HEURÍSTICAS CONSTRUTIVAS PARA A MINIMIZAÇÃO DO ATRASO TOTAL NO AMBIENTE *JOB SHOP* FLEXÍVEL

Everton Luiz de Melo
Universidade de São Paulo – Escola Politécnica
Av. Almeida Prado, 128, Cidade Universitária – São Paulo-SP – Brasil
everton.melo@usp.br

Débora Pretti Ronconi
Universidade de São Paulo – Escola Politécnica
Av. Almeida Prado, 128, Cidade Universitária – São Paulo-SP – Brasil
dronconi@usp.br

RESUMO

O ambiente de produção abordado neste trabalho é o *Job Shop* Flexível (JSF), uma generalização do problema NP-*Hard Job Shop*. O JSF é composto por um conjunto de tarefas constituídas por sequências de operações. Cada operação deve ser processada em uma máquina dentre aquelas capazes de processá-la. A medida de desempenho considerada é a minimização do atraso total. Novas heurísticas construtivas são propostas e implementadas, bem como adaptações de heurísticas da literatura. Testes computacionais utilizando um novo conjunto de instâncias realistas do problema são apresentados e os resultados obtidos pelas heurísticas propostas são comparados com os fornecidos pelas heurísticas adaptadas. Além disso, são realizadas comparações entre as soluções heurísticas e as obtidas através de um *software* comercial conhecido. Os resultados mostram que as heurísticas propostas apresentam melhor desempenho que as heurísticas da literatura, além de serem competitivas com a resolução exata limitada a uma hora de processamento.

PALAVRAS-CHAVE: *job shop* flexível, heurísticas construtivas, atraso total.

Área principal: PO na Administração & Gestão da Produção

ABSTRACT

This work addresses the production environment Flexible Job Shop (FJS), a generalization of the NP-Hard Job Shop problem. The FJS is composed of a set of jobs, each one with an operations sequence. Each operation must be processed on a machine among those able to process it. The performance measure is the minimization of the total tardiness. New constructive heuristics are proposed and implemented, as well as adaptations of literature heuristics. Computational tests using a new set of realistic instances are presented and comparisons among the heuristics are made. In addition, there are comparisons between the heuristics solutions and a known commercial software solutions. The results show that the proposed heuristics have better performance than the literature heuristics and they are competitive with exact resolution limited to an hour of execution time.

KEYWORDS: flexible job shop, constructive heuristics, total tardiness.

Main area: OR in Administration & Production Management

1 Introdução

O foco deste trabalho é o desenvolvimento de heurísticas construtivas para a versão flexível do problema de programação de tarefas *Job Shop* (JS). O JS consiste de um conjunto de tarefas, designadas por *jobs*, cada um formado por uma sequência de operações. Essas operações devem ser alocadas a máquinas otimizando algum critério. O *Job Shop* Flexível (JSF) difere do JS clássico por possuir operações que podem ser processadas alternativamente em mais de uma máquina. Minimizando o *makespan*, o JS é classificado por Garey *et al*. (1976) como NP-*Hard* e isso implica que não são conhecidos algoritmos exatos que o resolvam em tempo computacional aceitável. Por tal razão métodos alternativos, como heurísticas, são utilizados para gerar soluções de boa qualidade em tempo computacional razoável. Neste estudo o critério de desempenho considerado é a minimização do atraso total. O JSF é um problema de grande aplicabilidade prática como mostram Alvarez-Valdes *et al*. (2005) ao utilizar métodos heurísticos construtivos em um problema da indústria de vidro e Vilcot e Billaut (2008) ao usar Busca Tabu (BT) e Algoritmos Genéticos (AG) na indústria de impressão. Contudo a literatura sobre atraso total é escassa, sendo que o *makespan* é o critério de desempenho majoritariamente explorado.

Formalmente o JSF pode ser definido conforme segue. Considere um conjunto independente de *n jobs*, cada qual constituído por uma sequência específica de operações. Considere também um conjunto de *m* máquinas. A cada uma das operações está associado um subconjunto de máquinas capazes de realizar seu processamento, o que confere um caráter flexível ao problema. Cada operação deve ser processada individualmente e sem interrupção em uma única máquina pertencente ao subconjunto de máquinas a ela associado. Devem ser obtidas as atribuições de operações às máquinas e a sequência de processamentos que minimizem o critério estabelecido. A flexibilidade do problema torna possível um *job* ser processado através de diversos caminhos alternativos, o que aumenta significativamente o número de soluções factíveis. Desse modo as abordagens de resolução do JSF se distinguem pela decomposição ou não do problema em subproblemas menos complexos que podem ser atacados isoladamente em duas fases. A primeira dessas fases é a atribuição de cada uma das operações à uma das máquinas habilitadas a realizar seu processamento. Essa etapa é denominada roteamento, pois as atribuições determinam os caminhos que os *jobs* percorrerão através das máquinas. A segunda fase é o sequenciamento das operações nas máquinas, o que equivale à resolução do JS resultante da primeira fase. Essa estratégia de resolução é denominada abordagem hierárquica. Em contraste com essa estratégia existe a abordagem integrada, na qual se busca obter soluções para o JSF tratando simultaneamente atribuições e sequenciamentos.

