Gest. Prod., São Carlos, v. 17, n. 4, p. 683-692, 2010

Regras de despacho para a minimização do atraso total no ambiente *flowshop* flexível

Dispatching rules for the total tardiness minimization in flexible flowshops

Guilherme Barroso Mainieri[1]
Débora Pretti Ronconi[1]

Resumo: Embora existam muitos trabalhos que tratam do problema de programação de tarefas no ambiente *flowshop* flexível com o objetivo de minimizar o *makespan*, poucos abordam este problema envolvendo datas de entrega. Com o aumento do nível de exigência dos clientes, pesquisas que buscam o atendimento das datas de entrega têm se tornado de extrema importância em ambientes de manufatura. Este trabalho analisa o problema de minimização do atraso total no ambiente *flowshop* flexível. Novas regras de despacho (também conhecidas como regras de liberação), baseadas nas regras MDD e PRTT, são propostas e avaliadas em um grupo de 4.320 problemas teste. A segunda regra tem como característica principal considerar estados futuros do sistema. Comparações com outras regras conhecidas na literatura mostram que estas superam as melhores regras conhecidas para o problema considerado.

Palavras-chave: *Flowshop* flexível. Atraso. Heurística.

Abstract: *There are a number of articles on the flexible flowshop scheduling problem with the objective of minimizing makespan available in the literature. However, only a few articles have addressed this problem involving due dates. Therefore, research involving due dates in production environments has become extremely important due to the increasing competition and rising demand. This paper analyses the total tardiness minimization in flexible flowshops. Two new dispatching rules, based on the well known MDD rule and PRTT, are proposed and evaluated in 4.320 sets of instances. The second rule considers not only the jobs available at each scheduling moment, but also all jobs that have not been scheduled yet. Comparative tests with well known dispatching rules show that those rules outperformed the best rules known for the problem considered.*

Keywords: *Flexible flowshop. Tardiness. Heuristic.*

1 Introdução

Este trabalho trata o problema de minimização do atraso total no ambiente *flowshop* flexível. Neste ambiente existem *n* tarefas e *s* estágios em série, cada um com um certo número de máquinas. O tempo de processamento de uma tarefa depende apenas da tarefa e do estágio, ou seja, independe da máquina pertencente a um estágio. Todas as tarefas devem passar por todos os estágios na mesma ordem e cada tarefa deve ser processada por apenas uma máquina de cada estágio.

A Figura 1 mostra um exemplo de *flowshop* flexível com três estágios, três máquinas nos estágios 1 e 3 e duas máquinas no estágio 2. As setas entre os estágios representam as possíveis rotas das tarefas dentro do sistema.

Conforme Linn e Zhang (1999) e Quadt e Kuhn (2007), a maioria dos autores trata o problema de *flowshop* flexível com medidas de desempenho relacionadas à passagem das tarefas pelo sistema, tais como minimização do *makespan* ou do tempo médio de fluxo. Poucos trabalhos abordam este ambiente com medidas relacionadas com o atraso e, devido à sua complexidade, geralmente por meio de métodos heurísticos. Segundo Du e Leung (1990), o problema de minimização do atraso total para o caso particular de uma máquina é NP-*hard*.

Brah (1996) analisa o desempenho de dez regras de prioridade considerando atrasos médio e máximo. Este autor também examina os efeitos de características do problema, como, por exemplo, número de tarefas, número de estágios, número de máquinas em paralelo em cada estágio. O autor conclui que as regras *Earliest Due Date* (EDD) e *Modified Due Date* (MDD)

[1] Departamento de Engenharia de Produção, Escola Politécnica, Universidade de São Paulo – USP, Av. Prof. Almeida Prado, 128, Cidade Universitária, CEP 05508-900, São Paulo – SP, Brasil, E-mails: gmainieri@usp.br; dronconi@usp.br

Recebido em 18/4/2009 — Aceito em 18/8/2010

Suporte financeiro: CAPES, CNPq e FAPESP.

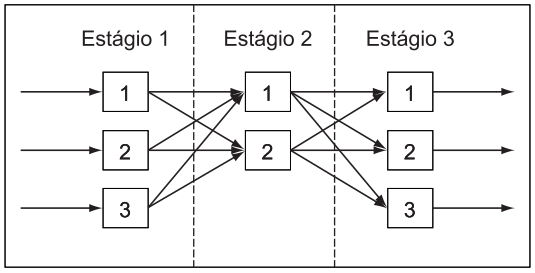

Figura 1. Exemplo do ambiente *flowshop* flexível.

apresentam os melhores desempenhos para máximo atraso e atraso médio, respectivamente.

Lee, Kim e Choi (2004) propõem uma heurística visando a minimização do atraso total com foco no estágio com maior carga de trabalho. O primeiro passo é identificar o estágio gargalo, em seguida o estágio identificado é programado como um problema de máquinas em paralelo por meio de uma heurística que é a modificação da heurística proposta por Koulamas (1994) conhecida como KPM. Em seguida, são determinadas as programações para os outros estágios, com base no resultado do gargalo. Os resultados são comparados com três métodos que utilizam como base oito regras de despacho: método de programação em ordem direta (do primeiro para o último estágio – *forward*), programação em ordem inversa (*backward*) e uma programação que mistura as duas primeiras (*multipass*). A heurística com foco no estágio gargalo apresenta os melhores resultados independentemente da regra de despacho considerada e, independentemente do modo de programação, as regras de despacho ATC (*Apparent Tardiness Cost*) e COVERT (*Cost Over Time*) apresentam os melhores resultados. Dentre as outras seis regras de despacho, que não dependem de parâmetros de ajuste, a regra MDD apresenta os melhores resultados para os quatro métodos de programação.

