram um novo limitante inferior para o *makespan* e uma análise assintótica. Ren, Diao e Luo (2012) consideraram o problema com dois objetivos separados: *makespan* e instante de término quadrático total. Para o primeiro, eles estudaram a otimalidade assintótica e para o segundo, uma estratégia de melhoria com busca local foi apresentada para melhorar o desempenho da heurística clássica SPT. A minimização do makespan também foi abordada por Ren et al. (2015), que propuseram uma heurística com um esquema de busca local.

Józefczyk, Markowski e Balgabaeva (2014) desenvolveram um método *branch-and-bound*, heurísticas construtivas e um algoritmo busca tabu para minimização do *makespan* no caso especial de *flow shop* com roteamento (*routing flow shop*), em que as máquinas podem se mover ao longo das tarefas.

Finalmente no trabalho mais recente, Amirian e Sahraeian (2016) propuseram um algoritmo de evolução diferencial baseado no *simulated annealing* para o problema de *flow shop* não permutacional com diferentes datas de liberação, paradas aleatórias de máquinas (*breakdowns*), tempos de *setup past-sequence-dependent* (cuja duração depende da soma dos tempos de processamento das tarefas anteriores) e *learning effect* (redução no tempo de processamento devido à melhora contínua no desempenho da produção).

2.2 Regras de prioridade para programação da produção

Regras de prioridade, também conhecidas como regras de sequenciamento ou regras de despacho, são procedimentos de extrema importância prática. São tecnicamente simples, fáceis de compreender e requerem pouco esforço para serem aplicadas. Geralmente a utilização de regras de prioridade é suficiente para a programação em diversos ambientes de produção, incluindo *flow shop* permutacional, como é o caso deste trabalho. Além disso, tais regras são fáceis de codificar em linguagens de programação modernas e seus cálculos são bastante rápidos (FUCHIGAMI, MOCCELLIN, RUIZ; 2015). Korytkowski (2013) afirmou que essas regras são amplamente aceitas em indústrias de computação, redes de telecomunicações e processos de produção, justamente pela sua fácil implementação, desempenho satisfatório, baixa exigência computacional e flexibilidade para incorporar conhecimento e expertise.

Há muitas décadas as regras de prioridade têm sido foco dos trabalhos de programação da produção, recebendo uma considerável atenção como importante técnica de solução para os problemas. Alguns resultados clássicos para o problema de máquina única são bem conhecidos por fornecer a solução ótima, como a regra SPT (*Shortest Processing Time*) para minimização do tempo total (ou médio) de fluxo, a EDD (*Earliest Due Date*) para minimização do atraso máximo e a MST (*Minimum Slack Time*) para a minimização do adiantamento máximo. Estes e outros conceitos básicos podem ser facilmente encontrados em obras de referência como Pinedo (2016), Baker e Tristsch (2009) e outros.

Jayamohan e Rajendran (2000) também salientaram as vantagens das regras de prioridade, afirmando que são mais populares em muitos sistemas reais de manufatura do que as heurísticas, especialmente por serem de simples implementação e utilização no ambiente produtivo.

S*urveys* de regras de prioridade para o problema de programação em *job shop* foram feitos por Blackstone, Phillips e Hogg (1982) e Haupt (1989), propondo classificação, caracterização e avaliação de regras elementares.

Barman (1998) analisou o impacto de 64 combinações de regras de prioridade no problema de *flow shop* com três máquinas e minimização do *lateness* (desvio de pontualidade) e do atraso das tarefas. Brah e Wheeler (1998) empreenderam um

estudo simulado do efeito de regras no tempo médio de fluxo e no *makespan* em um *flow shop* híbrido usando análise de regressão.

Novas regras foram propostas por Jayamohan e Rajendran (2000) para várias medidas de fluxo e de atraso para *flow shop* e *job shop* dinâmicos, ou seja, com chegadas de tarefas durante o período de programação. Chiang e Fu (2007) realizaram um estudo computacional de dezoito regras para *job shop* com medidas de atraso.

O problema de programação em *flow shop* dinâmico também foi estudado por El-Bouri, Balakrishnan e Popplewell (2008), Kianfar, Fatemi-Ghomi e Karimi (2009), El-Bouri (2012) e Kianfar, Fatemi-Ghomi e Oroojlooy-Jadid (2012), que propuseram as denominadas “regras cooperativas”, que estabelecem a prioridade em tempo real.

Esta revisão se restringiu a publicações que utilizaram *regras de prioridade propriamente como métodos de solução*. Evidentemente, existem muitos outros trabalhos que *incluem regras de prioridade nos métodos*, como um dos passos do algoritmo, por exemplo, para definir a solução inicial em heurísticas construtivas, meta-heurísticas e matheurísticas, como é o caso de Fuchigami e Moccellin (2016), Fuchigami (2014) e Ta, Billaut e Bouquard (2015), respectivamente.

Além disso, algumas pesquisas propuseram métodos de seleção de regras de prioridade, como é o caso de Zhang, Jiang e Guo (2009), que apresentaram uma integração entre simulação e a metodologia de superfície de resposta para verificar a melhor regra de prioridade em uma fábrica de pastilhas semicondutoras. Balasundaram, Baskar e Sankar (2012) introduziram uma metodologia para gerar regras de prioridade para *flow shop* usando a técnica *data mining*. E um método simulação evolucionária para seleção de regras de prioridade foi proposto por Korytkowski, Wisniewski e Rymaszewski (2013), obtendo soluções próximas da ótima em tempo de computação viável.

3 FORMULAÇÃO MATEMÁTICA E NOVAS REGRAS DE PRIORIDADE

Os dois problemas tratados nesta pesquisa podem ser formulados por meio de programação linear inteira mista. Para ambos os problemas, as restrições são idênticas, mudando apenas a função-objetivo, conforme descrito logo a seguir. Foi considerada a formulação baseada em variáveis de posição, que é conhecida como a

melhor para os diversos problemas de programação da produção, conforme observam Framinan e Perez-Gonzalez (2017).

Considerando a notação apresentada anteriormente, sejam as seguintes variáveis de decisão: X_{jh} é igual a 1 se a tarefa J_j é programada na posição h da sequência e 0 caso contrário; C_{hk} é o instante de término da tarefa programada na posição h da sequência na máquina M_k.

$$Min\, Z_1 = C_{nm} \tag{3.1}$$

$$Min\, Z_2 = \frac{\sum_{h=1}^{n} C_{hm}}{n} \tag{3.2}$$

sujeito a

$$\sum_{h=1}^{n} X_{jh} = 1, \qquad j = 1,...,n, \tag{3.3}$$

$$\sum_{j=1}^{n} X_{jh} = 1, \qquad h = 1,...,n, \tag{3.4}$$

$$C_{11} \geq \sum_{j=1}^{n} \left(s_{j1} + p_{j1}\right) X_{j1}, \tag{3.5}$$

$$C_{11} \geq \sum_{j=1}^{n} \left(r_{j} + p_{j1}\right) X_{j1}, \tag{3.6}$$

$$C_{h1} \geq C_{(h-1)1} + \sum_{j=1}^{n} \left(s_{j1} + p_{j1}\right) X_{j1}, \qquad h = 2,...,n, \tag{3.7}$$

$$C_{h1} \geq C_{(h-1)1} + \sum_{j=1}^{n} \left(r_{j} + p_{j1}\right) X_{j1}, \qquad h = 2,...,n, \tag{3.8}$$

$$C_{1k} \geq C_{1(k-1)} + \sum_{j=1}^{n} p_{jk} X_{jk}, \qquad k = 2,...,m, \tag{3.9}$$

$$C_{1k} \geq \sum_{j=1}^{n} \left(s_{jk} + p_{jk}\right) X_{jk}, \qquad k = 2,...,m, \tag{3.10}$$

$$C_{hk} \geq C_{(h-1)k} + \sum_{j=1}^{n} \left(s_{jk} + p_{jk}\right) X_{jk}, \qquad h = 2,...,n, k = 2,...,m, \tag{3.11}$$

$$C_{hk} \geq C_{h(k-1)} + \sum_{j=1}^{n} p_{jk} X_{jk}, \qquad h = 2,...,n, k = 2,...,m, \tag{3.12}$$

$$C_{hk} \geq 0, \qquad h = 1,...,n, k = 1,...,m, \tag{3.13}$$

$$X_{hk} \in \{0,1\}, \qquad h = 1,...,n, k = 1,...,m. \tag{3.14}$$

A expressão (3.1), ou seja, a função-objetivo Z_1 refere-se à minimização do *makespan*, enquanto a Z_2 na (3.2), à minimização do tempo médio de fluxo. As restrições (3.3) e (3.4) derivam do problema de designação clássico, garantindo

respectivamente que cada tarefa seja programada em uma única posição e que a cada posição seja associada uma única tarefa. As expressões (3.5) e (3.6) asseguram conjuntamente que o instante de término da primeira tarefa da sequência na primeira máquina (C_{11}) seja a maior dentre duas situações: a soma do tempo de *setup* e de processamento (restrição 3.5) e a soma da data de liberação da tarefa e o tempo de processamento (restrição 3.6).