A literatura acerca do JSF envolve diversos critérios de otimização e diferentes métodos de resolução, especialmente heurísticos. Brucker *et al*. (1990) está entre os primeiros trabalhos a mencionar o JSF. Nele é proposto um algoritmo de complexidade polinomial para o JSF limitado a dois *jobs*. Brandimarte (1993) é o primeiro a utilizar a abordagem hierárquica empregando regras de prioridade e BT. Kacem *et al.* (2002) classificam os problemas de JSF em JSF com flexibilidade total e JSF com flexibilidade parcial de acordo com o grau de sua flexibilidade. Na flexibilidade total cada operação pode ser executada em qualquer máquina e na flexibilidade parcial cada uma das operações está associada a um subconjunto de máquinas. Diversos trabalhos utilizam meta-heurísticas. AG são usados por Kacem *et al*. (2002), Zhang e Gen (2005), Chan *et al*. (2006), Ho *et al*. (2006), Pezzella *et al*. (2008) e Gutiérrez e García-Magariño (2011). A meta-heurística BT é empregada por Dauzère-Pérès e Paulli (1997). Zhang *et al*. (2009) hibridizam BT com *Particle Swarm Optimization* (PSO). Li *et al*. (2010) desenvolvem um método híbrido que envolve BT e *Variable Neighborhood Search* (VNS). Regras de prioridade são usadas por Tay e Ho (2008) que as aliam a *Genetic Programming* (GP). Baykasoğlu e Özbakir (2010) avaliam a influência de diversas regras de prioridade em problemas com diferentes níveis de flexibilidade no atraso médio.

Foram encontrados três trabalhos que propõem formulações matemáticas para o JSF com o objetivo de minimizar o *makespan*. Fattahi *et al*. (2007) propõem um modelo matemático e um conjunto de instâncias resolvidas por BT e *Simulated Annealing* (SA). Özgüven *et al*. (2010) apresentam outro modelo matemático para a minimização do *makespan* que é confrontado

com o proposto por Fattahi *et al*. (2007) e apresenta melhor desempenho. Birgin *et al*. (2011) propõem um modelo novo para o JSF, além de uma adaptação do modelo de Özgüven *et al*. (2010).

Com relação ao atraso total, Scrich *et al*. (2004) é, de acordo com o levantamento realizado neste estudo, o primeiro trabalho a considerar tal critério. Soluções iniciais são geradas pela aplicação de regras de prioridade e aprimoradas por BT. Dentre as regras utilizadas Scrich *et al*. (2004) indicam a *Modified Due Date* (MDD) como a que apresentou os melhores resultados. Uma aplicação do JSF minimizando atraso na indústria de armamentos militares é apresentada por Chen *et al*. (2008). Nesse caso o problema abordado permite que as operações dos *jobs* tenham sequências alternativas e os *jobs* podem ter relações de precedência entre si. O método de resolução hierárquica envolve uma etapa de seleção de máquinas e outra de sequenciamento, ambas baseadas na regra *Earliest Due Date* (EDD). Outras regras também são usadas para desempate. Gholami e Zandieh (2009) incorporam simulação num método baseado em AG. As soluções geradas são submetidas a repetidas simulações que envolvem probabilidades de quebra de máquinas. Com isso são obtidos o atraso médio e o *makespan* esperados.

Na pesquisa realizada foram encontrados poucos trabalhos que tratam da minimização do atraso total, o que reforça a existência de uma lacuna a respeito desse critério. Assim a proposta deste trabalho é propor novas regras de prioridade que permitam a obtenção de melhores resultados para o JSF com o critério de minimizar o atraso total. Na próxima Seção são descritas as heurísticas propostas. A Seção 3 traz os resultados obtidos com as instâncias geradas aleatoriamente e com algumas instâncias da literatura. Por fim, na Seção 4 são apresentadas as conclusões do trabalho.

2 Heurísticas para o *job shop* flexível

Pelo fato do JSF ser um dos problemas mais complexos da literatura, métodos simples e efetivos são empregados para a geração de soluções. Um desses métodos é o *list scheduling algorithm*. Sempre que uma máquina se torna ociosa o método seleciona para ocupá-la a operação de maior prioridade dentre aquelas que pode processar. A definição da operação prioritária se dá através de uma regra de prioridade. De acordo com Panwalkar e Iskander (1977) tais regras estão fortemente ligadas ao critério de desempenho que se busca. Na sequência são apresentadas as regras de prioridade para minimizar o atraso total implementadas neste trabalho.

2.1 Regras de prioridade adaptadas da literatura

Foram selecionadas e implementadas regras de prioridade que tiveram bom desempenho na minimização do atraso em outros trabalhos da literatura, não necessariamente sobre o JSF. Para que funcionassem com o JSF foram feitas algumas adaptações. As regras conhecidas da literatura implementadas foram: *Shortest Processing Time* (SPT), encontrada em Baker (1984); *Minimum Slack Time* (MST); *Smallest Critical Ratio* (SCR); EDD, explorada em Vepsalainen e Morton (1987) juntamente com as duas anteriores; MDD, de Baker e Bertrand (1982); *Modified Operation Due Date* (MOD), de Baker (1984); e *Priority Rule for Total Tardiness* adaptada (PRTTa), apresentada por Mainieri e Ronconi (2012).

As regras SPT, MST, SCR e EDD foram aplicadas diretamente ao JSF. No caso da regra MDD é necessário estimar a data de entrega modificada a partir do tempo de processamento das operações restantes do *job*. Para isso seria preciso saber em que máquina cada operação seria processada, uma informação ainda não disponível durante a resolução. Assim, para cada operação disponível mas ainda não alocada, foi considerado o tempo de processamento mínimo entre as máquinas habilitadas. A data de entrega modificada, d_i', é obtida com a equação a seguir:

$$d_i' = \max(d_i, t + \sum_{j \in G_i} \min_{k \in M_j} t_{jk}) \quad (1)$$

onde d_i é a data de entrega do *job* i; t é o instante atual considerado pelo *list scheduling algorithm*; G_i é o conjunto das operações ainda não alocadas do *job* i; M_j é o conjunto de máquinas habilitadas para a operação j; e t_{jk} é o tempo de processamento da operação j na máquina k.

