Production, v. 25, n. 1, p. 79-91, jan./mar. 2015
doi: http://dx.doi.org/10.1590/S0103-65132014005000016

Regras de prioridade eficientes que exploram características do *Job Shop* Flexível para a minimização do atraso total

Everton Luiz de Melo[a]*, Débora Pretti Ronconi[b]
[a]everton.melo@usp.br, USP, Brasil
[b]dronconi@usp.br, USP, Brasil

Resumo

Este trabalho aborda o ambiente de produção *Job Shop* Flexível (JSF), extensão do problema NP-Difícil *Job Shop*. O JSF envolve um conjunto de *jobs* compostos por operações e cada operação deve ser processada em uma das máquinas habilitadas. O critério considerado é a minimização do atraso total. Inicialmente são identificadas características relacionadas à flexibilidade do sistema de produção, mais especificamente às máquinas habilitadas por operação e aos seus tempos de processamento. A seguir são propostas novas regras que exploram tais características e que são capazes de antever estados futuros do sistema. São realizados experimentos computacionais com 600 instâncias. Comparações com regras da literatura mostram que a melhor heurística proposta supera a melhor regra conhecida em 81% das instâncias.

Palavras-chave

Job Shop. Heurística. Programação matemática. Programação da produção.

1. Introdução

Identificar características particulares de um problema de otimização pode ser um meio eficiente para obter soluções de boa qualidade. Isso é especialmente verdadeiro ao se lidar com problemas fortemente NP-Difíceis, como são vários problemas de escalonamento de tarefas. Assim, o objetivo deste trabalho é propor regras de prioridade eficientes que explorem características do ambiente *Job Shop* Flexível (JSF) para alcançar boas soluções com rapidez.

O JSF é a versão flexível do problema de programação de tarefas *Job Shop* (JS). O JS envolve um conjunto de tarefas, designadas por *jobs*, cada uma formada por uma sequência de operações. Essas operações devem ser sequenciadas em máquinas previamente determinadas otimizando algum critério. No JSF, as operações podem ser processadas alternativamente em mais de uma máquina. Minimizando o *makespan*, Garey et al. (1976) classificam o JS como NP-Difícil. Para a minimização do atraso total, Koulamas (1994) prova que a programação de tarefas no ambiente *flowshop* com apenas duas máquinas é fortemente NP-Difícil. Consequentemente, por redução, os problemas JS e JSF também recebem essa classificação. Tais provas implicam que não são conhecidos algoritmos exatos que resolvam o JSF em tempo computacional razoável. Por essa razão são utilizados métodos alternativos para gerar boas soluções em tempo computacional aceitável.

Neste trabalho, o critério de desempenho utilizado é a minimização do atraso total, o qual tem grande aderência prática. Alvarez-Valdes et al. (2005) utilizam métodos heurísticos construtivos na indústria de vidro. Vilcot & Billaut (2008) empregam Busca Tabu (BT) e Algoritmos Genéticos (AG) na indústria de impressão. Porém, a literatura sobre atraso total é limitada e o critério de desempenho majoritariamente explorado é o *makespan*.

A definição formal do JSF vem a seguir. Considere um conjunto independente de *n jobs*, cada um

*USP, São Paulo, SP, Brasil
Recebido 13/11/2012; Aceito 27/09/2013

formado por uma sequência determinada de operações. Considere também um conjunto de *m* máquinas. Cada operação é associada a um subconjunto de máquinas capazes de realizar seu processamento, o que torna o problema flexível. Cada operação deve ser processada isoladamente, sem interrupção, em uma única máquina do subconjunto de máquinas a ela associado. Devem ser determinadas as atribuições das operações às máquinas e as sequências de processamentos que otimizem o critério estabelecido. A flexibilidade do JSF faz com que um *job* possa ser processado por diferentes caminhos através das máquinas, aumentando muito o número de soluções factíveis.

A flexibilidade do problema possibilita que a resolução seja feita em duas etapas distintas, cada uma delas abordando subproblemas menos complexos. A primeira etapa é a atribuição de cada operação a uma máquina habilitada. Essa fase é denominada roteamento por determinar os caminhos que os *jobs* percorrerão. A segunda etapa é o sequenciamento das operações em cada máquina. Ela equivale à resolução do JS resultante das atribuições da etapa anterior. Essa estratégia de resolução é denominada abordagem hierárquica. Contrastando com ela existe a abordagem integrada, na qual atribuições e sequenciamentos são definidos simultaneamente.

O JSF foi investigado na literatura com diversos critérios de desempenho e através de diferentes métodos de resolução, principalmente heurísticos. Um dos primeiros trabalhos que mencionam o JSF é apresentado por Brucker & Schile (1990), que propõem um algoritmo polinomial para o JSF com até dois *jobs*. Brandimarte (1993) propõe a abordagem hierárquica aliando regras de prioridade e BT. Kacem et al. (2002) classificam os problemas em JSF com flexibilidade total e parcial, de acordo com o grau de flexibilidade das operações. Na flexibilidade total, cada operação pode ser processada por qualquer máquina. Na flexibilidade parcial, cada operação está associada a um subconjunto de máquinas.

Heurísticas e meta-heurísticas também são bastante exploradas na resolução do problema. Regras de prioridade são utilizadas por Tay & Ho (2008) juntamente com Programação Genética. Baykasoğlu & Özbakir (2010) investigam a influência da flexibilidade do problema sobre os resultados de diversas regras que minimizam o atraso médio. AG são utilizados por Kacem et al. (2002), Zhang & Gen (2005), Chan et al. (2006), Ho et al. (2006), Pezzella et al. (2008) e Gutiérrez & García-Magariño (2011). Uma BT tradicional é aplicada por Dauzère-Pérès & Paulli (1997). Uma BT hibridizada com *Particle Swarm Optimization* é empregada por Zhang et al. (2009) e outra BT associada à *Variable Neighborhood Search* é desenvolvida por Li et al. (2010).

Modelos matemáticos para minimização do *makespan* foram encontrados nos trabalhos recentes de Fattahi et al. (2007) e Özgüven et al. (2010). Além da formulação, Fattahi et al. (2007) propõem um conjunto de instâncias que também são resolvidas com BT e *Simulated Annealing*. Özgüven et al. (2010) comparam os resultados de seu modelo com os apresentados pelo modelo de Fattahi et al. (2007), mostrando melhor desempenho.

Especificamente sobre o atraso total, de acordo com o levantamento realizado, o trabalho de Scrich et al. (2004) é o primeiro a considerar tal critério no JSF. Os resultados indicam a regra *Modified Due Date* (MDD), a qual modifica a data de entrega do *job* baseada no tempo de processamento das operações restantes como a melhor regra de prioridade. Ela é utilizada para gerar soluções iniciais a serem melhoradas por BT. Uma aplicação do JSF minimizando atraso na indústria de armamentos é apresentada por Chen et al. (2008). Nessa aplicação, as operações têm sequências alternativas e os *jobs* podem ter relações de precedência. A resolução hierárquica é baseada na regra *Earliest Due Date* (EDD), a qual prioriza o escalonamento de *jobs* com menor data de entrega. Outras regras são usadas para desempate. Gholami & Zandieh (2009) visam obter soluções robustas que minimizem atraso médio e *makespan*. Para isso incorporam simulação a um método baseado em AG. As simulações envolvem probabilidades de quebra de máquina e são utilizadas para determinar o atraso médio e o *makespan* esperados.

Foram encontrados poucos trabalhos que tratam da minimização do atraso total no ambiente JSF, o que reforça a existência de uma lacuna na literatura a respeito desse critério. Por isso, este estudo propõe explorar características do JSF com minimização do atraso total usando novas regras de prioridade, que forneçam melhores soluções. Na próxima seção são descritas as adaptações do modelo matemático utilizado. Na seção 3 são propostas estratégias de exploração das características do problema e regras de prioridade. Na seção 4 são analisados os resultados obtidos pelas regras propostas com as instâncias geradas aleatoriamente. Por fim, a seção 5 traz as conclusões do trabalho.