Os conjuntos de restrições (3.7) e (3.8) garantem a consistência do instante de término da segunda tarefa em diante ainda na primeira máquina (C_{h1}), sendo maior do que o instante de término da tarefa anterior ($C_{(h-1)1}$) somado ao maior entre a soma do tempo de *setup* e do processamento (expressão 3.7) ou soma da liberação e o tempo de processamento (expressão 3.8). Já as restrições (3.9) e (3.10) asseveram a consistência do instante de término da primeira tarefa na segunda máquina em diante (C_{1k}, $k = 2, ..., m$), sendo maior do que duas situações: o instante de término da tarefa na máquina anterior ($C_{1(k-1)}$) somado ao tempo de processamento da tarefa (expressão 3.9) ou a soma do tempo de *setup* e do tempo de processamento da tarefa (expressão 3.10).

As restrições (3.11) e (3.12) são expressões genéricas dos instantes de término das tarefas (C_{hk}), respeitando as relações descritas anteriormente, ou seja, que precisam ser maiores do que o instante de término da tarefa anterior na mesma máquina ($C_{(h-1)k}$) somado aos tempos de *setup* e de processamento (conforme a expressão 3.11) ou o instante de término da operação da mesma tarefa na máquina anterior ($C_{h(k-1)}$) somado ao tempo de processamento (conforme a expressão 3.12). As expressões (3.13) e (3.14) definem o domínio das variáveis de decisão.

A partir das características do problema em estudo, foram desenvolvidas oito novas regras de prioridade. Os critérios de ordenação de cada regra são apresentados na Tabela 1.

A expressão $max\{r_j, s_{j1}\}$, utilizada no critério de ordenação de várias regras, representa o instante mais cedo para o início do processamento da tarefa J_j, quando tanto a atividade do *setup* já foi executada como o instante de liberação da tarefa foi respeitado. Esta expressão já desconsidera a parte antecipada do tempo de *setup*, de forma que este intervalo não seja somado duas vezes.

Tabela 1 – Critérios de ordenação das regras de prioridade propostas

Regra	Ordenação	Critério
R_1	Crescente	r_j
R_2	Crescente	$\max\{r_j, s_{j1}\} + p_{j1}$
R_3	Crescente	$s_{j1} + p_{j1}$
R_4	Crescente	$\max\{r_j, s_{j1}\} + \sum_{k=2}^{m} s_{jk}$
R_5	Crescente	$\max\{r_j, s_{j1}\} + \sum_{k=1}^{m} p_{jk}$
R_6	Crescente	$\max\{r_j, s_{j1}\} + \sum_{k=2}^{m} s_{jk} + \sum_{k=1}^{m} p_{jk}$
R_7	Decrescente	$s_{jm} + p_{jm}$
R_8	--	Aleatória

O raciocínio utilizado na elaboração de cada regra é explicado a seguir:

R1: ordenação em fila, ou seja, pela ordem crescente das datas de liberação das tarefas; é a conhecida disciplina FIFO (*First In First Out*) ou PEPS (Primeira que Entra, Primeira que Sai).

R2: ordenação crescente pela data de término da primeira operação de cada tarefa, ou seja, a soma do instante mais cedo de início da tarefa com o tempo de processamento da operação na primeira máquina ($pj1$).

R3: ordenação crescente pela soma dos tempos de processamento e de *setup* de cada tarefa na primeira máquina.

R4: ordenação crescente pela soma do instante mais cedo de início de cada tarefa e os seus tempos de *setup* da segunda máquina em diante.

R5: ordenação crescente pela soma do instante mais cedo de início de cada tarefa e os seus tempos de processamento em todas as máquinas.

R6: ordenação crescente pela soma do instante mais cedo de início de cada tarefa, os seus tempos de processamento em todas as máquinas e os seus tempos de *setup* da segunda máquina em diante.

R7: ordenação decrescente pela soma dos tempos de processamento e de *setup* das tarefas na última máquina; esta ordenação decrescente visa programar a

tarefa com a menor soma dos tempos de processamento e de *setup* na última posição da sequência.

R8: ordenação aleatória, para fins de comparação.

Estes critérios servem para estabelecer a sequência de tarefas que será executada nas máquinas do *flow shop*. Embora em algumas regras, o critério de ordenação não considere algumas variáveis do problema, como a data de liberação ou o tempo de *setup*, ao se programar as tarefas nas máquinas, evidentemente foram respeitadas todas as restrições do problema.

A ordenação crescente se refere à ordem “não decrescente”, pois pode haver empates no valor do critério de ordenação de duas ou mais tarefas. Da mesma forma, a ordenação decrescente indica a ordem “não crescente”. Os desempates são feitos pela menor soma dos tempos de processamento em todas as máquinas e, caso o empate persista, o desempate ocorre pela menor soma dos tempos de *setup* em todas as máquinas.

A ordenação crescente prevalecente, principalmente pela presença das diferentes datas de liberação, visa o processamento mais cedo possível das tarefas e deriva da regra SPT (*Shortest Processing Time*) que, como é bem conhecido, fornece boas soluções para o problema de minimização do tempo de fluxo. A utilização de ordenação decrescente das datas de liberação é inviável para a minimização do *makespan* e do tempo de fluxo, e é impraticável na maioria das situações reais.

4 EXPERIMENTAÇÃO COMPUTACIONAL E RESULTADOS

4.1 Metodologia da pesquisa

Segundo as definições de Martins (2010, p.45-49) e Nakano (2010, p.64), esta pesquisa possui *abordagem quantitativa*, pois há preocupação com mensurabilidade, causalidade, generalização e replicação. Pode também ser classificada como *experimento*, uma vez que testa o relacionamento entre as variáveis da pesquisa operacionalizada, manipulando variáveis independentes (neste caso, a programação do problema) para se observar o resultado na variável dependente (representada aqui pelas medidas de desempenho *makespan* e *flowtime*).

Além disso, de acordo com Jung (2004), este trabalho se classifica como *pesquisa aplicada* quanto à natureza, por gerar conhecimento com finalidades de aplicação prática, *pesquisa exploratória* quanto aos objetivos, pois visa melhoria teórico-prática de sistemas, processos e produtos, e inovação pela proposição de novos modelos, além de ser feita a partir de impulsos criativos, simulações e experimentações, podendo originar novos modelos destinados a invenções, inovações e a otimização, e *pesquisa experimental* quanto aos procedimentos, para a obtenção de novos conhecimentos e produtos tecnológicos, requerendo uma manipulação de variáveis detalhada e sistemática, e originando inovações a partir de ensaios e estudos dinâmicos em laboratório.

4.2 Planejamento do experimento

Na experimentação computacional, foram testados e avaliados 39.600 exemplares (ou problemas-teste), definidos pelo número de tarefas, número de máquinas, intervalos de datas de liberação das tarefas e intervalos de tempos de *setup*. Os exemplares foram divididos em dois grupos, sendo o primeiro (Grupo 1) com os de pequeno porte, tendo sido encontrada a solução ótima por meio de enumeração completa, e o segundo (Grupo 2) com exemplares de médio e grande porte. É importante salientar que a principal característica que difere o Grupo 1 do Grupo 2 é a comparação com a solução ótima e não o número de tarefas e de máquinas.

No Grupo 1, foram gerados 18.000 exemplares, cujos parâmetros foram: 5, 6, 7, 8 e 10 tarefas e 2, 3, 5 e 10 máquinas. E no Grupo 2, foram gerados 21.600 exemplares, com os seguintes parâmetros: 15, 20, 30, 50, 80 e 100 tarefas e 5, 10, 15 e 20 máquinas. Em ambos os grupos, todos os tempos foram gerados com valores inteiros com distribuição uniforme: três intervalos de datas de liberação, U[1, 49], U[1,99] e U[1,199], três intervalos de tempos de *setup*, U[1, 49], U[1, 99] e U[1, 149], e tempos de processamento em U[1, 99]. Para cada combinação de parâmetros, foram gerados 100 exemplares visando reduzir o erro amostral.

Os resultados obtidos na experimentação computacional foram analisados por meio de Desvio Relativo Percentual – RPD, em inglês (*relative percentage deviation*). O desvio relativo mede a variação correspondente à melhor solução obtida pelos métodos. Se o desvio relativo da solução de um método é igual a zero para um

determinado exemplar, significa que tal método forneceu o menor valor da função objetivo, ou seja, ele apresentou a melhor programação.

Para cada exemplar, o RPD de determinado método no Grupo 1 foi assim calculado:

$$RPD_{G1} = \frac{Z - Z^*}{Z^*}, \quad (4.1)$$

onde Z é o valor da função objetivo do método avaliado e o Z* é o valor da solução ótima do exemplar.

Enquanto para o Grupo 2, o cálculo foi elaborado da seguinte forma:

$$RPD_{G2} = \frac{Z - Z^b}{Z^b}, \quad (4.2)$$

onde Z é o valor da função objetivo do método avaliado e o Z^b a melhor solução obtida para o exemplar.

Na implementação computacional, a configuração do computador utilizado são as seguintes: processador Pentium Dual Core da Intel com 2 GHz de frequência e 3 GB de memória RAM. Utilizou-se o sistema operacional Windows e o ambiente de programação Delphi.

