MÉTODOS HEURÍSTICOS PARA MINIMIZAÇÃO DO NÚMERO DE TAREFAS ATRASADAS EM *FLOW SHOP* PERMUTACIONAL

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

Anayama Alves
Programa de Mestrado em Gestão Organizacional
Universidade Federal de Goiás – Regional Catalão
Av. Dr. Lamartine Pinto de Avelar, 1120, CEP 75704-020, Catalão/GO
anayama_alves_3011@hotmail.com

Hélio Yochihiro Fuchigami
Programa de Mestrado em Gestão Organizacional
Universidade Federal de Goiás – Regional Catalão
Av. Dr. Lamartine Pinto de Avelar, 1120, CEP 75704-020, Catalão/GO
heliofuchigami@yahoo.com.br

RESUMO

Este trabalho trata do problema de sequenciamento em *flow shop* permutacional com o objetivo de minimizar o número de tarefas atrasadas. Foram propostos e implementados computacionalmente oito métodos de solução heurística, sendo quatro exclusivamente construtivos e quatro incluindo etapas de melhoria para resolver o problema. As heurísticas foram avaliadas quanto à eficácia da solução (qualidade) por meio do índice de desvio relativo e quanto à eficiência computacional (tempo de CPU). Os resultados comprovaram a aplicabilidade dos métodos de solução. A melhor heurísticaobteve soluções com desvio relativo de apenas 0,3% da solução ótima, para problemas com até 10 tarefas e 5 máquinas e tempo de execução aceitável mesmo para problemas de grande porte.

PALAVRAS CHAVE. Programação da produção, *Flow shop* Permutacional, Heurísticas, Tarefas atrasadas.

AD&GP

ABSTRACT

This work addresses the scheduling problem in the permutation *flow shop* with the objective to minimize the number of tardy jobs. To solve the problem, were proposed and computationally implemented eight heuristics: four exclusively constructive and four including improvement steps. The heuristic methods were evaluated for efficacy of the solution (quality) by relative deviation index and for computational efficiency (CPU time). The results show the applicability of the solution methods. The best heuristic found solutions with a relative deviation index of only 0.3% of the optimal solution for problems with up to 10 jobs and 5 machines within an acceptable running time even for large problems.

KEYWORDS. Production scheduling, Permutation *Flow Shop,* Heuristics, Tardy jobs.

AD&GP

1.Introdução

Com o surgimento da revolução industrial, na segunda metade do século XVIII, o mundo presenciou o crescimento econômico e o aumento da complexidade das organizações. Com ela veio a divisão do trabalho, a segmentação da gerência, a necessidade de organização do processo produtivo e a busca por um processo eficiente visando maior lucro.

No atual cenário globalizado e extremamente competitivo em que as empresas disputam cada vez mais clientes, reconhecimento e condições de permanecer competitivas e lucrativas, é fundamental o desenvolvimento de mecanismos que as auxiliem na tomada de decisões.

O investimento em programação da produção *(scheduling)* pelas empresas tornou-se uma necessidade de sobrevivência. As empresas devem programar atividades de forma a utilizar os recursos disponíveis de maneira eficiente, a fim de minimizar custos e respeitar compromissos de entrega estabelecidos com os clientes.

Os problemas de programação de tarefas em sistemas de produção são, tradicionalmente, classificados em função do fluxo das operações nas máquinas, conforme segue:

- Máquina Única: existe uma única máquina disponível para o processamento das tarefas;
- Máquinas Paralelas: são disponíveis diversas máquinas, geralmente idênticas, para as mesmas operações;
- *Flow Shop*: todas as tarefas têm a mesma sequência de processamento no conjunto de máquinas;
- *Flow Shop* Permutacional: é um *Flow Shop* no qual a sequência das tarefas é a mesma em qualquer máquina;
- *JobShop:*cada tarefa tem sua própria sequência de processamento no conjunto de máquinas;
- *Open Shop*: não há uma sequência específica ou preestabelecida para o processamento das tarefas nas máquinas;
- *Flow Shop* com Máquinas Múltiplas: consiste em um *Flow Shop* no qual existe um conjunto de máquinas paralelas em cada estágio de produção.
- *Job Shop* com Máquinas Múltiplas: é um *Job Shop* no qual existe um conjunto de máquinas paralelas em cada estágio de produção;

O objetivo básico deste trabalho é apresentar métodos heurísticos construtivos de melhoria para a programação de tarefas em ambientes *flow shop* permutacionais com a presença de prazos de entrega. O critério de desempenho é a minimização do número de tarefas atrasadas. Segundo Pinedo (2010), o problema estudado pode ser representado por: $Fm|prmu,d_j|n_T$.

