MÉTODOS HEURÍSTICOS PARA A PROGRAMAÇÃO DE *FLOW SHOP* PERMUTACIONAL TRICRITÉRIO COM PRAZOS DE ENTREGA

Paulo Antônio Hordones
Programa de Mestrado em Gestão Organizacional
Universidade Federal de Goiás – Regional Catalão
Av. Dr. Lamartine Pinto Avelar, 1120, CEP 75704-020, Catalão/GO
ecn.pauloantonio@gmail.com

Hélio Yochihiro Fuchigami
Universidade Federal de Goiás – Regional Goiânia
Campus Aparecida de Goiânia – Engenharia de Produção
R. Mucuri, s/n, Área 3, Setor Conde dos Arcos, CEP 74968-755, Aparecida de Goiânia/GO
heliofuchigami@ufg.br

RESUMO

Este artigo trata do problema de *flow shop* permutacional tricritério envolvendo atraso total, atraso máximo e número de tarefas atrasadas. São propostas seis heurísticas construtivas e de melhoria para solução do problema. Por meio de uma vasta experimentação computacional, os métodos propostos são avaliados e comparados com dois métodos bastante consagrados na literatura. Para o conjunto de exemplares tratados, os resultados experimentais mostraram a superioridade dos novos métodos em relação à qualidade da solução como também ao esforço computacional.

PALAVRAS CHAVE. Sequenciamento da produção, *Flow Shop*, Heurísticas, Atraso.

Ad&GP

ABSTRACT

This paper addresses the permutation flow shop scheduling problem with a tri criteria objective with total tardiness, maximum tardiness and number of tardy jobs. Six constructive and improvement heuristics are proposed to solve the problem. Through a wide computational experiment, the proposed methods are evaluated and compared with two well-established methods reported in the literature. The experimental results showed the superiority of the new methods in relation to the quality of the solution as well as to the computational effort.

KEYWORDS. Scheduling. Flow shop. Heuristics. Tardiness.

Ad&GP

1. Introdução

O cenário de produção e consumo atual, marcado por uma disputa industrial cada vez mais feroz, levou as empresas a melhorarem seus sistemas, métodos de produção e técnicas de gerenciamento, em busca de maiores eficiências, melhores serviços e consequentemente, uma posição favorável no mercado [Nogueira et al. 2016].

Neste contexto, o *scheduling* é uma das atividades mais importantes (nível operacional) para que uma empresa se mantenha competitiva em mercados consumidores tão exigentes. Está relacionado à otimização de diversas medidas de desempenho que visam o bom funcionamento do sistema e a satisfação dos clientes, como a utilização eficiente dos recursos, a entrega dos produtos nos prazos estipulados e a redução dos custos de produção [Fuchigami e Rangel 2014].

Problemas de *scheduling* vêm sendo bastante estudados nos últimos 60 anos, recebendo atenção considerável por muitos pesquisadores em todo o mundo. São problemas que estão relacionados à alocação de recursos para a realização de um determinado conjunto de tarefas ao longo de um horizonte temporal [Kramer et al. 2015].

O principal objetivo deste trabalho é apresentar métodos heurísticos construtivos e de melhoria para a programação de tarefas em *flow shop* permutacional com a presença de prazos de entrega. A medida de desempenho é tricritério, envolvendo atraso total, atraso máximo e número de tarefas atrasadas. Segundo [Pinedo 2015], o problema estudado pode ser representado por: $Fm|prmu,d_j|\alpha T+\beta T_{\max}+\gamma n_T$. O nível de complexidade deste problema foi classificado por [Lenstra et al.1977] como NP-hard.

Critérios de otimalidade envolvendo datas de entrega são de grande importância nos sistemas de manufatura pois pode existir uma série de custos quando uma tarefa é entregue com atraso. Dentre estes podem ser citados: penalidades contratuais, perda de credibilidade e danos na reputação da empresa [Sen e Gupta 1984].

Nas próximas seções apresentamos: a revisão da literatura, descrevendo os trabalhos propostos para *flow shop*, que buscam minimizar o atraso total, o atraso máximo e o número de tarefas atrasadas (Seção 2); os métodos de solução propostos nesta pesquisa (Seção 3); a experimentação computacional e análise dos resultados obtidos (Seção 4); e as considerações finais (Seção 5).

2. Revisão bibliográfica

Critérios relacionados a atraso estão entre as condições mais importantes para se evitar perdas de clientes e contratos. [Wu 2011] destaca a importância deles, não só com o propósito de se evitar penalidades, mas também como condição para se criar clientes fiéis e satisfeitos em termos de qualidade do produto, custo e pontualidade.

Foi encontrado na literatura apenas um trabalho que aborda *flow shop* tricritério com atraso total, atraso máximo e número de tarefas atrasadas. [Lei e Guo 2011] estudaram o ambiente com a presença de máquinas de processamento em lote (BPM, em inglês). Os autores propuseram um método baseado na busca de vizinhança variável (VNS, em inglês), que se mostrou bastante eficiente durante os testes computacionais.

Considerando que lidamos com três objetivos diferentes, nas próximas subseções apresentamos uma breve revisão atualizada dos trabalhos propostos para *flow shop*, que buscam minimizar o atraso total, o atraso máximo e o número de tarefas atrasadas.

2.1. Minimização do atraso total

[Ow 1985] propôs uma heurística baseada no tempo ocioso das máquinas para o problema de *flow shop* proporcional. [Raman 1995] apresentou uma adaptação dessa heurística para o problema *flow shop* permutacional.

Um algoritmo baseado em *branch-and-bound* foi apresentado por [Kim 1995] para o problema *flow shop* permutacional com *m* máquinas genéricas. Para o mesmo problema, [Armentanto e Ronconi 1999] aplicaram a busca tabu em uma solução inicial gerada a partir da regra *Modified Due Date* (MDD). Em 94% dos problemas testados, a busca tabu demonstrou resultado superior ao algoritmo NEH de [Nawaz et al. 1983].

