UNIVERSIDADE CANDIDO MENDES
MESTRADO PROFISSIONAL EM PESQUISA OPERACIONAL
E INTELIGÊNCIA COMPUTACIONAL

Márcio de Souza Leite

Métodos Heurísticos para o Problema FlowShop Permutacional Multiobjetivo

CAMPOS DOS GOYTACAZES
2008

UNIVERSIDADE CANDIDO MENDES
MESTRADO PROFISSIONAL EM PESQUISA OPERACIONAL
E INTELIGÊNCIA COMPUTACIONAL

Márcio de Souza Leite

Métodos Heurísticos para o Problema FlowShop Permutacional Multiobjetivo

Dissertação apresentada ao Programa de Pós-Graduação, Universidades Candido Mendes – Campos/RJ, para a obtenção do grau de mestre em Pesquisa Operacional e Inteligência Computacional.

Orientador: Prof. Dalessandro Soares Vianna
Co-orientador: Prof. José Elias Cláudio Arroyo

CAMPOS DOS GOYTACAZES, RJ
2008

MÁRCIO DE SOUZA LEITE

Métodos Heurísticos para o Problema FlowShop Permutacional Multi-objetivo

Dissertação apresentada ao Programa de Pós-Graduação, Universidades Candido Mendes – Campos/RJ, para a obtenção do grau de mestre em PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL.

Aprovada em __/__/___

BANCA EXAMINADORA

_____

(Orientador: Dalessandro Soares Vianna)

(Universidade Cândido Mendes)

_____

(Coorientador: José Elias Claudio Arroyo)

(Universidade Federal de Voçosa)

_____

(Helder Gomes Costa)

(Universidade Federal Fluminense)

_____

(Fermin Alfredo Tang Montané)

(Universidade Cândido Mendes)

CAMPOS DOS GOYTACAZES, RJ

2008

Dedico esse trabalho a meu grande amigo Gabriel, exemplo de inteligência, força, honestidade e paciência. Amigo esse, que tenho o orgulho de chamar de PAI.

AGRADECIMENTOS

A Deus pelo dom da vida;

Aos meus Pais, pelo amor e pela presença em minha vida, sempre auxiliando meus caminhos e apoiando minhas decisões;

A minha esposa Joelma, que mesmo nos momentos mais difíceis não deixou de acreditar em mim.

A Alanis e Miguel, que mesmo sem entender, aceitaram meus períodos de ausência;

A Fábrica Boechat LTDA, por financiar parcialmente o desenvolvimento desse trabalho;

Ao professor Dalessandro Vianna, pela dedicação, entusiasmo e pelas valiosas contribuições a esse trabalho.

Ao professor José Elias, por me mostrar um mundo novo e fazer-me repensar vários conceitos;

Aos colegas do MIA2005, aonde expresso além do agradecimento o meu respeito;

Aos professores do programa de mestrado em Pesquisa Operacional e Inteligência Computacional da Universidade Cândido Mendes**.**

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), a Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), ao Parque de Alta Tecnologia do Norte Fluminense (TECNORTE) e a Fundação Estadual do Norte Fluminense (FENORTE) pelo apoio a realização deste trabalho.

RESUMO

O Problema FlowShop Permutacional é um problema clássico da literatura na área de seqüenciamento. Nesse trabalho é abordado o Problema FlowShop Permutacional Multi-objetivo, onde são otimizados dois objetivos: makespan (C_{max}) e Atraso Máximo (T_{max}). São utilizados métodos heurísticos baseados no conceito de Dominância de Pareto para demonstrar a aderências desse métodos a problemas de otimização Multi-objetivo.

Os métodos heurísticos são procedimentos que utilizam-se de estratégias inteligentes para buscar soluções em um espaço consideravelmente grande, como é o caso do Problema FlowShop Permutacional Multi-objetivo. São implementados, neste trabalho, métodos baseados nas Metaheurística Busca Tabu e GRASP. A questão Multi-objetivo aumenta a complexidade do problema, visto que, em geral os objetivos são conflitantes e quando se minimiza um, geralmente está se maximizando o outro. Por isso, o espaço de soluções em um problema Multi-objetivo não é completamente ordenado e sim parcialmente ordenado, pois não se pode afirmar que uma solução é melhor ou pior que uma outra solução levando em consideração todos os objetivos. Baseado nisso, foi usado nesse trabalho o conceito de Dominância de Pareto.

O Trabalho apresenta como contribuição: o desenvolvimento de uma estratégia de distribuição dos pesos entre os objetivos, aonde é usado um fator de influência baseado no limitante inferior e superior de cada objetivo; é proposta uma variação da heurística NEH; e uma implementação das Metaheurística GRASP e Busca Tabu. São realizados testes com um conjunto de 360 problemas, e os resultados comparados com uma referência da literatura científica.

PALAVRAS-CHAVES: Metaheurística, GRASP, Busca Tabu, Flowshop, Multi-objetivo.

ABSTRACT

The Flowshop Problem is a classical problem in scheduling area. This research has as focus the Flowshop Multiobjective problem, where are optimized two objective: makespan (C_{max}) and Maximum Tardiness (T_{max}). In order to resolve this problem is used heuristic methods based in Pareto´s Dominance.

Heuristics Methods are procedures that have smarts strategies to find solutions in the space considerably large, as the case of Flowshop Problem. The multiobjetive factor increases the problem's complexity because generally the objectives are conflicting, where one objective is minimizing, another is maximizing. In these Multobjetives problems the solutions space is partially ordered, because it can not say that one solution is better or worst than another one regarding all objectives.

In this work are presented as contributions: the developing of new strategies to distribute weights between objective, using the lower and upper bound; the proposal a variation of NEH procedure; and the implementation of metaheuristics GRASP and Tabu Search. Test with 360 problems was done and compared with results of scientific literature.

KEYWORDS: Metaheuristics, GRASP, Tabu Search, Flowshop, Multiobjective.

LISTA DE TABELAS

Lista de Figuras

SUMÁRIO

1 Introdução

Com a unificação dos mercados e a alto grau de concorrência as empresas, principalmente as indústrias, têm sido obrigadas a desenvolver novas ações de administração da produção com o intuito de otimizar em suas operações. Dentre esses desenvolvimentos pode-se citar a técnica de *Just-in-Time*, no que diz respeito a redução de estoques em processo e como conseqüência a redução do dispêndio financeiro para manter esse estoque. Outra técnica de otimização dos recursos da produção vem do campo da Pesquisa Operacional, principalmente da otimização no sequenciamento das tarefas no ambiente de chão-de-fábrica.

O problema de otimização de seqüência das tarefas correspondem a determinar a seqüência na qual as tarefas devem ser executadas com o intuito de otimizar um ou mais objetivos. É uma extensa área de pesquisa que tem merecido grande atenção devido a sua relevância na otimização dos processos produtivos. Os problemas de *scheduling* ou escalonamento, como são conhecidos os problemas de sequenciamento de tarefas, além de importantes para a indústria no que diz respeito em otimizar os recurso de produção (Homens e Máquinas), são matematicamente desafiadores devido a grande explosão combinatória, muita vezes de ordem exponencial.