2. Modelo matemático para minimização do atraso total

O modelo de Programação Linear Inteira Mista (PLIM), descrito a seguir, foi adaptado da formulação matemática apresentada por Özgüven et al. (2010) para minimização do *makespan*. Tal escolha se deve ao fato de esse modelo ter apresentado os melhores

resultados da literatura para esse critério, além da sua adaptação direta para o critério do atraso total. Ressalta-se que neste trabalho a formulação PLIM é utilizada com o intuito de obter soluções ótimas e limitantes que possam ser empregadas na avaliação das heurísticas. O modelo:

Índices e conjuntos:

J	conjunto dos *jobs*;
M	conjunto das máquinas;
O	conjunto das operações;
i, i'	*jobs* ($i, i' \in J$);
j, j'	operações ($j, j' \in O$);
k, k'	máquinas ($k, k' M$);
O_i	conjunto das operações do *job* i ($O_i \subseteq O$);
O_{ij}	operação j pertencente ao *job* i ($O_{ij} \in O$);
M_j	conjunto de máquinas alternativas da operação j ($M_j \subseteq M$);
$M_j \cap M_{j'}$	conjunto de máquinas onde ambas as operações j e j' podem ser processadas;
P_{ij}	conjunto das operações que devem preceder a operação O_{ij} no *job*.

Parâmetros:

t_{ijk}	tempo de processamento da operação O_{ij} na máquina k;
d_i	data de entrega do *job* i;
L	número adequadamente grande fornecido por $\sum_{O_{ij} \in O} \max_{k \in M_j}(t_{ijk})$.

Variáveis de decisão:

X_{ijk}	assume 1 se a máquina k é selecionada para a operação O_{ij} e 0, em caso contrário;
S_{ijk}	instante de início de processamento da operação O_{ij} na máquina k;
C_{ijk}	instante de término de processamento da operação O_{ij} na máquina k;
$Y_{iji'j'k}$	assume 1 se O_{ij} precede $O_{i'j'}$ na máquina k e 0, caso contrário;
C_i	instante de término de processamento do *job* i;
T_i	atraso do *job* i.

Função objetivo:

$$Minimizar \sum_{i \in J} T_i \tag{1}$$

Sujeito a:

$$\sum_{k \in M_j} X_{ijk} = 1 \quad \forall i \in J, \forall j \in O_i \tag{2}$$

$$S_{ijk} \leq X_{ijk} \cdot L \quad \forall i \in J, \forall j \in O_i, \forall k \in M_j \tag{3}$$

$$C_{ijk} \leq X_{ijk} \cdot L \quad \forall i \in J, \forall j \in O_i, \forall k \in M_j \tag{4}$$

$$C_{ijk} \geq S_{ijk} + t_{ijk} - (1 - X_{ijk}) \cdot L \quad \forall i \in J, \forall j \in O_i, \forall k \in M_j \tag{5}$$

$$S_{ijk} \geq C_{i'j'k} - (Y_{iji'j'k}) \cdot L \quad \forall i < i', \forall j \in O_i, \forall j' \in O_{i'}, \forall k \in M_j \cap M_{j'} \tag{6}$$

$$S_{i'j'k} \geq C_{ijk} - (1 - Y_{iji'j'k}) \cdot L \quad \forall i < i', \forall j \in O_i, \forall j' \in O_{i'}, \forall k \in M_j \cap M_{j'} \tag{7}$$

$$\sum_{k \in M_j} S_{ijk} \geq \sum_{k' \in M_{j'}} C_{ij'k'} \quad \forall i \in J, \forall j \in O_i, \forall O_{ij'} \in P_{ij} \tag{8}$$

$$C_i \geq \sum_{k \in M_j} C_{ijk} \quad \forall i \in J, \forall j \in O_i \tag{9}$$

$$T_i \geq C_i - d_i \quad \forall i \in J \tag{10}$$

$$X_{ijk} \in \{0,1\} \quad \forall i \in J, \forall j \in O_i, \forall k \in M_j \tag{11}$$

$$S_{ijk} \geq 0 \quad \forall i \in J, \forall j \in O_i, \forall k \in M_j \tag{12}$$

$$C_{ijk} \geq 0 \quad \forall i \in J, \forall j \in O_i, \forall k \in M_j \tag{13}$$

$$Y_{iji'j'k} \in \{0,1\} \quad \forall i < i', \forall j \in O_i, \forall j' \in O_{i'}, \forall k \in M_j \cap M_{j'} \tag{14}$$

$$C_i \geq 0 \quad \forall i \in J \tag{15}$$

$$T_i \geq 0 \quad \forall i \in J \tag{16}$$

O modelo minimiza o atraso total conforme a função objetivo (1); as restrições do tipo (2) garantem a atribuição de cada operação a exatamente uma máquina habilitada; as restrições (3 e 4) asseguram que, caso uma operação O_{ij} não seja alocada a uma máquina k, seus instantes de início e término de processamento nessa máquina sejam nulos; as restrições (5) garantem que, caso uma determinada operação seja atribuída a certa máquina, seu instante de término seja, no mínimo, a soma de seu instante de início de processamento e de seu tempo de processamento naquela máquina; as restrições (6 e 7) evitam a execução simultânea de operações alocadas em uma mesma máquina; as restrições (8) estabelecem que as precedências das operações sejam respeitadas; as restrições (9) determinam que o instante de término de cada *job* seja maior ou igual ao instante de término de sua última operação; as restrições (10) determinam o atraso de cada *job*; e as restrições de (11 a 16) definem os domínios das variáveis, sendo que as restrições (11 e 14) tratam a condição de integralidade.

3. Identificação das características do *Job Shop* Flexível e regras de prioridade

Pelo fato de o JSF estar entre os problemas de otimização mais difíceis da literatura é frequente o uso de métodos efetivos e rápidos para a geração de soluções. Um desses métodos é o *List Scheduling Algorithm*. Conforme Kim (1995), sempre que uma máquina fica ociosa, o método seleciona a operação de maior prioridade, dentre as disponíveis, para

ocupá-la. Essa escolha é realizada com uma regra de prioridade. De acordo com Panwalkar & Iskander (1977), as regras de prioridade estão fortemente ligadas aos critérios de desempenho. Além de utilizar um método eficiente, é extremamente interessante quando informações a respeito do problema podem ser usadas para aprimorar os resultados. Dessa forma, a exploração de algum conhecimento sobre o problema pode permitir que soluções de qualidade bastante superior sejam alcançadas sem aumento do custo computacional.

Com esse intuito, nesta seção são propostas formas de identificar e quantificar características do JSF. Primeiramente são relacionadas regras da literatura indicadas para minimização do atraso, bem como as adaptações necessárias para que elas possam ser aplicadas ao JSF. A seguir são descritas as estratégias que identificam as características do JSF e as regras de prioridade propostas.

3.1. *Regras de prioridade adaptadas da literatura*

Regras de prioridade com bom desempenho na minimização do atraso na literatura, não necessariamente abordando o JSF, foram implementadas. Tais regras são descritas na sequência, assim como as respectivas adaptações propostas, quando necessárias.