[Onwubulu e Mutingi 1999] propuseram um método de solução baseado no algoritmo genético, que se mostrou eficiente para problemas de médio e grande porte.

[Hasija e Rajendran 2004] apresentaram uma heurística baseada no *simulated annealing* (SA), que demonstrou, desempenho superior às heurísticas até então existentes baseadas em busca tabu. [Chung et al 2006] incorporaram melhorias ao método proposto por [Kim 1995]. O algoritmo desenvolvido demonstrou desempenho superior em testes com até 20 tarefas.

Um método baseado em um algoritmo de evolução diferencial foi proposto por [Onwubulu e Davendra 2006]. Comparado ao de [Onwubulu e Mutingi 1999], demonstrou desempenho superior especialmente em problemas de pequeno porte. [Framinam e Leisten 2008] sugeriram, um algoritmo *greedy* que apresentou desempenhos superiores à heurística de [Hasija e Rajendran 2004].

[Vallada e Ruiz 2009] desenvolveram um método baseado em algoritmo genético cooperativo (CGA, em inglês) com o objetivo de minimizar o *makespan* e o atraso total. O método proposto apresentou melhorias ao redor de 10% em relação aos métodos tradicionais da literatura. [Vallada e Ruiz 2010] propuseram três métodos baseados em AG. Uma heurística baseada em AG também foi proposta por [Kellegöz et al. 2010]. Resultados experimentais e comparações com os trabalhos de [Hasija e Rajendran 2004] e [Vallada e Ruiz 2010] atestaram a qualidade e a robustez das soluções.

A meta-heurística PSO (*particle swarm optimization*) combinada com busca local foi apresentada e comparada com o SA por [Bank et al. 2012]. A PSO apresentou menor atraso total, enquanto o SA solucionou os problemas em menor tempo computacional.

A fim de solucionar o problema de *flow shop* permutacional com tempos *setup*, [Schaller 2012] propôs 15 heurísticas baseadas nas melhores encontradas na literatura. O algoritmo *greedy* obteve os melhores resultados em problemas de pequeno porte, enquanto o procedimento de busca na vizinhança apresentou os melhores resultados para problemas de grande porte.

[Tasgetiren et al. 2013] apresentaram um método baseado em colônia de abelhas para o *flow shop* permutacional com a restrição *no-idle*. A heurística demonstrou-se altamente competitiva em relação ao algoritmo genético. Para o mesmo problema, [Shen et al. 2015] sugeriram um algoritmo de estimação de distribuição bipopulacional (*BEDA*, em inglês), que demonstrou melhor qualidade da solução em relação ao algoritmo genético e à colônia de abelhas.

[Fernandez-Viegas e Framinan 2015] propuseram oito mecanismos de desempate para a aplicação da heurística NEHedd de [Kim 1993]. Os autores demonstraram que esses mecanismos podem melhorar em até 25% o resultado alcançado pela NEHedd original.

Um procedimento denominado busca interativa local foi proposto por [Li *at al.* 2015]. A heurística proposta superou significativamente os três melhores algoritmos existentes, com o mesmo tempo computacional.

2.2. Minimização do atraso máximo

No que diz respeito à minimização do atraso máximo em *flow shop*, algumas pesquisas focaram em encontrar soluções exatas para o problema, como [Townsend 1977] e [Grabowski et al. 1983].

Para o problema envolvendo duas máquinas em *flow shop*, [Dileepan e Sem 1991] apresentaram um método baseado em *branch-and-bound* e duas heurísticas baseadas na EDD. Um algoritmo também baseado na EDD foi utilizado por [Xiang et al. 2000] para *flow shop* com *m* máquinas. [Lin 2001] analisou em sua pesquisa a complexidade desse problema.

A pesquisa de [Koulamas 1998] demonstrou que quando são impostas certas restrições sobre o tempo de processamento das tarefas e suas datas de entrega, os problemas se tornam polinomialmente solucionáveis. Quando as restrições são ligeiramente enfraquecidas, os problemas permanecem NP-completos [Garey et al. 1976]. [Wu et al. 2007] propuseram um método baseado em *branch-and-bound* e outro baseado em SA. O método baseado no SA apresentou desempenho superior à regra EDD para problemas de grande porte.

[Portougal e Scott 2001] sugeriram uma regra de convergência assintótica para *flow shop* permutacional, que demonstrou desempenho satisfatório.

[Chung et al. 2006] incorporaram melhorias ao método proposto por [Kim 1995]. O algoritmo dos autores demonstrou desempenho superior em exemplares com até 20 tarefas. [Liao et al. 2006] estudaram *flow shop* permutacional e não permutacional, utilizando a busca tabu e o AG como métodos de solução.

2.3. Minimização do número de tarefas atrasadas

[Lawler e Moore 1969] propuseram um algoritmo de programação dinâmica para resolver o problema de *flow shop* com duas máquinas e prazos de entrega comuns. O problema com duas máquinas em *flow shop* permutacional foi investigado por [Ardakan et al 2013]. Os autores propuseram um método baseado em *branch-and-bound*, que pode resolver os problemas em tempo de computação razoável.

A pesquisa de [Koulamas 1998] demonstrou que quando são impostas certas restrições sobre o tempo de processamento das tarefas e suas datas de entrega, esses problemas se tornam polinomialmente solucionáveis. Quando as restrições são ligeiramente enfraquecidas os problemas permanecem NP-complete [Garey et al. 1976].

[Hariri e Potts 1989] desenvolveram um método baseado no algoritmo *branch and bound*, que forneceu a solução em tempo computacional aceitável para problemas com até 20 tarefas e 3 máquinas. [Bulfin e M'Hallah 2003] apresentaram um algoritmo semelhante para minimizar o número mínimo ponderado de tarefas atrasadas em um *flow shop* com duas máquinas. [Choi e Lee 2009] também apresentaram um algoritmo baseado em *branch-and-bound* em um *flow shop* híbrido com dois estágios.