Muitas vezes no mundo real existe a necessidade de se otimizar mais de um objetivo, buscando dessa forma a eficácia do sistema produtivo. Entretanto é comum encontrar objetivos conflitantes aonde a melhoria de um objetivo significa a piora de outros objetivos. Como exemplo: a relação entre os objetivos *makespan* (C_{max}) e Atraso Máximo (T_{max}). O primeiro tem como objetivo minimizar o tempo total de

produção, enquanto o segundo está relacionado maximizar a satisfação dos clientes em receber seus produtos na data previamente acordada.

A otimização de problemas multiobjetivo tem merecido grande atenção dos pesquisadores nos últimos anos, aonde são aplicadas diversas técnicas para se conseguir bons resultados em tempo computacional factível ao dinamismo do setor produtivo. Ao contrário da otimização Monoobjetivo, na otimização de problemas Multiobjetivos, em geral, não existem soluções ótimas no sentido de minimizarem (ou maximizarem) individualmente todos os objetivos. A característica principal da otimização Multi-objetivo (quando todos os objetivos são de igual importância) é a existência de um conjunto de soluções aceitáveis que são superiores às demais. Quanto a perspectiva do decisor (*decison maker*), existem basicamente três formas de resolução de problemas Multi-objetivos. Na primeira o decisor define pesos para os objetivos e a busca por soluções é guiada estritamente naquela ponderação. Na segundo forma o decisor guia a busca de maneira iterativa, a todo momento que é necessário uma decisão entre a ponderação dos objetivos. Na terceira forma são geradas diversas soluções de qualidade em diversas direções e após a busca o decisor escolhe a solução que melhor adere a sua necessidade, seguindo o conceito de dominância proposto por Pareto (1896, *apud* ARROYO, 2002). Estas soluções de qualidade são denominadas soluções *Pareto-ótimas* ou *eficientes*.

É tratado nesse trabalho um problema relacionado ao ambiente de produção visando otimizar mais de um objetivo, denominado Problema FlowShop Permutacional Multi-objetivo. É utilizado o conceito de dominância de pareto caracterizando assim o uso do método *a-posteriori* aonde o decisor faz a escolha das soluções após a busca.

A otimização do Problema FlowShop Permutacional Mono objetivo é considerado um desafio devido ao grande número de soluções a serem avaliadas, e a questão Multi-objetivo aumenta ainda mais o grau de complexidade do problema. É de grande importância para essa classe de problemas de otimização a escolha do método de resolução, ponderando sempre a razão da qualidade da solução com o esforço computacional necessário para encontrar essa solução. Não é conhecido um algoritmo, que possa em tempo polinomial para o Problema FlowShop Permutacional. Para tanto é necessário o uso de métodos heurísticos. Métodos esses que quando bem adaptados ao problema conseguem gerar soluções de boa qualidade em tempo compatível com a necessidade de rapidez (ARROYO, 2002).

Nesse trabalho as heurísticas são classificadas em três grupos, classificação essa dada pela forma que as heurísticas executam a busca no espaço de soluções. O primeiro grupo é chamado de Heurísticas Construtivas - algoritmos esses que utilizam de uma regra adaptada ao problema para que seja construída uma solução factível com um mínimo de qualidade. O segundo grupo é classificado como Heurística de Busca Local ou Busca em Vizinhanças - algoritmos que utilizam de uma solução inicial gerada por uma heurística construtiva para melhorar o resultado através de exploração dentro de uma estrutura de vizinhança. A qualidade da solução encontrada por uma heurística de Busca Local depende da qualidade da solução inicial e está limitado ao vale no espaço de soluções conhecido como ótimo local. O terceiro grupo é conhecido como Metaheurísticas. As Metaheurística são algoritmos que utilizam certas técnicas que permitem que a busca escape de ótimos locais e continue a procura pelo ponto ótimo – ou mais próximo - dentro do espaço de soluções, ponto esse conhecido como ótimo global. As metaheurísticas são métodos flexíveis e de fácil adaptação aos problemas Multiobjetivo, por essas características é grande o número de pesquisadores que utilizam essa técnica para encontrar soluções para problemas de otimização Multi-objetivo (ARROYO, 2002).

O objetivo desse trabalho é demonstrar a aderência dos métodos heurísticos a problemas ligados a produção, especificamente ao Problema do FlowShop Permutacional Multi-objetivo. São implementados diversas heurísticas, iniciando com uma adaptação do heurística construtiva NEH (NAWAZ *et al.*, 1983), onde é adicionado um fator de aleatoriedade. São implementadas duas estruturas de vizinhanças que são utilizadas no algoritmo de Busca Local. Quanto a metaheurística, são implementadas duas diferentes metaheurística, uma baseada em múltiplos reinícios denominada GRASP (FEO E RESENDE, 1995) e uma baseada em Busca Local com memória adaptativa denominada Busca Tabu (GLOVER, 1990).

O trabalho está dividido em seis Capítulos, onde no primeiro Capítulo é descrito os principais conceitos e métodos de otimização Multi-objetivo, as principais técnicas de utilizadas e as medidas de cardinalidade e distância usadas para comparar conjuntos de soluções *pareto-ótimos*.

O Capítulo 2 apresenta a formulação matemática do problema FlowShop Permutacional Multi-objetivo, as características que norteiam esse problema e uma revisão da literatura.

Os métodos heurísticos construtivos e de busca local são descritos no Capítulo 3, aonde é proposto uma variação da heurística construtiva NEH. Também é proposta uma nova estratégia de geração dos pesos necessários para direcionar a busca dentro do espaço de soluções factível. Por fim, é apresentado o resultado computacional comparando duas regras de despacho utilizadas na heurística NEH: LTP (*longest processing time*) proposto NAWAZ *et al.* (1983) e TLB (*tardiness lower bound*) proposto por Armentato e Ronconi (1999).

No Capítulo 4 são apresentadas as metaheurísticas GRASP e Busca Tabu e os detalhes de implementação destas.

No Capítulo 5 são apresentados os resultados computacionais das heurísticas implementadas e uma comparação com o algoritmo da literatura MOGLS (ARROYO, 2005).

As conclusões e os pontos para desenvolvimento futuro são enumeradas no Capítulo 6.

2 Otimização Multiobjetivo

2.1 Problema de Otimização Multiobjetivo

A resolução problema de otimização multiobjetivo consiste em obter um conjunto de soluções factíveis que satisfaçam o conjunto de restrições e otimize (Maximize/Minimize) um vetor de funções objetivas. Assim o problema de minimização multiobjetivo pode ser formalmente descrito da seguinte forma:

$$\text{Minimizar } z = f_1(x), f_2(x),..., f_r(x), z \in Z$$
$$\text{Sujeito a } x \in X$$

onde, *x* é o vetor de decisão, *z é o* vetor de critérios*, X* denota o espaço de soluções factíveis e $Z = f(X) = \{ z = f(x) \mid x \in X\}$ é a imagem de *X* denominado espaço objetivo factível. Note que a imagem de uma solução $x = (x_1, x_2, ..., x_n) \in X$ no espaço objetivo é um ponto $z = (z_1, z_2, ..., z_r) = f(x)$, tal que $z_j = f_j(x)$, $j = 1,..., r$ (*r* é o número de critérios).