Segundo Ronconi (1997), 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. De acordo com a autora, embora poucos trabalhos abordem problemas envolvendo atraso, este critério é muito importante em muitos sistemas reais como: manufatura de papel, processamento e refinamento de metais, fabricação de pães e indústrias químicas.

Nas próximas seções apresentamos a revisão da literatura, expondo os trabalhos que abordaram especificamente o mesmo problema objeto desta pesquisa (Seção 2), os métodos de solução propostos (Seção 3) e a experimentação computacional e análise dos resultados encontrados (Seção 4). Na Seção 5 apresentamos as considerações finais do trabalho.

2. Revisão Bibliográfica

A programação da produção ou s*cheduling* surgiu como uma área de pesquisa na década de 1950 com a publicação do trabalho de Johnson (1954). Desde então, a teoria de *scheduling* vem recebendo considerável atenção de estudiosos da pesquisa operacional, matemáticos e pesquisadores da área de produção e operações. Nos últimos 60 anos uma significativaquantidade de obras que tratam dos problemas de programação foi formulada utilizando diversas configurações de máquinas, diferentes restrições e variadas funções objetivo.

A maioria das pesquisas que tratam do problema em ambiente *flow shop* permutacional considera como objetivo a minimização da duração total da programação (*makespan*), associada à

utilização eficiente dos recursos produtivos, e a minimização do tempo médio de fluxo (*mean flow time*), equivalente à redução do estoque em processamento.

Foram encontrados poucos trabalhos na literatura que tratam do ambiente *flow shop* permutacional enfocando aminimização do número de tarefas atrasadas (*tardy jobs*). Este critério é uma medida de desempenho que deve ser monitorada e pode ser facilmente calculada, é também uma das condições mais importantes para se evitar perdas de clientes e contratos. As pesquisas publicadas que consideraram a minimização do número de tarefas atrasadas em *flow shop* são descritas a seguir.

Hariri e Potts (1989) desenvolveram um método baseado no algoritmo *branch and bound* para o problema de minimização do número de tarefas atrasadas em um ambiente *flow shop* permutacional. Por se tratar de um método exato, algoritmo proposto fornece a solução em tempo computacional aceitável para problemas com até 20 tarefas e 3 máquinas. Um algoritmo baseado no método *branch and bound* também foi apresentado por Bulfin e M'Hallah (2003) para minimizar o número mínimo ponderado de tarefas atrasadas em um ambiente *flow shop* com duas maquinas. Uma extensiva experimentação computacional foi implementada e indicou que problemas com cem tarefas podem ser solucionados rapidamente.

Gupta e Hariri (1997) apresentaram quatro casos especiais em *flow shop* com duas máquinas. Foram propostas várias heurísticas construtivas, sendo que duas delas apresentaram soluções muito próximas da ótima durante a experimentação computacional. Os autores apresentaram ainda 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, considerando três objetivos em um sistema de produção *flow shop*: minimização do atraso total, do *makespan* e do número de tarefas atrasadas. Os resultados indicaram que o algoritmo genético foi eficiente para solucionar problemas de médio e grande porte com aceitável esforço computacional.

A meta-heurística PSO (*Particle Swarm Optimization*), com base na regra SPV (*Smallest Position Value*), foi apresentada e comparada com o algoritmo genético por Ucar e Tasgetiren (2006). A PSO apresentou o menor número de tarefas atrasadas, enquanto o algoritmo genético solucionou os problemas em menor tempo computacional.

Um problema muito particular é abordado por Mosheiov e Sarig (2010). Os autores estudaram um *flow shop* 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, baseada em um algoritmo de programação pseudopolinomial demonstrou desempenho eficiente, fornecendo programações muito próximas da ótima.

Varmazyar e Salmasi (2012a) enfocaram a minimização do número de tarefas atrasadas em um ambiente *Flow Shop* com tempos de *setup* dependentes da sequência. Vários algoritmos baseados em busca tabu e no algoritmo competitivo imperialista (ICA, em inglês) foram desenvolvidos. 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.

Três meta-heurísticasbaseadas na busca tabu foram propostas por Varmazyar e Salmasi (2012) para resolver o problema de minimização do número de tarefas atrasadas em um ambiente *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.