[Gupta e Hariri 1997] apresentaram quatro casos especiais em *flow shop* com duas máquinas. Foram propostas várias heurísticas construtivas e um algoritmo de melhoria, que apresentou média de desvio em relação a solução ótima inferior a 3%.

[Onwubolu e Mutingi 1999] propuseram um método de solução baseado no algoritmo genético. Os resultados indicaram que o algoritmo genético foi eficiente para solucionar problemas de médio e grande porte com aceitável esforço computacional.

Um algoritmo meta-heurístico PSO, com base na regra SPV (*smallest position value*), foi apresentada e comparada com o algoritmo genético por [Ucar e Tasgetiren 2006]. O PSO apresentou o menor número de tarefas atrasadas, enquanto o algoritmo genético solucionou os problemas em menor tempo computacional.

Um problema muito particular foi abordado por [Mosheiov e Sarig 2010]. Os autores estudaram um *flow shop* híbrido com dois estágios de produção, contendo no primeiro uma máquina denominada crítica e no segundo, *m* máquinas disponíveis. O objetivo é minimizar o número ponderado de tarefas atrasadas. A heurística proposta, demonstrou desempenho eficiente, fornecendo programações muito próximas da ótima.

[Wu 2011] apresentou um algoritmo genético híbrido (HGA, em inglês) para minimizar o número de tarefas atrasadas em *flow shop*. As análises dos resultados do experimento computacional indicaram que o algoritmo proposto é robusto e confiável.

[Aldowaisan e Allahverdi 2012] desenvolveram três heurísticas baseadas no SA em *no-wait flow shop*. Os tempos computacionais das três heurísticas se mantiveram muito próximos uns dos outros, todos menores do que um segundo. Para a mesma função objetivo, [Dhouib et al. 2013] utilizaram duas formulações de programação matemática e o algoritmo SA em um *flow shop* com restrições de setup e de tempo entre cada par de operações sucessivas da mesma tarefa.

Três meta-heurísticas baseadas na busca tabu foram propostas por [Varmazyar e Salmasi 2012a] em *flow shop* com tempos de *setup* dependentes da sequência. O experimento computacional mostrou que o algoritmo que em sua lista tabu mantém o controle das posições em que as tarefas são designadas possui melhor desempenho comparado com os outros. Para este mesmo problema, vários algoritmos baseados em busca tabu e no algoritmo competitivo imperialista (ICA, em inglês) foram desenvolvidos por [Varmazyar e Salmasi 2012b]. Os resultados da experimentação computacional mostraram que o desempenho da ICA é pior do que o dos outros algoritmos para problemas de pequeno e médio porte. Um modelo híbrido de ICA e busca tabu ofereceu melhor desempenho do que os demais algoritmos propostos para problemas de grande porte.

Métodos heurísticos construtivos e de melhoria foram sugeridos por [Hordones et al. 2015]. A melhor heurística obteve soluções com desvio relativo de apenas 0,3% da solução ótima para exemplares com até dez tarefas e cinco máquinas. [Allahverdi et al. 2016] desenvolveram diversas heurísticas baseadas em algoritmo genético, algoritmo genético melhorado (IGA, em inglês), SA e SA baseado na teoria da nuvem (CSA, em inglês). As heurísticas baseadas no SA apresentaram os melhores resultados.

3. Heurísticas propostas

O ambiente de produção *flow shop* permutacional é ilustrado na Figura 1.

O problema consiste em programar um conjunto de n tarefas, definido como $J = \{J_1, ...,J_n\}$, onde se mantém a mesma ordem de programação destas n tarefas em todas as máquinas. Entre as $n!$ possíveis sequências, o objetivo do problema é tricritério, buscando minimizar o atraso total, o atraso máximo e o número de tarefas atrasadas.

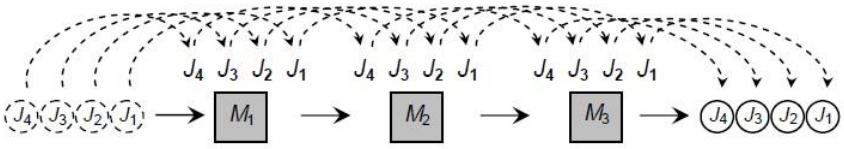

Figura 1: Ilustração de um *flow shop* permutacional

Foram propostas seis heurísticas construtivas e de melhoria. As ideias que nortearam a concepção dos métodos baseiam-se em algoritmos clássicos para outros problemas, como a heurística NEHedd de [Kim 1993], conhecida pelos bons resultados em minimização do atraso médio e o algoritmo de [Hodgson 1977], que fornece a solução ótima para minimização do número de tarefas atrasadas em máquina única.

No estabelecimento da ordenação inicial, foram empregadas duas regras bastante famosas, a EDD (*Earliest Due Date*) e a MST (*Minimum Slack Time*), ambas utilizando os prazos das tarefas em seu critério de prioridade. A EDD faz a ordenação crescente pelos prazos (d_j) e a MST pela folga ($d_j - \sum p_{jk}$), onde p_{jk} é o tempo de processamento da tarefa j na máquina k.

Nas heurísticas H1, H3, H4 e H6, foram implementadas as buscas em vizinhanças de inserção e de permutação.

Será apresentado a seguir o funcionamento de cada heurística desenvolvida.

HEURÍSTICA H1:

(1) Ordene as tarefas pela regra EDD.
(2) Com as duas primeiras tarefas da ordenação obtida, encontre a subsequência (entre as duas possíveis) que fornece o melhor valor da função objetivo.
(3) Sem alterar as posições relativas das tarefas já alocadas, insira a próxima tarefa da ordenação obtida em (1) em todas as posições possíveis da subsequência atual e considere aquela que fornece o melhor valor da função objetivo.
(4) Repita (3) até que todas tarefas estejam programadas.
(5) Considerando toda a Vizinhança de Inserção da sequência com $(n–1)^2$ soluções, considere aquela que fornece o melhor valor da função objetivo.
Considerando toda a Vizinhança de Permutação da sequência com $n(n–1)/2$, considere aquela que fornece o melhor valor da função objetivo.