4.3 Análise dos resultados do problema de minimização do *makespan*

Na análise global dos resultados do problema de minimização do *makespan*, a regra R2 obteve os melhores resultados, com 8,2% de desvio em relação à solução ótima, no Grupo 1, e 2,2% em relação à melhor solução do Grupo 2. A R2 faz a ordenação crescente pela soma do instante mais cedo de início de cada tarefa com o tempo de processamento da operação na primeira máquina, obteve os melhores resultados. Os gráficos das Figuras 1 e 2 apresentam o RPD médio com intervalos com 95% de confiança.

Como pode ser visto na Figura 1, após a regra R2, veio a R7, que utiliza a ordenação decrescente pela soma dos tempos de processamento e de *setup* das tarefas na última máquina, com 9,4% de desvio no Grupo 1 e 2,7% no Grupo 2, e a regra R3, que ordena de forma crescente pela soma dos tempos de processamento e de *setup* de cada tarefa na primeira máquina, com 10,3% de desvio no Grupo 1 e empatou com 2,7% no Grupo 2.

Figura 1 – Comparação do desempenho global (RPD) das regras – Grupo 1

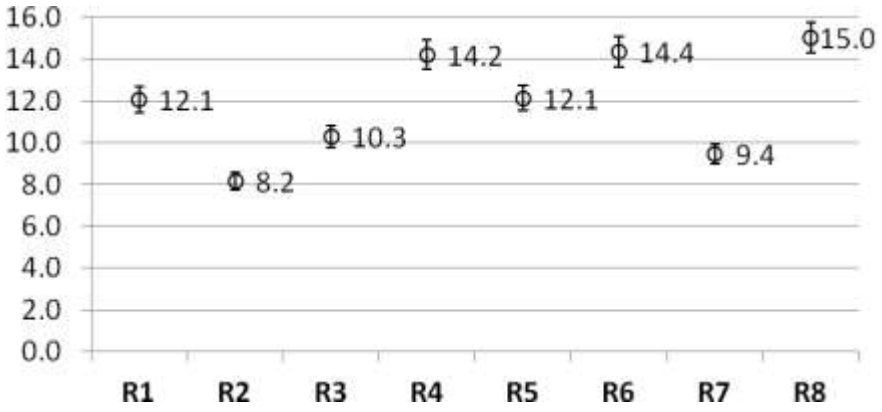
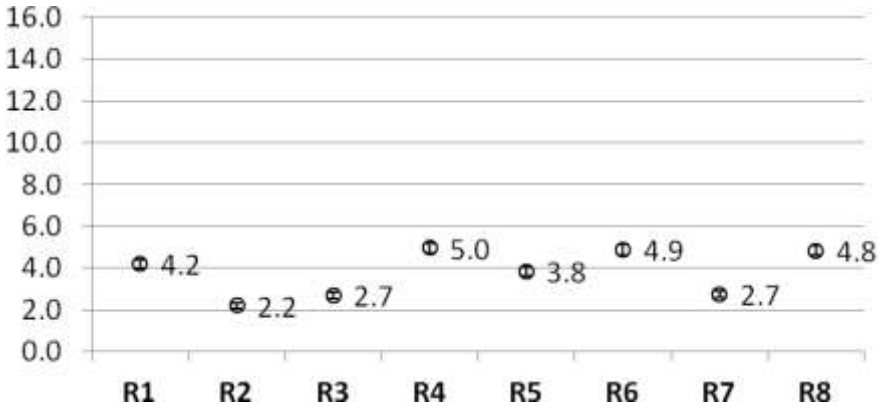


Figura 2 – Comparação do desempenho global (RPD) das regras – Grupo 2



A regra de ordenação aleatória R8 ficou em último lugar no Grupo 1 (Figura 1), com 15% de desvio, o que já era esperado, e em penúltima posição no Grupo 2 (Figura 2), com 4,8% de desvio, bastante próximo dos valores da regra R6, com 4,9%, e da R4, com 5,0%. A regra R6 faz a ordenação crescente pela soma dos tempos de processamento em todas as máquinas e os tempos de *setup* da segunda máquina em diante, e a R4 utiliza a ordenação crescente pela soma dos tempos de *setup* da segunda máquina em diante.

Nota-se ainda nas Figuras 1 e 2 que a dispersão dos desvios das regras não foi significativa, ficando na faixa dos 8,2% a 15,0% no Grupo 1 e entre 2,2% a 5,0% no Grupo 2. É natural que os desvios no Grupo 1 sejam maiores que no 2 por causa da comparação com a solução ótima.

A Tabela 2 a seguir apresenta o número de soluções ótimas encontradas por cada uma das heurísticas e o percentual equivalente, além do total de soluções ótimas encontradas conjuntamente pelas dez heurísticas sem considerar os empates.

Tabela 2 – Soluções ótimas obtidas por cada regra – Grupo 1

	R1	R2	R3	R4	R5	R6	R7	R8	Total
núm. soluções ótimas	661	1634	1129	318	736	357	1555	347	**3450**
% de soluções ótimas	3.7	9.1	6.3	1.8	4.1	2.0	8.6	1.9	**19.2**

É bastante notório que as regras de prioridade, procedimentos de solução classificados como os de menor complexidade e consequentemente de custo computacional insignificante (como será discorrido à frente), sejam capazes de fornecer a solução ótima em quase 20% da totalidade dos problemas resolvidos no Grupo 1 (conforme a Tabela 2). A melhor regra R2 sozinha atingiu 9,1% de soluções ótimas.

Para maior detalhamento da análise dos resultados, as Tabelas 3 e 4, a seguir apresentam os valores do RPD médio para cada opção do porte do problema, por número de tarefas (*n*) e de máquinas (*m*), respectivamente, para o Grupo 1 e o Grupo 2. Os melhores valores estão destacados em azul e os piores em laranja.

Tabela 3 – Desempenho (RPD) por porte do problema – Grupo 1

n	*m*	R1	R2	R3	R4	R5	R6	R7	R8
5	2	8.7	5.4	9.3	11.9	9.2	12.0	6.4	13.7
5	3	11.5	7.0	10.4	14.0	11.7	14.7	9.2	15.3
5	5	11.8	8.1	10.2	14.4	12.7	14.7	9.7	15.1
5	10	10.6	8.2	9.6	12.3	11.3	12.5	9.5	12.8
6	2	8.5	5.1	8.6	11.1	8.9	11.6	6.1	13.3
6	3	12.5	7.8	10.6	14.7	12.3	15.2	8.9	16.1
6	5	13.4	9.5	11.4	15.7	13.6	15.6	10.8	16.6
6	10	12.2	9.4	10.3	13.6	12.9	13.7	11.0	14.6
7	2	8.5	5.0	8.3	11.3	8.4	11.1	5.3	11.5
7	3	12.4	7.8	9.9	14.6	12.2	14.6	8.8	15.4
7	5	14.8	10.2	11.7	16.9	14.9	17.2	11.9	17.0
7	10	13.3	10.2	11.4	14.8	13.7	15.0	12.2	15.6
8	2	7.7	4.1	7.0	10.5	7.7	10.7	4.6	11.5
8	3	12.7	7.4	9.7	14.8	12.1	14.8	9.0	15.7
8	5	15.5	11.0	12.4	17.1	15.2	17.3	12.3	17.8
8	10	14.5	11.4	12.4	15.9	15.1	16.3	13.0	16.4
10	2	7.7	3.6	6.3	10.6	6.8	10.1	4.0	10.6
10	3	12.2	7.2	9.0	14.6	11.4	14.5	8.4	15.0
10	5	16.1	11.6	12.8	17.7	16.1	18.2	13.1	18.5
10	10	16.5	13.3	14.3	17.7	16.4	17.6	14.7	17.9

As Tabelas 3 e 4 confirmam a superioridade da regra R2 para o problema de minimização do *makespan*, curo melhor desempenho predomina em todas as opções de porte do problema em ambos os grupos.

Em relação ao pior resultado, no Grupo 1, houve ainda a predominância da regra aleatória R8, com exceção de duas classes de problemas, como pode ser visto na Tabela 3. Já no Grupo 2, o maior número de classes com o pior resultado ocorreu com a regra R4 (16 classes), em seguida com a regra aleatória R8 (11 classes) e por fim com a R6 (4 classes), conforme a Tabela 4.