Na MOD o cálculo das datas de entrega das operações considerou seus tempos médios nas diversas máquinas alternativas. Esses mesmos tempos médios foram usados na PRTTa para as datas de entrega das operações que substituíram as datas de entrega dos estágios do *flowshop* flexível, o ambiente original da regra.

2.2 Estratégias e regras de prioridade propostas

A seguir são apresentadas as regras de prioridade desenvolvidas neste trabalho. Algumas derivam de regras conhecidas da literatura. Também são propostas estratégias que visam conduzir os passos das heurísticas construtivas de forma conveniente.

2.2.1 Grau de flexibilidade das operações

Um fator que pode ser considerado pelas heurísticas ao selecionar a operação que ocupará uma máquina é o grau de flexibilidade das operações candidatas. O intuito dessa estratégia é favorecer operações com menor grau de flexibilidade, já que essas possuem possibilidades de alocação mais restritivas. A flexibilidade de uma operação é fornecida por $1/|M_j|$, onde $|M_j|$ indica o número de máquinas habilitadas para a operação j.

2.2.2 Afinidade entre operações e máquinas

Considerar a afinidade existente entre operações e máquinas tem o intuito de direcionar cada operação para a máquina que pode efetuar seu processamento em menor tempo. A afinidade representa a conveniência de se atribuir a operação j à máquina k e é expressa pela relação:

$$t_{jk} \Big/ \sum_{w \in M_j} t_{jw} \qquad (2)$$

onde w é um índice de máquinas. Essa relação varia de 0 a 1 e quanto menor seu valor, maior a afinidade. A afinidade também pode ser usada como uma estratégia associada a uma regra de prioridade.

2.2.3 Regras compostas propostas

A primeira regra proposta mescla as regras PRTTa e MDD, sendo denominada *Composite Rule* (CR). Ela é semelhante à regra PRTTa mas em vez de utilizar datas de entrega por operação utiliza a data de entrega modificada do *job*, d_i', fornecida pela MDD. Nela se considera:

r_j instante de liberação da operação j;
α constante que pondera o primeiro termo da regra.

A regra então pode ser usada para calcular o valor CR(j,t) que determina a prioridade da operação j, no instante t, da maneira que segue:

$$\text{CR}(j,t) = \alpha \cdot \max(r_j, t) + \max[\max(r_j, t) + t_{jk}, d_i'] \qquad (3)$$

Como na PRTTa, o primeiro termo da regra, com α>0, busca penalizar os *jobs* que não estão disponíveis no instante t. O segundo termo visa, ao mesmo tempo, considerar a priorização da data de entrega da operação em análise e penalizar operações não disponíveis. A CR considera adicionalmente como candidatas as operações cujas predecessoras já estejam alocadas, embora não finalizadas. A regra antevê em que instante no futuro cada operação se tornará disponível e faz seu instante de liberação r_j igual ao instante em que sua operação predecessora será concluída. Operações disponíveis em t, aquelas cujas predecessoras já estejam concluídas, têm $r_j \leq t$. Operações que possuem predecessoras ainda não alocadas não entram na lista de operações candidatas. É selecionada a operação com menor valor CR(j,t).

Uma derivação dessa regra proposta associa a CR à afinidade e considera o grau de flexibilidade das operações em análise. O objetivo é tentar induzir o *list scheduling algorithm* a atribuir cada operação a uma das máquinas que a processem com maior rapidez, além de favorecer operações com menor número de máquinas alternativas. Essa regra é denominada *Composite Rule with Affinity and Flexibility* (CRAF). A regra proposta incorpora as estratégias relacionas a afinidade e a flexibilidade do seguinte modo:

$$\text{CRAF}(j,t) = \text{CR}(j,t) \cdot \left[1 + \beta \cdot \left(\frac{t_{jk}}{\sum_{w \in M_j} t_{jw}} - \frac{1}{|M_j|} \right) \right] \quad (4)$$

Nessa equação os valores da afinidade e da flexibilidade estão limitados ao intervalo [0,1]. A diferença entre afinidade e flexibilidade é multiplicada por uma constante β, cuja função é ponderar a importância da afinidade e da flexibilidade na CRAF. O resultado dessa multiplicação é somado a 1 para evitar valores negativos. Então se pode multiplicar o termo entre colchetes pelo valor fornecido pela CR sem descaracterizar a regra original.

2.3 Heurística Construtiva

Baseada nas regras de prioridade mencionadas, foi implementada uma heurística construtiva que busca obter soluções viáveis de qualidade razoável com rapidez. A heurística construtiva aqui utilizada é derivada das apresentadas por Scrich (1997) e Mainieri e Ronconi (2012) e utiliza a abordagem de resolução integrada.

O método tem início no instante igual a zero, no qual todas máquinas estão disponíveis e todas operações são conhecidas. À medida que o tempo avança, as operações são atribuídas às máquinas, até que todas as operações estejam alocadas. Uma alocação é definida a cada vez que uma máquina se torna disponível. Nesse momento são levantadas todas as operações candidatas para ocupar a máquina que se tornou vaga, sendo selecionada a operação de maior prioridade, conforme a regra de prioridade vigente. Neste trabalho a operação de maior prioridade sempre é aquela que permita à regra gerar o menor valor. Se houver mais de uma máquina disponível no mesmo instante, a primeira máquina a receber uma operação é a que possuir o menor carregamento estimado.