Os dois trabalhos apresentados acima consideram um ambiente de *flowshop* flexível com as mesmas características do ambiente considerado neste trabalho. Os próximos trabalhos também consideram o *flowshop* flexível, porém com algumas características diferentes das consideradas neste trabalho.

Choi, Kim e Lee (2005) abordam o *flowshop* flexível com lotes reentrantes no sistema já que algumas tarefas podem ser processadas mais de uma vez em um estágio. A medida de desempenho considerada é o atraso total e os autores propõem quatro heurísticas que utilizam o método de programação em ordem direta. A primeira heurística é uma adaptação da heurística de Nawaz, Enscore e Ham (1983), conhecida como NEH. Na adaptação de Choi, Kim e Lee (2005), o algoritmo utiliza o procedimento de inserção de NEH com os lotes inicialmente ordenados conforme a regra EDD. À cada iteração, a melhor posição na sequência parcial é determinada por meio da avaliação do atraso total das tarefas já alocadas. Ao final deste procedimento, é fornecida uma ordenação final das tarefas que é mantida para todo o sistema e é usada para a programação por meio do método da lista de prioridades. A segunda heurística é similar à primeira, no entanto, o método para avaliação de uma sequência parcial é diferente: à cada iteração do procedimento de inserção de NEH, quando uma sequência parcial é avaliada, os lotes da sequência parcial são programados, em seguida, os lotes que não estão na sequência parcial também são programados pelo método da lista de prioridades, com a prioridade definida de acordo com a regra EDD. Os dois algoritmos adicionais consideram pedidos e são adaptações das duas últimas heurísticas. Ambos consideram as datas de entrega dos lotes como a mesma data do pedido ao qual o lote pertence. Os resultados das quatro heurísticas são comparados com oito regras de despacho e apresentam melhor desempenho.

Yang, Kreipl e Pinedo (2000) consideram um *flowshop* flexível no qual as tarefas têm diferentes datas de liberação, datas de entrega, tempos de processamento e pesos, visando a minimização do atraso total ponderado. No artigo, são apresentadas três heurísticas. A primeira é uma heurística de decomposição que pode ser dividida em quatro partes (detecção de gargalos, formulação de subproblemas, solução dos subproblemas, reotimização). A segunda heurística é uma heurística de busca local. Cada iteração da busca local consiste em duas fases. A primeira fase modifica a carga de trabalho e gera novas sequências para as máquinas. Basicamente, isto é feito como no método usado na solução de subproblemas da primeira heurística. A segunda fase tenta procurar por melhores sequências para cada máquina em cada estágio por meio de perturbações na solução. Por último, é proposta uma heurística que é uma mistura das duas primeiras: a solução final da heurística de decomposição é usada como solução inicial para a heurística de busca local. O trabalho é concluído com uma análise comparativa das três heurísticas e, como esperado, a terceira heurística teve melhor desempenho que as duas primeiras, mostrando a importância de uma boa solução inicial para a segunda heurística.

O presente trabalho tem como objetivo analisar e propor heurísticas construtivas para o problema de minimização do atraso total no ambiente *flowshop* flexível. O pequeno esforço computacional desta estratégia, que pode ser valioso em aplicações práticas, é uma das razões que motivaram este estudo. Além disso, de acordo com Ronconi (2004) que também considera o ambiente *flowshop*, heurísticas construtivas geralmente são mais fáceis de implementar e não

exigem extensos testes computacionais para ajustá-las. As regras de despacho propostas consideram o *flowshop* flexível como uma série de problemas de máquinas em paralelo e baseiam-se nos métodos de programação em ordem direta. Uma dessas regras considera tarefas que ainda não estão disponíveis para a programação. Não encontramos, para o problema considerado, estudos que utilizam esta estratégia, ou seja, que incorporem uma visão do futuro. Para avaliarmos o desempenho dos métodos sugeridos, apresentamos uma comparação com outras regras conhecidas na literatura em problemas de média e grande dimensão.

A próxima seção contém a descrição do problema. A Seção 3 apresenta as heurísticas desenvolvidas para o problema em questão. Na Seção 4, são apresentados os resultados computacionais obtidos com os métodos propostos e a última seção resume os principais resultados deste trabalho.

2 Descrição do problema

O ambiente abordado neste trabalho se caracteriza pelo processamento de tarefas em linha, ou seja, n tarefas que são processadas em s estágios na mesma ordem. Cada estágio possui m_l máquinas e uma tarefa pode ser processada por qualquer máquina deste estágio. O tempo de processamento de uma tarefa depende apenas da tarefa e do estágio. Além disso, o número de tarefas e máquinas, os tempos de processamento e as datas de entrega de cada tarefa são previamente conhecidos.

Consideramos que todas as tarefas estão disponíveis para processamento no instante zero, os tempos de *setup* são independentes da sequência, as máquinas estão continuamente disponíveis e não são permitidas interrupções no processamento. A medida de desempenho a ser avaliada é o atraso total das tarefas.

Apresentamos a seguir a formulação matemática adaptada de Guinet et al. (1996) e Yang (1998).

Índices:

j, k: índice para tarefas (0,..., n + 1);
i: índice para máquina (1,..., m_l);
l: índice para estágio (1,..., s).

Parâmetros:

n: número de tarefas;
m_l: número de máquinas do estágio l;
s: número de estágios;
p_{jl}: tempo de processamento da tarefa j no estágio l;
d_j: data de entrega da tarefa j.

Variáveis:

x_{jkil}: assume o valor 1 se a tarefa k é designada após a tarefa j na máquina i do estágio l, caso contrário, assume valor zero;
C_{jl}: instante de conclusão da tarefa j ao término do processamento no estágio l;
T_j: atraso da tarefa j.