HEURÍSTICA H2:

(1) Ordene as tarefas pela regra MST.
(2) Com as duas primeiras tarefas da ordenação obtida, encontre a subsequência (entre as duas possíveis) que fornece o melhor valor da função objetivo.
(3) Sem alterar as posições relativas das tarefas já alocadas, insira a próxima tarefa da ordenação obtida em (1) em todas as posições possíveis da subsequência atual e considere a que fornece o melhor valor da função objetivo.
(4) Repita o (3) até que todas tarefas estejam programadas.

HEURÍSTICA H4:

(1) Ordene as tarefas pela regra EDD.
(2) Se não houver tarefas atrasadas na sequência atual, a sequência total é a sequência ótima. Caso contrário, identifique a primeira tarefa atrasada na sequência atual (J_T).

(3) Da primeira tarefa até J_T, remova a tarefa com maior soma dos tempos de processamento em todas as máquinas para a lista de tarefas removidas (sequência de tarefas após a sequência atual). Vá para (2).
(4) Considerando toda a Vizinhança de Inserção da sequência com $(n–1)^2$ soluções, considere aquela que fornece o melhor valor da função objetivo.
Considerando toda a Vizinhança de Permutação da sequência com $n(n–1)/2$, considere aquela que fornece o melhor valor da função objetivo.

HEURÍSTICA H5:
(1) Ordene as tarefas pela regra MST.
(2) Se não houver tarefas atrasadas na sequência atual, a sequência total é a sequência ótima. Caso contrário, identifique a primeira tarefa atrasada na sequência atual (J_T).
(3) Da primeira tarefa até J_T, remova a tarefa com maior soma dos tempos de processamento em todas as máquinas para a lista de tarefas removidas (sequência de tarefas após a sequência atual). Vá para (2).

HEURÍSTICA H6:
(1) Ordene as tarefas pela regra MST.
(2) a (3) Idem à heurística H5.
(4) Considerando toda a Vizinhança de Inserção da sequência com $(n–1)^2$ soluções, considere aquela que fornece o melhor valor da função objetivo.
Considerando toda a Vizinhança de Permutação da sequência com $n(n–1)/2$, considere aquela que fornece o melhor valor da função objetivo.

As seis heurísticas propostas foram comparadas com duas da literatura: a NEHedd de [Kim 1993] e o algoritmo de [Hodgson 1977].

4. Experimentação computacional e resultados

4.1. Metodologia da pesquisa

Segundo as definições de [Martins 2010] e [Nakano 2010], esta pesquisa possui abordagem quantitativa, pois há preocupação com mensurabilidade, causalidade, generalização e replicação. Pode também ser classificada como experimento, uma vez que testa o relacionamento entre as variáveis da pesquisa operacionalizada, manipulando variáveis independentes (neste caso, a programação do problema) para se observar o resultado na variável dependente (representada aqui pelas medidas de desempenho, atraso total, atraso máximo e número de tarefas atrasadas).

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

4.2. Planejamento do experimento

Na experimentação computacional foram testados e avaliados 15.600 exemplares, divididos em dois grupos: Grupo 1, com exemplares de pequeno porte, tendo sido encontrada a solução ótima por meio da enumeração completa e Grupo 2, com exemplares de médio e grande portes. As classes de exemplares foram definidas pelo número de tarefas (n), número de máquinas (m) e pelo cenário relativo aos prazos das tarefas. Para cada classe, foram gerados aleatoriamente 100 exemplares visando reduzir o erro amostral.

Conforma a maioria dos trabalhos de sequenciamento da produção, os tempos de processamento foram gerados no intervalo $U[1,99]$. No Grupo 1, os parâmetros foram: $n \in \{5, 6, 7, 8, 10\}$ e $m \in \{2, 3, 5\}$. E no Grupo 2, os parâmetros consistiram de: $n \in \{15, 20, 30, 50, 80, 100\}$ e $m \in \{5, 10, 15, 20\}$. Estes valores foram escolhidos de forma a abranger uma significativa gama de exemplares de diversos tamanhos.

Os prazos das tarefas foram gerados seguindo o método utilizado por [Armentano e Ronconi 2000] e [Ronconi e Birgin 2012], que utiliza a distribuição uniforme no intervalo $[P(1–T–R/2), P(1–T+R/2)]$, onde T e R são dois parâmetros denominados fator de atraso e faixa de

prazos, respectivamente, e *P* é o limitante inferior de [Taillard 1993] para o *makespan*, definido como:

$$P = \max\left\{\max_{1\le k\le m}\left\{\sum_{j=1}^{n} p_{jk} + \min_{j}\sum_{q=1}^{k-1} p_{jq} + \min_{j}\sum_{q=k+1}^{m} p_{jq}\right\}, \max_{j}\sum_{k=1}^{m} p_{jk}\right\}.$$

A partir da variação de *T* e *R*, foram obtidos os seguintes cenários:

- Cenário 1: baixo fator de atraso (*T*=0,2) e pequena faixa de prazos (*R*=0,6);
- Cenário 2: baixo fator de atraso (*T*=0,2) e ampla faixa de prazos (*R*=1,2);
- Cenário 3: alto fator de atraso (*T*=0,4) e pequena faixa de prazos (*R*=0,6);
- Cenário 4: alto fator de atraso (*T*=0,4) e ampla faixa de prazos (*R*=1,2).