No caso das regras PRTTa, CR e CRAF operações ainda não liberadas podem ser alocadas, causando ociosidade na máquina. Caso isso ocorra, a heurística construtiva tenta preencher esse tempo ocioso com operações que possam ser processadas nessa máquina sem interferir no instante de início de processamento da operação não liberada que foi alocada. As operações a serem inseridas no tempo ocioso são aquelas com menor instante de liberação, r_j.

A heurística construtiva proposta permite a geração de soluções para o JSF através de diferentes regras de prioridade. De acordo com a regra de prioridade em uso, a operação prioritária a ser alocada à máquina pode variar. Desse modo se pode considerar que o método apresentado fornece diferentes possibilidades de se obter soluções para o problema.

3 Experimentos Computacionais

A seguir é feita a descrição das instâncias utilizadas nos experimentos, tanto as geradas aleatoriamente quanto as obtidas na literatura. Os resultados obtidos pelas heurísticas e pela resolução exata são apresentados e analisados.

3.1 Instâncias Utilizadas

A geração de instâncias aleatórias se baseou em Scrich (1997) e Scrich *et al*. (2004). Com relação aos níveis de flexibilidade, em vez de se ter o mesmo número de máquinas alternativas para todas as operações, optou-se por trabalhar com faixas de flexibilidade. Todos os valores aleatórios utilizados são fornecidos pelo gerador de Taillard (1993), o qual permite a obtenção de valores inteiros em uma distribuição uniforme $U[a,b]$. Cada instância gerada possui 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;
$\bar{t}_j$ tempo médio de processamento da operação j, considerando 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.

Tabela 1: Valores dos componentes das instâncias geradas.

Componente	Valores utilizados
n	5, 10, 15, 30, 50, 100
m	5, 10, 15
$\|O_i\|$	$U[m/2, m]$
$\|M_j\|$	$U[\lceil 0{,}1 \cdot m \rceil; \lceil 0{,}3 \cdot m \rceil]$; $U[\lceil 0{,}3 \cdot m \rceil; \lceil 0{,}7 \cdot m \rceil]$; $U[\lceil 0{,}6 \cdot m \rceil; m]$
M_j	$\|M_j\|$ máquinas sorteadas utilizando $U[1, m]$
t_{jk}	$U[1, 99]$, para $k=1$; e $U[t_{j1}, 3 \cdot t_{j1}]$, para $1<k\leq\|M_j\|$, com $t_{jk}\leq 99$
δ	Se $n \geq m$, $\delta=(n/m)^{1/2}$, senão $\delta=1$
d_i	$0{,}8 \cdot \delta \cdot \sum_{j=1}^{\|O_i\|} \bar{t}_j$; $1{,}0 \cdot \delta \cdot \sum_{j=1}^{\|O_i\|} \bar{t}_j$; $1{,}2 \cdot \delta \cdot \sum_{j=1}^{\|O_i\|} \bar{t}_j$; CA

Os valores de n e m são definidos previamente para cada instância. $|O_i|$ varia de $m/2$ a m. O tempo de processamento da operação j na sua primeira máquina alternativa, t_{j1}, é obtido do intervalo [1,99]. Nas demais máquinas os tempos são limitados a intervalos compreendidos em $[t_{j1}, 3 \cdot t_{j1}]$, limitados também a 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 somente uma dessas três faixas com o intuito de obter instâncias com diferentes $|M_j|$ para cada operação j, sendo os elementos de M_j obtidos aleatoriamente em $U[1,m]$. Em Scrich *et al*. (2004) todas operações de uma mesma instância possuem exatamente a mesma quantidade de máquinas alternativas. Outra diferença em relação à Scrich *et al*. (2004) envolve as datas de entrega. Elas são 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 $\delta=1$ se $n<m$, e $\delta=(n/m)^{1/2}$, caso contrário. Esse cálculo de δ 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 uma combinação aleatória desses três valores, CA.

Foram geradas instâncias com dimensões expressas pelas seguintes relações $n \times m$: 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 10 combinações foi gerado um grupo de 5 instâncias, formando um subconjunto de 50 instâncias. Como existem 3 faixas de flexibilidade e 4 tipos de data de entrega, foram criados 12 subconjuntos de 50 instâncias, cada qual com uma combinação distinta de faixa de flexibilidade e opção de data de entrega. Posteriormente essas combinações poderão ser observadas na Tabela 4. No total foram geradas 600 instâncias para o JSF.

As instâncias da literatura foram propostas por Brandimarte (1993) para o JSF. São 5 instâncias para o critério de desempenho atraso ponderado, denominadas WT1 a WT5. Nos experimentos deste trabalho o fator de ponderação para cada *job* não foi considerado, já que o objetivo é minimizar o atraso total. Então tais instâncias passam a ser chamadas WT1' a WT5'.

3.2 Resultados

Os experimentos foram realizados em um computador com processador Intel Core i3 350M de 2,26GHz, 4GB de RAM e sistema operacional Windows 7 de 64 bits. As heurísticas foram implementadas em linguagem C. Após testes preliminares a constante de ponderação α foi fixada em 13 na PRTTa e em 91 na CR.