Todos os trabalhos descritos consideraram prazos individuais para cada tarefa. Entretanto, existem também pesquisas que utilizaram o mesmo prazo para todas as tarefas (*common due date*), como é o caso de Józefowska, Jurisch e Kubiak (1994), Della Croce, Gupta e Tadei (2000) e Panwalkar e Koulamas (2012), todos considerando o problema de *flow shop* com duas máquinas.

3. Métodos de Solução Propostos

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 minimizar o número de tarefas atrasadas.

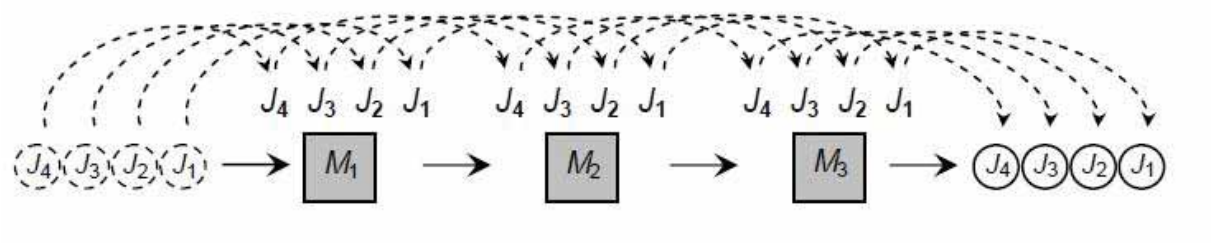

Figura 1: Ilustração de um *flow shop* permutacional

Foram propostas oito heurísticas, sendo quatro exclusivamente construtivas (H1, H3, H5 e H7) e quatro incluindo etapas de melhoria (H2, H4, H6 e H8). As idéias que nortearam a concepção dos métodos baseiam-se em algoritmos clássicos para outros problemas, como a heurística NEH, de Nawaz, Enscore Jr. e Ham (1983), conhecida pelos bons resultados em minimização do *makespan* em *flow shop* permutacional, 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, também 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 que consideraram etapas de melhorias, foram implementadas as buscas em vizinhanças de inserção e de permutação.

Em resumo, as oito heurísticas resultam da combinação de duas estratégias de ordenação (EDD e MST), de duas estratégias construtivas (NEH e Hodgson) e da aplicação ou não do método de busca local (inserção e permutação).

Será apresentado a seguir o funcionamento de cada heurística desenvolvida.

HEURÍSTICA H1:

PASSO 1–Ordene astarefas pela regra EDD.

PASSO 2– Com as duas primeiras tarefas da ordenação obtida, encontre a subsequência (entre as duas possíveis) com o menor número de tarefas atrasadas.Desempate pelo maior adiantamento total e se houver um segundo empate, pelo maior atraso total.

PASSO 3– Sem alterar as posições relativas das tarefas já alocadas, insira a próxima tarefa da ordenação obtida no Passo 1 em todas as posições possíveis da subsequência atual e considere a que fornece o menor número de tarefas atrasadas. Desempate pelo maior adiantamento total e se houver um segundo empate, pelo menor atraso total.

PASSO 4– Repita o Passo 3 até que todas tarefas estejam programadas.

HEURÍSTICA H2:

PASSOS 1 A 4–Idem à heurística H1.

PASSO 5–Considerando toda a Vizinhança de Inserção da sequência com $(n-1)^2$ soluções, determine aquela com melhor n_T.
Considerando toda a Vizinhança de Permutação da sequência com $n(n-1)/2$, determine aquela com melhor n_T.

HEURÍSTICA H3:

PASSO 1–Ordene as tarefas pela regra MST.

PASSO 2 a 4– Idem à heurística H1.

HEURÍSTICA H4:

PASSO 1– Ordene as tarefas pela regra MST.

PASSO 2 a 4– Idem à heurística H1.

PASSO 5–Considerando toda a Vizinhança de Inserção da sequência com $(n-1)^2$ soluções, determine aquela com melhor n_T.

Considerando toda a Vizinhança de Permutação da sequência com $n(n–1)/2$, determine aquela com melhor n_T.

HEURÍSTICA H5:

PASSO 1– Ordene astarefas pela regra EDD.

PASSO 2– Se não houver tarefas atrasadas na sequência atual, a sequência total é a sequência ótima. Caso contrário, identifique a primeira tarefa atrasada nasequência atual (J_T).