Formulação (Equações 1 a 11):

$$Minimizar \sum_{j=1}^{n} T_j \tag{1}$$

Sujeito a:

$$\sum_{i=1}^{m_l} \sum_{j=0}^{n} x_{jkil} = 1 \quad l = 1,...,s; \quad k = 1,...,n \tag{2}$$

$$\sum_{k=1}^{n+1} x_{jkil} \le 1 \quad l = 1,...,s; \quad j = 0,...,n; \quad i = 1,...,m_l \tag{3}$$

$$\sum_{k=1}^{n+1} x_{jkil} - \sum_{k=0}^{n} x_{kjil} = 0 \quad l = 1,...,s \quad j = 1,...,n \quad i = 1,...,m_l \tag{4}$$

$$T_j \ge C_{js} - d_j \quad j = 1,...,n \tag{5}$$

$$T_j \ge 0 \quad j = 1,...,n \tag{6}$$

$$C_{kl} - C_{jl} \ge p_{kl} \sum_{i=1}^{m_l} x_{jkil} + M\left(\sum_{i=1}^{m_l} x_{jkil} - 1 \right) \quad l = 1,...,s; \; j,k = 1,...,n; \; j \ne k \tag{7}$$

$$C_{j,l+1} - C_{jl} \ge p_{j,l+1} \quad j = 1,...,n; \quad l = 1,...,s-1 \tag{8}$$

$$C_{j1} \ge p_{j1} \quad j = 1,...,n \tag{9}$$

$$x_{jkil} \in \{0,1\} \quad \begin{matrix} j = 0,...,n; \quad k = 1,...,n+1 \\ i = 1,...,m_l; \quad l = 1,...,s \end{matrix} \tag{10}$$

$$x_{jjil} = 0 \quad j = 0,...,n; \quad i = 1,...,m_l; \quad l = 1,...,s \tag{11}$$

Considere que a variável x_{0kil} assume o valor 1 se a tarefa k está programada como a primeira tarefa na máquina i do estágio l, e $x_{j(n+1)il}$ assume o valor 1, se a tarefa j está programada como a última tarefa na máquina i do estágio l. Ou seja, as tarefas n+1 e zero são tarefas fictícias e representam, respectivamente, a tarefa posterior à última e a tarefa anterior à primeira. O instante de conclusão da tarefa j no sistema é igual ao instante de conclusão da tarefa j no último estágio. A constante M é escolhida como um escalar maior que o maior instante de término de qualquer solução factível.

A Equação 1 representa a minimização da soma dos atrasos das tarefas. O conjunto de restrições (2) garante que apenas uma tarefa *j* preceda uma tarefa *k* em alguma das máquinas do estágio *l*. As restrições (3) garantem que no máximo uma tarefa *k* suceda uma tarefa *j* em uma certa máquina *i* do estágio *l*. As restrições (4) garantem que cada tarefa tenha uma tarefa antecessora e uma sucessora na máquina *i* do estágio *l* na qual ela seja programada. Os conjuntos de restrições (5) e (6) garantem que, para cada tarefa *j*, o atraso seja $\max\{0, C_{js}\text{-}d_j\}$. As restrições (7) garantem que se uma tarefa *j* e outra *k* estão designadas para a mesma máquina de um certo estágio, então a tarefa *j* deve ser processada antes ou depois da tarefa *k*. Caso contrário, não existe restrição de precedência entre as duas tarefas. As restrições (8) garantem a precedência de uma tarefa entre estágios, ou seja, uma tarefa não pode ser processada no estágio $l + 1$ antes de ter sido processada no estágio *l*. As restrições (9) garantem que o instante de conclusão de uma tarefa no primeiro estágio seja maior ou igual ao seu tempo de processamento. As restrições (10) garantem que uma tarefa não seja precedida por ela mesma e as restrições (11) garantem que as variáveis sejam binárias.

3 Heurística proposta

Nesta seção, apresentamos novas regras de despacho propostas para minimização do atraso total no ambiente *flowshop* flexível. Estas regras utilizam um método bem conhecido na literatura para a programação de tarefas em ambientes produtivos: o método de programação em ordem direta (*forward list scheduling*). Em muitos ambientes de produção, especialmente em sistemas complexos como o considerado neste trabalho, este método de programação das operações é amplamente utilizado. Neste algoritmo, sempre que uma máquina de certo estágio torna-se disponível, seleciona-se a tarefa de maior prioridade que pode ser processada na máquina. Esta prioridade é determinada por uma regra de despacho. Existem diversas regras de despacho conhecidas na literatura para problemas de programação de tarefas. Por exemplo, as regras SPT (*Shortest Processing Time*) e EDD, são regras simples que apresentam bom desempenho para o problema considerado, veja Lee, Kim e Choi (2004).

Outra regra conhecida pelo seu bom desempenho para a minimização do atraso total no ambiente *flowshop* flexível é a regra MDD. Esta regra foi originalmente proposta por Baker e Bertrand (1982) para o problema de uma máquina e foi adaptada para o *flowshop* flexível da seguinte maneira: sendo *t* o tempo no qual a regra é aplicada, para uma tarefa *j* num certo estágio *l*, MDD é o máximo entre a data de entrega e a soma dos tempos de processamentos da tarefa em *s*-*l*+1 estágios restantes, ou seja, $max\left(d_j, t + \sum_{u=l}^{s} p_{ju}\right)$ Atribui-se maior prioridade para tarefas com menor MDD.

Estas três regras citadas são regras clássicas para a programação de tarefas. Em especial, a regra MDD é importante para este trabalho, pois ela não depende de parâmetros de ajuste e apresenta os melhores resultados para a minimização do atraso total no ambiente *flowshop* flexível, veja Brah (1996) e Lee, Kim e Choi (2004). Por isto as duas regras propostas neste trabalho são baseadas na regra MDD.