Os resultados obtidos com as heurísticas construtivas básicas são apresentados na Tabela 2. Essas heurísticas são chamadas básicas pois não incorporam as estratégias de afinidade e flexibilidade das operações. São elas: SPT, MST, SCR, EDD, MDD, MOD, PRTTa e CR. Cada linha da Tabela 2 equivale a um conjunto de 60 instâncias. Os resultados são estratificados pelas

dimensões das instâncias, permitindo verificar como cada método funcionou para diferentes combinações $n \times m$, explicitadas na primeira coluna. Da segunda coluna em diante é apresentada a média do atraso total, T_{total}, que cada regra de prioridade obteve para o respectivo conjunto de 60 instâncias. Imediatamente abaixo desse valor é apresentado, entre parênteses, o número de vezes em que essa mesma regra foi capaz de obter o menor atraso total dentre os obtidos por todas heurísticas da tabela. Caso duas ou mais regras empatem com a melhor solução, cada uma delas soma um melhor resultado.

Tabela 2: Resultados das heurísticas construtivas.

Dimensão ($n \times m$)	**T_{total} (Número de melhores soluções)**							
	SPT	**MST**	**SCR**	**EDD**	**MDD**	**MOD**	**PRTTa**	**CR**
5×5	**190,0** (35)	231,6 (17)	229,1 (15)	208,5 (25)	205,0 (24)	**190,0** (36)	192,7 (32)	205,0 (24)
10×5	715,5 (9)	842,5 (6)	722,8 (6)	645,2 (12)	639,7 (10)	708,2 (6)	**625,8** (22)	643,0 (9)
10×10	593,4 (22)	662,4 (10)	655,4 (9)	668,3 (10)	663,4 (6)	593,7 (18)	**560,5** (33)	661,6 (8)
15×10	**2129,3** (12)	2677,0 (1)	2315,3 (5)	2212,0 (6)	2242,8 (6)	2163,6 (9)	2129,4 (20)	2268,5 (4)
15×15	1201,1 (21)	1346,3 (14)	1314,9 (11)	1277,3 (16)	1282,7 (17)	1227,3 (14)	**1193,6** (27)	1287,3 (13)
30×15	5383,2 (0)	5786,2 (6)	4645,9 (3)	**4056,5** (12)	4082,7 (10)	4948,3 (2)	4304,2 (21)	4132,3 (6)
50×5	25883,2 (3)	29293,6 (0)	22506,9 (4)	21915,2 (8)	21526,4 (26)	27471,7 (0)	25141,6 (0)	**21439,6** (28)
50×10	22187,1 (0)	25302,1 (0)	17806,6 (2)	16598,1 (12)	16390,9 (22)	22976,4 (0)	20289,8 (0)	**16358,8** (24)
50×15	21450,7 (0)	28009,3 (0)	18807,6 (3)	16742,2 (10)	16553,9 (18)	22357,7 (0)	20741,4 (0)	**16430,5** (29)
100×10	118898,5 (0)	123230,3 (0)	89425,6 (5)	87544,1 (15)	87643,4 (15)	123985,4 (0)	111861,1 (0)	**86945,0** (25)
Média (Total)	19863,2 (102)	21738,1 (54)	15843,0 (63)	15186,7 (126)	15123,1 (154)	20662,2 (85)	18704,0 (155)	**15037,2** (170)
Tempo total (s)	5,7	6,7	7,0	5,5	6,9	5,7	6,0	9,6

Considerando somente as regras básicas, CR, MDD e EDD se destacaram. Com atraso total médio geral igual a 15037,2 a CR foi a melhor regra. Em seguida aparece a MDD, regra com melhores resultados em Scrich *et al*. (2004), com média geral 15123,1. A EDD ficou em terceiro lugar com atraso médio 15186,7. Se observa que a CR também foi a regra que mais vezes obteve a melhor solução, 170 ocasiões. A segunda regra com maior quantidade de melhores soluções foi a PRTTa com 155 melhores resultados, apesar de seu atraso médio geral ter sido 18704,0. Isso pode ser explicado ao se notar que nas instâncias de menor porte, com até 30 *jobs*, a PRTTa foi uma das melhores regras, tendo conseguido menor atraso nas dimensões 10×5, 10×10 e 15×15. A PRTTa também foi a regra com maiores quantidades de melhores soluções em todas dimensões com menos de 30 *jobs*, exceto em 5×5. Porém cabe observar que somente nas instâncias com 30 *jobs* ou menos a PRTTa obteve os menores atrasos. Nas demais instâncias o desempenho da PRTTa ficou aquém das demais regras. Uma justificativa para isso pode ser o crescimento da falta de precisão das datas de entrega estimadas por operação à medida que os problemas crescem. Um comportamento similar foi observado com a MOD, outra regra que estima datas de entrega para operações. Com relação às instâncias de maior porte, com 50 *jobs* ou mais, a CR apresentou um melhor desempenho. Obteve atraso menor em todas essas

dimensões e também obteve maior número de vitórias. Sobre o tempo computacional se constata que as heurísticas foram rápidas, gastando poucos segundos para resolver todas as 600 instâncias.

A partir dos resultados da Tabela 2 foram identificadas as três regras que geraram os melhores resultados: CR, MDD e EDD. Então foram propostas regras compostas que utilizam as regras básicas EDD e MDD associadas à afinidade e à flexibilidade das operações. O princípio é o mesmo utilizado na elaboração da CRAF. A diferença é que o termo entre colchetes da Equação 4 passa a multiplicar o valor fornecido pela EDD, equivalente à data de entrega original, ou pela MDD, equivalente à data de entrega modificada da Equação (1), em vez de multiplicar o valor fornecido pela CR. Essas regras são denominadas EDD com Afinidade e Flexibilidade (EDDAF) e MDD com Afinidade e Flexibilidade (MDDAF). Após experimentos iniciais a constante β foi fixada em 0,1 na CRAF, enquanto na EDDAF e na MDDAF foi fixada em 5,0. Os resultados obtidos com essas novas regras são apresentados na Tabela 3.