- Shortest Processing Time (SPT): regra que prioriza a operação com menor tempo de processamento e é recomendada por Baker (1984) para o critério do atraso;
- Minimum Slack Time (MST): prioriza o *job* com a menor folga em relação à sua data de entrega. Neste trabalho seu cálculo utilizou os tempos médios de processamento das operações em suas diversas máquinas alternativas;
- Smallest Critical Ratio (SCR): prioriza o *job* com menor relação entre o tempo disponível até sua data de entrega e o tempo ainda demandado de processamento. Esse cálculo também utilizou os tempos médios de processamento das operações;
- EDD: prioriza *jobs* com menor data de entrega. Uma descrição mais detalhada das regras MST, SCR e EDD originais se encontra em Vepsalainen & Morton (1987);
- Modified Operation Due Date (MOD): proposta de Baker (1984) que estima datas de entrega para as operações priorizando o *job* com a menor data de entrega. Essas estimativas utilizaram os tempos médios de processamento das operações;
- MDD*:* proposta de Baker & Bertrand (1982) que modifica a data de entrega do *job* baseada no tempo de processamento das operações restantes. Como essas operações ainda não têm máquinas definidas, foi utilizado o tempo de processamento mínimo entre as máquinas habilitadas. A data de entrega modificada, d_i' para o JSF é obtida com a equação a seguir:

$$d_i' = \max\left(d_i, t + \sum_{j \in G_i} \min_{k \in M_j} t_{jk}\right) \quad (17)$$

onde t é o instante atual considerado pelo *List Scheduling Algorithm*; G_i é o conjunto das operações ainda não alocadas do *job i*; e t_{jk} é o tempo de processamento da operação j na máquina k. Desse modo d_i' indica o menor instante possível para a conclusão do *job i*. Na aplicação dessa regra, prioriza-se o *job* com menor data de entrega modificada d_i'.

Priority Rule for Total Tardiness adaptada (PRTTa): proposta de Mainieri & Ronconi (2013) que estima datas de entrega para os estágios do *flowshop* flexível e antevê estados futuros do sistema. A adaptação proposta para o JSF utiliza os tempos médios das operações para estimar suas datas de entrega. Tais estimativas são calculadas de modo similar ao empregado por Mainieri & Ronconi (2013), ou seja, a data de entrega do *job* menos o tempo médio de processamento das operações restantes. A PRTTa para o JSF é definida como:

$$PRTTa(j,k,t) = \alpha \cdot \max(r_j, t) + \max[\max(r_j, t) + t_{jk}, d_j] \quad (18)$$

sendo d_j sua data de entrega estimada e r_j o instante de conclusão da predecessora de j, já que a regra considera operações não liberadas. No primeiro termo, o valor $\alpha > 0$ penaliza operações ainda não disponíveis. No segundo termo, o potencial instante de término da operação é comparado com sua data de entrega estimada. É selecionada a operação com menor valor *PRTTa*(j, k, t).

3.2. *Identificação e utilização das características do problema*

A seguir são propostas duas estratégias para explorar características específicas do JSF. Essas estratégias podem ser associadas a diferentes regras de prioridade ou a outros métodos de resolução, guiando as atribuições e os sequenciamentos.

3.2.1. *Afinidade entre operações e máquinas*

A primeira estratégia proposta está relacionada à ideia intuitiva de atribuir as operações às máquinas que possam processá-las mais rapidamente. Contudo é preciso que essa percepção seja explorada na medida correta para que bons resultados sejam alcançados. Assim, a atribuição de certa operação a uma máquina pode ser induzida, mas não deve ser imposta. Neste trabalho é denominada afinidade, A_{jk}, a conveniência

de atribuir uma operação *j* a uma máquina *k*. Essa conveniência baseia-se na relação entre o tempo de processamento de *j* em *k* e os tempos de processamento de *j* em suas demais máquinas alternativas. A afinidade A_{jk} é expressa pela relação:

$$A_{jk} = \frac{t_{jk}}{\sum_{k' \in M_j} t_{jk'}} \quad (19)$$

Essa relação varia de 0 a 1 e quanto menor seu valor maior a afinidade entre *j* e *k*. Essa medida pode ser aplicada para direcionar cada operação para alguma máquina que possa efetuar seu processamento em menor tempo. A afinidade entre uma operação e uma máquina não habilitada para ela não é definida.

3.2.2. *Grau de flexibilidade das operações*

Outra característica do JSF a ser identificada é o grau de flexibilidade das operações candidatas. Considerar o grau de flexibilidade permite favorecer operações com menor número de alternativas de processamento, ou seja, aquelas com menor grau de flexibilidade. Isso pode ser conveniente, pois essas operações possuem possibilidades de atribuição mais restritivas e, se não escolhidas, podem dificultar o restante da resolução. A flexibilidade F_j de uma operação *j* é fornecida por:

$$F_j = \frac{|M_j|}{m} \quad (20)$$

onde $|M_j|$ indica o número de máquinas habilitadas para a operação *j*. Como a afinidade, a flexibilidade varia de 0 a 1. Quanto menor o valor de F_j maior deve ser a possibilidade de que a operação *j* seja selecionada.

3.3. *Regras propostas*

A primeira regra proposta é denominada Data de entrega Modificada com Antevisão (DMA) e utiliza elementos das regras PRTTa e MDD. O intuito da DMA é aliar o bom desempenho da regra MDD com a antevisão da PRTTa. A antevisão permite que quando uma máquina se torna ociosa, uma operação ainda não disponível, mas com instante de liberação r_j conhecido, seja atribuída a ela. Como essa escolha gera tempo ocioso na máquina, ela deve ser realizada com cautela. Isso é feito através de uma constante α que pondera a penalização da escolha de uma operação ainda não liberada. Além disso, a data de entrega de cada *job* é atualizada quando ele não pode ser entregue no prazo. Essa atualização leva em consideração a possibilidade de as operações candidatas ainda não estarem disponíveis. A regra então pode ser usada para calcular o valor DMA (*i*, *j*, *t*) que determina a prioridade da operação *j* no instante *t*:

$$DMA(i,j,t) = \alpha \cdot \max(r_j, t) + \max\left[d_i, \max(r_j, t) + \sum_{j' \in G_i} \min_{k' \in M_{j'}} t_{j'k'} \right] \quad (21)$$

O primeiro termo, com $\alpha > 0$, penaliza as operações não disponíveis no instante *t*, evitando inserir ociosidades excessivas na programação. O segundo termo atualiza a data de entrega do *job i*. Para tanto, o tempo mínimo de processamento das operações restantes do *job i* é somado ao máximo entre *t* e r_j. Então esse valor é comparado à data de entrega original do *job*. Assim, a DMA é capaz de considerar como candidatas não somente as operações já disponíveis mas também aquelas cujas predecessoras estejam alocadas, embora não finalizadas. Para isso, ela antevê em que instante do futuro essas operações se tornarão disponíveis. Operações que possuem predecessoras ainda não alocadas não são candidatas. Sempre é selecionada a operação com menor valor *DMA*(*i*, *j*, *t*).

A segunda regra proposta associa a DMA à afinidade e à flexibilidade. O objetivo é induzir o *List Scheduling Algorithm* a atribuir cada operação a uma das máquinas que a processem com maior rapidez, além de favorecer operações com menos máquinas alternativas. Essa regra é denominada DMA com Afinidade e Flexibilidade (DMA-AF) e incorpora as estratégias do seguinte modo:

$$DMA-AF(i,j,k,t) = \alpha \cdot \max(r_j, t) + A_{jk} \cdot F_j \cdot \max\left[d_i, \max(r_j, t) + \sum_{j' \in G_i} \min_{k' \in M_{j'}} t_{j'k'} \right] \quad (22)$$

Deve ser selecionada a operação com menor valor *DMA-AF*(*i*, *j*, *k*, *t*). Quanto maior for a afinidade entre *j* e *k*, menor será A_{jk} e, portanto, menor resultará o valor da regra. Do mesmo modo, quanto menor for a flexibilidade de *j*, menor será F_j, tornando *DMA-AF*(*i*, *j*, *k*, *t*) menor. A incorporação dessas estratégias por regras da literatura é abordada na seção dos experimentos computacionais.

3.4. *Algoritmo construtivo*

A seguir é apresentado um algoritmo construtivo que visa gerar soluções viáveis de boa qualidade com rapidez. Esse procedimento seleciona as operações de acordo com uma regra de prioridade utilizando a abordagem integrada. Ele é baseado nos procedimentos propostos por Scrich (1997) e Mainieri & Ronconi (2013).