Tabela 4 – Desempenho (RPD) por porte do problema – Grupo 2

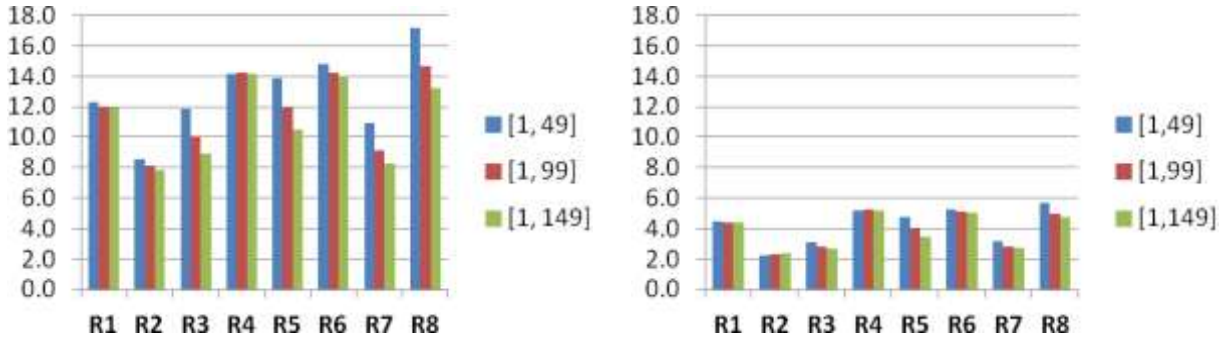
n	*m*	R1	R2	R3	R4	R5	R6	R7	R8
15	**5**	7.0	3.1	3.9	8.3	6.5	8.3	3.9	8.4
15	**10**	5.9	3.2	3.7	6.6	6.0	7.0	4.3	7.4
15	**15**	5.3	2.9	3.6	5.8	5.1	5.8	3.7	6.1
15	**20**	4.8	2.8	3.4	5.4	4.9	5.5	3.7	5.5
20	**5**	6.6	2.7	3.5	7.9	5.7	7.7	3.3	7.6
20	**10**	5.4	2.8	3.6	6.3	5.3	6.4	3.7	6.5
20	**15**	4.7	2.7	3.3	5.3	4.5	5.3	3.7	5.7
20	**20**	4.2	2.6	3.0	5.0	4.2	5.0	3.3	4.9
30	**5**	5.5	2.2	2.9	6.9	4.6	6.6	2.7	6.4
30	**10**	4.6	2.5	3.2	5.6	4.3	5.4	3.1	5.6
30	**15**	4.2	2.4	2.9	4.7	4.0	4.7	2.9	4.8
30	**20**	3.7	2.2	2.9	4.4	3.7	4.2	2.8	4.4
50	**5**	4.7	1.9	2.3	5.8	3.6	5.6	2.2	5.2
50	**10**	4.0	2.1	2.7	4.4	3.4	4.4	2.6	4.4
50	**15**	3.4	2.0	2.3	4.0	3.1	3.8	2.5	3.7
50	**20**	3.2	1.9	2.3	3.6	3.0	3.5	2.3	3.5
80	**5**	3.8	1.5	1.9	4.9	2.8	4.6	1.9	4.2
80	**10**	3.3	1.7	2.0	3.9	2.9	3.9	2.1	3.7
80	**15**	2.8	1.7	2.0	3.4	2.6	3.3	1.9	3.1
80	**20**	2.5	1.6	1.9	2.8	2.4	2.8	1.8	2.8
100	**5**	3.4	1.7	1.8	4.6	2.5	4.3	1.7	3.6
100	**10**	2.9	1.6	1.9	3.5	2.5	3.4	1.9	3.2
100	**15**	2.5	1.5	1.8	3.1	2.3	3.0	1.9	2.9
100	**20**	2.4	1.5	1.6	2.7	2.3	2.7	1.8	2.6

Foram feitas também análises dos resultados para as variáveis do problema considerado – tempos de *setup* e datas de liberação. O desempenho por grupo para cada regra e opções de intervalo de tempos de *setup* pode ser observado na Tabela 5 e nos gráficos da Figura 3.

Tabela 5 – Desempenho (RPD) por opção de intervalo de tempos de *setup*

	Intervalo	R1	R2	R3	R4	R5	R6	R7	R8
Grupo 1	[1,49]	12.3	8.6	11.9	14.2	13.9	14.8	10.9	17.2
	[1,99]	12.0	8.1	10.1	14.3	11.9	14.2	9.1	14.7
	[1,149]	11.9	7.8	8.9	14.1	10.5	14.0	8.3	13.2
Grupo 2	[1,49]	4.5	2.3	3.1	5.2	4.8	5.2	3.1	5.6
	[1,99]	4.4	2.3	2.8	5.3	4.0	5.1	2.8	5.0
	[1,149]	4.4	2.4	2.7	5.2	3.4	5.1	2.8	4.8

Figura 3 – Desempenho (RPD) por opção de intervalo de tempos de *setup*



Grupo 1 **Grupo 2**

Como pode ser observado na Figura 3, da mesma forma que os resultados globais, na análise por intervalos de *setup* os valores do RPD das regras no Grupo 1 foram maiores do que no Grupo 1 devido à comparação com a solução ótima no primeiro e a comparação relativa entre as regras no segundo. Entretanto, percebe-se que o comportamento das curvas foi proporcionalmente bastante similar.

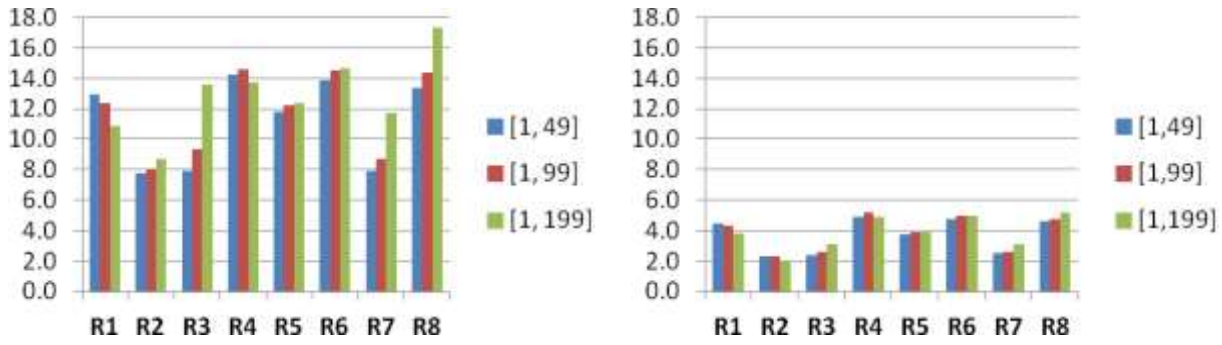
As regras R1, R2, R4 e R6 tiveram uma variação muito suave com a mudança do intervalo de *setup*, enquanto as regras R3, R5, R7 e R8 mostram um claro decrescimento nos desvios com o aumento dos intervalos de *setup*. Assim, as regras que consideram em seu critério a soma dos tempos de *setup* em todos os estágios, R4 e R6, tiveram menos influência da variação dos intervalos de *setup* considerados do que as regras que consideram a soma dos tempos de processamento em todos os estágios (R5) ou a soma do *setup* e processamento no primeiro (R3) ou no último estágio (R7).

A seguir, na Tabela 6 e nos gráficos da Figura 4, são apresentados os resultados por grupo e para cada opção de intervalo de datas de liberação considerado.

Tabela 6 – Desempenho (RPD) por opção de intervalo de datas de liberação

	Intervalo	R1	R2	R3	R4	R5	R6	R7	R8
Grupo 1	[1,49]	12.9	7.7	7.9	14.2	11.8	13.8	7.9	13.4
	[1,99]	12.4	8.0	9.3	14.6	12.2	14.6	8.7	14.4
	[1,199]	10.9	8.7	13.6	13.8	12.4	14.7	11.7	17.3
Grupo 2	[1,49]	4.5	2.3	2.4	4.9	3.7	4.7	2.5	4.6
	[1,99]	4.3	2.3	2.6	5.1	3.9	4.9	2.6	4.8
	[1,199]	3.8	2.0	3.1	4.9	3.9	5.0	3.1	5.2

Figura 4 – Desempenho (RPD) por opção de intervalo de datas de liberação



Grupo 1 **Grupo 2**

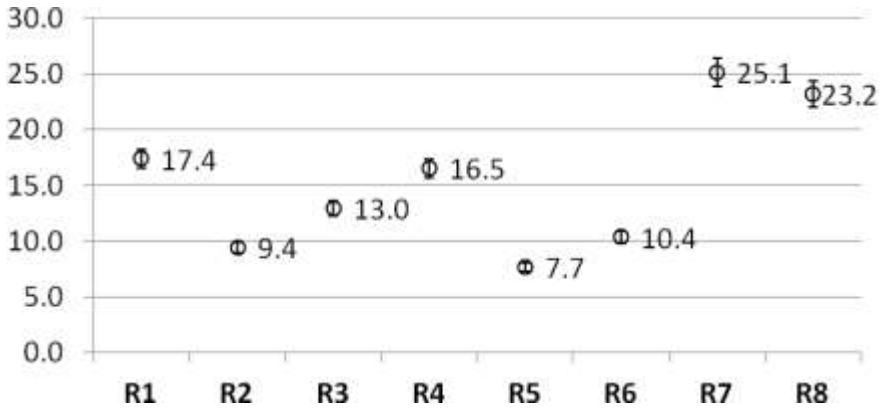
Na análise dos intervalos de datas de liberação, conforme a Figura 4, os valores dos RPD das regras no Grupo 1 foram também superiores aos do Grupo 2 (pela comparação com a solução ótima o primeiro). O comportamento das curvas foi relativamente bem parecido nos dois grupos. O destaque está nas regras R3, R7 e na aleatória R8, que tiveram desvios bastante superiores com o intervalo de datas de liberação em [1,199]. As regras R3 e R7 não consideram os tempos de processamento e *setup* em todos os estágios mas, respectivamente, apenas no primeiro e no último.