Em ambos os grupos, foram utilizadas quatro opções de valores para os pesos α, β, e γ, que representam respectivamente as prioridades das componentes da função objetivo: atraso total, atraso máximo e número de tarefas atrasadas. As relações de peso seguiram o método utilizado por [Eren e Güner 2008], que abordaram o problema de *flow shop* tricritério com duas máquinas e minimização da soma ponderada do *makespan*, *flowtime* e atraso total A Tabela 1 apresenta os valores considerados.

	α	**β**	**γ**
R1	0,33	0,33	0,33
R2	0,25	0,25	0,50
R3	0,25	0,50	0,25
R4	0,50	0,25	0,25

Tabela 1 – Opções de valores para alfa, beta e gama.

Todas as heurísticas propostas neste estudo consideram durante a fase de construção e/ou de melhoria, a seleção de subsequências com o melhor valor da função objetivo (utilizando os pesos da relação executada), estabelecendo a ponderação dos valores de α, β, e γ.

Com estes parâmetros foram obtidos 6.000 exemplares do Grupo 1: 5 opções de número de tarefas, 3 opções de número de máquinas, 4 cenários e 100 exemplares por classe (5*3*4*100=6.000). E foram gerados 9.600 exemplares do Grupo 2: 6 opções de número de tarefas, 4 opções de número de máquinas, 4 cenários e 100 exemplares por classe (6*4*4*100=9.600). Ambos os grupos totalizam os 15.600 exemplares resolvidos.

Foi utilizado o sistema operacional Windows de 64 bits e o ambiente de programação Delphi. As configurações da máquina foram as seguintes: processador Intel Core i5-5200U com 2.2 GHz de frequência e 4.0 GB de memória RAM.

4.3. Medidas de comparação

Os resultados obtidos na experimentação computacional foram avaliados em termos da eficácia, ou seja, a qualidade da solução dos métodos, e também da eficiência computacional, verificada por meio do tempo médio de computação medido em milissegundos (ms).

A eficácia dos métodos de solução foi analisada por meio do índice de desvio relativo (*relative deviation index* – RDI), tal como em [Varmazyar e Salmasi 2012b]. Conforme observaram os autores, embora seja mais comum se utilizar o desvio relativo percentual (*relative percentage deviation* – RPD) na comparação de desempenho entre os métodos de solução, no caso de critérios que envolvem minimização de atraso, o valor ótimo da função-objetivo em muitos exemplares-testes pode ser zero. Por este motivo, o RDI é mais apropriado, sendo calculado para um determinado método de solução da seguinte forma:

$$RDI = \left(\frac{fmethod - fbest}{fworst - fbest}\right).100\%, \qquad (4.1)$$

onde $fmethod$ é o valor da função objetivo obtido por um dado algoritmo, $fbest$ é a melhor solução obtida entre todos os algoritmos comparados e $fworst$ é a pior solução obtida entre todos os algoritmos comparados. Com esta medida, um índice entre 0 e 100 é obtido para cada método.

Quanto menor o RDI de um método, melhor o seu desempenho. E se o RDI de um método é igual a zero, significa que ele forneceu a solução ótima ou a melhor solução dentre os métodos avaliados. Vale ressaltar que: se a pior e a melhor solução têm o mesmo valor, todos os métodos fornecem a melhor (mesma) solução e o valor do RDI será 0 para todos os métodos.

A comparação de desempenho foi feita conjuntamente entre as seis heurísticas propostas e os dois métodos da literatura (NEHedd e Hodgson). Além disso, no Grupo 1, os desvios foram comparados coma solução ótima obtida pelo método de enumeração completa, enquanto no Grupo 2, foi considerada a melhor solução fornecida pelos métodos implementados.

4.4. Análise dos resultados globais

O teste ANOVA foi executado para todas as opções de relações e grupos. Infere-se, em todos os casos que há diferença significativa nos resultados dos métodos de solução para o nível de significância 0,05.

Os resultados globais das heurísticas propostas mostraram que as heurísticas H4 e H3 obtiveram os melhores desempenhos, próximos da solução ótima em alguns cenários. Este é um resultado bastante satisfatório para heurísticas, que além disso consumiram tempo computacional bastante viável, próximo de zero

A Tabela 2 expressa numericamente os dados globais, destacando os melhores e os piores resultados. Ilustrados na cor verde estão os melhores resultados e de azul os segundos melhores. Em vermelho estão os piores desempenhos.

Nos dois grupos analisados e também em cada uma das opções de pesos α, β, e γ, a heurística H4 obteve o melhor desempenho, em muitos casos com RDI próximo de zero. A heurística H3 obteve o segundo melhor resultado nas 4 opções de α, β, e γ do Grupo 1, com desvios relativamente próximos da H4. Já no Grupo 2, o segundo melhor método foi o algoritmo de Hodgson, porém com desvios bem mais elevados do que a melhor heurística H4.

Dentre as oito heurísticas implementadas, a NEHedd atingiu os piores resultados em termos de RDI, em todas as opções de α, β, e γ, nos dois grupos. Pode-se notar também, que em todas as heurísticas que incluem etapas de melhoria (H1, H3, H4 e H6) houve redução do RDI em relação às heurísticas construtivas (NEHedd, H2, Hodgson e H5), fato que comprova a eficácia da etapa de buscas nas vizinhanças de inserção e de permutação.

	Opções de pesos para α, β e γ	Heurísticas							
		NEHedd	H1	H2	H3	Hodgson	H4	H5	H6
Grupo 1	R1	84,2	20,8	60,6	11,8	57,4	11,1	68,5	13,3
	R2	84,2	20,8	60,7	11,8	57,3	11,1	68,4	13,3
	R3	84,0	19,7	61,2	11,6	54,5	9,9	66,8	12,5
	R4	84,0	19,7	61,2	11,6	54,5	9,9	66,8	12,5
Grupo 2	R1	87,7	51,8	66,5	33,0	28,4	3,4	62,6	30,0
	R2	87,7	51,8	66,5	33,0	28,4	3,4	62,6	30,0
	R3	87,6	52,4	66,0	32,7	28,9	3,6	63,6	30,5
	R4	87,7	51,4	66,8	33,2	28,2	3,4	62,1	29,6

Tabela 2 - Resultados da Análise Global do RDI das Heurísticas

Para melhor visualização dos resultados globais, os gráficos das Figuras 2 e 3 apresentam os RDI do Grupo 1 e do Grupo 2 separadamente, para cada opção de peso de α, β, e γ.