PASSO 3– Da primeira tarefa até J_T, remova a tarefa com maior soma dos tempos deprocessamento em todas as máquinas para a lista de tarefas removidas (sequência de tarefasapós a sequência atual). Vá para o Passo 2.

HEURÍSTICA H6:

PASSO 1 A 3– Idem à heurística H5.

PASSO 4–Considerando toda a Vizinhança de Inserção da sequência com $(n–1)^2$ soluções, determine aquela com melhor n_T.

Considerando toda a Vizinhança de Permutação da sequência com $n(n–1)/2$, determine aquela com melhor n_T.

HEURÍSTICA H7:

PASSO 1– Ordene astarefas pela regra MST.

PASSO 2 a 3– Idem à heurística H5.

HEURÍSTICA H8:

PASSO 1– Ordene as tarefas pela regra MST.

PASSO 2 a 3– Idem à heurística H5.

PASSO 4–Considerando toda a Vizinhança de Inserção da sequência com $(n–1)^2$ soluções, determine aquela com melhor n_T.

Considerando toda a Vizinhança de Permutação da sequência com $n(n–1)/2$, determine aquela com melhor n_T.

4. Experimentação computacional e resultados

4.1 Metodologia da pesquisa

Segundo as definições de Martins (2010, p.45-49) e Nakano (2010, p.64), esta pesquisa possui abordagem *quantitativa*, pois há preocupação com mensurabilidade, causalidade, generalização e replicação. Pode também ser classificada como *experimento*, uma vez que testa o relacionamento entre as variáveis da pesquisa operacionalizada, manipulando variáveis independentes (neste caso, a programação do problema) para se observar o resultado na variável dependente (representada aqui pela medida de desempenho, ou seja, o 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 problemas, divididos em dois grupos: Grupo 1, com problemas de pequeno porte e Grupo 2, com problemas de médio e grande portes. As classes de problemas 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 problemas 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 problemas 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).

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

Foi utilizado o sistema operacional Windows e o ambiente de programação Delphi. As configurações da máquina foram as seguintes: processador Pentium Dual-Core com 2.0 GHz de frequência e 3.0 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 minimização do número de tarefas atrasadas, o valor ótimo da função-objetivo em muitos problemas-testes pode ser zero. Por este motivo, o RDI é mais apropriado, sendo calculado para um determinado método de solução da seguinte forma:

$$RDI = \left(\frac{(n - n^{*}) - (n - n^{m})}{n - n^{*}}\right).100, \qquad (4.1)$$

Onde n é o número total de tarefas do problema, n^{*} é o número mínimo de tarefas atrasadas (sendo o valor da solução ótima no Grupo 1 e o melhor valor obtido dentre os métodos no Grupo 2) e n^{m} é o número de tarefas atrasadas obtido pelo método de solução que está sendo avaliado. Assim, $n - n^{m}$ representa o número de tarefas sem atraso do problema e $n - n^{*}$, o número máximo de tarefas sem atraso. Simplificadamente, tem-se a expressão:

$$RDI = \left(\frac{f_h - f^*}{f - f^*}\right).100. \quad (4.2)$$

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.

O valor do RDI indica o desvio percentual da solução do método avaliado, tanto em relação ao número mínimo de tarefas atrasadas como também relativo ao número total de tarefas do problema. Isto permite a comparação de desempenho dos métodos nos problemas com diferentes números de tarefas, pois considerar-se simplesmente o número absoluto de tarefas atrasadas não fornece base equitativa de comparação entre os problemas-testes.

O RDI também é eficaz na identificação dos casos em que o método heurístico forneceu a solução ótima ou a melhor solução, obtendo valor zero (pois $f_h = f^*$). Isto não ocorre, por exemplo, ao se considerar como medida comparação o percentual de tarefas atrasadas ($\%T_h = (T_h/n).100$).

Para a comparação do desempenho dos problemas do Grupo 1, a solução ótima foi obtida pelo método de enumeração completa, enquanto no Grupo 2, foi considerada a melhor solução fornecida pelos métodos implementados.

4.4 Análise dos resultados

Os resultados obtidos na experimentação computacional foram submetidos à análisedo RDI e o tempo médio de computaçãodos métodos apresentados.É importante observar que já era previsto que na análise dos pares de heurísticas com e sem busca local, ou seja, na comparação de H1 com H2, H3 com H4, H5 com H6 e H7 com H8, o método com busca local seria no mínimo tão bom quanto o seu equivalente sem busca local e com custo computacional maior. Contudo, a comparação entre os pares citados visou *quantificar* a eficácia e o custo do procedimento de busca local nos métodos empregados.