O método inicia no instante zero, quando todas as máquinas estão disponíveis e todas operações são conhecidas. À medida que o tempo avança, as operações são atribuídas às máquinas, até que todas sejam alocadas. Cada vez que uma máquina fica ociosa

são levantadas as operações candidatas a ocupá-la. É selecionada a operação de maior prioridade, conforme a regra vigente. Neste trabalho, a operação de maior prioridade é a que permita à regra gerar o menor valor. Quando duas ou mais máquinas estão disponíveis ao mesmo tempo, a primeira máquina a receber uma operação é aquela que possuir o menor carregamento estimado. Esse carregamento é fornecido pela soma dos tempos de processamento de todas as operações ainda não alocadas que a máquina pode executar. O intuito dessa seleção é reduzir parte da sobrecarga de máquinas com carregamento estimado elevado, facilitando futuras alocações.

As regras PRTTa, DMA e DMA-AF podem alocar operações ainda não liberadas, inserindo tempo ocioso na máquina. Caso isso aconteça, o algoritmo construtivo tenta preencher esse tempo com outras operações sem interferir no instante de início de processamento da operação previamente alocada. As operações usadas para preencher o tempo ocioso são as de menor instante de liberação. O algoritmo envolve:

t	instante da programação que está sendo considerado pelo algoritmo;
R_k	instante em que a máquina k se torna disponível;
m_j	máquina selecionada para processar a operação j;
t_{jk}	tempo de processamento da operação j na máquina k;
r_j	instante de liberação da operação j;
s_j	instante de início de processamento da operação j;
u_k	assume 1 se a máquina k, ociosa, não possui operação candidata e 0, caso contrário;
m	número de máquinas;
$\|O\|$	número total de operações do problema;
M_t	conjunto das máquinas disponíveis no instante t;
$\|M_t\|$	número de máquinas disponíveis no instante t;
$M_t[w]$	indica a w-ésima máquina com menor carregamento estimado de M_t;
A	conjunto das operações já alocadas a alguma máquina;
$\|A\|$	número de operações já alocadas a alguma máquina;
T^{ocioso}	tempo ocioso gerado pela alocação de uma operação ainda não disponível;
t'	instante no qual tem início um tempo ocioso;
L	número grande.

O algoritmo implementado é detalhado na sequência.

procedimento Algoritmo Construtivo (entrada: *Inst_JSF*, *Regra*)
início
$t \leftarrow 0$
$A \leftarrow \emptyset$
para $k \leftarrow 1$ até m faça:
  $R_k \leftarrow 0$
  $u_k \leftarrow 0$
fim para
para $j \leftarrow 1$ até $|O|$ faça:
  se (a operação j não tem predecessora no job) faça:
    $r_j \leftarrow 0$
  senão
    $r_j \leftarrow \infty$
  fim senão
fim para
enquanto ($|A| \neq |O|$) faça:
  $M_t \leftarrow \emptyset$
  para $k \leftarrow 1$ até m faça:
    se (($R_k \leq t$) ou ($u_k = 1$)) faça:
      $M_t \leftarrow M_t \cup$ máquina k
    fim se
  fim para
  Ordene as máquinas de M_t por carregamento estimado crescente
  para $j \leftarrow 1$ até $|O|$ faça:
    se (as predecessoras de j estiverem concluídas) faça:
      $r_j \leftarrow$ instante de término da última predecessora
    fim se
  fim para
  para $w \leftarrow 1$ até $|M_t|$ faça:
    $k = M_t[w]$
    Identifique as operações candidatas para k em t segundo *Regra*
    se (houver operação candidata) faça:
      Calcule as prioridades das operações candidatas segundo *Regra*
      Selecione a operação j de maior prioridade
      $m_j \leftarrow k$
      $s_j \leftarrow \max\{t, r_j\}$
      $A \leftarrow A \cup$ operação j
      $R_k \leftarrow s_j + t_{jk}$
      $u_k \leftarrow 0$
      Retire a operação j da lista de candidatas
      se ($r_j > t$) faça:
        $T^{ocioso} \leftarrow r_j - t$
        $t' \leftarrow t$
        enquanto (houver operação candidata j' com $t_{j'k} \leq T^{ocioso}$) faça:
          Selecione a operação candidata j' com menor r_j
          $m_{j'} \leftarrow k$
          $s_{j'} \leftarrow \max\{t', r_{j'}\}$
          $A \leftarrow A \cup$ operação j'
          $t' \leftarrow t' + s_{j'} + t_{j'k}$
          $T^{ocioso} \leftarrow r_{j'} - t'$
          Retire a operação j' da lista de candidatas
        fim enquanto
      fim se
    senão
      $u_k \leftarrow 1$

55. $R_k \leftarrow L$
56. **fim senão**
57. **fim para**
58. $t \leftarrow \min_{k=1...m}\{R_k\}$
59. **fim enquanto**
60. Devolva (**saída:** solução estabelecida por m_j, s_j, j=1,...,|O|)
61. **fim.**

Esse algoritmo construtivo tem como entradas uma instância do problema (*Inst_JSF*) e uma regra de prioridade (*Regra*). A identificação das operações candidatas (linha 30) para uma máquina k no instante t depende da regra utilizada. Se a regra não possuir antevisão, somente as operações liberadas se tornam candidatas. Se a regra possuir antevisão, as operações com instante de liberação conhecido também se tornam candidatas. Havendo operações candidatas, a prioridade de cada uma delas é calculada (linha 32) conforme a regra utilizada. De acordo com a regra fornecida como argumento de entrada, a operação prioritária j pode variar (linha 33). Desse modo se pode considerar que o algoritmo apresentado fornece diferentes possibilidades de obter soluções, representando diversas heurísticas construtivas. Caso ocorra empate na seleção da linha 33, o desempate é feito priorizando-se a operação que possa liberar a máquina a ser ocupada mais rapidamente. No caso das regras com antevisão, as linhas de 40 a 52 preenchem os tempos ociosos gerados pelas alocações de operações ainda não liberadas. Isso ocorre enquanto, dentre as candidatas, houver operações (linha 43) que possam ser alocadas sem interferir no processamento da operação previamente alocada. Entre essas operações é selecionada a operação j' com menor instante de liberação. Em caso de empate na seleção da linha 44, é selecionada a operação com maior prioridade. O algoritmo gera soluções para o JSF de acordo com a regra de entrada e fornece como saída uma solução para o problema.

4. Experimentos computacionais

Nesta seção são apresentadas a descrição da geração das instâncias utilizadas e a análise dos resultados obtidos pelas heurísticas construtivas.

4.1. *Geração de instâncias*

A geração aleatória de instâncias foi baseada em Scrich (1997) e Scrich et al. (2004). Esse processo e as alterações propostas são detalhados no decorrer desta subseção. Os valores aleatórios foram fornecidos pelo gerador de Taillard (1993), que permite obter valores inteiros em uma distribuição uniforme $U[a, b]$. Cada instância gerada tem os seguintes componentes:

n	número de *jobs*;
m	número de máquinas;
$\|O_i\|$	número de operações do *job* i;
M_j	conjunto de máquinas que podem processar a operação j;
$\|M_j\|$	número de máquinas alternativas para a operação j;
t_{jk}	tempo de processamento da operação j na máquina k;
$\overline{t}_j$	tempo médio de processamento da operação j em suas máquinas alternativas;
d_i	data de entrega do *job* i;
δ	fator usado no cálculo das datas de entrega dos *jobs*.