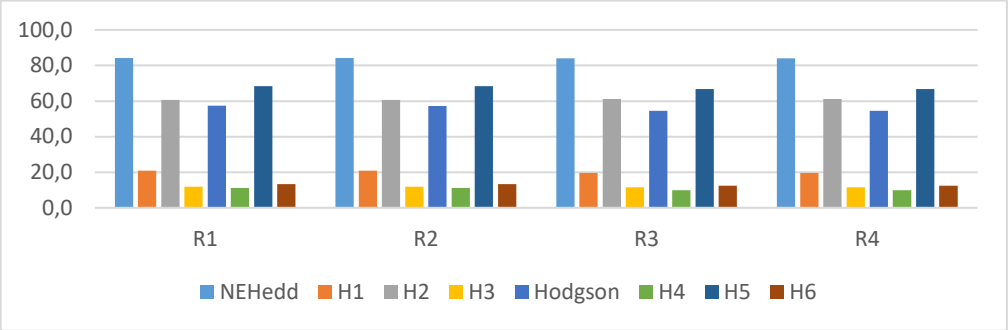

Figura 2 – Comparação dos resultados globais do RDI das heurísticas no Grupo 1

Como pode ser visto na Figura 2, no Grupo 1 os desempenhos das heurísticas H4 e H3 possuem RDI pequenos, apresentando assim resultados próximos à solução ótima, principalmente nas relações R3 e R4. É possível visualizar, de forma clara, a discrepância dos desempenhos das heurísticas que incluem etapas de melhoria, em relação às exclusivamente construtivas. Constata-se também, que o desempenho das heurísticas se mantém praticamente inalterado em todas as opções de peso de α, β, e γ.

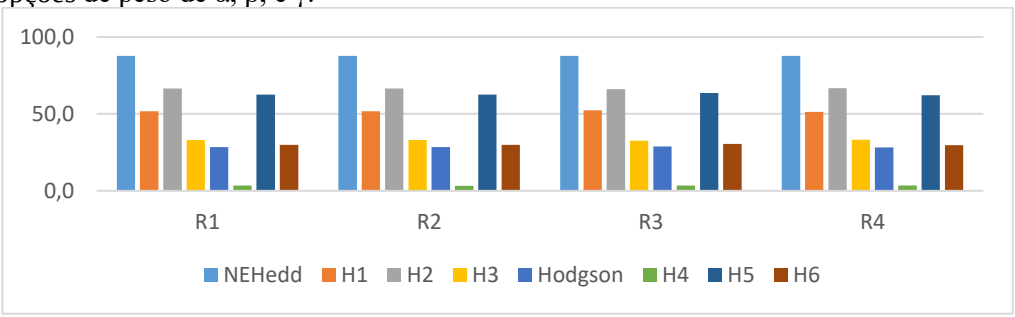

Figura 3 – Comparação dos resultados globais do RDI das heurísticas no Grupo 2

A Figura 3, com os resultados do Grupo 2, expõe discrepância dos resultados da heurística H4 em relação às demais. Assim como no Grupo 1, no Grupo 2 o desempenho das heurísticas se mantém praticamente inalterado em todas as relações.

Nos quatro problemas desdobrados a partir das opções dos pesos dos componentes da função objetivo (nas análises das diferentes relações de prioridade), não foram verificadas anomalias ou conclusões divergentes da análise global, podendo indicar que os pesos considerados não influenciaram suficientemente os problemas a ponto de alterar significativamente os resultados nas quatro opções consideradas

Para os tempos computacionais de execução das heurísticas foi elaborada a Tabela 3. O tempo médio de computação é obtido realizando a soma do tempo de computação de todos os exemplares, dividido pelo total de exemplares. A utilização deste critério de comparação fornece informações sobre qual dos métodos é mais rápido na busca por soluções. O método que apresentar a menor média é considerado o mais rápido, apresentando maior eficiência computacional.

Grupo 1	**Hodgson**	**H5**	**H2**	**NEHedd**	**H4**	**H6**	**H3**	**H1**
	0,0007	0,0019	0,0265	0,0338	0,0514	0,0608	0,0705	0,0933
Grupo 2	**Hodgson**	**H5**	**H2**	**NEHedd**	**H4**	**H6**	**H3**	**H1**
	0,1178	0,2975	16,3945	17,2296	59,5675	59,9791	76,2821	76,3641

Tabela 3 – *Ranking* dos tempos médios de CPU das heurísticas do Grupo 1 e Grupo 2 (em milisegundos)

Como pode ser observado, o tempo médio de CPU do algoritmo Hodgson foi bem próximo a zero. Isso ocorreu por se tratar de uma heurística de estrutura simples aplicada a exemplares de pequeno porte.

Na resolução do Grupo 2, naturalmente as heurísticas levaram mais tempo de CPU do que as do Grupo 1, já que se trata de exemplares de médio e grande portes.

Conforme esperado, nas heurísticas H1, H3, H4 e H6, o tempo de CPU foi maior do que nas demais, devido à etapa de melhoria inserida nestas. Sugere-se, portanto a aplicação da heurística H4, que obteve o melhor desempenho global em todas quatro opções de pesos α, β, e γ (R1, R2, R3 e R4), com tempo de CPU aceitável, mesmo para exemplares de grande porte.

5. Considerações finais

Este estudo atingiu seu objetivo de desenvolver e implementar métodos heurísticos construtivos e de melhoria eficazes, em termos da qualidade da solução, e eficientes computacionalmente. As heurísticas propostas apresentaram resultados melhores em comparação com métodos consagrados.