No problema de otimização monoobjetivo, o espaço das soluções factíveis é completamente ordenado, o que permite saber se uma solução *x* é melhor ou pior que a solução *y*, visto que é sempre verdade que $f(x) \leq f(y)$ ou $f(x) \geq f(y)$. O objetivo então é otimizar o valor de *f*. Diferentement do problema monoobjetivo, os problemas multiobjetivo não têm seu espaço de soluções completamente ordenado e sim parcialmente ordenado (PARETO, 1896 *apud* ARROYO, 2002), pois geralmente os objetivos são conflitantes, onde minimizar um objetivo pode causar o

acréscimo de outros objetivos. A ordenação do espaço de soluções é a diferença básica entre os problemas de otimização monoobjetivos e multiobjetivos. Para os problemas multiobjetivo utiliza-se o conceito de relação de ordem parcial denominada *Dominância de Pareto*, definido a seguir:

Definição 1:

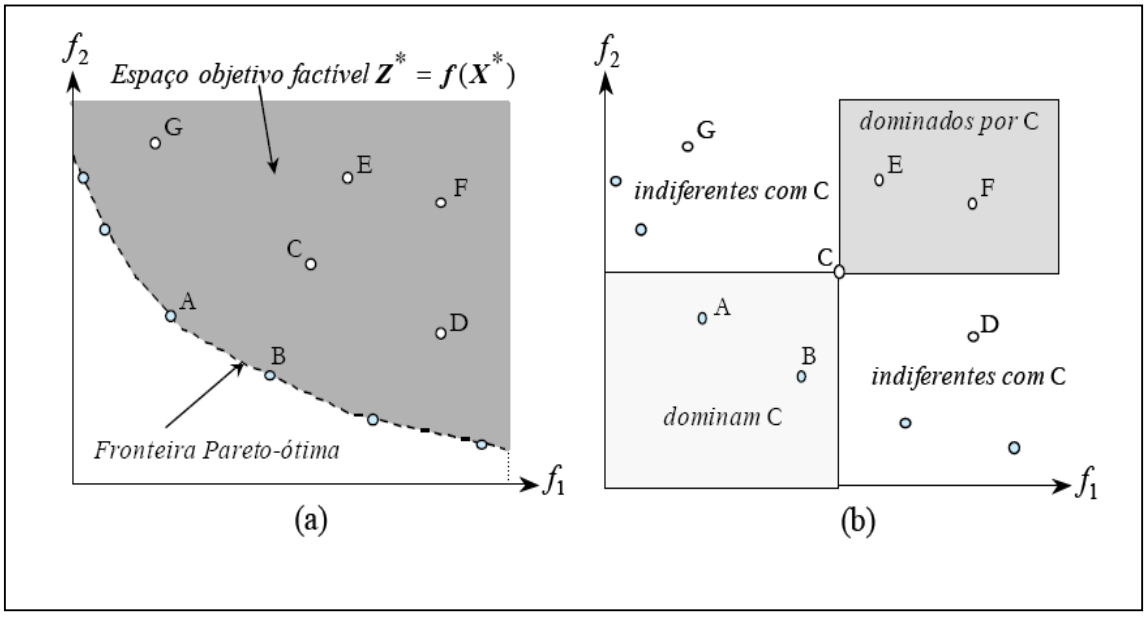
Dadas duas soluções x, $y \in X$ existem três possibilidades para seus correspondentes vetores objetivos:

1. Se $f_i(x) \leq f_i(y)\ \forall i \in \{1,...,r\}$: diz-se que o vetor objetivo x = $f_i(x)$ domina y = $f_i(\boldsymbol{y})$ e que a solução x domina a solução $\boldsymbol{y}$, se pelo menos um $f_i(x) < f_i(y)$;
2. Se $f_i(x) \geq f_i(y)\ \forall i \in \{1,...,r\}$: diz-se que o vetor objetivo x = $f_i(x)$ é dominado x = $f(y)$ e que a solução x dominado a solução y; ou,
3. Neste caso dizemos que x e y são indiferentes ou possuem o mesmo grau de dominância, visto que x não domina y**,** nem é dominado por y.

Definição 2:

Uma solução x é *Pareto-ótima* ou *eficiente* se ela não é dominada por nenhuma solução do espaço de soluções factíveis.

Figura 1: Dominância de Pareto no espaço objetivo. Fonte: (ARROYO, 2002)



Exemplo:

Sejam s_1, s_2 e s_3 três soluções com valores objetivos (70, 79), (62, 92) e (67, 106) respectivamente. Observa-se que s_2 domina s_3, s_1 e s_2 não são dominados (s_1 possui o menor valor de f_2 e s_2 possui o menor valor de f_1). Caso não existam outras soluções, s_1 e s_2 seriam as soluções *Pareto-ótimas*.

Na figura 1(a) é demonstrado que os pontos A e B delimitam a *Fronteira Pareto-ótima.* Na figura 1(b), o ponto C domina os pontos pertecentes ao retângulo superior direito (subconjunto de espaço objetivo). Os pontos pertencentes ao retângulo inferior esquerdo dominam o ponto C. Os pontos G, C e D são indiferentes.

O conjunto de todas as soluções dominantes é chamado de conjunto *pareto-ótimo* ou conjunto *eficiente*.

2.2 Classificação de Métodos Multi-objetivos

O principal aspecto na classificação dos métodos multiobjetivos é a tomada de decisão aonde a perspectiva do decisor é de suma importância. Cabe ao decisor fazer uma ponderação (*trade*-off) dos objetivos conflitante, pode-se classificar os métodos multiobjetivos segundo a perspectiva do decisor em duas categorias (MINELLA *et al*, 2008), abaixo descritas:

2.2.1 Métodos a-priori

Os métodos pertencentes a essa categoria são caracterizados pela participação do decisor antes do início do processo de busca por soluções. Para tanto, o decisor deve fazer a ponderação dos objetivos antes de resolver o problema. Os métodos *a-priori* são em geral mais simples de serem implementados e também mais eficientes, contudo é necessário que o decisor tenha o conhecimento das preferências antes da utilização do método, o que nem sempre é possível.

2.2.2 Métodos a-posteriori

Nos métodos *a-priori* a busca se concentra em um único ponto seguindo a direção determinada pelo decisor no *trade-off*, enquanto nos métodos *a-posteriori* procuram gerar o conjunto de soluções eficientes ou parte relevante deste conjunto para ser apresentada ao julgamento do decisor quanto a aderência de cada solução ao problema apresentado. De certo, os métodos *a-posteriori* requerem um maior esforço, uma vez que a geração do conjunto eficiente obtém todas as soluções que podem ser escolhidas pelo decisor. Esses métodos dependem menos do conhecimento do decisor, uma vez que a procura por soluções é de forma geral e uma vez desenvolvida uma metodologia para gerar o conjunto de soluções não-dominadas pode-se aplicá-la em diversas situações, independente de mudanças nas preferências do decisor.

2.3 Métodos Tradicionais de Otimização Multiobjetivo

A principal dificuldade na busca por soluções em um problema de otimização multiobjetivo é o conflito entre os objetivos. Os métodos tradicionais (COHON, 1978; STEUER, 1986) formulam o problema reduzindo a otimização vetorial a um problema de otimização escalar. Para que isso seja possível, os métodos tradicionais escalarizam os objetivos em um único objetivo. É importante ressaltar que tal formulação dos problemas multiobjetivos possuem restrições adicionais.