Os resultados globais das heurísticas propostas mostraram que a H1 e H2 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 viável.

A tabela 1, expressa os dados globais numericamente, destacando os melhores e os piores resultados. Destacados 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 um dos cenários considerados, a heurística H2 obteve o melhor desempenho, em muitos casos com RDI próximo de zero. Além disso, a análise dos resultados também revelou que a heurística H2 forneceu a solução ótima em 91,9% dos problemas do Grupo 1. E considerando conjuntamente as oito heurísticas, a solução ótima foi obtida em 96,6% dos casos. Também pode-se notar na Tabela 1 que em todos os pares de heurísticas, houve redução do RDI da heurística com melhoria em relação à apenas construtiva. Isto comprova a eficácia da etapa de buscas nas vizinhanças de inserção e de permutação.

Dentre as oito heurísticas propostas, a heurística H7 obteve os piores resultados em termos de RDI, com desempenhos bem distantes da solução ótima.

	Cenário	H1	H2	H3	H4	H5	H6	H7	H8
Grupo 1	1	**3,22**	1,83	7,85	4,28	14,42	4,52	30,68	9,12
	2	**0,73**	0,48	5,75	3,34	5,27	1,35	27,82	6,40
	3	**3,85**	2,45	7,10	4,42	20,22	5,73	39,97	11,46
	4	**0,73**	0,45	4,06	2,52	6,80	0,99	34,55	6,61
Grupo 2	1	**1,77**	0,35	11,94	9,29	10,70	6,77	38,44	27,87
	2	**0,37**	0,21	15,43	13,24	2,89	1,52	40,33	30,35
	3	**2,61**	0,69	11,32	8,49	13,12	7,76	49,68	35,98

	4	**0,50**	0,29	15,76	14,08	4,03	2,05	51,73	39,27

Tabela 1 - Resultados da Análise Global do RDI das Heurísticas

Para melhor visualização e entendimento dos resultados globais, os gráficos das Figuras 2 e 3 apresentam os RDI do Grupo 1 e do Grupo 2 respectivamente e de forma separada para cada cenário.