A melhor heurística H4 obteve os resultados mais promissores na maioria das classes de problemas, principalmente naqueles de grande porte. Fato que demonstra a sua aplicabilidade prática. Para exemplares com até 10 tarefas e 5 máquinas, as melhores heurísticas H4 e H3, tiveram desvios médios da melhor solução encontrada próximos a 10%.

A experimentação computacional evidenciou, a importância das etapas de buscas em vizinhanças no desempenho das heurísticas. Toda a heurística que aplicou a etapa de melhoria teve desempenho superior à sua equivalente que não a aplicou. Isso sem inviabilizar o tempo computacional.

Como sugestão para trabalhos futuros, indica-se a adição de novas restrições ao problema, como tempos de *setup* e datas de liberação das tarefas.

Agradecimentos

Agradecemos às agências de fomento CAPES, CNPq e FAPEG pelo apoio e auxílio financeiro, primordiais para a realização do presente estudo e aos revisores anônimos pela leitura cuidadosa do texto.

Referências

Aldowaisan, T. A., Allahverdi, A. (2012). No-wait flowshop scheduling problem to minimize the number of tardy jobs. *The International Journal of Advanced Manufacturing Technology*, *61*(1-4), 311-323.

Allahverdi, A., Aydilek, A., Aydilek, H. (2016). Minimizing the number of tardy jobs on a two-stage assembly flowshop. *Journal of Industrial and Production Engineering*, *33*(6), 391-403.

Armentano, V. A., Ronconi, D. P. (1999). Tabu search for total tardiness minimization in flowshop scheduling problems. *Computers & operations research*, *26*(3), 219-235.

Bank, M., Ghomi, S. F., Jolai, F., Behnamian, J. (2012). Application of particle swarm optimization and simulated annealing algorithms in flow shop scheduling problem under linear deterioration. *Advances in Engineering Software*, *47*(1), 1-6.

Bulfin, R. L., M'Hallah, R. (2003). Minimizing the weighted number of tardy jobs on a two-machine flow shop. *Computers & Operations Research*, *30*(12), 1887-1900.

Choi, H. S., Lee, D. H. (2009). Scheduling algorithms to minimize the number of tardy jobs in two-stage hybrid flow shops. *Computers & Industrial Engineering*, *56*(1), 113-120.

Chung, C. S., Flynn, J., Kirca, Ö. (2006). A branch and bound algorithm to minimize the total tardiness for m-machine permutation flowshop problems. *European Journal of Operational Research*, *174*(1), 1-10.

Dhouib, E., Teghem, J., Loukil, T. (2013). Minimizing the number of tardy jobs in a permutation flowshop scheduling problem with setup times and time lags constraints. *Journal of Mathematical Modelling and Algorithms in Operations Research*, *12*(1), 85-99.

Dileepan, P., Sen, T. (1991). Job lateness in a two-machine flowshop with setup times separated. *Computers & operations research*, *18*(6), 549-556.

Eren, T., Güner, E. (2008). The tricriteria flowshop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, *36*(11-12), 1210-1220.

Fernandez-Viagas, V., Framinan, J. M. (2015). NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness. *Computers & Operations Research*, *60*, 27-36.
Framinan, J. M., Leisten, R. (2008). Total tardiness minimization in permutation flow shops: a simple approach based on a variable greedy algorithm. *International Journal of Production Research*, *46*(22), 6479-6498.
Fuchigami, H. Y., Rangel, S. (2014). Uma análise de estudos de casos em sequenciamento da produção. *XLVI Simpósio Brasileiro de Pesquisa Operacional (XLVI SBPO)*, 159-170.
Garey, M. R., Johnson, D. S., Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, *1*(2), 117-129.
Gupta J.N.D., & Hariri A.M.A. (1997), Two machine flow shop to minimize number of tardy jobs. *Journal of the Operational Research Society*, 48, 212–20.
Grabowski, J., Skubalska, E., Smutnicki, C. (1983). On flow shop scheduling with release and due dates to minimize maximum lateness. *Journal of the Operational Research Society*, *34*(7), 615-620.
Hariri A.M.A., Potts C.N. (1989), A branch and bound algorithm to minimize number of late jobs in a permutation flow-shop *Eur J Opl Res*,38, 228–237.
Hasija, S., Rajendran, C. (2004). Scheduling in flowshops to minimize total tardiness of jobs. *International Journal of Production Research*, *42*(11), 2289-2301.
Hodgson, T.J. (1977), Note - A Note on Single Machine Sequencing with Random Processing Times. *Management Science*,23(10), 1144-1146.
Jung, C.F. (2004), Metodologia *para pesquisa & desenvolvimento: aplicada a novas tecnologias, produtos e processos*, Rio de Janeiro: Axcel Books.
Hordones, P. A. M.; Silva, A. A ; Fuchigami, H. Y (2015). Métodos heurísticos para minimização do número de tarefas atrasadas em flow shop permutacional. *XLVII Simpósio Brasileiro de Pesquisa Operacional (XLVII SBPO)*, 86-97.
Kellegöz, T., Toklu, B., Wilson, J. (2010). Elite guided steady-state genetic algorithm for minimizing total tardiness in flowshops. *Computers & Industrial Engineering*, *58*(2), 300-306.
Kim, Y. D. (1993). A new branch and bound algorithm for minimizing mean tardiness in two-machine flowshops. *Computers & Operations Research*, *20*(4), 391-401.
Kim, Y. D. (1995). Minimizing total tardiness in permutation flowshops. *European Journal of Operational Research*, *85*(3), 541-555.
Koulamas, C. (1998). On the complexity of two-machine flowshop problems with due date related objectives. *European journal of operational research*, *106*(1), 95-100.
Kramer, R., Subramanian, A., Vidal, T., & Lucídio dos Anjos, F. C. (2015). A matheuristic approach for the pollution-routing problem. *European Journal of Operational Research*, *243*(2), 523-539.
Lawler, E. L., & Moore, J. M. (1969). A functional equation and its application to resource allocation and sequencing problems. *Management Science*, *16*(1), 77-84.
Lei, D., Guo, X. P. (2011). Variable neighbourhood search for minimising tardiness objectives on flow shop with batch processing machines. *International Journal of Production Research*, *49*(2), 519-529.
Lenstra, J. K., Kan, A. R., & Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of discrete mathematics*, *1*, 343-362.
Li, X., Chen, L., Xu, H., Gupta, J. N. (2015). Trajectory Scheduling Methods for minimizing total tardiness in a flowshop. *Operations Research Perspectives*, *2*, 13-23.
Liao, C. J., Liao, L. M., Tseng, C. T. (2006). A performance evaluation of permutation vs. non-permutation schedules in a flowshop. *International Journal of Production Research*, *44*(20), 4297-4309.
Lin, B. M. (2001). Scheduling in the two-machine flowshop with due date constraints. *International Journal of Production Economics*, *70*(2), 117-123.
Martins, R.A. (2010), Abordagens Quantitativa e Qualitativa. In: MIGUEL, P.A.C. (Org.). *Metodologia de pesquisa em Engenharia de Produção e Gestão de Operações*, Rio de Janeiro: Elsevier; 45-61, cap. 3.
Mosheiov, G., Sarig, A. (2010), Minimum weighted number of tardy jobs on an m-machine flow-shop with a critical machine. *European Journal of Operational Research*, 201, 404-408.