Os valores dos componentes das instâncias geradas são apresentados na Tabela 1. Os parâmetros n e m são definidos previamente. O valor de $|O_i|$ varia de $\lceil m/2 \rceil$ a m. O tempo de processamento da operação j em uma máquina alternativa k', t_{jk}, é obtido no intervalo [1, 99]. Nas demais máquinas, os tempos são limitados a intervalos compreendidos em [t_{jk}, min(3·t_{jk}, 99)]. Uma diferença em relação a Scrich et al. (2004) é que $|M_j|$ é obtido em três faixas distintas de flexibilidade, com médias de 20%, 50% e 80%. Cada instância é gerada utilizando-se somente uma dessas três faixas. O intuito é ter operações com diferentes $|M_j|$, sendo os elementos de M_j obtidos aleatoriamente em $U[1, m]$. Em Scrich et al. (2004), todas as operações de uma dada instância têm o mesmo $|M_j|$. Outra diferença em relação a Scrich et al. (2004) envolve as datas de entrega, calculadas a partir da soma dos tempos médios de processamento das operações dos *jobs*. O tempo médio do *job* é multiplicado por um fator δ definido a partir de n e m, sendo que $\delta = (n/m)^{1/2}$ se $n \geq m$, e $\delta = 1$, caso contrário. Esse cálculo visa gerar soluções mais realistas, com datas de entrega menos restritivas quando o número de máquinas do problema é pequeno. O resultado obtido é multiplicado por 0,8; 1,0; 1,2 ou por uma combinação aleatória desses três valores, CA. Assim existem quatro tipos de datas de entrega mais ou menos restritivas. Cada instância utiliza um desses quatro tipos de data de entrega. Em Scrich et al. (2004) existem dois tipos de data de entrega. Um deles é obtido pela multiplicação do tempo médio do *job* por (n·m)/1.000+0,5 e o outro por (n·m)/1.000+1,0.

Foram geradas instâncias com as seguintes dimensões expressas pela relação n×m: 5×5, 10×5, 10×10, 15×10, 15×15, 30×15, 50×5, 50×10, 50×15 e 100×10. Para cada uma dessas dez combinações foi gerado um grupo de cinco instâncias, formando um subconjunto de 50 instâncias. Como existem três faixas de flexibilidade e quatro tipos de data de entrega, foram criados 12 subconjuntos de 50 instâncias. Cada subconjunto tem uma combinação distinta de faixa de flexibilidade e opção de data de entrega. No total foram geradas 600 instâncias para o JSF.

4.2. *Obtenção e análise dos resultados*

Os experimentos foram conduzidos em um computador com processador Intel Core i7 870 de 2,93GHz, 16GB de RAM e sistema operacional Windows 7. O algoritmo construtivo foi implementado em linguagem C.

4.2.1. *Calibragem do parâmetro* α

Testes preliminares foram realizados para definir o valor da constante de ponderação α presente nas regras com antevisão. Observou-se que utilizar uma constante relacionada à dimensão das instâncias fornecia melhores resultados. Então a constante α foi composta por um valor relacionado à dimensão da instância multiplicado por outro valor denominado α'. O valor usado para relacionar α à dimensão das instâncias foi m. Isso se mostrou interessante porque, independentemente do nível de flexibilidade, quanto maior o número de máquinas do problema, mais máquinas alternativas cada operação possui. Por isso, com mais máquinas é menos interessante inserir tempo ocioso, já que as operações têm outras alternativas de processamento. Assim, a constante foi definida como $\alpha = m \cdot \alpha'$ e o valor de α variou somente pela variação de α'. O gráfico da Figura 1 exibe os valores do atraso total médio das 600 instâncias à medida que α' varia na DMA.

Foram feitos experimentos com α' variando de 0 a 100 com passo 0,1. Contudo, com $\alpha' > 50$, o atraso não variou significativamente. Assim, o eixo horizontal do gráfico da Figura 1 foi restrito para melhor visualização. Observa-se que, após grande redução inicial, o atraso teve pequenas oscilações. Contudo, acima de um determinado valor de penalização, o atraso total médio se estabiliza, tornando a calibragem menos crítica. Para a DMA, o melhor valor para o parâmetro foi $\alpha = m \cdot 16{,}8$, com o qual a média do atraso total foi igual a 14.681,3. Com $\alpha = \alpha' = 0$, o valor médio do atraso total foi 19.766,7, mostrando que a inserção de tempos ociosos pode ser prejudicial se não for controlada. Processos de calibragem semelhantes foram realizados para as demais regras de prioridade. Na PRTTa, o valor usado foi $\alpha = m \cdot 5{,}1$.

4.2.2. *Regras básicas*

Os resultados das heurísticas construtivas básicas são apresentados na Tabela 2. São chamadas básicas as heurísticas que utilizam regras de prioridade que não incorporam as estratégias de afinidade e flexibilidade. Os valores de cada linha da Tabela 2 equivalem a 60 instâncias e estão estratificados pelas dimensões $n \times m$. Para cada regra é apresentada a média do atraso total no respectivo subconjunto de 60 instâncias, sendo que em negrito são identificados os melhores resultados. Abaixo desse valor está, entre parênteses, o número de vezes em que a regra obteve o menor atraso total, mesmo empatando com uma ou mais regras. Na última linha é indicado o tempo total consumido por cada regra para resolver as 600 instâncias.

Entre as regras básicas se destacaram DMA, MDD e EDD. Com atraso total médio geral igual a 14.681,3, a DMA foi a melhor regra. Em seguida veio a MDD, melhor regra conforme Scrich et al. (2004), com média geral 14.781,8, seguida pela EDD, em terceiro lugar. A DMA também foi a regra que mais vezes obteve a melhor solução, 172 ocasiões. Depois aparece a PRTTa, com 154 melhores soluções, mas com atraso médio geral igual a 18.277,3. Uma justificativa pode ser a falta de precisão das datas de entrega estimadas por operação, que se torna crítica quando os problemas crescem, o mesmo servindo para a MOD. Com relação às instâncias com 50 *jobs* ou mais, a DMA apresentou o melhor desempenho, com atraso total médio menor em todas essas dimensões. Nota-se que as heurísticas baseadas em antevisão, DMA e PRTTa, conseguiram menores médias em sete das dez dimensões.

O tempo computacional das heurísticas pode ser considerado satisfatório, pois cada uma consumiu menos de cinco segundos para resolver todas as

Tabela 1. Valores dos componentes das instâncias geradas.

Componente	Valores utilizados
n	5, 10, 15, 30, 50, 100
m	5, 10, 15
$\lvert O_j \rvert$	$U\left[\lceil m/2 \rceil, m\right]$
$\lvert M_j \rvert$	$U\left[\lceil 0{,}1 \cdot m \rceil, \lceil 0{,}3 \cdot m \rceil\right]; U\left[\lceil 0{,}3 \cdot m \rceil, \lceil 0{,}7 \cdot m \rceil\right]; U\left[\lceil 0{,}6 \cdot m \rceil, m\right]$
M_j	$\lvert M_j \rvert$ máquinas sorteadas utilizando $U[1, m]$
t_{jk}	$U[1, 99]$, para uma máquina $k' \in M_j$; e $U[t_{jk}$, min(3. t_{jk}, 99)], para as demais
δ	Se $n \geq m$, $\delta = (n/m)^{1/2}$, senão $\delta = 1$
d_i	$0{,}8 \cdot \delta \cdot \sum_{j=1}^{\lvert O_i \rvert} \bar{t}_j; 1{,}0 \cdot \delta \cdot \sum_{j=1}^{\lvert O_i \rvert} \bar{t}_j; 1{,}2 \cdot \delta \cdot \sum_{j=1}^{\lvert O_i \rvert} \bar{t}_j; CA$

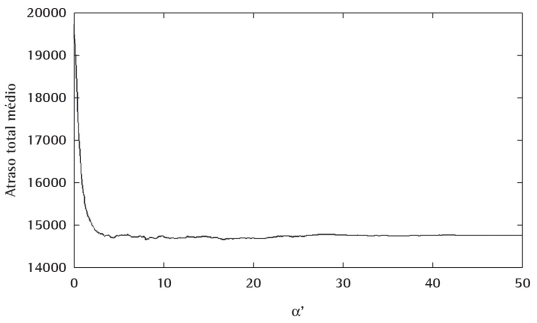

Figura 1. Calibragem da regra DMA.

Tabela 2. Comparação entre as regras básicas.