Nas subseções 1.3.1 e 1.3.2 são descritas duas abordagens tradicionais para otimização multiobjetivo. Na primeira abordagem atribuí-se pesos para cada critério e em seguida otimiza-se uma função definida pela combinação dos critérios (SRIDHAR e RAJENDRAN, 1996; CHAKRAVARTHY e RAJENDRAN, 1999; ALLAHVERDI, 2001; RAJENDRAN e ZIEGLER; 2003; ALLAHVERDI, 2003; ALLAHVERDI, 2004;). Na segunda abordagem os critérios são classificados em ordem decrescente de prioridade e sempre se otimiza o critério de menor prioridade respeitando o valor ótimo do critério de maior prioridade (GUPTA *et al*., 1999; GUPTA *et al*., 2000a; GUPTA *et al*., 2000b; FRAMINAN *et al*., 2002).

2.3.1 Método da Soma Ponderada

Esse método consiste em definir diferentes pesos para cada objetivo e então formar uma função *f* que combine linearmente os objetivos e seus pesos, transformando assim o problema multiobjeivo em um problema monoobjetivo. Formalmente, pode-se escrever a função de soma ponderada conforme formulação a seguir:

$$\text{Minimizar} \sum_{i=1}^{r} w_i . f_i(x)$$

$$\text{Sujeito a} \quad x \in X^*$$

onde $w_i \geq 0$ é o peso que representa a importância relativa do objetivo f_i comparado com os outros objetivos, geralmente os pesos são normalizados de forma que:

$$\sum_{i=1}^{r} w_i = 1$$

Para a resolução do problema multiobjetivo utilizando o método acima descrito, o decisor deve resolvê-lo iterativamente utilizando diversos vetores de pesos positivos. A definição dos pesos deve demonstrar a importância dos objetivos para o decisor. Apesar de ser um método simples, ele traz consigo a desvantagem de não conseguir gerar todas as soluções *Pareto-ótimas* quando o espaço objetivo não se trata de um espaço convexo (ARROYO, 2002). Isso porque o método consiste basicamente em gerar diferentes retas suportes definidas pelo vetor de pesos e quando o espaço de soluções é convexo nem todos os pontos *Pareto-ótimos* admitem tais retas.

2.3.2 Método do $\xi - \text{Re}\,strito$

Esse método prioriza os objetivos e resolve o problema para o primeiro objetivo – mais importante, em seguida resolve o problema para os demais objetivos sempre acrescentando ao próximo problema um limitante com o resultado do problema anterior. Formalmente pode ser definido assim:

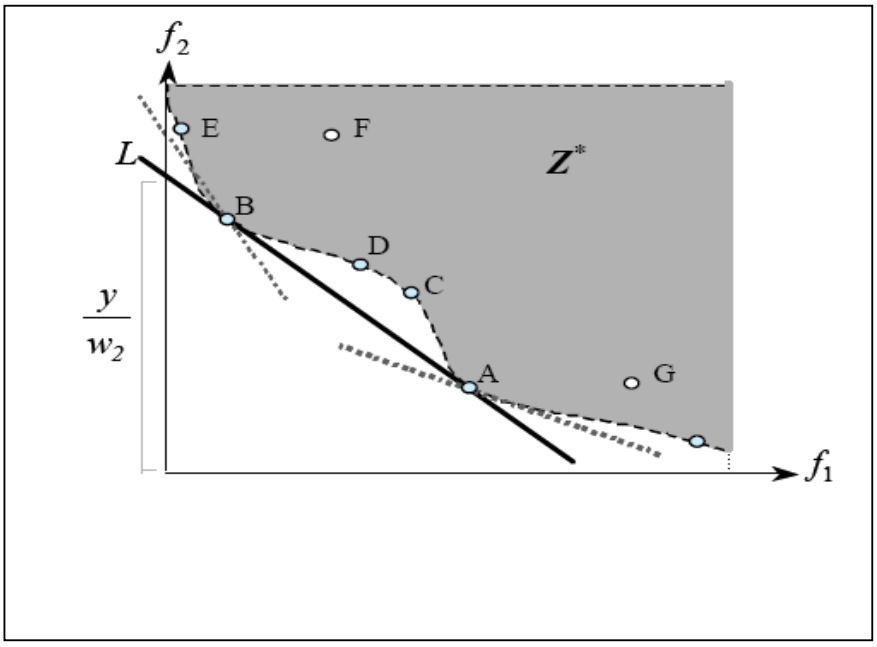
$$\text{Minimizar } f_1(x)$$

$$\text{Sujeito a } \quad f_i(x) \leq \xi_i, i = 2,...,r$$

$$x \in X^*$$

Variando os limitantes é possível gerar o conjunto *Pareto-ótimo*, mesmo em um espaço de soluções não-convexo. A desvantagem desse método é quando os limitantes não são bem selecionados, o que acaba acarretando em uma busca em subespaço vazio.

Figura 2: Interpretação gráfica do método da Soma Ponderada. Fonte (ARROYO, 2002)



2.4 Critérios de Avaliação de Heurísticas Multi-objetivo

Geralmente, os problemas de otimização Multi-objetivo são combinatórios de ordem exponencial, por isso, o esforço computacional é demasiadamente grande para se chegar a um conjunto de pontos-ideais. As heurísticas apresentam-se como uma solução com boa relação qualidade *versus* tempo de processamento para resolver de forma satisfatória problemas de otimização NP-difíceis (PINEDO, 1995).

Em um problema monoobjetivo é relativamente fácil a avaliação de qualidade de uma solução aproximada em relação ao ponto ótimo, bastando para isso calcular uma diferença relativa entre os valores. Mas tal premissa não pode ser usada em problema multiobjetivos, pois não existe uma medida simples que demonstre a qualidade de um conjunto aproximado ***H*** em relação ao conjunto de referência *Pareto-ótimo* ***R***.

Em (ARROYO, 2002) são apresentadas em detalhes várias técnicas de avaliação de heurísticas multiobjetivos, abaixo é apresentado um resumo de tais técnicas.

Avaliação Analítica de Daniels

Daniels (1992) apresenta um algoritmo polinomial que determina a qualidade de aproximação de pontos heurísticos em relação ao conjunto referência. Cada ponto eficiente é convertido em um valor dependente da importância associada em cada objetivo. Baseado nessa premissa é realizada uma varredura de pesos, e os pontos são comparados através do erro de aproximação relativo. O critério de avaliação baseia-se na compilação do erro médio e do erro máximo.

Medida de Cardinalidade

É a medida mais simples e natural; tem como base a porcentagem de pontos de referência – conjunto R – encontrados pelo método heurístico: $C_1(H) = \frac{|H \cap R|}{|H|} \text{x}\, 100$. A desvantagem desse método é a necessidade de se conhecer o conjunto de todos os pontos *Pareto-ótimo*, visto que em problema de tamanhos reais pode ser impossível obter em tempo factível uma porcentagem significativa de pontos *Pareto-ótimo*. Ainda para avaliação, é importante obter pontos aproximados distribuídos por toda a fronteira *Pareto-ótimo*, a medida C_1 não é apropriada para medir com qualidade tal distribuição, pois C_1 pode ser zero mesmo que *H* seja uma boa aproximação. Existe ainda uma segunda medida envolvendo cardinalidade, onde é definida a porcentagem de pontos não dominados pelos pontos de referência: $C_2(H) = \frac{|\{z \in H : \text{não existe } z' \in R \text{ tal que } z' \text{domina } z\}|}{|H|} \text{x}100$, tal medida também não considera a distribuição dos pontos e a distância em relação

aos pontos de referência, portanto não é considerada uma medida apropriada.