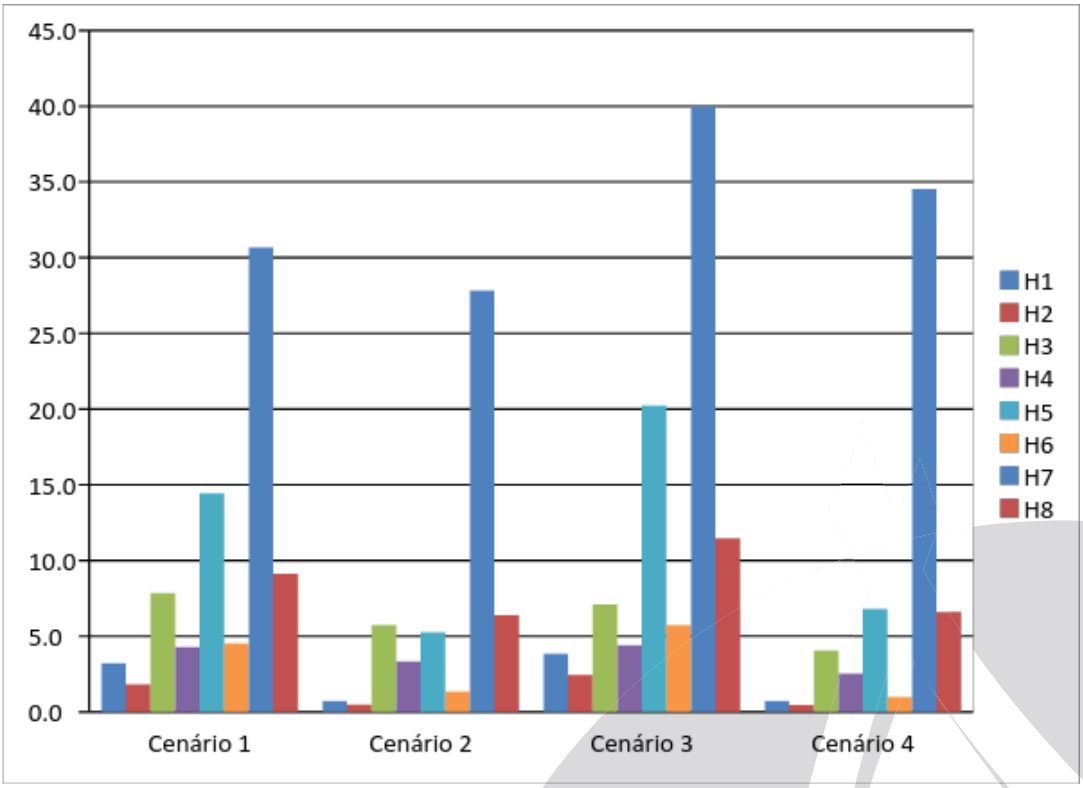

Figura 2 – Comparação dos resultados globais do RDI das heurísticas no Grupo 1

Como pode ser visto na Figura 2, no Grupo 1 os desempenhos das heurísticas H1 e H2 possuem valores de RDI muito pequenos, ou seja, resultados bastante próximos à solução ótima, principalmente nos Cenários 2 e 4. Além disso, é possível visualizar melhor a discrepância dos desempenhos das heurísticas H7 e H5, principalmente no Cenário 3.

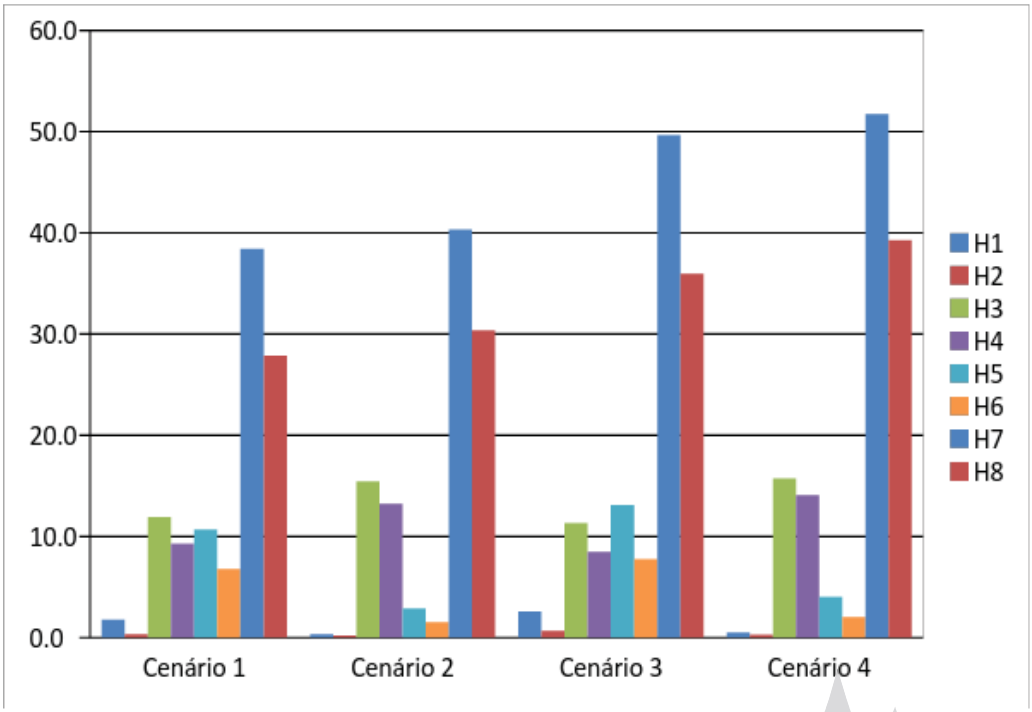

Figura 3 – Comparação dos resultados globais do RDI das heurísticas no Grupo 2

A Figura 3, com os resultados do Grupo 2, expõe a diferença no desempenhodas heurísticas H7 e H8, mais visíveis nos Cenários 3 e 4. Já os resultados das heurísticas H1 e H2 ficaram bem próximos entre si principalmente nos Cenários 2 e 4.

Visando o melhor entendimento dos resultados das heurísticas propostas, a seguir é apresentada a Tabela 2 que ilustra numericamente as oito heurísticas construtivas propostas e os respectivos valores do RDI, para cada parâmetro (número de tarefas e de máquinas, por grupo). Esta análise visa detectar a existência de resultados diferentes dos analisados anteriormente para alguma classe específica de problemas e que não se possa perceber em resultados baseados em média.