A primeira regra proposta é uma nova adaptação da regra MDD para o ambiente considerado e a denominamos por MDDa. Tendo em vista que o *flowshop* flexível pode ser visto como uma série de problemas de máquinas em paralelo, é promissor desenvolver uma regra que considere esta propriedade deste ambiente de máquinas. Para isso é necessário definir data de entrega para cada um dos problemas de máquinas em paralelo. Assim sendo, a data de entrega da tarefa *j* no estágio *l* (d_{jl}) é simplesmente definida como a data de entrega real menos o trabalho remanescente, ou seja (Equação 12):

$$d_{jl} = d_j - \sum_{u=l+1}^{s} p_{ju} \tag{12}$$

Por fim, define-se a regra MDDa como (Equação 13):

$$MDDa\left(j,l,t\right) = max\left(d_{jl},\ t + p_{jl}\right) \tag{13}$$

sendo *j* a tarefa considerada, *l* o estágio, *t* o instante de aplicação da regra.

A segunda regra proposta apresenta relação com as regras MDD, MDDa e PRTT. A regra PRTT (*Priority Rule for Total Tardiness*), proposta por Chu e Portmann (1992), foi originalmente proposta para minimizar o atraso total em um ambiente com uma única máquina e tarefas que possuem diferentes instantes de liberação. Na regra PRTT, as tarefas que estarão disponíveis em instantes posteriores ao atual também são consideradas. Dado que podemos analisar o ambiente *flowshop* flexível como uma série de problemas isolados de máquinas em paralelo, nos quais as tarefas ficam disponíveis em diferentes instantes de tempo, o estudo de funções de prioridades que consideram estados futuros do sistema é relevante. Assim sendo, a adaptação da regra PRTT também considera o *flowshop* flexível como uma série de máquinas em paralelo, assim como a regra MDDa. Portanto, ela utiliza a mesma definição de data de entrega da tarefa *j* no estágio *l* (*djl*). A regra PRTT adaptada para os problemas de máquinas em paralelo do *flowshop* flexível (PRTTa) foi definida como (Equação 14):

$$\begin{aligned} PRTTa\left(j,l,t\right) &= \alpha \cdot max(r_{jl}, t) + \\ &+ max\left[\ max\left(r_{jl}, t\right) + p_{jl},\ d_{jl}\ \right] \end{aligned} \tag{14}$$

Sendo j a tarefa considerada, l o estágio, t o instante de aplicação da regra, r_{jl} o instante de liberação das tarefas j no estágio l e α um parâmetro constante que pondera a prioridade do primeiro termo da regra.

O primeiro termo da soma, $\alpha \max(r_{jl},t)$, procura penalizar tarefas que não estão disponíveis no instante t ($r_{jl} > t$), ou seja, estas tarefas receberão um valor maior para este termo do que tarefas disponíveis para programação ($r_{jl} \leq t$). O segundo termo procura simultaneamente considerar o princípio de priorização da regra MDDa e penalizar as tarefas não disponíveis. Note que se $\max(r_{jl},t) = t$, o segundo termo torna-se idêntico à regra MDDa. Esta característica é valiosa para a resolução do problema considerado, visto que a regra MDD original apresenta bons resultados para a minimização do atraso total no ambiente *flowshop* flexível.

No método de programação, em todo instante t, no qual existe uma máquina disponível para processar uma tarefa, programa-se a tarefa disponível com maior prioridade para ser processada naquela máquina, sendo esta prioridade dada por meio da regra de despacho considerada. A regra proposta (PRTTa) exige uma adaptação no método de programação em ordem direta tradicional. Nesta regra, quando tarefas que não estão disponíveis no instante t são programadas, ocorre a inclusão de tempo ocioso. Um procedimento para o preenchimento deste tempo ocioso é proposto a seguir.

- Método de programação em ordem direta – PRTTa: em todo instante t, no qual existe uma máquina disponível, programe a tarefa j ainda não programada com menor PRTTa, desempate pelo menor instante de conclusão na máquina considerada. Caso a tarefa programada ainda não esteja disponível no instante t, preencha o tempo ocioso gerado com tarefas que possam ser programadas antes do início da tarefa j, sem alterar o instante de início de processamento da tarefa j. Se existirem tais tarefas, deve-se programar a tarefa com menor instante de liberação (r_{jl}), desempate pelo menor PRTTa, até que não existam mais tarefas que possam ser programadas no tempo ocioso.

A Figura 2 apresenta um exemplo de uma programação gerada pelo método proposto. Este exemplo possui quatro tarefas e dois estágios com duas máquinas no primeiro e uma máquina no segundo estágio. Os parâmetros para este exemplo são apresentados na Tabela 1.

O resultado da programação no estágio 1 é o mesmo resultado da programação em ordem direta original para a regra MDDa, visto que todas as tarefas estão disponíveis no instante inicial. O primeiro instante de decisão do estágio $l = 1$ é o instante $t = 0$. A Tabela 2 mostra os valores necessários para o cálculo da regra PRTTa para este instante, sendo $\alpha = 1$. Observe que, $r_{j1} = 0$ ($\forall j$), visto que todas as tarefas estão disponíveis no instante inicial. A última coluna apresenta o valor calculado para a regra PRTTa(j,l,t) por meio de (14), por exemplo. PRTTa (1,1,0) = 1.0 + Max [0 + 3,1] = 3.

Com isto, são escolhidas as duas tarefas com menor valor de PRTTa ($j = 1$ e $j = 2$) para serem programadas nas duas primeiras máquinas do estágio $l = 1$. Seguindo o mesmo procedimento, o algoritmo segue para o próximo instante de disponibilização de

Tabela 1. Parâmetros para o exemplo.