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

Dimensão (n×m)	T_{total} **(Número de melhores soluções)**		
	EDDAF	**MDDAF**	**CRAF**
5×5	202,9 (50)	**197,1** (53)	209,5 (44)
10×5	550,7 (42)	**546,0** (32)	568,2 (24)
10×10	616,7 (28)	624,1 (29)	**604,6** (34)
15×10	**1853,1** (34)	1894,0 (19)	2016,3 (10)
15×15	1228,7 (28)	1217,8 (23)	**1145,2** (38)
30×15	3158,6 (17)	3189,1 (16)	**3106,2** (31)
50×5	17699,2 (27)	**17527,6** (27)	18236,3 (14)
50×10	**11803,5** (32)	11818,6 (19)	12000,2 (9)
50×15	12107,9 (27)	12260,0 (10)	**12107,0** (23)
100×10	**66190,0** (46)	67203,9 (14)	70332,7 ()
Média (Total)	**11541,1** (331)	11647,8 (242)	12032,6 (227)
Tempo total(s)	5,6	7,1	9,7

É possível observar que a consideração da afinidade e da flexibilidade das operações permitiu que as regras básicas melhorassem consideravelmente seus resultados. A incorporação das estratégias relacionadas à afinidade e à flexibilidade permitiu que os resultados da CRAF fossem 19,98% melhores que os da CR e que a MDDAF gerasse atrasos 22,97% inferiores ao da MDD. Já a inclusão dessas estratégias à EDD possibilitou as maiores melhorias. A EDDAF obteve redução de 24% do atraso total em relação à EDD. A análise das diferentes dimensões de problemas permite observar que em 4 das 10 dimensões a CRAF conseguiu melhor atraso total, enquanto EDDAF e MDDAF geraram menor média em 3 dimensões cada. No geral a EDDAF foi a regra com melhores resultados, alcançando atraso total igual a 11541,1. Na sequência aparece a MDDAF, com 11647,8, e CRAF, com 12032,6. Com relação ao número de melhores resultados a EDDAF foi superior em 331 instâncias, seguida da MDDAF com 242. A CRAF ficou com 227 melhores resultados. A Tabela 4 apresenta os resultados desses três métodos estratificados por tipo de data de entrega. Além disso as médias são separadas de acordo com o

grau de flexibilidade de cada conjunto de instâncias, conforme a segunda coluna. Cada linha equivale a um dos 12 subconjuntos mencionados na geração das instâncias.

Tabela 4: Resultados de EDDAF, MDDAF e CRAF em diferentes graus de flexibilidade.

Multiplicador	Grau de flexibilidade	T_{total}		
		EDDAF	MDDAF	CRAF
0,8	20	20362,0	**20087,2**	20439,0
	50	**15619,0**	15998,4	16458,7
	80	**15987,5**	16560,6	16941,2
1,0	20	14606,7	**14589,3**	14834,0
	50	**10129,5**	10313,4	10778,6
	80	**10466,6**	10775,1	11291,1
1,2	20	9738,8	**9668,7**	10048,2
	50	**5769,3**	5843,5	6374,7
	80	**6271,8**	6379,3	6778,2
CA	20	12755,3	**12618,2**	12966,8
	50	**8216,5**	8247,8	8531,2
	80	**8570,3**	8692,3	8949,7

Esses resultados mostram que no menor grau de flexibilidade, 20%, a MDDAF foi sempre a regra com os melhores resultados, independente do tipo de data de entrega. Já nas instâncias com grau de flexibilidade 50% e 80% a EDDAF obteve melhores resultados.

Uma comparação complementar pode ser realizada através de perfis de desempenho. Essa técnica de Dolan e Moré (2002) usa uma representação compacta que permite rápida comparação 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:

$$\sigma_{p,s} = \frac{T_{p,s}}{\min\{T_{p,s} : s \in S\}} \quad (5)$$

onde $T_{p,s}$ é o atraso total obtido pelo método *s* no problema *p*, e *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:

$$\rho_s(\tau) = \frac{\left|\{p \in P : \sigma_{p,s} \leq \tau\}\right|}{|P|} \quad (6)$$

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' se tem $\rho_s(\tau')=1$, significa que em 100% dos problemas o método *s* obtém resultado, no máximo, τ' vezes pior que o melhor dos resultados. A Figura 1 apresenta os perfis de desempenho dos métodos EDD, MDD, CR, EDDAF, MDDAF e CRAF para que se possa avaliar o uso das estratégias de afinidade e de flexibilidade. Para permitir melhor visualização o valor de τ foi limitado a 4.