Medida de Distâncias de Czyzak e Jaszkiewics

Com o objetivo de evitar as desvantagens das medidas de cardinalidade, Czyzak e Jaskiewicz (1998) utilizam duas medidas de distâncias – D_{med} e D_{max} - para medir a proximidade do conjunto *H* e o conjunto de referência *R*. *H* é uma boa aproximação de *R*, se *H* fornece informações sobre todas as regiões do conjunto *R*, isto quer dizer, se para cada ponto $z \in R$ existe um ponto $z' \in H$ tal que a distância entre eles é pequena. As medidas de distância proposta por Czyzak e Jaskiewicz são definidas abaixo:

$$D_{med} = \frac{1}{|R|}\sum_{z\in R} \min_{z'\in H} d(z', z) \text{ e } D_{\max} = \max_{z\in R}\left\{\min_{z'\in H} d(z', z)\right\}$$

onde |*R*| é a cardinalidade do conjunto *R* e *d* é definido por

$$d(z', z) = \max_{j1,\ldots,r}\left\{\frac{1}{\Delta_j}\left(z'_j - z_j\right)\right\},$$

$z' = (z'_1,\ldots,z'_r) \in H,\ z = (z_1,\ldots,z_r) \in R$

onde: $\Delta_j = \max f_j - \min f_j$

com $\max f_j = \max\{z_j = f_j(x), f(x) \in H \cup R\}$ e

$\min f_j = mim\{z_j = f_j(x), f(x) \in H \cup R\}$

Nesse trabalho foram usados os métodos de *medida de cardinalidade* para definir os pontos *Pareto-ótimos* encontrados por uma heurística. Para determinar a distância entre os pontos foi utilizado o método de distância D_{med} e D_{max} proposto por Czyzak e Jaszkiewicz (1998), sendo diferente a forma de calcular a variação dos objetivos conforme Arroyo (2002).

Quando não se é conhecido o conjunto de soluções *Pareto-ótimo*, e é usado um outro conjunto de soluções H' aproximadas gerados por uma outra heurística, o conjunto referência R é definido pelas soluções dominantes de $H \cup H'$.

No Capítulo 2 é apresentada a formulação do Problema do Flowshop

Permutacional Multiobjetivo, objeto de estudo desse trabalho. É descrito formalmente as características do problema e apresentado as técnicas que serão utilizadas na busca por soluções para o problema.

3 FlowShop Permutacional Multiobjetivo

3.1 Introdução

Os problemas de escalonamento de tarefas são problemas de alocação de recursos para execução de atividades concorrentes. Esses problemas encontram-se intimamente ligados ao planejamento e programação da produção nas indústrias, por isso é conveniente que seja adotado a mesma termologia utilizada na indústria, aonde *tarefas* são atividades a serem executadas em *máquinas* que são recursos a serem alocados. Segundo Moccellin (1994), a programação refere-se à ordenação das tarefas a serem executadas, em uma ou diversas máquinas considerando-se uma base de tempo, ou seja, determinando-se, principalmente, as datas de início e fim de cada tarefa.

Em um ambiente real de chão-de-fábrica o numero de variáveis que podem compor o problema de escalonamento de tarefas é numerosa, complexa e específica variando de indústria a indústria. Por isso se faz necessário recorrer a modelos já definidos pela literatura para que seja possível comparar os resultados encontrados nesse trabalho com outro já publicado na literatura e também que seja de fácil entendimento as características do problema abordado. Dentre os mais estudados e conhecidos problemas de escalonamento existentes na literatura pode-se citar:

- **Job Shop**: cada tarefa tem ordem própria de processamento nas máquinas;

- **Flowshop**: todas as tarefas têm o mesmo fluxo de processamento nas máquinas;

- **Open Shop**: não existe um fluxo definido para as tarefas nas máquinas;

- **Flowshop Permutacional**: a ordem de todas as tarefas é a mesma em todas as máquinas;

Estes problemas, além de matematicamente desafiadores, são extremamente difíceis de serem resolvidos. A explosão combinatória decorrente da necessidade de se examinar as várias alternativas de escalonamento existentes nesses problemas torna difícil e custosa a obtenção de uma solução ótima ou aproximada, principalmente quando é utilizado a busca por soluções que tentam otimizar dois ou mais objetivos, conhecidos como problemas multiobjetivo que é o foco desse trabalho.

Este Capítulo descreve o problema de escalonamento de tarefas em um ambiente *Flowshop Permutacional*, os objetivos que norteiam a otimização desse problema, bem como uma revisão da literatura.

3.2 Problema FlowShop Permutacional (PFSP)

Quando a seqüência de processamento das tarefas é a mesma em todas as máquinas, tem-se o ambiente de escalonamento de tarefas do tipo *FlowShop* Permutacional. O estudo desse tipo de problema tem crescido exponencialmente nos últimos anos (REISMAN *et al*, 1997), pois trata-se de uma área de estudo abrangente e é encontrado em muitas indústrias como as de plástico, onde o produto requer uma série de processos seguidos a fim de prevenir a sua degradação (ALLAHVERDI E ALDOWAISAN, 2002). Segundo Moccellin e Nagano (1998), o PFSP pode ter as seguintes características:

- os tempos de processamento das tarefas nas máquinas são conhecidos, fixos e assumen o valor zero quando não existe processamento em determinada máquina;

- os tempos de *setup* estão incluídos no tempo de processamento e independem da posição da tarefa na ordenação das tarefas – *setup* independente da seqüência;
- toda tarefa é processada em somente uma máquina por vez e toda máquina só e somente só processa uma tarefa por vez;
- as tarefas não são interrompidas nas máquinas; e
- não existe tempo de deslocamento entre as tarefas.

O PFSP é definido formalmente da seguinte maneira: Considere um conjunto de n tarefas J_1, J_1, ..., J_n e um conjunto de m máquinas M_1, M_2, ..., M_m. Cada tarefa J_i é constituído de m operações consecutivas O_{i1}, O_{i2}, ...,O_{im}. A operação O_{ij} da tarefa J_i é processada na máquina M_j em um tempo p_{ij}. Cada tarefa J_i possui uma data de entrega d_i e todas as tarefas são disponíveis no tempo zero.

Mesmo para problemas relativamente pequenos em ternos de tarefas e máquinas, o número de soluções factíveis é extremamente elevado o que demanda um alto custo computacional para se encontrar a solução exata. Outra característica do PFSP é que o numero de soluções cresce exponencialmente ao numero de tarefas a serem ordenadas, este número é igual a $n!$, onde n é o numero de tarefas a serem ordenadas.

Tabela 1: Número de soluções, exponencialmente relacionados ao numero de tarefas.