4.4 Análise dos resultados do problema de minimização do *flowtime*

As regras de prioridade desta pesquisa foram concebidas de forma que pudessem ser aplicadas e avaliadas nos dois problemas tratados (de *makespan* e *flowtime*). Assim, as mesmas regras foram testadas nos dois problemas considerados. Como será apresentado logo adiante, é interessante notar a diferença dos resultados das mesmas regras quando aplicados em problemas diferentes.

Os gráficos das Figuras 5 e 6 mostram a comparação de desempenho global das oito regras aplicadas ao problema de minimização do *flowtime*, respectivamente, nos Grupos 1 e 2. Nos dois gráficos, que apresentam os RPD médios com intervalos com 95% de confiança, nota-se que o melhor critério de sequenciamento para o problema de *flowtime* é a regra R5, que faz a ordenação crescente pela soma do instante mais cedo de início de cada tarefa e os tempos de processamento em todas as máquinas. Esta regra teve desvio de 7,7% no Grupo 1 e apenas 1,2% no Grupo 2.

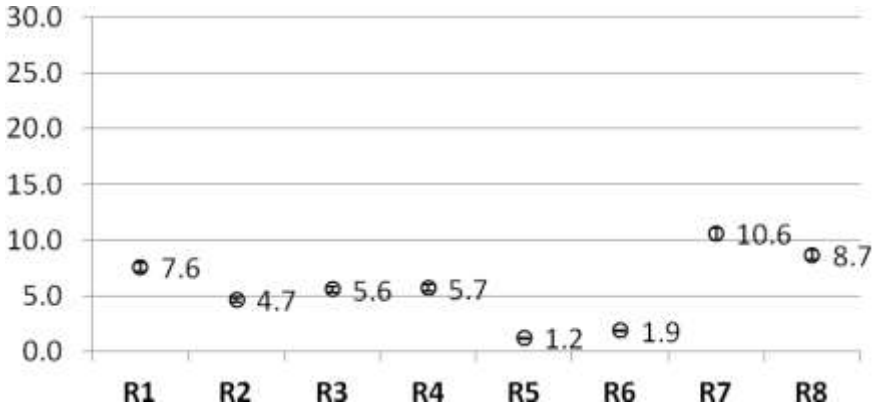
Figura 5 – Comparação do desempenho global (RPD) das regras – Grupo 1



Na Figura 5 é possível ainda observar que no Grupo 1 a regra R2 ficou em segundo lugar na ordem de melhor desempenho, com 9,4% de desvio, e a R6 em terceiro, com 10,4%. Já no Grupo 2, pela Figura 6, houve a inversão: a regra R6 ficou em segundo lugar, com 1,9% de desvio, e a regra R2 em terceiro, com 4,7%. A regra R2 faz a ordenação crescente pela soma do instante mais cedo de início com o tempo de processamento da tarefa na primeira máquina, enquanto a regra R6 sequencia pela ordem crescente da soma do instante mais cedo de início, os tempos de processamento em todas as máquinas e os tempos de *setup* da segunda máquina em diante.

Como pode ser observado na Figura 6, no problema de *flowtime*, a regra R7 foi claramente a de pior de desempenho, com 25,1% de desvio no Grupo 1 e 10,6% no Grupo 2. Ela foi inferior até à regra aleatória R8, que obteve 23,2% de desvio no Grupo 1 e 8,7% no Grupo 2. A regra R7 utiliza como critério a ordenação decrescente da soma dos tempos de processamento e de *setup* na última máquina.

Figura 6 – Comparação do desempenho global (RPD) das regras – Grupo 2



Da mesma forma que no problema de *makespan*, os desvios das regras em ambos os grupos ficaram numa baixa relativamente pequena, compreendida entre 7,7% a 25,1% no Grupo 1 e 1,2% a 10,6% no Grupo 2.

A Tabela 7 a seguir apresenta o número de soluções ótimas encontradas por cada uma das heurísticas e o percentual equivalente, além do total de soluções ótimas encontradas conjuntamente pelas dez heurísticas sem considerar os empates.

Tabela 7 – Soluções ótimas obtidas por cada regra

	R1	R2	R3	R4	R5	R6	R7	R8	Total
núm. soluções ótimas	65	489	512	93	642	395	20	41	**1626**
% de soluções ótimas	0.4	2.7	2.8	0.5	3.6	2.2	0.1	0.2	**9.0**

Como pode ser visto na Tabela 7, no problema de *flowtime* as regras forneceram a solução ótima de 1.626 exemplares, o que equivalente a 9,0% do total resolvido no Grupo 1, um resultado bastante relevante em se tratando de métodos de solução heurística com a simplicidade das regras de prioridade, conforme já salientado. A melhor regra R5 obteve a solução ótima em 3,6% dos exemplares.

A análise detalhada por porte do problema pode ser verificada por meio das Tabelas 8 e 9 a seguir. Os melhores valores estão destacados em azul e os piores em laranja.

Tabela 8 – Desempenho (RPD) por porte do problema – Grupo 1

n	*m*	R1	R2	R3	R4	R5	R6	R7	R8
5	**2**	17.3	6.5	13.4	16.0	6.6	9.4	31.9	28.0
5	**3**	15.6	7.1	12.5	16.1	6.4	10.3	25.8	23.9
5	**5**	13.2	7.5	11.0	14.7	5.7	9.0	19.3	18.9
5	**10**	10.6	7.1	9.3	11.5	4.2	6.7	13.8	14.0
6	**2**	19.1	7.7	13.0	16.9	7.2	9.8	33.9	29.2
6	**3**	18.2	8.6	12.7	16.4	7.6	10.6	26.6	25.1
6	**5**	15.4	9.2	12.3	15.9	7.0	9.8	20.4	20.8
6	**10**	12.3	8.5	10.4	13.0	5.5	7.7	15.2	15.8
7	**2**	20.9	9.0	13.5	18.3	7.9	10.4	34.9	28.0
7	**3**	19.1	9.4	13.3	17.4	8.5	11.3	28.0	25.9
7	**5**	16.4	10.0	13.0	16.9	7.9	11.0	21.9	20.8
7	**10**	13.6	9.4	11.4	14.1	5.9	8.6	16.4	16.7
8	**2**	21.5	9.8	13.9	18.3	9.0	11.5	34.9	29.2
8	**3**	20.6	10.6	14.6	18.6	9.5	11.9	29.3	26.8
8	**5**	18.2	11.2	13.7	17.7	8.7	11.6	23.3	22.5
8	**10**	13.9	9.9	11.9	14.7	7.0	9.4	17.1	17.2
10	**2**	24.0	11.0	15.0	20.1	9.7	12.1	37.3	30.6
10	**3**	21.8	11.8	15.3	19.5	10.3	12.9	30.0	27.5
10	**5**	19.7	12.8	15.3	18.5	10.5	12.9	24.4	24.2
10	**10**	16.2	11.8	13.5	16.1	8.5	10.7	18.3	18.5

A regra mais eficaz na análise global, R5, obteve também os melhores resultados na maioria das classes por porte do problema, em ambos os grupos. As poucas exceções podem ser observadas nas Tabelas 8 e 9, em que a regra R2 obteve 0,1 ponto percentual de desvio a menos no Grupo 1 em exemplares com 5 tarefas e 2 máquinas e no Grupo 2 a regra R6 empatou em duas classes e foi levemente superior em outras três.

Em relação à pior regra, embora a maioria das classes mostre que a regra R7 ainda obteve os maiores desvios, houve algum revezamento com a regra aleatória R8, principalmente nos exemplares do Grupo 1.

A Tabela 10 e a Figura 7 apresentam os resultados de cada regra por opção de intervalo de tempos de *setup*.