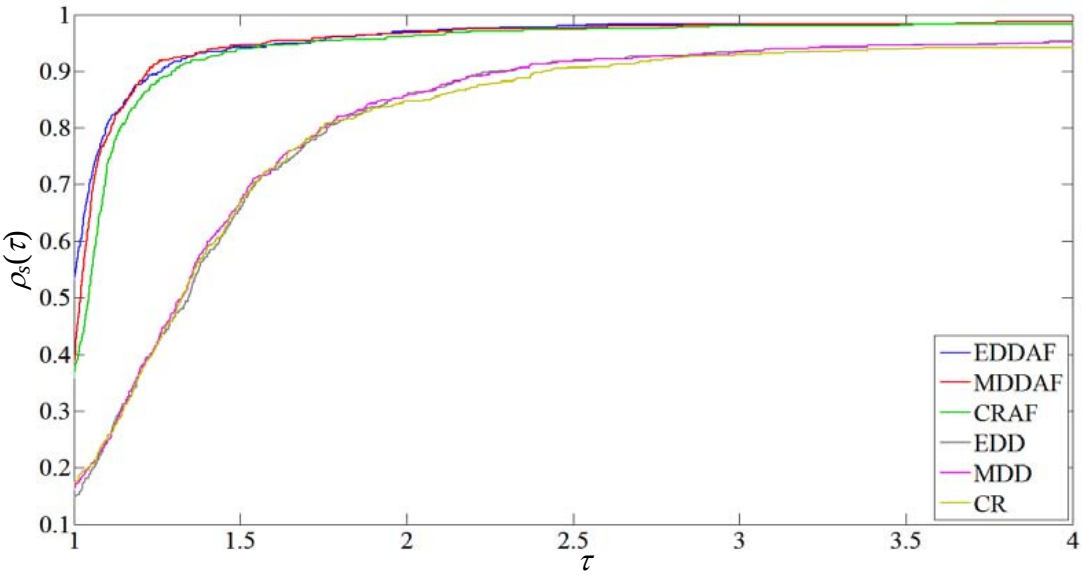

Figura 1: Perfis de desempenho das melhores regras.

É possível observar a dominância das regras que utilizam as estratégias de afinidade e de flexibilidade das operações sobre as demais. Considerando apenas as seis regras, a EDDAF obtém melhor solução em mais de 50% das instâncias, como indica o extremo esquerdo de sua curva. Juntamente com as regras MDDAF e CRAF, a EDDAF exerce dominância sobre as regras básicas até a extremidade direita do gráfico. Os perfis das regras EDD e MDD, quase se sobrepõem, indicando que elas têm comportamento muito similar. Já a curva da CR inicia acima das curvas de EDD e de MDD e depois evoluiu em posição ligeiramente inferior.

Uma avaliação complementar pode ser feita com o auxílio da Tabela 5. Nela constam os resultados de experimentos com as instâncias adaptadas de Brandimarte (1993), listadas na primeira coluna. Nesses experimentos foi utilizado o modelo matemático apresentado por Birgin *et al*. (2011) para minimização do *makespan*, denominado modelo adaptado. Esse modelo foi selecionado por considerar explicitamente os instantes de término de processamento de cada *job*. Isso foi conveniente para as alterações que permitiram seu emprego para a minimização do atraso total. O modelo implementado foi executado em CPLEX 12.2 limitado a uma hora de processamento. Na Tabela 5 também constam resultados das melhores heurísticas implementadas, aquelas que consideram afinidade e flexibilidade. O *software* de programação matemática no qual o modelo foi implementado obteve, para cada instância, um limitante inferior e um limitante superior. Esses valores estão, nesta ordem, entre colchetes na segunda coluna da tabela, sendo que o limitante superior indica o atraso total da melhor solução inteira alcançada pela resolução exata. Na quarta coluna está o *gap* fornecido pelo *software*. Por não haver *gap* igual a zero se verifica que nenhuma solução ótima foi obtida, mesmo após uma hora de processamento. Nas demais colunas, para cada heurística, são apresentados o atraso total e o tempo de processamento em milissegundos.

Tabela 5: Resultados exatos e heurísticos para instâncias de Brandimarte (1993).

Instância	**Dimensão (*n*, *m*)**	**CPLEX**		**EDDAF**		**MDDAF**		**CRAF**	
		T_{total}	***gap* (%)**	T_{total}	**Tempo (ms)**	T_{total}	**Tempo (ms)**	T_{total}	**Tempo (ms)**
WT1’	10, 5	[0,00; 72]	100,00	80	0	67	0	97	0
WT2’	20, 5	[18,00; 332]	94,58	407	0	448	0	547	0
WT3’	20, 10	[89,00; 621]	85,67	383	16	436	0	221	0
WT4’	20, 10	[51,00; 56]	8,93	275	16	268	0	205	15
WT5’	30, 15	[83,00; 911]	90,89	488	0	417	16	247	16
Média	-	398,4	-	326,6	-	327,2	-	263,4	

Através da Tabela 5 se verifica que as heurísticas implementadas apresentaram resultados competitivos com a implementação em CPLEX. Em 2 das 5 instâncias, WT3' e WT5', todas heurísticas consideradas encontraram melhor solução que a resolução exata limitada a 1 hora. Para tanto cada heurística não exigiu mais que poucos milésimos de segundo de processamento. Na instância WT1' o CPLEX foi superado pela MDDAF. Nas demais instâncias, apesar dos menores valores de atraso indicados pelo CPLEX, os resultados das heurísticas não ficaram muito distantes. No geral a média do atraso total da CRAF foi a menor com valor 263,4, sendo 26,25% inferior ao atraso médio do CPLEX.

4 Conclusões

Neste trabalho foram propostas regras de prioridade para o JSF usando como critério de desempenho o atraso total. As regras de prioridade implementadas cumpriram seu propósito. Em todos os casos geraram soluções factíveis de qualidade aceitável demandando mínimo tempo computacional. Na comparação entre as regras básicas foi possível constatar a superioridade da regra proposta CR sobre as demais, inclusive sobre a regra indicada por Scrich *et al*. (2004) como melhor regra da literatura, a MDD. Essa superioridade ocorreu especialmente nos problemas de maior porte. A utilização das estratégias baseadas na afinidade e na flexibilidade permitiu que as regras propostas EDDAF, MDDAF e CRAF obtivessem atraso médio até 24% inferior ao das regras básicas correspondentes.