Numero de Tarefas	**Soluções Factíveis**
5	120
6	720
7	5040
8	40.320
9	362.880
10	3.628.800

Conforme a Tabela 1, é inviável a busca pela solução ótima através de geração de todas as soluções factíveis. No caso de um problema com 10 tarefas a serem ordenadas, se o computador leva 0,1 segundo para avaliar cada solução

possível seria gasto 4,2 dias de processamento para terminar a procura. Como mencionado em Pinedo (1995) os problemas de otimização podem ser classificado em: problemas simples no qual, mesmo com grandes instâncias garantem a solução ótima em tempo polinomial e problemas complexos aonde não são conhecidos métodos que garantem encontrar a solução ótima em tempo aceitável, conhecidos como NP-difícil. O PFSP é definido como NP-difícil (GAREY e JOHNSON, 1979; LAWLER *et al*., 1993; CHEN *et al*., 1998), por isso, métodos aproximativos ou heurísticos são comumente utilizados neste tipo de problema. Estes métodos, quando bem desenvolvidos e adaptados aos problemas que se deseja resolver, são capazes de apresentar soluções de boa qualidade e em tempo compatível com a necessidade de rapidez (ARROYO, 2002). Recentemente, o PFSP tem sido estudado otimizando mais de um objetivo simultaneamente. T'kindt e Billaut (2001) apresenta uma revisão do estado da arte sobre problemas de escalonamento de tarefas em máquinas otimizando múltiplos critérios, incluindo máquinas simples e máquinas paralelas. Uma lista de 115 artigos é apresentada, o que mostra o crescente interesse nesta importante área. Na realidade, a área de otimização combinatória multiobjetivo teve um grande impulso na década de 90 com a aplicação das metaheurísticas algoritmos genéticos, busca tabu e *simulated annealing* em diversos tipos de problemas (EHRGOTT e GANDIBLEUX, 2000; JONES *et al*. 2002).

Mais recentemente, Minella *et al*. (2008) fizeram uma completa e atualizada revisão com o objetivo de comparar os diversos métodos publicados na literatura para o PFSP Multiobjetivo, aonde são citados 23 diferentes métodos heurísticos. Apesar desse grande interesse, principalmente nos últimos anos, o número de trabalhos publicados sobre o PFSP multimobjetivo é relativamente pequena quando comparada com o numero de trabalhos publicados para o PFSP monoobjetivo (MINELLA *et al*, 2008).

3.3 Objetivos para o Problema FlowShop Permutacional

A solução para o problema *flowshop* permutacional é dada por um vetor *v* de solução que demonstra a seqüência na qual as tarefas devem ser executadas objetivando otimizar um ou mais objetivos. Esses objetivos estão basicamente relacionados: a) com o estoque, que é produzido em tempo de processamento e que

tem sua importância devido a necessidade da indústria de alocar recursos financeiros de outras áreas, i.e, investimento em máquinas, em estoque o que aumenta a necessidade de capital de giro e, em alguns casos, torna a empresa dependente de capital de terceiros, o que onera o custo do produto em questão e conseqüentemente o preço de venda; b) data de entrega - a satisfação do cliente está intrinsecamente ligada a expectativa de receber o produto comprado em tempo hábil de ser utilizado ou revendido. Tal objetivo pode determinar o sucesso ou não da empresa, uma vez que o mercado moderno tem se preocupado cada vez mais com a capacidade da indústria em honrar seus compromissos de entrega, como exemplo pode-se citar as empresas montadores de veículos que utilizam o sistema de produção *just-in-time* para reduzir seus estoques, e depende totalmente da confiabilidade da indústria fornecedora em entregar seus pedidos na data acordada.

Desde o primeiro algoritmo proposto por Jonhson (1954), diferente métodos heurísticos e exatos têm sido apresentados na literatura para resolver esse problema, otimizando principalmente os seguintes objetivos (BAKER, 1974):

MakeSpan

A otimização de *Makespam* (C_{max}) está relacionada com a alocação de recursos da indústria e tem como conseqüência a otimização do estoque. Formalmente, seja um vetor v = { 1,2,..., n} com a sequência das tarefas a serem executadas , e seja C_i o tempo de término da tarefa i, então, $C_{max} = \max_i\{C_i\}$.

Atraso Total e Máximo

O Atraso Máximo (T_{max}) e o Atraso Total (T) estão relacionados com o atendimento das tarefas em tempos previamente definidos, tendo muitas vezes na indústria uma relação com a satisfação do cliente com a entrega do produto dentro do prazo acordado. Definindo formalmente, $T_{max} = \max_i\{T_i\}$, onde $T_i = \max\{C_{i,m} - d_i, 0\}$ é o atraso da tarefa J_i e $T = \Sigma T_i$.

Para melhor exemplificar, é definida a seguir uma matriz de tempo de processamento de cada tarefa, bem como as datas de entregas programadas para um problema com n = 4 tarefas e m = 3 máquinas. Nas linhas estão enumerados as Tarefas e as colunas trazem as maquinas as quais as tarefas devem ser executadas, o produto cartesiano (Linha x Coluna) determina o tempo de

processamento da tarefa *J* na máquina *M*. Por exemplo, o tempo de processamento da tarefa J_1 sendo executado na máquina M_1 é de seis unidades de tempo (segundo, minuto, etc), já a tarefa J_4 na máquina M_2 tem o tempo de processamento de duas unidades de tempo. A última coluna da tabela determina o tempo em que a tarefa deve ser terminada.

Tabela 2: Exemplo de uma matriz de tempo de processamento e data de entrega.

Máquina / **Tarefa**	**M_1**	**M_2**	**M_3**	***d***
J_1	6	10	11	36
J_2	10	3	12	30
J_3	8	4	5	50
J_4	2	2	12	20

A Figura 2a demonstra que a solução para o problema utilizando a seqüência J_3, J_1, J_4, J_2 apresenta como resultado C_{max} = 59 e T_{max} = 29. Aonde, C_{max} é o tempo gasto para executar a ultima tarefa da seqüência e, T_{max} é o maior atraso entre a data de encerramento da tarefa e sua respectiva data de entrega. Na seqüência, apresentada na Figura 2(a) o Atraso Máximo (T_{max}) é verificado em J_2.

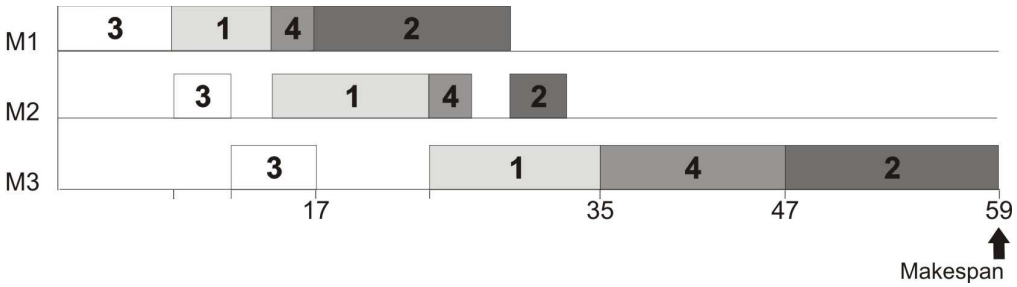


Figura 3(a). Tarefas dispostas em máquinas no ambiente FlowShop. Seqüência (J3, J1, J4, J2).

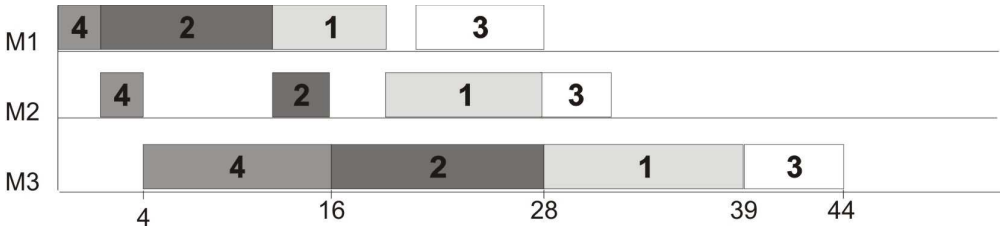


Figura 3(b). Tarefas dispostas em máquinas no ambiente FlowShop. Seqüência (J4, J2, J1, J2).