Tarefas j	$\mathbf{d_j}$	$\mathbf{p_{jl}}$	
		$l = 1$	$l = 2$
1	4	3	3
2	14	2	5
3	1	10	6
4	15	7	2

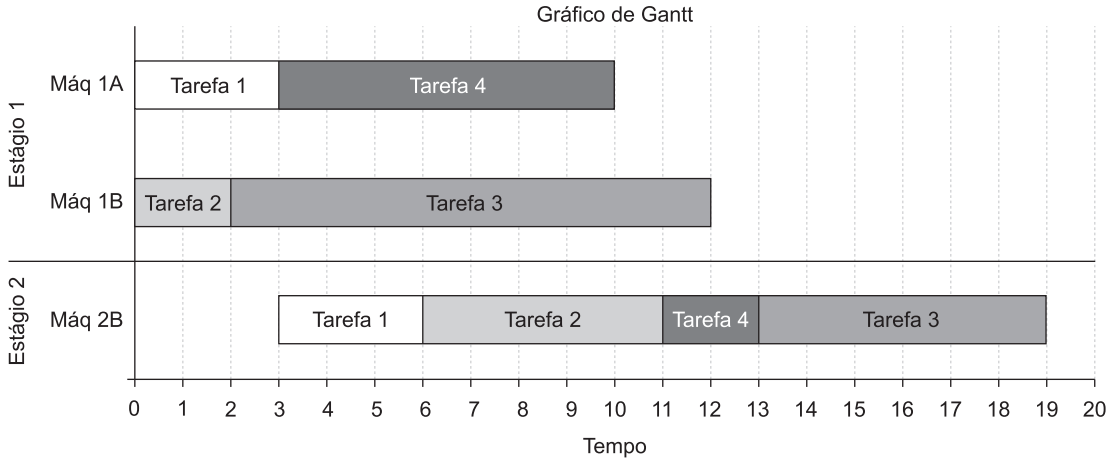

Figura 2. Programação gerada pelo método de programação em ordem direta – PRTTa.

máquinas no estágio ($t = 2$) até que todas as tarefas tenham sido programadas no estágio considerado.

A programação no segundo estágio começa no primeiro instante de disponibilização de uma tarefa no estágio, ou seja, $t = 2$. O instante $t = 2$ do estágio 2 ilustra a diferença entre os dois métodos de programação. Neste instante, apenas a tarefa 2 está disponível, no entanto, o método decidiu programar a tarefa 1 no instante de sua disponibilidade ($t = 3$). Esta programação foi realizada, pois a tarefa 1 apresenta um menor valor para a regra PRTTa no instante $t = 2$ do estágio 2 (vide Tabela 3). Com isso, foi inserido um tempo ocioso de 2 até 3, portanto deve-se procurar por tarefas que possam ser programadas antes da tarefa 1. A tarefa 2 é a única tarefa ainda não programada que apresenta disponibilidade durante este período de tempo, no entanto ela não é programada, pois seu tempo de processamento é maior que o tempo ocioso. Como não foram encontradas tarefas que possam preencher este período de tempo, o método de programação segue para o próximo instante de disponibilização de máquina no estágio ($t = 6$). As demais tarefas foram programadas seguindo o mesmo procedimento.

O atraso total da programação ilustrada na Figura 2 é igual a 20. Por meio da comparação com o valor ótimo da soma dos atrasos para este exemplo (obtida pela resolução da formulação apresentada nos softwares *Xpress* e *What's Best*) verificamos que a heurística proposta encontrou a solução ótima.

Este mesmo exemplo foi resolvido por meio de três regras de despacho: SPT, EDD, MDD e MDDa. O atraso total foi, respectivamente, 24, 22, 24, 23, resultados superiores ao valor encontrado pela heurística proposta, ilustrando que a inserção de tempo ocioso pode ser importante para a qualidade da solução encontrada. Conforme Kanet e Sridharan (2000), quando existem tarefas com diferentes instantes de liberação, como no problema considerado, a inserção de tempo ocioso na programação pode ser vantajosa.

É importante ressaltar que, pelo método de programação em ordem direta tradicional, não é possível chegar à programação da Figura 2, pois qualquer regra, por meio do método tradicional, programaria a tarefa $j = 2$ no instante $t = 2$ do estágio $l = 2$.

A próxima seção apresenta os testes preliminares e os resultados computacionais.

Tabela 2. Dado para o cálculo da regra PRTTa para $l = 1$ e $t = 0$.

Tarefa j	r_{j1}	max (r_{j1}, t)	p_{j1}	max(rj_1, t) + p_{j1}	d_{j1}	PRTTa $(j, 1, 0)$
1	0	0	3	3	1	3
2	0	0	2	2	9	9
3	0	0	10	10	–5	10
4	0	0	7	7	13	13

Tabela 3. Dados para o cálculo da regra PRTTa para $l = 2$ e $t = 2$.

Tarefa j	r_{j2}	max (r_{j2}, t)	p_{j2}	max(rj_2, t) + p_{j2}	d_{j2}	PRTTa $(j, 2, 0)$
1	3	3	3	6	4	9
2	2	2	5	7	14	16
3	12	12	6	18	1	30
4	10	10	2	12	15	25

4 Resultados computacionais

Os métodos propostos foram implementados em linguagem de programação C++ e os testes realizados em um computador com processador Athlon 64 X2 2,36 GHz com 2,5 Gb de memória RAM.