Os resultados envolvendo programação matemática forneceram limitantes para instâncias de porte razoável, balizando a avaliação dos métodos não exatos. Os resultados indicam que as heurísticas implementadas são rápidas e competitivas com o modelo implementado em um *software* de programação matemática com processamento limitado a uma hora. Desse modo é possível afirmar que tais heurísticas construtivas, especialmente as regras propostas EDDAF, MDDAF e CRAF, são capazes de gerar soluções de qualidade, as quais podem ser usadas como soluções iniciais para métodos de melhoria mais sofisticados, como meta-heurísticas.

Agradecimentos: Este trabalho foi financiado parcialmente por CAPES e CNPq.

Referências

Alvarez-Valdes, R.; Fuertes, A.; Tamarit, J.; Giménez, G.; Ramos, R. (2005),A heuristic to schedule flexible job-shop in a glass factory, *European Journal of Operational Research*, v. 165, n. 2, 525–534.

Baker, K. (1984), Sequencing rules and due-date assignments in a job shop, *Management Science*, v. 30, n. 9, 1093–1104.

Baker, K.; Bertrand, J. (1982), A Dynamic Priority Rule for Scheduling Against Due-Dates, *Journal of Operations Management*, v. 3, n. 1, 37–42.

Baykasoğlu, A.; Özbakir, L. (2010), Analyzing the effect of dispatching rules on the scheduling performance through grammar based flexible scheduling system, *Int. J. Production Economics*, v. 124, n. 2, 369–381.

Birgin, E.; Feofiloff, P.; Fernandes, C; Melo, E; Oshiro, M.; Ronconi, D. (2011), Um estudo sobre formulações de programação inteira mista para o problemas de *job shop* flexível, *Anais do XLIII Simpósio Brasileiro de Pesquisa Operacional*, Ubatuba.

Brandimarte, P. (1993), Routing and scheduling in a flexible job shop by tabu search, *Annals of Operations Research*, v. 41, n. 3, 157–183.

Brucker, P.; Schile, R. (1990), Job-shop scheduling with multi-purpose machines, *Computing*, v. 45, n. 4, 369–375.

Chan, F.; Wong, T; Chan, L. (2006), Flexible job-shop scheduling problem under resource constraints, *International Journal of Production Research*, v. 44, n. 11, 2071–2089.

Chen, J.; Chen, K.; Wu, J.; Chen, C. (2008), A study of the flexible job shop scheduling problem with parallel machines and reentrant process, Int. J. Adv. Manuf. Technol., v. 39, 344–354.

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*, v. 70, 281–306.
Dolan, E.; Moré, J. (2002), Benchmarking optimization software with performance profiles. In.: *Mathematical programming*, v. 91, n. 2, 201–213.
Fattahi, P.; Mehrabad, M.; Jolai, F. (2007), Mathematical modeling and heuristic approaches to flexible job shop scheduling problems, *Journal of Intelligent Manufacturing*, v. 18, 331–342.
Garey, M; Johnson, D.; Sethi, R. (1976), The Complexity of Flowshop and Jobshop Scheduling, *Mathematics of Operations Research*, v. 1, n. 2, 117–129.
Gholami, M.; Zandieh, M. (2009), Integrating simulation and genetic algorithm to schedule a dynamic flexible job shop, *Journal of Intelligent Manufacturing*, v. 20, 481–498.
Gutiérrez, C.; García-Magariño, I. (2011), Modular design of a hybrid genetic algorithm for a flexible job–shop scheduling problem, *Knowledge-Based Systems*, v. 24, 102–112.
Ho, N.; Tay, J.; Lai, E. (2006), An effective architecture for learning and evolving flexible job-shop schedules, *European Journal of Operational Research*, v. 179, 316–333.
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*, v. 32, n. 1, 1–13.
Li, J.; Pan, Q.; Liang, Y. (2010), An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems, *Computers & Industrial Engineering*, v. 59, 647–662.
Mainieri, G.; Ronconi, D. (2012), New heuristics for total tardiness minimization in a flexible flowshop, *Optimization Letters*, aceito para publicação.
Özgüven, C.; Özbakır, L.; Yavuz, Y. (2010), Mathematical models for job-shop scheduling problems with routing and process plan flexibility, *Applied Mathematical Modelling*, v. 34, 1539–1548.
Panwalkar, S.; Iskander, W. (1977), A survey of scheduling rules, *Operations Research*, v. 25, n. 1, 45–61.
Pezzella, F.; Morganti, G.; Ciaschetti, G. (2008), A genetic algorithm for the flexible job-shop scheduling problem, *Computers & Operations Research*, v. 35, n. 10, 3201–3212.
Scrich, C. (1997), Busca Tabu para a programação de tarefas em *job shop* com datas de entrega, Tese (Doutorado em Engenharia Elétrica) – Departamento de Engenharia de Sistemas, Unicamp.
Scrich, C.; Armentano, V.; Laguna, M. (2004), Tardiness minimization in a flexible job shop: A tabu search approach, *Journal of Intelligent Manufacturing*, v. 15, 103–115.
Taillard, E. (1993), Benchmarks for basic scheduling problems, *European Journal of Operational Research*, v. 64, n. 2, 278–285.
Tay, J.; Ho, N. (2008), Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems, *Computers & Industrial Engineering*, v. 54, 453–473.
Vepsalainen, A.; Morton, T. (1987), Priority rules for job shop with weighted tardiness costs, *Management Science*, v. 33, n. 8, 1035–1047.
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*, v. 190, 398–411.
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*, v. 56, 1309–1318.
Zhang, H.; Gen, M. (2005), Multistage-based genetic algorithm for flexible job-shop scheduling problem, *Complexity International*, v. 11, 223–232.