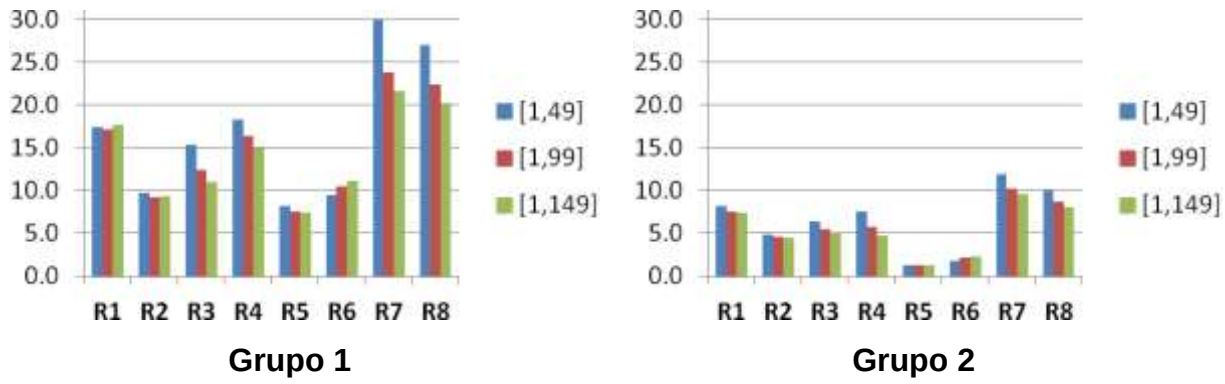
Tabela 9 – Desempenho (RPD) por porte do problema – Grupo 2

n	*m*	R1	R2	R3	R4	R5	R6	R7	R8
15	**5**	11.2	5.1	6.8	9.3	2.3	4.0	15.2	14.1
15	**10**	8.6	4.9	6.2	7.9	1.8	3.3	10.9	11.0
15	**15**	7.7	4.5	5.7	7.2	1.3	2.9	9.0	9.1
15	**20**	7.1	4.6	5.6	6.9	1.1	2.7	8.3	8.3
20	**5**	11.3	5.1	6.6	8.5	1.9	3.3	15.3	13.1
20	**10**	8.4	4.7	6.3	7.1	1.6	3.0	10.3	10.1
20	**15**	7.3	4.6	5.7	6.7	1.1	2.6	8.8	8.7
20	**20**	6.8	4.4	5.5	6.4	1.0	2.6	8.0	8.0
30	**5**	10.2	5.2	6.4	7.2	1.5	2.3	15.4	12.0
30	**10**	7.8	4.7	5.9	6.4	1.3	2.2	9.9	9.2
30	**15**	6.7	4.1	5.2	5.5	1.1	2.0	7.8	7.7
30	**20**	5.9	3.8	4.8	5.6	0.9	1.9	6.9	7.0
50	**5**	9.8	5.7	6.5	6.1	1.3	1.5	16.4	10.9
50	**10**	7.3	4.7	5.7	4.8	1.1	1.3	9.7	8.1
50	**15**	6.0	4.1	4.7	4.5	0.9	1.4	7.4	6.7
50	**20**	5.5	3.8	4.4	4.3	0.7	1.3	6.3	6.1
80	**5**	9.6	6.3	7.0	5.4	1.4	0.9	18.4	10.4
80	**10**	6.8	4.7	5.2	4.1	1.0	1.0	10.6	7.6
80	**15**	5.7	4.0	4.6	3.8	0.8	1.0	7.5	6.2
80	**20**	5.0	3.6	4.1	3.4	0.8	0.9	6.2	5.5
100	**5**	9.6	6.8	7.2	5.2	1.4	0.8	19.7	10.2
100	**10**	6.8	5.0	5.4	3.9	1.2	0.8	11.5	7.5
100	**15**	5.6	4.1	4.5	3.5	0.9	0.8	8.1	6.0
100	**20**	4.8	3.7	4.0	3.1	0.8	0.8	6.2	5.2

Tabela 10 – Desempenho (RPD) por opção de intervalo de tempos de *setup*

	Intervalo	R1	R2	R3	R4	R5	R6	R7	R8
Grupo 1	**[1,49]**	17.4	9.8	15.4	18.3	8.1	9.5	29.9	27.0
	[1,99]	17.1	9.3	12.4	16.3	7.5	10.5	23.8	22.3
	[1,149]	17.7	9.3	11.0	15.0	7.4	11.1	21.6	20.2
Grupo 2	**[1,49]**	8.2	4.9	6.4	7.5	1.3	1.7	11.9	10.2
	[1,99]	7.6	4.6	5.5	5.8	1.2	2.1	10.2	8.6
	[1,149]	7.4	4.5	5.0	4.7	1.3	2.4	9.6	8.1

Figura 7 – Desempenho (RPD) por opção de intervalo de tempos de *setup*



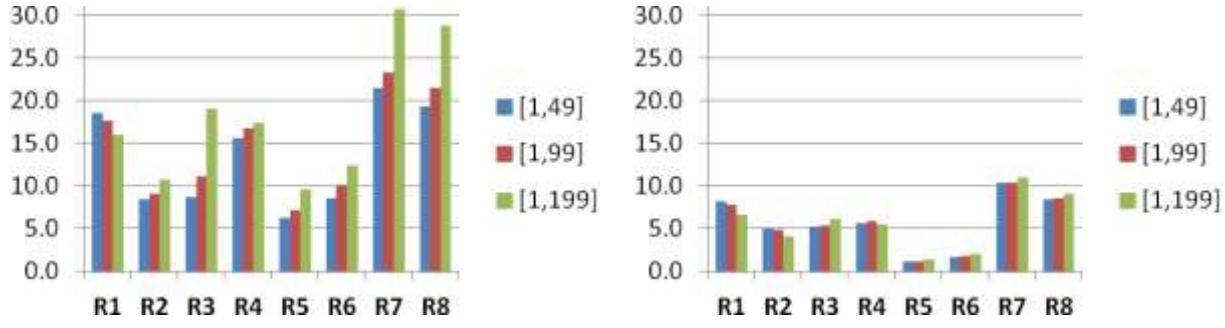
Conforme a Figura 7, respeitadas as proporções dos desvios nos Grupos 1 e 2, os comportamentos das curvas das regras em ambos os grupos foi bastante parecido. Na maioria dos casos, os desvios de cada regra são reduzidos com o aumento do intervalo dos tempos de *setup*, exceto a regra R6 (nos dois grupos) em que aumentam os desvios com a ampliação dos intervalos, e as regras R1, R2 (no Grupo 1) e R5 (no Grupo 2), que mantém os desvios praticamente constantes.

A análise por opção dos intervalos de datas de liberação é apresentada por meio da Tabela 11 e da Figura 8.

Tabela 11 – Desempenho (RPD) por opção de intervalo de datas de liberação

	Intervalo	R1	R2	R3	R4	R5	R6	R7	R8
Grupo 1	[1,49]	18.6	8.4	8.7	15.6	6.3	8.6	21.5	19.3
	[1,99]	17.7	9.1	11.1	16.7	7.2	10.1	23.2	21.4
	[1,199]	15.9	10.8	19.0	17.3	9.5	12.4	30.7	28.7
Grupo 2	[1,49]	8.2	5.1	5.3	5.7	1.1	1.7	10.4	8.4
	[1,99]	7.8	4.8	5.4	5.9	1.2	1.8	10.4	8.5
	[1,199]	6.7	4.0	6.1	5.6	1.4	2.1	11.0	9.1

Figura 8 – Desempenho (RPD) por opção de intervalo de datas de liberação



Grupo 1 **Grupo 2**

Em relação aos intervalos das datas de liberação, em geral, percebe-se na Figura 8 que as regras tiveram um aumento nos desvios com a ampliação do intervalo, com exceção das regras R1 (ambos os grupos) e R2 (no Grupo 2), que reduziram os desvios com o aumento do intervalo de datas de liberação, e as regras R4 e R5 (no Grupo 2) que mantiveram os desvios praticamente constantes. Além disso, nota-se que os desvios das regras R3, R7 e a aleatória R8 foram bem mais elevados com o

intervalo de datas de liberação em [1,199], comportamento que também ocorreu no problema de minimização do *makespan*.

5 CONSIDERAÇÕES FINAIS

Este trabalho cumpriu com o seu objetivo de conceber, implementar computacionalmente e avaliar estatisticamente o desempenho de regras de prioridade para programação da produção de dois problemas de *flow shop* com *setup* e datas de liberação das tarefas: minimização do *makespan* e do *flowtime*. Foram desenvolvidas sete novas regras de prioridade, cuja eficácia (qualidade da solução) foi comparada com a da ordenação aleatória e com o método exato de enumeração completa.

No problema de minimização do *makespan*, os melhores resultados em todos os portes de exemplares foram obtidos pela regra R2, que faz a ordenação crescente pela soma do instante mais cedo de início de cada tarefa com o tempo de processamento da operação na primeira máquina. Já no caso da minimização do *flowtime*, o melhor desempenho foi atingido pela regra R5, cujo critério de ordenação é de forma crescente pela soma do instante mais cedo de início de cada tarefa e os tempos de processamento em todas as máquinas.

Em ambos os problemas, a faixa de dispersão dos desvios de todas as regras foi relativamente pequeno, demonstrando baixa variabilidade dos resultados, inclusive nas análises por opções de intervalos de tempos de *setup* e de datas de liberação.

Além disso, conjuntamente as regras forneceram a solução ótima em 19,2% dos exemplares de pequeno porte resolvidos para *makespan* e 9,0% dos exemplares para *flowtime*, o que é bastante significativo, dada a simplicidade das regras de prioridade como métodos de solução heurística.

Conforme a previsto, as regras exploraram a vantagem da sua simplicidade e rapidez de execução, e tiveram tempos de CPU irrelevantes, sendo praticamente zero em média e no máximo 16 milissegundos, mesmo para os exemplares de grande porte. Para o método de enumeração completa, a média foi de 1,6 segundos, tanto no caso do *makespan* como no do *flowtime*.