Dimensão ($n \times m$)	Atraso total médio (número de melhores soluções)							
	SPT	MST	SCR	EDD	MDD	MOD	PRTTa	DMA
5×5	**190,0** (36)	231,6 (17)	229,1 (15)	208,5 (25)	205,0 (24)	**190,0** (37)	190,7 (33)	205,0 (24)
10×5	715,5 (9)	842,5 (6)	722,8 (5)	645,2 (13)	639,7 (11)	708,2 (6)	**629,8** (22)	640,4 (10)
10×10	**593,4** (28)	662,4 (9)	655,4 (12)	668,3 (10)	663,4 (8)	593,7 (22)	595,9 (26)	669,3 (10)
15×10	1.108,9 (8)	1.222,3 (14)	1.123,6 (8)	1.099,4 (3)	1.099,5 (8)	1.086,0 (10)	**966,6** (26)	1.118,3 (3)
15×15	1.201,1 (24)	1.346,3 (17)	1.314,9 (13)	1.277,3 (17)	1.282,7 (18)	1.227,3 (16)	**1.192,5** (25)	1.284,9 (12)
30×15	5.383,2 (0)	5.786,2 (5)	4.645,9 (4)	**4.056,5** (14)	4.082,7 (7)	4.948,3 (2)	4.324,4 (19)	4.085,7 (9)
50×5	25.883,2 (3)	29.293,6 (0)	22.506,9 (4)	21.915,2 (9)	21.526,4 (26)	27.471,7 (0)	25.230,1 (0)	**21.468,1** (26)
50×10	22.187,1 (0)	25.302,1 (0)	17.806,6 (2)	16.598,1 (15)	16.390,9 (17)	22.976,4 (0)	20.386,8 (1)	**16.312,5** (25)
50×15	19.077,3 (1)	23.936,4 (0)	16.209,0 (1)	14.331,0 (13)	14.284,3 (16)	19.647,6 (0)	17.698,6 (2)	**14.075,0** (27)
100×10	118.898,5 (0)	123.230,3 (0)	89.425,6 (5)	87.544,1 (15)	87.643,4 (14)	123.985,4 (0)	111.557,4 (0)	**86.953,4** (26)
Média (total)	19.523,8 (109)	21.185,4 (68)	15.464,0 (69)	14.834,3 (134)	14.781,8 (149)	20.283,5 (93)	18.277,3 (154)	**14.681,3** (172)
Tempo total (s)	3,5	3,6	3,9	3,8	3,8	3,8	4,0	4,5

600 instâncias. Além disso, conforme esperado, as regras que possuem antevisão, com provavelmente maior número de operações candidatas em cada instante de avaliação, apresentaram um tempo computacional levemente superior ao das regras sem antevisão.

4.2.3. *Regras com afinidade e flexibilidade*

Identificadas as regras básicas com os melhores resultados, puderam ser propostas regras mais elaboradas que utilizam EDD e MDD e que exploram a afinidade e a flexibilidade. Num segundo momento, constatou-se que a inclusão da antevisão permitiria melhorias que, apesar de modestas, não poderiam ser desprezadas. O princípio é o mesmo usado na elaboração da DMA-AF, com a diferença de que A_{jk} e F_j multiplicam, respectivamente, os valores fornecidos pelas regras básicas. As equações da EDD com Antevisão, Afinidade e Flexibilidade (EDD-AAF) e da MDD com Antevisão, Afinidade e Flexibilidade (MDD-AAF) são apresentadas a seguir:

$$EDD - AAF(i,j,k,t) = \alpha \cdot \max(r_j, t) + A_{jk} \cdot F_j \cdot d_i \quad (23)$$

$$MDD - AAF(i,j,k,t) = \alpha \cdot \max(r_j, t) + A_{jk} \cdot F_j \cdot d_i' \quad (24)$$

A constante de penalização foi fixada em $\alpha = m \cdot 0{,}5$ na EDD-AAF e em $\alpha = m \cdot 0{,}6$ na MDD-AAF e na DMA-AF. Os resultados obtidos com essas regras estão na Tabela 3.

É possível observar que a consideração da afinidade e da flexibilidade das operações permitiu que as regras básicas melhorassem consideravelmente seus resultados. No caso da EDD-AAF e da MDD-AAF, a antevisão também contribuiu. EDD-AAF e MDD-AAF conseguiram melhor média de atraso total em quatro dimensões cada e a DMA-AF, em duas. No geral, a EDD-AAF foi a regra com melhores resultados, alcançando média geral de atraso total igual a 10.810,2. Seguida pela MDD-AAF, com média 10.898,7, e pela DMA-AF, com 10.900,4. Para evidenciar a contribuição das estratégias de afinidade e de flexibilidade é preciso mencionar que as versões sem antevisão da EDD-AAF e da MDD-AAF obtiveram médias de atraso total iguais a 11.003,6 e 11.106,9, respectivamente. Ambos os resultados são bastante inferiores aos das regras EDD e MDD originais e mostram que a contribuição das estratégias de exploração foi mais decisiva que a contribuição da antevisão. Com relação ao número de melhores resultados, a EDD-AAF alcançou a melhor solução em 389 ocasiões. O tempo computacional se manteve reduzido e nenhum método ultrapassou cinco segundos para resolver todo o conjunto de 600 instâncias. As melhorias relativas obtidas pela inclusão das estratégias de afinidade e de flexibilidade, além da antevisão no caso da EDD-AAF e da MDD-AAF,

podem ser consultadas na Tabela 4. Os valores são obtidos por:

$$\%Dif = \left[\frac{T^{Básica} - T^{AF}}{T^{Básica}}\right] \cdot 100 \quad (25)$$

onde T^{AF} representa o atraso total médio de cada regra com afinidade e flexibilidade e $T^{Básica}$ indica o atraso total médio de sua regra original correspondente.

Nota-se que utilizar o conhecimento a respeito do problema permitiu que os resultados da DMA-AF fossem, no geral, 25,75% melhores que os da DMA. Com a MDD-AAF, a redução geral de atraso foi de 26,27%. Na EDD-AAF, a exploração das características do problema possibilitou as maiores melhorias, chegando a 27,13% no geral. Observa-se que as melhorias foram significativas em todas as dimensões e atingiram, no mínimo, 8,50%. A maior melhoria foi constatada com a EDD-AAF, na dimensão 50×15, sendo 38,97%. De modo geral, todas as regras com afinidade e antevisão conseguiram as maiores melhorias com as maiores dimensões. Essa constatação é interessante, pois é exatamente nessas instâncias que reside a maior dificuldade dos métodos de resolução. Isso ocorre especialmente com os métodos de melhoria, que passam a consumir muito tempo de processamento quando muitos *jobs* ou máquinas são envolvidos. Na Tabela 5 estão as melhorias alcançadas por tais regras em diferentes níveis médios de flexibilidade.

Nota-se claramente que melhorias ainda mais significativas puderam ser alcançadas em níveis de flexibilidade mais altos. Quanto maior o nível de flexibilidade, maior é o número de máquinas alternativas. Também é maior o número de soluções possíveis e mais complexa é a resolução do problema. Dessa forma, a exploração da afinidade e da flexibilidade foi capaz de distanciar os resultados das regras básicas e das regras com estratégias. Essa tendência foi a inversa da observada nos experimentos de Baykasoğlu & Özbakir (2010), quando as diferenças entre as regras diminuíram à medida que a flexibilidade aumentou.

A utilização da afinidade e da flexibilidade também permitiu diferentes melhorias com diferentes tipos de data de entrega. Com as datas de entrega mais apertadas, a melhoria da EDD-AAF sobre a EDD foi de 21,08%. Com datas de entrega intermediárias, essa melhoria foi de 26,99%. Já com as maiores folgas chegou a 34,40%. No subconjunto CA, a melhoria foi igual a 31,17%.

Para aprimorar a comparação entre os métodos propostos é utilizada uma técnica de Dolan & Moré (2002) denominada perfis de desempenho. Essa técnica usa uma representação compacta para comparação rápida entre diferentes métodos de resolução. O resultado obtido pelo método *s* no problema *p* é comparado com o melhor resultado para esse problema dentre os obtidos por cada um dos métodos em análise. Assim é usada uma relação de desempenho, dada por:

Tabela 3. Resultados das heurísticas construtivas com afinidade e flexibilidade.