Para avaliação das heurísticas apresentadas, utilizamos o conjunto de problemas gerados em Lee, Kim e Choi (2004). Nestas instâncias o número de máquinas por estágio foi gerado de acordo com uma distribuição uniforme discreta (entre 1 e 10). Os tempos de processamento também foram gerados de acordo com uma distribuição uniforme, mas de duas maneiras diferentes: entre $1 \cdot m_l$ e $50 \cdot m_l$ para o estágio gargalo e entre $1 \cdot m_l \cdot L$ e $50 \cdot m_l \cdot L$ para os demais estágios, sendo L o nível de diferença da carga de trabalho entre o estágio gargalo e os demais estágios. As datas de entrega foram geradas por uma distribuição uniforme dentro do intervalo (Equação 15):

$$P\left(1 - T - \frac{R}{2}\right) \leq d_j \leq P\left(1 - T + \frac{R}{2}\right) \qquad (15)$$

No qual T e R são fatores de atraso e amplitude, respectivamente, e P é um limitante inferior do *makespan* proposto por Santos, Hunsucker e Deal(1995).

O conjunto de problemas é composto por 4.320 instâncias de teste, 10 problemas para cada combinação do número de tarefas (30, 50, 100), do número de estágios (5, 10, 30), da posição do estágio gargalo no sistema (1, 2, 3 e 4), da diferença da carga de trabalho entre o estágio gargalo e os demais estágios (0,5, 0,7 e 0,9), da amplitude da data de entrega (0,8 e 1,8) e do fator de atraso (0,1 e 0,5).

4.1 Seleção do parâmetro α

Primeiramente realizamos testes para selecionar um valor para o parâmetro α que pondera a importância do primeiro termo da regra PRTTa. Estes testes consistiram em variar α e registrar o resultado da média do atraso total das 4.320 instâncias para cada valor de α. O Gráfico 1 apresenta o resultado destes testes.

Por meio da análise do Gráfico 1, pode-se perceber que a regra apresenta melhor desempenho para $\alpha = 12$. No entanto é importante observar que resultados similares são obtidos para $\alpha \geq 5$. Este comportamento mostra a robustez da metodologia no que se refere à

escolha do parâmetro, ou seja, não são necessários grandes esforços para ajustá-lo.

4.2 Experimentos computacionais – MDDa

Devido às dificuldades de encontrar a solução ótima para os problemas, utilizamos o RDI (*Relative Deviation Index*) para medir o desempenho das diferentes regras consideradas. O RDI é calculado da seguinte maneira (Equação 16):

$$RDI = \frac{S_a - S_b}{S_w - S_b} \tag{16}$$

no qual S_a é a solução do método avaliado, S_b e S_w são, respectivamente, a melhor e pior soluções encontradas. Quanto menor o valor do RDI, melhor o desempenho do método avaliado.

Para avaliar o desempenho da primeira regra proposta (MDDa), utilizamos cinco regras de despacho conhecidas na literatura que utilizam o mesmo método de programação: SPT, EDD, MDD, *Apparent Tardiness Cost* (ATC) e *Cost Over Time* (COVERT). Brah (1996). Vale observar que as regras ATC e COVERT, implementadas por Lee, Kim e Choi (2004), necessitam de parâmetros de ajuste, no entanto apresentaram os melhores resultados para o método de programação em ordem direta. Vale ressaltar que ATC e COVERT são regras mais elaboradas, visto que elas dependem de dois parâmetros de ajuste.

A Tabela 4 apresenta uma comparação da regra MDDa com as regras de despacho consideradas. Os valores destacados em cinza indicam as melhores médias do RDI. Analisando os resultados em relação aos estágios/tarefas, a regra proposta (MDDa) apresenta os melhores resultados em 7 das 9 dimensões consideradas.

Além disso, podemos observar que a regra MDDa (RDI médio de 0,05) tem desempenho superior às demais regras (0,07 da ATC e 0,13 da COVERT). Ao mesmo tempo, apresenta a vantagem de possuir fórmula de cálculo relativamente mais simples do que as regras ATC e COVERT, conhecidas como as regras com melhores resultados para a minimização do atraso total no *flowshop* flexível. Além disso, estas regras apresentam a desvantagem de cada uma necessitar de dois parâmetros de difícil ajuste.

Comparando-se apenas as regras que não dependem de parâmetros de ajuste (SPT, EDD, MDD e MDDa), pode-se notar o desempenho bem superior da regra proposta, lembrando que a MDD é conhecida como a regra que não depende de parâmetros de ajuste e que apresenta os melhores resultados para o problema considerado. A superioridade também está presente no número de melhores soluções: 2(SPT), 215(EDD), 297(MDD), 2358(MDDa), 1269(ATC), 502 (COVERT).

4.3 Experimentos computacionais – PRTTa

Nesta seção, consideramos as quatro melhores regras analisadas no método de programação em ordem direta (MDD, MDDa, ATC e COVERT) e a regra PRTTa com seu método particular de programação em ordem direta. A Tabela 5 apresenta uma comparação das regras propostas com as melhores regras de despacho consideradas.

Pela análise dos resultados apresentados na Tabela 5, pode-se perceber que a regra PRTTa tem desempenho superior às demais. Apesar de a regra PRTTa apresentar um parâmetro de ajuste (α),

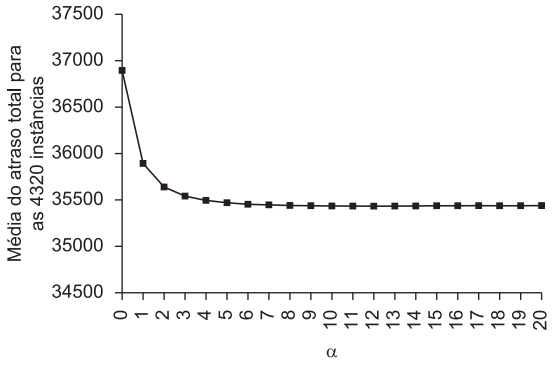

Gráfico 1. Comportamento do atraso total em função de α.

Tabela 4. Comparação entre as regras para o método em ordem direta.