Esta ampla experimentação computacional comprovou a eficácia e aplicabilidade das regras de prioridade analisadas nos ambientes considerados, principalmente em problemas de grande porte. Tais regras podem também ser

incorporadas em novos métodos de solução mais estruturados, como meta-heurísticas, por exemplo, em pesquisas futuras.

AGRADECIMENTOS

A pesquisa relatada neste artigo teve o apoio do CNPq (processos 443464/2014-6 e 233654/2014-3) e da FAPEG em cooperação com a CAPES (processo 201510267000983).

REFERÊNCIAS

ABEDI, M.; SEIDGAR, H.; FAZLOLLAHTABAR, H.; BIJANI, R. Bi-objective optimisation for scheduling the identical parallel batch-processing machines with arbitrary job sizes, unequal job release times and capacity limits. **International Journal of Production Research**, v. 53, n. 6, p. 1680-1711, 2015. http://dx.doi.org/10.1080/00207543.2014.952795

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

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

ALLAHVERDI, A.; NG, C.T.; CHENG, T.C.E.; KOVALYOV, M.Y. A survey of scheduling problems with setup times or costs. **European Journal of Operational Research**, v. 187, p. 985-1032, 2008. https://doi.org/10.1016/j.ejor.2006.06.060

AMIRIAN, H.; SAHRAEIAN, R. A hybrid differential evolution for general multi-objective flow shop problem with a modified learning effect. **Journal of Engineering** Manufacturing, v. 230, n. 12, p. 2275-2285, 2016. http://dx.doi.org/10.1177/0954405416673094

BAI, D. Asymptotic analysis of online algorithms and improved scheme for the flow shop scheduling problem with release dates. **International Journal of Systems Science**, v. 46, n. 11, p. 1994-2005, 2015. http://dx.doi.org/10.1080/00207721.2013.843736

BAI, D.; HUO, M.; TANG, L. A new lower bound for flow shop makespan with release dates. **IEEE**, p. 276-280, 2008. https://doi.org/10.1109/SOLI.2008.4686405

BAKER, K.R.; TRIETSCH, D. **Principles of sequencing and scheduling**. New Jersey: John Wiley & Sons, 2009.

BALASUNDARAM, R.; BASKAR, N.; SANKAR, R.S. A new approach to generate dispatching rules for two machine flow shop scheduling using data mining. **Procedia Engineering**, v. 38, p. 238-245, 2012. https://doi.org/10.1016/j.proeng.2012.06.031

BARMAN, S. The impact of priority rule combinations on lateness and tardiness. **IIE Transactions**, v. 30, n. 5, p. 495-504, 1998. https://doi.org/10.1023/A:1007503508080

BIANCO, L.; DELL'OLMO, P.; GIORDANI, S. Flow shop no-wait scheduling with sequence dependent setup times and release dates. **INFOR**, v. 37, n. 9, p. 3-19, 1999. http://dx.doi.org/10.1080/03155986.1999.11732365

BLACKSTONE, J.H.; PHILLIPS, D.T.; HOGG, G.L. A state-of-the-art survey of dispatching rules for manufacturing job shop operations. **International Journal of Production Research**, v. 20, n. 1, p. 27-45, 1982. http://dx.doi.org/10.1080/00207548208947745

BRAH, S.A; WHEELER, G.E. Comparison of scheduling rules in a flow shop with multiple processors: a simulation. **Simulation**, v. 71, n. 5, p. 302-311, 1998. http://journals.sagepub.com/doi/abs/10.1177/003754979807100501

BÜLBÜL, K.; KAMINSKY, P.; YANO, C. Flow shop scheduling with earliness, tardiness, and intermediate inventory holding costs. **Naval Research Logistics**, v. 51, p. 407-445, 2004. http://dx.doi.org/10.1002/nav.20000

CHEN, H.; ZHOU, S.; LI, X.; XU, R. A hybrid differential evolution algorithm for a two-stage flow shop on batch processing machines with arbitrary release times and blocking. **International Journal of Production Research**, v. 52, n. 19, p. 5714-5734, 2014. http://dx.doi.org/10.1080/00207543.2014.910625

CHENG, J.; STEINER, G.; STEPHENSON, P. A computational study with a new algorithm for the three-machine permutation flow-shop problem with release times. **European Journal of Operational Research**, v. 130, p. 559-575, 2001. https://doi.org/10.1016/S0377-2217(99)00415-4

CHENG, T.C.E.; GUPTA, J.N.D.; WANG, G. A review of flowshop scheduling research with setup times. **Production and Operations Management**, v. 9, n. 3, p. 262-282, 2000. https://doi.org/10.1111/j.1937-5956.2000.tb00137.x

CHIANG, T.C.; FU, L.C. Using dispatching rules for job shop scheduling with due date-based objectives. **International Journal of Production Research**, v. 45, n. 14, p. 3245-3262, 2007. https://doi.org/10.1109/ROBOT.2006.1641909

CHIHAOUI, F.B.; KACEM, I.; HADJ-ALOUANE, A.B.; DRIDI, N.; REZG, N. No-wait scheduling of a two-machine flow-shop to minimize the makespan under non availability constraints and different release dates. **International Journal of Production Research**, v. 49, n. 21, p. 6273-6286, 2011. http://dx.doi.org/10.1080/00207543.2010.531775

EL-BOURI, A. A cooperative dispatching approach for minimizing mean tardiness in a dynamic flowshop. **Computers & Operations Research**, v. 39, p. 1305-1314, 2012. https://doi.org/10.1016/j.cor.2011.07.004

EL-BOURI, A.; BALAKRISHNAN, S.; POPPLEWELL, N. Cooperative dispatching for minimizing mean flowtime in a dynamic flowshop. **International Journal of Production Economics**, v. 113, p. 819-833, 2008. https://doi.org/10.1016/j.ijpe.2007.11.005

FRAMINAN, J.M.; GUPTA, J.N.D.; LEISTEN, R. A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. **Journal of the Operational Research Society**, v. 55, p. 1243-1255, 2004. https://link.springer.com/article/10.1057/palgrave.jors.2601784

FRAMINAN, J.M.; LEISTEN, R.; RUIZ-USANO, R. Comparison of heuristics for flowtime minimization in permutation flowshops. **Computers & Operations Research**, v. 32, p. 1237-1254, 2005. https://doi.org/10.1016/j.cor.2003.11.002

FUCHIGAMI, H.Y. Proposição de algoritmo *Simulated Annealing* para programação em *flow shops* paralelos proporcionais com tempos de *setup* explícitos. **Produção Online**, v. 14, n. 3, p. 997-1023, 2014. http://dx.doi.org/10.14488/1676-1901.v14i3.1631

FUCHIGAMI, H.Y.; MOCCELLIN, J.V. Desempenho relativo de regras de prioridade para programação de *flow shop* híbrido com tempos de *setup*. **Revista Produção Online**, v. 15, n. 4, p. 1174-1194, 2015. http://dx.doi.org/10.14488/1676-1901.v15i4.1791

FUCHIGAMI, H.Y.; MOCCELLIN, J.V. Efeitos de regras de prioridade para programação da produção em sistemas industriais complexos. **Produção Online**, v. 16, n. 1, p. 3-25, 2016. http://dx.doi.org/10.14488/1676-1901.v16i1.1720

FUCHIGAMI, H.Y.; MOCCELLIN, J.V.; RUIZ, R. Novas regras de prioridade para programação em flexible flow line com tempos de setup explícitos. **Production**, v. 25, n. 4, p. 779-790, 2015. http://dx.doi.org/10.1590/0103-6513.089212

FRAMINAN, J.M.; PEREZ-GONZALEZ, P. Order scheduling with tardiness objective: improved approximate solutions. **European Journal of Operational Research**, *in press*, 2017. http://doi.org/10.1016/j.ejor.2017.10.064

GRABOWSKI, J.; SKUBALSKA, E.; SMUTNICKI, C. On flow shop scheduling with release and due dates to minimize maximum lateness. **The Journal of the Operational Research Society**, v. 34, n. 7, p. 615-620, 1983. http://www.jstor.org/stable/2581775

GRAHAM, R.L.; LAWLER, E.L.; LENSTRA, I.K.; RINNOOY KAN, A.H.G. Optimization and approximation in deterministic machine scheduling: a survey. **Annals of Discrete Mathematics**, v. 5, p. 287-326, 1979. http://dx.doi.org/10.1016/S0167-5060(08)70356-X

GUPTA, J.N.D.; STAFFORD JR., E.F. Flowshop scheduling research after five decades. **European Journal of Operational Research**, v. 169, p. 699-711, 2006. https://doi.org/10.1016/j.ejor.2005.02.001

HALL, L. A polynomial approximation scheme for a constrained flow-shop scheduling problem. **Mathematics of Operations Research**, v. 19, n. 1, p. 68-85, 1994. https://doi.org/10.1287/moor.19.1.68

HALL, L.A. Approximability of flow shop scheduling. **Mathematical Programming**, v. 82, n. 1, p. 175-190, 1998. https://doi.org/10.1007/BF01585870

HAOUARI, M.; LADHARI, T. Minimizing maximum lateness in a flow shop subject to release dates. **Journal of the Operational Research Society**, v. 58, p. 62-72, 2007. http://dx.doi.org/10.1057/palgrave.jors.2602092

HAUPT, R. A survey of priority rule-based scheduling. **OR Spektrum**, v. 11, p. 3-16, 1989. https://doi.org/10.1007/BF01721162

JAYAMOHAN, M.; RAJENDRAN, C. New dispatching rules for shop scheduling: a step forward. **International Journal of Production Research**, 38:3, p. 563-586, 2000. http://dx.doi.org/10.1080/002075400189301

JÓZEFCZYK, J.; MARKOWSKI, M.; BALGABAEVA, L. Routing flow-shop with buffers and ready times – comparison of selected solution algorithms. **Management and Production Engineering Review**, v. 5, n. 4, p. 26-35, 2014. http://dx.doi.org/10.2478/mper-2014-0033

JUNG, C.F. **Metodologia para pesquisa & desenvolvimento**: aplicada a novas tecnologias, produtos e processos. Rio de Janeiro: Axcel Books, 2004.