			H1	H2	H3	H4	H5	H6	H7	H8
G1	n	5	1,12	0,40	2,93	1,18	12,01	1,20	31,30	3,97
		6	1,70	0,89	4,50	2,26	11,96	2,23	32,62	6,28
		7	1,99	1,15	6,20	3,63	11,80	3,23	33,01	8,28
		8	2,86	1,88	7,95	4,88	11,75	4,22	34,85	10,17
		10	3,01	2,19	9,37	6,26	10,87	4,85	34,49	13,28
G2		15	2,02	1,03	10,11	7,47	11,70	5,33	49,63	25,67
		20	1,78	0,63	12,01	9,08	10,39	5,48	49,30	30,77
		30	1,53	0,40	14,35	11,48	8,43	5,04	46,64	34,51
		50	1,06	0,15	16,06	13,65	6,40	4,32	43,93	36,64
		80	0,81	0,05	17,20	15,31	4,98	3,72	41,26	36,63
		100	0,69	0,05	11,94	10,66	4,21	3,26	39,52	35,98
G1	m	2	1,80	1,22	6,33	3,47	9,61	2,58	28,82	6,93
		3	2,32	1,39	6,36	3,94	12,00	3,50	33,62	8,96
		5	2,29	1,30	5,88	3,51	13,43	3,36	37,32	9,30
G2		5	0,85	0,22	14,35	11,72	6,42	3,98	38,95	29,00
		10	1,34	0,40	12,98	10,74	7,89	4,75	41,78	30,95

		15	1,51	0,46	13,54	11,31	8,29	4,86	47,81	35,50
		20	1,56	0,46	13,58	11,34	8,14	4,51	51,65	38,03

Tabela 2 - Valores do RDI das heurísticas detalhadas por parâmetros

A análise detalha apenas comprovou as inferências observadas nas análises anteriores, especialmente em relação aos dois melhores métodos e também ao pior.

Para os tempos computacionais das heurísticas foram elaboradas as Tabelas 3 e 4. A seguir, os tempos de execução dos problemas do Grupo 1.O tempo médio de computação é obtido realizando a soma do tempo de computação detodos os problemas, dividido pelo total de problemas.A utilização deste critério de comparação fornece informações sobre qual dos métodos é maisrápido na busca por soluções, o método que apresentar a menor média é considerado o maisrápido, apresentando maior eficiência computacional.

Grupo 1								
H1	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**	**EC**
0,042000	0,134833	0,031667	0,122833	0,002667	0,098667	0,000000	0,112667	1178,955000

Tabela 3 – Tempos Médios de CPU das Heurísticas do Grupo 1 (em milissegundos)

Como pode ser visto na Tabela 3, o tempo de CPU da heurística H7 foi zero. Isso por se tratar de uma heurística de estrutura simples e no caso de problemas de pequeno porte. Neste caso,percebe-se que o computador não foi capaz de medir a precisão decimal o tempo de CPU gasto.

Ainda sobre a Tabela 3, a última coluna mostra o tempo de CPU consumido pelo método de enumeração completa, bem maior do que os demais.

A Tabela 4 apresenta os tempos de CPU das heurísticas do Grupo 2, em milissegundos. Na resolução do Grupo 2, naturalmente as heurísticas levaram mais tempo de CPU do que as do Grupo 1, já que se trata de problemas de médio e grande portes.

Grupo 2							
H1	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**
30,447500	144,345313	29,720104	144,653021	0,188021	113,015938	0,488542	114,318333

Tabela 4 – Tempos Médios de CPU das Heurísticas do Grupo 2 (em milissegundos)

Conforme esperado, nas heurísticas pares o tempo de CPU foi maior do que para as ímpares, isso devido à etapa de melhoria inserida nestas.Portanto, sugere-se a aplicação da heurística H2, que obteve o melhor desempenho em todas as classes de problemas, com tempo de CPU aceitável, mesmo para problemas de grande porte.

5. Considerações Finais

Neste trabalho foram apresentados e avaliados quatro métodos heurísticos construtivos e quatro métodos heurísticos melhorativos para o problema de programação da produção em sistemas *flow shop* permutacionais, em que há presença de prazos de entrega com o objetivo de minimizar o número de tarefas atrasadas.