Estágios	Tarefas	SPT	EDD	MDD	MDDa	ATC	COVERT
5	30	0,81	0,51	0,31	0,08	0,15	0,11
	50	0,78	0,55	0,20	0,08	0,13	0,07
	100	0,89	0,44	0,08	0,03	0,05	0,04
10	30	0,91	0,44	0,38	0,04	0,06	0,16
	50	0,77	0,59	0,31	0,07	0,10	0,13
	100	0,67	0,66	0,15	0,09	0,10	0,07
30	30	0,97	0,36	0,40	0,01	0,02	0,20
	50	0,95	0,47	0,38	0,01	0,02	0,20
	100	0,81	0,62	0,30	0,02	0,03	0,14
Média		0,84	0,52	0,28	0,05	0,07	0,13

este ajuste mostrou-se relativamente simples. A superioridade também aparece no número de melhores soluções: 351 (MDD), 759 (MDDa), 2607 (PRTTa), 520 (ATC), 468 (COVERT).

A Tabela 6 mostra que as regras propostas também são superiores em relação à média do atraso total. Para as 4.320 instâncias teste, a regra PRTTa teve uma média do atraso total de 35.432 em contraposição a 35.552 da MDDa, 35.663 (ATC), 36.081 (COVERT) e 37.121 (MDD). No entanto, pode-se notar que os valores não são expressivamente menores, tanto na média total quanto nas médias por estágios/tarefas. Levando-se isto em consideração e os números de melhores soluções, concluímos que o desempenho é superior porque as regras propostas são consistentemente melhores, ou seja, são levemente melhores, porém superiores na maior parte das 4.320 instâncias. Por exemplo, as regras MDDa e PRTTa apresentam as melhores soluções para 3.152 instâncias, o que corresponde a 73% do total de problemas teste.

A Tabela 7 mostra a média de tempo de resolução para as duas regras propostas e para a regra MDD. O tempo computacional médio da regra PRTTa foi de 3,79 ms para as 4.320 instâncias que é um tempo computacional maior, em comparação com a MDDa. No entanto, é um tempo considerado satisfatório, tendo em vista a melhoria na qualidade dos resultados e sua ordem de grandeza (10^{-3}s), aceitável para a aplicação em casos reais.

Com o objetivo de aprimorar a avaliação das heurísticas propostas, realizamos uma comparação dos resultados obtidos por estes métodos com os resultados obtidos por meio da resolução do modelo de programação linear inteira mista apresentado na Seção 2. Este modelo foi resolvido utilizando-se o pacote computacional *Xpress Optimization Suite 7* com tempo limite de execução estabelecido em 1 hora.

Dada a dificuldade em se obter soluções ótimas para problemas com as dimensões do conjunto original, geramos um novo conjunto de instâncias com dimensões menores para a realização desta avaliação. Nestas instâncias, o número de estágios é fixo e igual a 2 ($s = 2$), assim como o número de máquinas por estágio ($m_1 = m_2 = 2$). Os tempos de processamento foram gerados de acordo com uma distribuição uniforme: entre $1 \cdot m_l$ e $50 \cdot m_l$ para o estágio gargalo e entre $1 \cdot m_l \cdot L$ e $50 \cdot m_l \cdot L$ para os demais estágios. As datas de entrega foram geradas por uma distribuição uniforme dentro do intervalo (Equação 17):

$$P\left(1 - T - \frac{R}{2}\right) \leq d_j \leq P\left(1 - T + \frac{R}{2}\right) \qquad (17)$$

Este conjunto de problemas é composto por 720 instâncias de teste, cinco problemas para cada

Tabela 5. Comparação entre as regras para o método em ordem direta.

Estágios	Tarefas	MDD	MDDa	PRTTa	ATC	COVERT
5	30	0,86	0,20	0,17	0,39	0,34
	50	0,77	0,26	0,25	0,45	0,32
	100	0,67	0,21	0,22	0,39	0,37
10	30	0,95	0,18	0,07	0,23	0,45
	50	0,88	0,20	0,16	0,28	0,41
	100	0,75	0,30	0,30	0,35	0,34
30	30	0,98	0,21	0,01	0,22	0,58
	50	0,99	0,18	0,01	0,19	0,60
	100	0,97	0,14	0,07	0,15	0,51
Média		0,87	0,21	0,14	0,29	0,43

Tabela 6. Média do atraso total.

Estágios	Tarefas	MDD	MDDa	PRTTa	ATC	COVERT
5	30	7.202	6.820	6.806	6.918	6.895
	50	13.611	13.002	12.997	13.215	13.032
	100	26.911	25.839	25.854	26.184	25.918
10	30	11.483	10.955	10.898	10.984	11.146
	50	22.532	21.309	21.250	21.427	21.685
	100	56.558	54.602	54.562	54.716	54.715
30	30	28.744	28.153	28.036	28.167	28.444
	50	50.636	49.009	48.729	49.035	49.849
	100	116.413	110.274	109.760	110.320	113.041
Média		37.121	35.552	35.432	35.663	36.081

Tabela 7. Tempo computacional para o método em ordem direta.

Estágios	Tarefas	MDD	MDDa	PRTTa
5	30	0,13	0,16	0,16
	50	0,32	0,26	0,49
	100	1,24	1,14	3,12
10	30	0,16	0,16	0,16
	50	0,29	0,42	0,91
	100	1,17	1,60	6,54
30	30	0,45	0,43	0,75
	50	0,85	0,81	2,51
	100	3,32	3,27	19,43
Média		0,88	0,92	3,79

combinação do número de tarefas (4, 5, 6, 7, 8, 9), da posição do estágio gargalo no sistema (1 e 2), da diferença da carga de trabalho entre o estágio gargalo e os demais estágios (0,5, 0,7 e 0,9), da amplitude da data de entrega (0,8 e 1,8) e do fator de atraso (0,1 e 0,5).