Na Figura 3(b) é demonstrado a seqüência J_4, J_2, J_1, J_3 que otimiza o resultado para 44 e 3, respectivamente para C_{max} e T_{max}. O T_{max} é encontrado na tarefa J_1.

São apresentados nesse trabalho dois métodos heurísticos baseado nas metaheurísticas GRASP e Busca Tabu com o objetivo minimizar o *makespan* (C_{max}) e o Atraso Máximo (T_{max}). A busca se deu para encontrar com conjunto de soluções eficientes segundo o conceito de *dominância de Pareto* (PARETO, 1896 *apud* ARROYO, 2002), caracterizando assim o uso de métodos *a-posteriori*. São realizados testes e os resultados das heurísticas propostas foram comparados com o trabalho de Arroyo e Armentano (2005).

No Capítulo 3 são discutidos os métodos heurísticos Construtivos e de Busca Local, bem como a apresentação de uma proposta para a distribuição das direções de busca dentro do espaço de soluções factível utilizando limitantes.

4 Métodos Heurísticos Construtivos e de Busca Local

4.1 Introdução

É normal para a resolução de problemas a busca por uma solução exata para o mesmo. Porém em algumas classes de problema a busca pela solução exata demanda grande esforço computacional, o que reflete no tempo para que essa solução seja encontrada; tempo esse que em alguns casos inviabiliza a solução. Isso ocorre porque não se conhece um algoritmo polinomial para a solução do problema, fazendo que o tempo aumente exponencialmente ao tamanho do dado de entrada. Uma técnica bastante utilizada nesse tipo de problema, denominados problemas NP-Difíceis, são os métodos heurísticos, que basicamente se dividem em três grupos, diferenciando-se na forma de como explorar o espaço de soluções; são eles: Heurísticas Construtivas, Heurísticas de Busca Local e Metaheurísticas (ARROYO, 2002). As heurísticas construtivas são procedimentos heurísticos baseados nas propriedades estruturais dos problemas e sua principal função é fornecer como resposta uma solução viável para o problema sem se preocupar com a garantia da qualidade. Os métodos de busca local ou busca por vizinhança são procedimentos que utilizam como base uma solução gerada por uma heurística construtiva. Esses métodos dependem da definição de uma estrutura de vizinhos no qual é estabelecida uma relação entre as soluções (vizinhos) em um espaço de decisões, denominado vizinhança (ARROYO, 2002). Os métodos de busca local apresentam geralmente uma melhora considerável na solução gerada pela heurística construtiva, entretanto esses métodos limitam-se a estrutura de

vizinhança definida, não permitindo que sejam explorados espaços mais distantes, mesmo que promissores, da solução dada como ponto de partida. Para solucionar essa limitação são utilizadas técnicas denominadas de métodos metaheurísticos. As metaheurísticas são métodos flexíveis e de fácil implementação que possibilitam a busca por soluções além da estrutura da vizinhança, escapando assim dos ótimos locais. É possível encontrar diversos métodos metaheurísticos publicados na literatura, dentre eles os mais estudados são:

- Algoritmos Genéticos: proposto por Holland (1975), utiliza o conceito de evolução natural, no qual os indivíduos (soluções) com uma função de aptidão são selecionados e modificados mediante operadores de cruzamento e mutação;
- Recozimento Simulado (*Simulated Annealing*): utiliza a busca aleatória de vizinhança, inspirado no comportamento termodinâmico da matéria.
- Busca Tabu: utiliza conjuntamente o conceito de busca local e memória adaptativa, proibindo movimentos para que a busca seja guiada para um espaço ainda não explorado; algumas vezes, são aceitas soluções que piorem a função objetivo para que seja possível fugir de um ótimo local.
- Busca por Vizinhança Variável (VNS): utiliza-se de uma troca sistemática de estrutura de vizinhanças com o objetivo de explorar o maior número de vizinhos possíveis.
- Procedimento Aleatório Adaptativo Guloso (GRASP): utiliza a estratégia de *multi-start*, ou seja, repetição de criação de uma solução inicial e melhoria sucessiva usando uma heurística construtiva com um fator *alfa* de aleatoriedade e busca local a cada iteração;

As metaheurísticas têm como característica possuírem grande facilidade de incorporar novas estratégias e explorar regiões do espaço de soluções factíveis na tentativa de superar um ótimo local.

Procurando fazer uso dos benefícios de cada método supracitado, são encontrados na literatura diversos trabalhos explorando a técnica de combinação

entre metaheurísticas; técnica essa denominada de metaheurística híbrida (CHAVES, 2007; SOUZA e SOUZA, 2005; RIBEIRO e VIANNA, 2005; ALVIM e RIBEIRO; 2004).

Será apresentado nesse Capítulo o conceito de métodos heurísticos construtivos e de busca local. Será apresentada na Seção 3.2 a heurística construtiva NEH proposta por Nawaz *et al.* (1983) que foi implementada nesse trabalho com o objetivo de gerar uma solução inicial para as heurística de Busca Local e Metaheurística também implementadas. Na Seção 3.3 são apresentadas as estruturas de vizinhanças e o método de busca local. Testes computacionais foram realizados com o intuito de definir a melhor estratégia para abordar o Problema do Flowshop Permutacional Multiobjetivo. Ainda nesse Capítulo é proposto um método de definição de pesos para os problemas multiobjetivos, baseado no método tradicional da soma ponderada. Tal método utiliza-se de um fator de influência para definir o peso para cada objetivo levando em consideração limitante superior e inferior encontrados na busca por extremos do espaço de soluções factíveis.

4.2 Heurística Construtiva

Heurísticas construtivas são procedimentos que constroem uma solução a partir de uma ou mais regras específicas para um dado problema de otimização. Os métodos construtivos, geralmente, são rápidos e os resultados obtidos através deles podem ser utilizados como ponto de partida para algoritmos de melhoria e/ou metaheurísticas.

Nesse trabalho foi utilizada a heurística construtiva NEH (NAWAZ *et al*, 1983) que é aclamada na literatura como uma das melhores heurísticas construtivas para minimizar o *makespan* (C_{max}) (ARROYO, 2002). Este mesmo tipo de algoritmo foi também usado com sucesso por Armentano e Ronconi (1999) para a minimização do tempo total. Este procedimento constrói uma solução *x* baseada na inserção de tarefas numa seqüência determinada segundo uma ordem de despacho (ordenação).

A seguir é apresentado o pseudocódigo da heurística construtiva NEH, onde pode ser observado que no primeiro passo as tarefas são organizadas segundo uma ordem e em seguida, é construída uma solução não factível utilizando a primeira tarefa do conjunto de tarefas ordenado. Logo após é inserido a segunda tarefa antes e depois da primeira tarefa selecionada formando assim duas soluções diferentes; a melhor solução é guardada e a próxima tarefa é selecionada para que seja inserida na primeira posição e em todas as outras posições possíveis, não alterando a seqüência selecionada no passo anterior. O procedimento é executado dessa forma até que a última tarefa seja inserida e uma solução factível seja gerada. A seguir é descrito o pseudocódigo do procedimento NEH.