Foram propostas oito heurísticas, sendo quatro exclusivamente construtivas (H1, H3, H5 e H7) e quatro incluindo etapas de melhoria (H2, H4, H6 e H8). As idéias que nortearam a concepção dos métodos baseiam-se em algoritmos clássicos para outros problemas, como a heurística NEH, de Nawaz, Enscore Jr. e Ham (1983), conhecida pelos bons resultados em minimização do *makespan* em *flow shop* permutacional, e o algoritmo de Hodgson (1977), que fornece a solução ótima para minimização do número de tarefas atrasadas em máquina única.

Foram testados e avaliados 15.600 problemas, de pequeno, médio e grande portes e osresultados experimentais indicaram quepara o Grupo 1, com problemas com até 10 tarefas e 5 máquinas, as heurísticas H1 e H2 possuem desvios relativosmuito pequenos, ou seja, a solução é bem

próximada ótima, principalmente nos Cenários 2 e 4.No Grupo 2 as heurísticas H2, H1, H6 e H5, apresentaram na média, nesta ordem, resultados muito próximos ao ótimo.

A pior heurística construtiva (H7) teve índices de desvios relativos bem maiores do que as demais, porém tempo de CPU muito menor, sem, entretanto, justificar o seu uso pela falta de qualidade da solução.

Para o desenvolvimento de futuros trabalhos, sugere-se que o ambiente de produção seja avaliado com outras medidas de desempenho, como atraso máximoe atraso total, ou até mesmo com bicritério.

Referências

Armentano, V.A. e Ronconi, D.P.(2000), Minimização do Tempo Total de Atraso no Problema de Flowshop com Buffer Zero Utilizando Busca Tabu, *Gestão & Produção*,7 (3), 353-363.

Della Croce, F., Gupta, J.N.D. e Tadei, R. (2000), Minimizing tardy jobs in a flow shop with common due date, *European Journal of Operational Research*, 120, 375–81.

Fuchigami, H.Y. (2005), Métodos heurísticos construtivos para o problema deprogramação da produção em sistemas flow shop híbridos com tempos de preparação dasmáquinas assimétricos e dependentes da sequência. *Dissertação (Mestrado)* – Escola deEngenharia de São Carlos, Universidade de São Paulo, São Carlos.

Gupta J.N.D. e Hariri A.M.A. (1997), Two machine flow shop to minimize number of tardy jobs. *Journal of the Operational Research Society*, 48, 212–20.

Hariri A.M.A. e Potts C.N. (1989), A branch and bound algorithm to minimize number of late jobs in a permutationflow-shop *Eur J Opl Res*,38, 228–237.

Hodgson, T.J. (1977), Note - A Note on Single Machine Sequencing with Random Processing Times. *Management Science*,23(10), 1144-1146.

Johnson, S.M. (1954), Optimal two and three-stage production schedules with set-uptimes included. *Naval Research Logistics Quarterly*,1–68.

Jozefowska, J., Jurisch, B. eKubiak, W., (1994), Scheduling shops to minimize the weighted number of late jobs. *OperationsResearchLetters*, 10, 27–33.

Jung, C.F. (2004), Metodologia *para pesquisa & desenvolvimento: aplicada a novastecnologias, produtos e processos*, Rio de Janeiro: Axcel Books.

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. e Sarig, A. (2010), Minimum weighted number of tardy jobs on an m-machine flow-shop with a critical machine.*EuropeanJournalofOperationalResearch*, 201, 404-408.

Nakano, D. (2010), Métodos de Pesquisa Adotados na Engenharia de Produção eGestão 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. 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.

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

Ucar, H. e Tasgetiren, M.F. (2006), A particle swarm optimization algorithm for permutation flow shop sequencing problem with the number of tardy jobs criterion, *5th International Symposium on intelligent manufacturing systems: Agents and virtual worlds*, May 29-31, 2006, Sakarya, Turkey.

Panwalkar, S. S. e Koulamas, C. (2012),An O(n^2) algorithm for the variable common due date, minimal tardy jobs bicriteria two-machine flow shop problem with ordered machines. *European Journal of Operational Research*. 211, 7-13.

Ronconi, D.P. (1997), Uma contribuição para oestudo do problema de flow shop com buffer ilimitado e zero para minimizar a soma dos atrasos. *Tese (Doutorado).*Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas.

Ronconi, D.P. e 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.
Taillard, E.D. (1993), Parallel Iterative Search Methods for Vehicle Routing Problems, *Networks 23*, 661 – 676.
Varmazyar, M. e 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. e 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.