Dimensão (*n*×*m*)	Atraso total médio (número de melhores soluções)		
	EDD-AAF	MDD-AAF	DMA-AF
5×5	187,7 (54)	**183,3** (51)	187,6 (48)
10×5	554,7 (43)	**543,6** (33)	546,8 (32)
10×10	587,2 (43)	**585,1** (39)	585,7 (38)
15×10	**866,1** (37)	876,1 (29)	881,4 (27)
15×15	1.130,1 (40)	**1.101,9** (36)	1.103,9 (36)
30×15	**2.750,6** (36)	2.816,4 (26)	2.825,8 (24)
50×5	17.355,3 (32)	17.266,9 (21)	**17.248,7** (22)
50×10	11.242,6 (25)	11.168,6 (27)	**11.142,6** (30)
50×15	**8.746,2** (36)	8.806,9 (24)	8.839,0 (17)
100×10	**64.681,6** (43)	65.638,3 (14)	65.642,4 (11)
Média (total)	**10.810,2** (389)	10.898,7 (300)	10.900,4 (285)
Tempo total (s)	4,2	4,6	4,4

Tabela 4. Melhorias obtidas ao explorar afinidade e flexibilidade.

Dimensão (*n*×*m*)	%*Dif*		
	EDD-AAF	MDD-AAF	DMA-AF
5×5	−9,98	−10,60	−8,50
10×5	−14,02	−15,02	−14,61
10×10	−12,14	−11,81	−12,50
15×10	−21,22	−20,32	−21,18
15×15	−11,53	−14,10	−14,09
30×15	−32,19	−31,02	−30,84
50×5	−20,81	−19,79	−19,65
50×10	−32,27	−31,86	−31,69
50×15	−38,97	−38,35	−37,20
100×10	−26,12	−25,11	−24,51
Geral	−27,13	−26,27	−25,75

Tabela 5. Melhorias obtidas em cada nível médio de flexibilidade.

Nível médio de flexibilidade	%*Dif*		
	EDD-AAF	MDD-AAF	DMA-AF
20	−13,42	−13,04	−12,43
50	−34,36	−33,41	−33,12
80	−34,65	−33,19	−32,54

$$\sigma_p(s) = \frac{f_p(s)}{\min\left\{f_p(s) : s \in S\right\}} \quad (26)$$

onde S é o conjunto de métodos. Então pode ser calculada a probabilidade $\rho_s(\tau)$ de que a relação de desempenho $\sigma_p(s)$ do método s esteja limitada a um fator $\tau \in \Re$, do seguinte modo:

$$\rho_s(\tau) = \frac{\left|\left\{p \in P : \sigma_p(s) \leq \tau\right\}\right|}{|P|} \quad (27)$$

onde $|P|$ indica o número de elementos do conjunto de problemas P. Desse modo, ρ_s: $\Re \rightarrow [0,1]$ é uma função de distribuição acumulada que expressa o desempenho relativo de s frente aos demais métodos na resolução dos problemas de P. Assim, dado um valor τ, $\rho_s(\tau)$ indica a porcentagem de problemas de P em que o método s consegue obter solução até τ vezes pior que a melhor solução alcançada. Com $\tau = 1$, ponto inicial da curva, o valor de $\rho_s(\tau)$ indica a porcentagem de melhores soluções do método s. Se para um determinado τ' tem-se $\rho_s(\tau') = 1$, significa que em 100% dos problemas o método s obtém resultado, no máximo, τ' vezes pior que o melhor dos resultados. A Figura 2 apresenta os perfis de desempenho da EDD-AAF, método com melhores resultados, e da EDD, para que se possa avaliar o uso das estratégias de afinidade e de flexibilidade. Para evitar divisões por zero foi adicionada uma unidade a cada $f_p(s)$. Para permitir melhor visualização o valor de τ foi limitado a 5.

É possível observar a dominância da regra que utiliza as estratégias de afinidade e de flexibilidade sobre a regra original. Considerando-se apenas essas duas regras, a EDD-AAF obtém melhor solução em mais de 90% das instâncias, extremo esquerdo da curva. Ela também exerce dominância sobre a EDD até a extremidade direita do gráfico. O desempenho de MDD-AAF e DMA-AF, quando comparado ao de MDD e DMA, respectivamente, é similar.

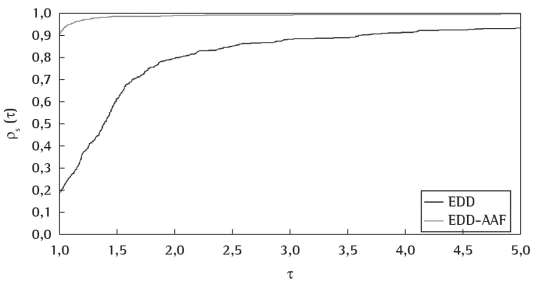

Figura 2. Perfis de desempenho de EDD e EDD-AAF.

4.2.4. *Comparações com resolução exata*

A avaliação final é feita ao comparar os resultados das regras com afinidade e flexibilidade com soluções ótimas ou limitantes superiores na Tabela 6. Para isso, o modelo matemático da seção 2 foi implementado em CPLEX 12.2 com limite de uma hora de processamento. Nos casos em que o CPLEX não encontrou a solução ótima, o limitante superior foi utilizado como atraso total. O *gap* médio obtido pela resolução exata está na terceira coluna, mas esse cálculo exclui *gaps* infinitos, casos em que o limitante inferior foi zero e o superior foi maior que zero. Subconjuntos com *gaps* infinitos excluídos são identificados por ∞. Também são apresentados os tempos médios de resolução. A tabela envolve apenas as dimensões para as quais o CPLEX conseguiu obter limitantes superiores no tempo estipulado.

Verifica-se que as heurísticas apresentaram, na média geral, melhores resultados que a resolução exata limitada a uma hora, apesar de atrasos mais elevados nas três menores dimensões. Esses atrasos mais elevados se devem ao fato de o método exato ter alcançado 109 soluções ótimas nessas três dimensões. Por outro lado, com a dimensão 15×10, todas as heurísticas propostas tiveram, na média, melhor desempenho que o CPLEX, o qual obteve oito soluções ótimas. Isso mostra como a complexidade do JSF torna os métodos heurísticos mais vantajosos, mesmo com dimensões não muito grandes. Nessa dimensão, as heurísticas obtiveram maior quantidade de melhores soluções que o CPLEX. Na dimensão 15×15 o método exato, mesmo após uma hora, não foi capaz de obter limitantes superiores para 18 das 60 instâncias do subconjunto, indicando suas limitações. Enquanto isso, as heurísticas resolveram até os problemas 100×10 gastando, em média, menos de 35ms e mostrando que são mais adequadas para problemas de grande porte.

Tabela 6. Comparação entre resultados exatos e heurísticos.

Dimensão ($n \times m$)	CPLEX			EDD-AAF		MDD-AAF		DMA-AF	
	Atraso total	*gap* (%)	Tempo (s)	Atraso total	Tempo (s)	Atraso total	Tempo (s)	Atraso total	Tempo (s)
5×5	120,3	0,00	1,68	187,7	0,000	183,3	0,000	187,6	0,000
10×5	404,6	0,00∞	1.984,00	554,7	0,000	543,6	0,000	546,8	0,000
10×10	550,8	603,51∞	3.053,41	587,2	0,001	585,1	0,001	585,7	0,001
15×10	1.248,7	3.866,45∞	3.382,77	**866,1**	0,001	876,1	0,001	881,4	0,001
Média	581,1	1.117,49	2.049,88	548,9	0,001	**547,0**	0,001	550,4	0,001

Nota: ∞ Subconjunto com *gaps* infinitos excluídos do cálculo do *gap* médio.