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

Nawaz, M., Enscore JR., E. E., Ham I. (1983), A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega, The International Journal of Management cience, 11, 91-95.

Nogueira, T. H.; Guimarães Neto, E. P.; Morais, G. ; Ravetti Martin, G.(2016) . Two-dedicated-parallel-machine sheduling approach for a two-dock truck sheduling problem in a cross-docking centre. *In: SBPO 2016*, Vitória/ES. XLVIII Simpósio Brasileiro de Pesquisa Operacional.

Onwubolu, G., Davendra, D. (2006). Scheduling flow shops using differential evolution algorithm. *European Journal of Operational Research*, *171*(2), 674-692.

Onwubolu, G., Mutingi, M. (1999), Genetic algorithm for minimizing tardiness in flow-shop scheduling. *Production Planning & Control*, *10*, 462–471.

Ow, P. S. (1985). Focused scheduling in proportionate flowshops. Management Science, 31(7), 852-869.

Pinedo, M.L. (2015), Scheduling: theory, algorithms, and systems, Prentice-Hall, New Jersey, 6ªed.

Portougal, V., Scott, J. L. (2001). The asymptotic convergence of some flow-shop scheduling heuristics. *Asia-Pacific Journal of Operational Research*, *18*(2), 243.

Raman, N. (1995). Minimum tardiness scheduling in flow shops: construction and evaluation of alternative solution approaches. *Journal of Operations Management*, *12*(2), 131-151.

Ronconi, D.P. Birgin, E.G. (2012), Mixed-integer programming models for flow shop scheduling problems minimizing the total earliness and tardiness. *In:*Ríos-Solís, Y.A. e Ríos-Mercado, R.Z. (Eds.) *Just-in-Time Systems*, Springer Sciences, New York.

Schaller, J. (2012). Scheduling a permutation flow shop with family setups to minimise total tardiness. *International Journal of Production Research*, *50*(8), 2204-2217.

Sen, T., Gupta, S. K. (1984). A state-of-art survey of static scheduling research involving due dates. *Omega*, *12*(1), 63-76.

Shen, J. N., Wang, L., Wang, S. Y. (2015). A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion. *Knowledge-Based Systems*, *74*, 167-175.

Taillard, E.D. (1993), Parallel Iterative Search Methods for Vehicle Routing Problems, Networks 23, 661 – 676.

Tasgetiren, M. F., Pan, Q. K., Suganthan, P. N., Oner, A. (2013). A discrete artificial bee colony algorithm for the no-idle permutation flowshop scheduling problem with the total tardiness criterion. *Applied Mathematical Modelling*, *37*(10), 6758-6779.

Townsend, W. (1977). Note—Sequencing n Jobs on m Machines to Minimise Maximum Tardiness: A Branch-and-Bound Solution. *Management Science*, *23*(9), 1016-1019.

Ucar, H., Tasgetiren, M. F. (2006). A particle swarm optimization algorithm for permutation flow shop sequencing problem with the number of tardy jobs criterion. In *Fifth International Symposium on Intelligent Manufacturing Systems*, pp. 237-250.

Vallada, E., Ruiz, R. (2009). Cooperative metaheuristics for the permutation flowshop scheduling problem. *European Journal of Operational Research*, *193*(2), 365-376.

Vallada, E., Ruiz, R. (2010). Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. *Omega*, *38*(1), 57-67.

Varmazyar, M., Salmasi, N. (2012), Sequence-dependent flow shop scheduling problem minimizing the number of tardy jobs, *International Journal of Production Research*, 50(20), 5843-5858.

Varmazyar, M., Salmasi, N. (2012), Minimizing the number of tardy jobs in flow shop sequence dependent setup times scheduling problem, Applied Mechanics and Materials, 110-116, 4063-4069.

Wu, J. J. (2011). Using a Hybrid Genetic Algorithm to Minimize the Number of Tardy Jobs in the Flow Shop. In *Advanced Materials Research* (Vol. 201, pp. 1070-1074). Trans Tech Publications.

Wu, C. C., Lee, W. C., Wang, W. C. (2007). A two-machine flowshop maximum tardiness scheduling problem with a learning effect. *The International Journal of Advanced Manufacturing Technology*, *31*(7-8), 743-750.

Xiang, S., Tang, G., Cheng, T. C. E. (2000). Solvable cases of permutation flowshop scheduling with dominating machines. *International Journal of Production Economics*, *66*(1), 53-57.