Neste novo conjunto-teste, o *software Xpress*® encontrou a solução ótima para todos os problemas com até 7 tarefas, 85% dos problemas com 8 tarefas e 56% dos problemas com 9 tarefas. O valores ótimos encontrados estão em média a 10,3% e 9,9% respectivamente dos valores obtidos pelas heurísticas MDDa e PRTTa. No cálculo desta diferença percentual não foram considerados os problemas em que somente o algoritmo exato obteve a soma dos atrasos zero (25 problemas). Isto porque a diferença seria de 100% independentemente do valor encontrado pelas heurísticas. O tempo computacional médio dos métodos propostos para cada problema foi inferior a 0,1 milissegundo enquanto o pacote de otimização apresentou um tempo médio de 162,6 segundos.

5 Comentários finais

Neste trabalho, apresentamos novas regras para o problema de minimização do atraso total no *flowshop* flexível. As duas novas regras propostas consideram o problema como uma série de problemas de máquinas em paralelo e baseiam-se nos métodos de programação em ordem direta. Além disso, a regra PRTTa considera, em cada decisão, todas as tarefas que passarão pelo sistema e não somente as tarefas disponíveis no momento atual. Para tanto foi necessário um método modificado de lista de prioridades. Considerando as 4.320 instâncias avaliadas, a regra PRTTa proporcionou 2.607 melhores soluções, mais de cinco vezes o número de melhores soluções da regra ATC, conhecida por apresentar os melhores resultados para a minimização do atraso total no *flowshop* flexível. Em relação ao RDI, a ATC apresenta um RDI médio de 0,29, mais do que o dobro do RDI médio da regra PRTTa (0,14) e 38% maior que o RDI médio da regra MDDa. Vale destacar que as regras propostas são relativamente simples de serem implementadas e possuem um tempo computacional baixo compatível com aplicações práticas de grande dimensão.

Apesar de a regra PRTTa apresentar um parâmetro de ajuste, foi mostrado que é um parâmetro robusto, portanto não são necessários muitos esforços para ajustá-lo. Além disso, foi desenvolvida a regra MDDa que não depende de parâmetros de ajuste e que também apresentou resultados superiores à regra ATC.

Como futura pesquisa, sugere-se a aplicação de meta-heurísticas que considerem a inserção de tempo ocioso. Conforme Kanet e Li (2004), a utilização da informação sobre a liberação das tarefas para permitir a inserção de tempo ocioso é uma interessante área de pesquisa.

Agradecimentos

Os autores são gratos aos revisores anônimos, pelos seus úteis comentários e sugestões, e a Geun C. Lee, por gentilmente prover as instâncias utilizadas neste trabalho. Esta pesquisa teve o apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

Referências

BAKER, K. R.; BERTRAND, J. W. M. A dynamic priority rule for scheduling against due-dates. **Journal of Operations Management**, v. 3, p. 37-42, 1982.

BRAH, S. A. A comparative analysis of due date based job sequencing rules in a flow shop with multiple processors. **Production Planning and Control**, v. 7, p. 362-373, 1996.

CHOI, S. W.; KIM, Y. D.; LEE, G. C. Minimizing total tardiness of orders with reentrant lots in a hybrid flowshop. **International Journal of Production Research**, v. 43, p. 2149-2167, 2005.

CHU, C.; PORTMANN, M. C. Some new efficient methods to solve the n/1/ri/∑Ti scheduling problem. **European Journal of Operational Research**, v. 58, p. 404-413, 1992.

DU, J.; LEUNG, J. Y. T. Minimizing total tardiness on one machine is NP-hard. **Mathematics of Operations Research**, v. 15, p. 483-495, 1990.

GUINET, A. et al. computational study of heuristics for two-stage flexible flowshops. **International Journal of Production Research**, v. 34, p. 1399-1415, 1996.

KANET, J. J.; LI, X. A weighted modified due date rule for sequencing to minimize weighted tardiness. **Journal of Scheduling**, v. 7, p. 261-276, 2004.

KANET, J. J.; SRIDHARAN, V. Scheduling with inserted idle time: problem taxonomy and literature review. **Operations Research**, v. 48, p. 99-110, 2000.

KOULAMAS, C. P. The total tardiness problem: review and extensions. **Operations Research**, v. 42, p. 1025-1041, 1994.

LEE, G. C.; KIM, Y. D.; CHOI, S. W. Bottleneck-focused scheduling for a hybrid flowshop. **International Journal of Production Research**, v. 42, p. 165-181, 2004.

LINN, R.; ZHANG, W. Hybrid flow shop scheduling: a survey. **Computers and Industrial Engineering**, v. 37, p. 57-61, 1999.

NAWAZ, M.; ENSCORE, E. E.; HAM, I. A heuristic algorithm for the m-machine, n-job flowshop scheduling problem. **Omega**, v. 11, p. 91-95, 1983.

QUADT, D.; KUHN, H. A taxonomy of flexible flow line scheduling procedures. **European Journal of Operations Research**, v. 178, p. 686-698, 2007.

RONCONI, D. P. A note on constructive heuristics for the flowshop problem with blocking. **International Journal of Production Economics**, v. 87, p. 39-48, 2004.

SANTOS, D. L.; HUNSUCKER, J. L.; DEAL, D. E. Global lower bound for flow shops with multiple processors. **European Journal of Operational Research**, v. 80, p. 112-120, 1995.

YANG, Y. Optimization and heuristic algorithms for flexible flow shop scheduling. Tese (Doutorado)–Columbia University, New York, 1998.

YANG, Y.; KREIPL, S.; PINEDO, M. Heuristics for minimizing total weighted tardiness in flexible flow shops. **Journal of Scheduling**, v. 3, p. 89-108, 2000.