5. Conclusões

Este trabalho propôs estratégias baseadas na afinidade e na flexibilidade das operações para explorar características específicas do ambiente JSF minimizando o atraso total. Essas estratégias foram incorporadas a regras de prioridade clássicas e propostas. As regras cumpriram seu propósito, pois geraram soluções factíveis de qualidade aceitável, demandando tempo computacional reduzido. Entre as regras que não exploram as características do JSF, a regra proposta DMA conseguiu, em geral, melhor desempenho que as demais. Essa superioridade ocorreu especialmente nos problemas de maior porte.

Utilizar as estratégias de afinidade e de flexibilidade, além da antevisão, permitiu explorar características específicas de cada problema e reduzir o atraso médio geral em até 27,13%, sem aumento significativo do tempo computacional. A melhor regra proposta conseguiu superar a melhor regra da literatura em 81% dos problemas. Isso deixa claro o quão vantajoso pode ser utilizar conhecimento a respeito do problema.

Finalmente, os resultados envolvendo resolução exata indicaram que as regras e estratégias propostas são capazes de gerar soluções iniciais aceitáveis com rapidez. Tais soluções podem servir de ponto de partida para métodos de melhoria mais sofisticados.

Referências

Alvarez-Valdes, R., Fuertes, A., Tamarit, J., Giménez, G., & Ramos, R. (2005). A heuristic to schedule flexible job-shop in a grass factory. *European Journal of Operational Research*, *165*(2), 525-534. http://dx.doi.org/10.1016/j.ejor.2004.04.020

Baker, K. (1984). Sequencing rules and due-date assignments in a job shop. *Management Science*, *30*(9), 1093-1104. http://dx.doi.org/10.1287/mnsc.30.9.1093

Baker, K., & Bertrand, J. (1982). A dynamic priority rule for scheduling against due-dates. *Journal of Operations Management*, *3*(1), 37-42. http://dx.doi.org/10.1016/0272-6963(82)90020-1

Baykasoğlu, A., & Özbakir, L. (2010). Analyzing the effect of dispatching rules on the scheduling performance through grammar based flexible scheduling system. *International Journal Production Economics*, *124*(2), 369-381. http://dx.doi.org/10.1016/j.ijpe.2009.11.032

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, *41*(3), 157-183. http://dx.doi.org/10.1007/BF02023073

Brucker, P., & Schile, R. (1990). Job-shop scheduling with multi-purpose machines. *Computing*, *45*(4), 369-375. http://dx.doi.org/10.1007/BF02238804

Chan, F., Wong, T., & Chan, L. (2006). Flexible job-shop scheduling problem under resource constraints. *International Journal of Production Research*, *44*(11), 2071-2089. http://dx.doi.org/10.1080/00207540500386012

Chen, J., Chen, K., Wu, J., & Chen, C. (2008). A study of the flexible job shop scheduling problem with parallel machines and reentrant process. *International Journal of Advanced Manufacturing Technology*, *39*, 344-354. http://dx.doi.org/10.1007/s00170-007-1227-1

Dauzère-Pérès, S., & Paulli, J. (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, *70*, 281-306. http://dx.doi.org/10.1023/A:1018930406487

Dolan, E., & Moré, J. (2002). Benchmarking optimization software with performance profiles, *Mathematical programming*, *91*(2), 201-213. http://dx.doi.org/10.1007/s101070100263

Fattahi, P., Mehrabad, M., & Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, *18*, 331-342. http://dx.doi.org/10.1007/s10845-007-0026-8

Garey, M., Johnson, D., & Sethi, R. (1976). The complexity of flowshop and jobshop Scheduling. *Mathematics of Operations Research*, *1*(2), 117-129. http://dx.doi.org/10.1287/moor.1.2.117

Gholami, M., & Zandieh, M. (2009). Integrating simulation and genetic algorithm to schedule a dynamic flexible job shop. *Journal of Intelligent Manufacturing*, *20*, 481-498. http://dx.doi.org/10.1007/s10845-008-0150-0

Gutiérrez, C., & García-Magariño, I. (2011). Modular design of a hybrid genetic algorithm for a flexible job-shop scheduling problem. *Knowledge-Based Systems*, *24*, 102-112. http://dx.doi.org/10.1016/j.knosys.2010.07.010

Ho, N., Tay, J., & Lai, E. (2006). An effective architecture for learning and evolving flexible job-shop schedules. *European Journal of Operational Research*, *179*, 316-333. http://dx.doi.org/10.1016/j.ejor.2006.04.007

Kacem, I., Hammadi, S., & Borne, P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics*, *32*(1), 1-13.

Kim, Y. (1995). A backward approach in list scheduling algorithms for multi-machine tardiness problems. *Computers & Operations Research*, 22(3), 307-319. http://dx.doi.org/10.1016/0305-0548(94)E0019-4

Koulamas, C. (1994). The total tardiness problem: review and extensions. *Operations Research*, *42*(6), 1025-1041. http://dx.doi.org/10.1287/opre.42.6.1025

Li, J., Pan, Q., & Liang, Y. (2010). An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, *59*, 647-662. http://dx.doi.org/10.1016/j.cie.2010.07.014

Mainieri, G., & Ronconi, D. (2013). New heuristics for total tardiness minimization in a flexible flowshop. *Optimization Letters*, *7*(4), 665-684. http://dx.doi.org/10.1007/s11590-012-0448-x

Özgüven, C., Özbakir, L., & Yavuz, Y. (2010). Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling*, *34*, 1539-1548. http://dx.doi.org/10.1016/j.apm.2009.09.002

Panwalkar, S., & Iskander, W. (1977). A survey of scheduling rules. *Operations Research*, *25*(1), 45-61. http://dx.doi.org/10.1287/opre.25.1.45

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, *35*(10), 3201-3212. http://dx.doi.org/10.1016/j.cor.2007.02.014

Scrich, C. (1997). *Busca Tabu para a programação de tarefas em job shop com datas de entrega* (Tese de doutorado). Universidade Estadual de Campinas, Campinas.

Scrich, C., Armentano, V., & Laguna, M. (2004). Tardiness minimization in a flexible job shop: A tabu search approach. *Journal of Intelligent Manufacturing*, *15*, 103-115. http://dx.doi.org/10.1023/B:JIMS.0000010078.30713.e9

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*(2), 278-285. http://dx.doi.org/10.1016/0377-2217(93)90182-M

Tay, J., & Ho, N. (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering*, *54*, 453-473. http://dx.doi.org/10.1016/j.cie.2007.08.008

Vepsalainen, A., & Morton, T. (1987). Priority rules for job shop with weighted tardiness costs. *Management Science*, *33*(8), 1035-1047. http://dx.doi.org/10.1287/mnsc.33.8.1035

Vilcot, G., & Billaut, J. (2008). A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem. *European Journal of Operational Research*, *190*, 398-411. http://dx.doi.org/10.1016/j.ejor.2007.06.039

Zhang, G.,Shao, X., Li, P., & Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, *56*, 1309-1318. http://dx.doi.org/10.1016/j.cie.2008.07.021

Zhang, H., & Gen, M. (2005). Multistage-based genetic algorithm for flexible job-shop scheduling problem. *Complexity International*, *11*, 223-232.

Agradecimentos

Os autores são gratos aos revisores anônimos pelos seus úteis comentários e sugestões. 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 – Processo 477203/2012-4) e da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP – Processos 2010/10133-0 e 2013/07375-0).

Efficient priority rules that explore flexible job shop characteristics for minimizing total tardiness

Abstract

This paper presents heuristic strategies that exploit characteristics of the Flexible Job Shop (FJS) environment, an extended version of the NP-hard job-shop problem. The FJS involves a set of jobs composed of operations, and each operation must be processed on a machine that can process it. The criterion is the minimization of total tardiness. Initially, characteristics related to the production system flexibility or, more precisely, characteristics related to the machines that can process each operation and the machines' processing times are identified. Therefore, rules that explore these characteristics and foresee future states of the system are proposed. Computational experiments are conducted with 600 instances. Comparisons with rules from the literature show that the best heuristic proposed outperforms the best known rule in approximately 81 percent of instances.

keywords

Job shop. Heuristic. Mathematical programming. Production scheduling.