KALCZYNSKI, P.J.; KAMBUROWSKI, J. An empirical analysis of heuristics for solving the two-machine flow shop problem with job release dates. **Computers & Operations Research**, v. 39, p. 2659-2665, 2012. https://doi.org/10.1016/j.cor.2012.01.011

KAMINSKY, P.; SIMCHI-LEVI, D. Asymptotic analysis of an on-line algorithm for the single machine completion time problem with release dates. **Operations Research Letters**, v. 29, p. 141-148, 2001. http://dx.doi.org/10.1080/00207721.2013.843736

KASHYRSKIKH, K.N.; POTTS, C.N.; SEVASTIANOV, S.V. A 3/2-approximation algorithm for two-machine flow-shop sequencing subject to release dates. **Discrete Applied Mathematics**, v. 114, p. 255-271, 2001. https://doi.org/10.1016/S0166-218X(00)00374-7

KIANFAR, K.; FATEMI GHOMI, S.M.T.; KARIMI, B. New dispatching rules to minimize rejection and tardiness costs in a dynamic flexible flow shop. **The International Journal of Advanced Manufacturing Technology**, v. 45, p. 759-771, 2009. https://doi.org/10.1007/s00170-009-2015-x

KIANFAR, K.; FATEMI GHOMI, S.M.T.; OROOJLOOY JADID, A. Study of stochastic sequence-dependent flexible flow shop via developing a dispatching rule and a hybrid GA. **Engineering Applications of Artificial Intelligence**, v. 25, p. 494-506, 2012. https://doi.org/10.1016/j.engappai.2011.12.004

KORYTKOWSKI, P.; WISNIEWSKI, T.; RYMASZEWSKI, S. An evolutionary simulation-based optimization approach for dispatching scheduling. **Simulation Modelling Practice and Theory**, v. 35, p. 69-85, 2013. http://dx.doi.org/10.1080/00207540110064938

KURZ, M.E.; ASKIN, R.G. Heuristic scheduling of parallel machines with sequence-dependent set-up times. **International Journal of Production Research**, v. 39, n. 16, p. 3747-3769, 2001. http://dx.doi.org/10.1080/00207540110064938

LADHARI, T.; HAOUARI, M. A branch-and-bound algorithm for the permutation flow shop scheduling problem subject to release dates and delivery times. **IEEE**, p. 1-5, 2006. https://doi.org/10.1109/ICSSSM.2006.320673

LENSTRA, J.K.; RINNOOY KAN, A.H.G.; BRUCKER, P. Complexity of machine scheduling problems. **Annals of Operations Research**, v. 1, p. 343-362, 1977. https://doi.org/10.1016/S0167-5060(08)70743-X

LIU, H.; QUEYRANNE, M.; SIMCHI-LEVI, D. On the asymptotic optimality of algorithms for the flow shop problem with release dates. **Naval Research Logistics**, v. 52, p. 232-242, 2005. https://doi.org/10.1002/nav.20066

MA, T.; CHU, C.; ZUO, C. A survey of scheduling with deterministic machine availability constraints. **Computers & Industrial Engineering**, v. 58, p. 199-211, 2010. https://doi.org/10.1016/j.cie.2009.04.014

MacCARTHY, B.L.; LIU, J. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. **International Journal of Production Research**, v. 31, n. 1, p. 59-79, 1993. http://dx.doi.org/10.1080/00207549308956713

MARTINS, R.A. Abordagens Quantitativa e Qualitativa. In: MIGUEL, P.A.C.(Org.). **Metodologia de pesquisa em Engenharia de Produção e Gestão de Operações**. Rio de Janeiro: Elsevier, 2010. p. 45-61, cap. 3.

MUTHUSWAMY, S.; VÉLEZ-GALLEGO, M.; MAYA, J.; ROJAS-SANTIAGO, M. Minimizing makespan in a two-machine no-wait flow shop with batch processing machines. **The International Journal of Advanced Manufacturing Technology**, p. 1-10, 2012. http://dx.doi.org/10.1007/s00170-012-3906-9

NAKANO, D. Métodos de Pesquisa Adotados na Engenharia de Produção e Gestão de Operações. In: MIGUEL, P.A.C.(Org.). **Metodologia de pesquisa em Engenharia de Produção e Gestão de Operações**. Rio de Janeiro: Elsevier, 2010. p. 63-72, cap. 4.

PAN, Q.-K.; RUIZ, R. A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. **Computers & Operations Research**, v. 40, p. 117-128, 2013. https://doi.org/10.1016/j.cor.2012.05.018

PINEDO, M.L. **Scheduling**: theory, algorithms, and systems. New Jersey: Prentice-Hall, 5ªed., 2016.

POTTS, C.N.; STRUSEVICH, V.A. Fifty years of scheduling: a survey of milestones. **Journal of the Operational Research Society**, v. 60, p. S41-S68, 2009. http://www.jstor.org/stable/40206725

REN, T.; DIAO, Y.; LUO, X. Optimal results and numerical simulations for flow shop scheduling problem. **Journal of Applied Mathematics**, v. 2012, p. 1-9, 2012. http://dx.doi.org/10.1155/2012/395947

REN, T.; GUO, M.; LIN, L.; MIAO, Y. A local search algorithm for the flow shop scheduling problem with release dates. **Discrete Dynamics in Nature and Society**, v. 2015, p. 1-8, 2015. http://dx.doi.org/10.1155/2015/320140

REN, T.; ZHANG, C.; LIN, L.; GUO, M.; XIE, X. Asymptotic analysis of SPTA-based algorithm for no-wait flow shop scheduling problem with release dates. **The Scientific World Journal**, v. 2014, p. 1-7, 2014. http://dx.doi.org/10.1155/2014/979238

REZA HEJAZI, S.; SAGHAFIAN, S. Flowshop-scheduling problems with makespan criterion: a review. **International Journal of Production Research**, v. 43, n. 14, p. 2895-2929, 2005. http://dx.doi.org/10.1080/0020754050056417

RUIZ, R.; MAROTO, C. A comprehensive review and evaluation of permutation flowshop heuristics. **European Journal of Operational Research**, v. 165, p. 479-494, 2005. https://doi.org/10.1016/j.ejor.2004.04.017

SUNG, C.S.; KIM, Y.H. Minimizing makespan in a two-machine flowshop with dynamic arrivals allowed. **Computers & Operations Research**, v. 29, p. 275-294, 2002. https://doi.org/10.1016/S0305-0548(00)00071-X

T'KINDT, V.; BILLAUT, J.-C. **Multicriteria scheduling:** theory, models and algorithms. Berlin Heidelberg: Springer-Verlag, 2006. 2nd ed.

TA, Q.C.; BILLAUT, J.-C.; BOUQUARD, J.-L. Matheuristic algorithms for minimizing total tardiness in the *m*-machine flow shop scheduling problem. **Journal of Intelligent Manufacturing**, p. 1-12, 2015. http://dx.doi.org/10.1007/s10845-015-1046-4

TADEI, R.; GUPTA, J.N.D.; DELLA-CROCE, F.; CORTESI, M. Minimising makespan in the two-machine flow-shop with release times. **Journal of the Operational Research Society**, v. 49, p. 77-85, 1998. https://doi.org/10.2307/3010655

TANG, L.; LIU, P. Minimizing makespan in a two-machine flowshop scheduling with batching and release time. **Mathematical and Computer Modelling**, v. 49, p. 1071-1077, 2009. https://doi.org/10.1016/j.mcm.2008.09.012

TYAGI, N.; VARSHNEY, R.G.; CHANDRAMOULI, A.B. Six decades of flowshop scheduling research. **International Journal of Scientific & Engineering Research**, v. 4, n. 9, p. 854-864, 2013.

ZHOU, Z.; LI, X.; CHEN, H.; GUO, C. Minimizing makespan in a no-wait flowshop with two batch processing machines using estimation of distribution algorithm. **International Journal of Production Research**, p. 1-19, 2016. http://dx.doi.org/10.1080/00207543.2016.1140920

Artigo recebido em 01/03/2017 e aceito para publicação em 10/10/2017
DOI: http://dx.doi.org/10.14488/1676-1901.v18i1.2747