Procedimento NEH

Inicio

Seja {$J_1, J_2,..., J_n$} uma lista de tarefas ordenada de acordo com a regra de despacho;

Seja x = (J_1) a seqüência parcial formada pela primeira tarefa da lista;

Para i = 2 até n faça

Insira a tarefa J_i em cada uma das posições de x, obtendo i seqüências parciais com i tarefas;

Faça x = à seqüência com menor valor de $f(x)$;

Fim para.

Retorne x uma seqüência com n tarefas;

Fim.

Na Figura 4 é demonstrado um exemplo do procedimento heurístico NEH. Seja S_1 = (1 2 3 4) uma seqüência de n = 4 tarefas ordenadas segundo uma regra de despacho qualquer. A partir das tarefas 1 e 2 são geradas as seqüências parciais (1 2) e (2 1), então é determinada a melhor seqüência pela soma ponderada das função de cada objetivo. Supondo que a seqüência (1 2) seja selecionada, então S_2 = { (1 2) }. Inserindo a tarefa 3 em cada uma das posições da seqüência em S_2, obtém-se o conjunto de seqüências {(**3** 1 2), (1 **3** 2), (1 2 **3**)}, novamente é

determinado a melhor solução através da função de soma ponderada. Neste exemplo a seqüência (3 1 2) é selecionada. Este processo é continuado até encontrar o conjunto de seqüências completas dominantes S4 = {(4 3 1 2), (3 1 4 2), (3 1 2 4), (1 2 4 3)}, aonde a seqüência (4 3 1 2) é determinada a melhor solução.

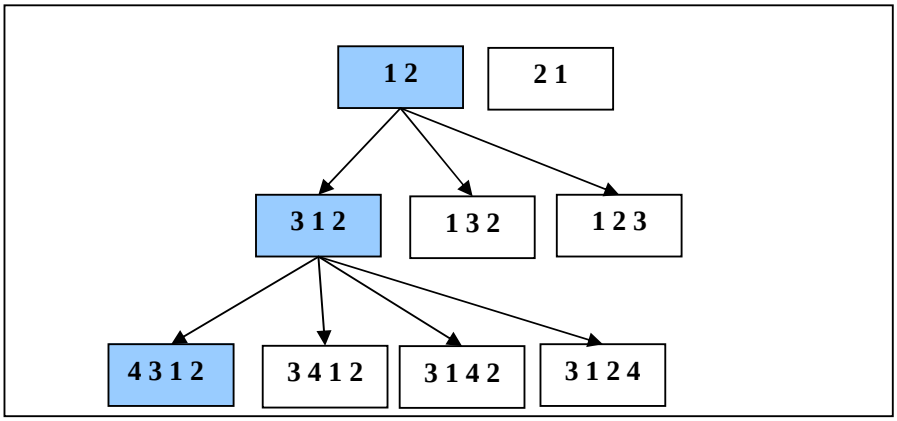


Figura 4. Exemplo do procedimento heurístico NEH.

A forma que as tarefas são ordenadas é de grande importância para os resultados encontrados pela heurística construtiva NEH. Neste trabalho foram testadas duas diferentes ordenações ou regras de despacho: LPT (*longest processing time*) proposta por Nawaz *et al*. (1983), na qual as tarefas são ordenadas de forma decrescente de acordo com o tempo total de processamento sobre as máquinas, $\sum_j p_{ij}$, onde p_{ij} é o tempo de processamento da tarefa ina máquina *j*, e TLB (*tardiness lower bound*), proposta por Armentano e Ronconni (1999) onde as tarefas são ordenadas de forma não-decrescente com o limitante inferior do atraso da cada tarefa *i*, $d_i - \sum_j p_{ij}$, onde d_i é a data de entrega da tarefa *i*.

4.2.1 Definição dos pesos

No caso de problemas multiobjetivo as soluções geradas pelas permutações da heurística construtiva NEH são avaliadas através de uma função *f* que é formada por uma combinação linear dos objetivos. Neste trabalho são tratados os objetivos *Makespan* (C_{max}) e Atraso Máximo (T_{max}) atribuindo a cada um desses um fator de importância denominado *peso*. Pode se definir formalmente tal técnica da seguinte

forma: $f(x) = f_1(x) \cdot \lambda_1 + f_2(x) \cdot \lambda_2$, tal que $\lambda_1 + \lambda_2 = 1$ (normalizando assim os pesos que ponderam a função). Dessa forma a combinação dos pesos determina a direção da busca no espaço de soluções factíveis. Para que seja possível explorar as soluções, em diversos caminhos dentro do espaço de soluções é necessário variar os pesos para cada objetivo utilizando, portanto, diferentes conjuntos de pesos são avaliados a cada iteração. Comumente os pesos são distribuídos de formar linear, entretanto é verificado neste trabalho que, aplicando essa estratégia, algumas regiões são mais exploradas que outras, isso porque geralmente os pesos tem diferentes graus de grandeza. É proposta nesse trabalho uma estratégia de distribuição dos pesos utilizando limitantes de acordo com os objetivos a serem otimizados e um fator de influência baseado na média dos limitantes. A seguir é detalhada esta estratégia.

4.2.2 Estratégia Proposta de Definição dos pesos

Nessa Seção é apresentada uma estratégia de variação dos pesos levando em consideração os limitantes inferiores e superiores de cada objetivo. Essa proposta de estratégia, denominada *método influenciado*, baseia-se na determinação de um fator de influência k_i para cada objetivo *i* que compõe o conjunto de objetivos a ser otimizado. O fator de influência *k* é definido pela média entre os limitantes inferiores e superiores de cada objetivo, após uma busca considerando os extremos no espaço de soluções factível. Por exemplo, para o Problema Flowshop Permutacional (PSFP) Biobjetivo tratado nesse trabalho, visando otimizar *makespan* (C_{max}) e Atraso Total (T_{max}) os extremos seriam definidos fixando a direção da busca somente considerando o primeiro objetivo, neste caso C_{max}, e em um segundo momento direcionado a busca somente para o segundo objetivo T_{max}. Basicamente são executados três passos para definir o *step* ou salto em cada iteração, onde a cada *step* é definida a diferença de prioridade entre os objetivos e consequentemente a direção a ser explorada. No primeiro passo é realizada uma busca apenas na direção de um objetivo (peso atribuído ao segundo objetivo igual a zero), no final dessa primeira busca são armazenados o limitante inferior do primeiro objetivo (min_1) e o correspondente valor encontrado pela solução para o segundo objetivo (max_2), uma vez sabido que os objetivos são conflitantes é

possível afirmar que a solução que determina o limitante inferior do primeiro, tem como conseqüência encontrar um valor suficientemente alto para ser determinado como limitante superior para o segundo objetivo. Em um segundo passo a busca é realizada na direção inversa, objetivando encontrar o limitante inferior do segundo objetivo (min_2) (peso atribuído ao primeiro objetivo igual a zero) e também o correspondente valor encontrado pela o primeiro objetivo (max_1). No terceiro passo os limitantes armazenados são utilizados para que seja calculada a média do primeiro objetivo, através da equação $med_1 = \frac{\min_1 + \max_1}{2}$, e a média do segundo objetivo, utilizando a equação $med_2 = \frac{\min_2 + \max_2}{2}$. Após encontrar a média dos objetivos é gerado uniformemente variando 0 a 1 um fator k_1 definindo um percentual de influência a cada iteração, consequentemente encontra-se k_2 visto que $k_2 = 1 - k_1$. Então é calculado o peso do primeiro objetivo utilizando o percentual de influência k_1 através da equação $\lambda_1 = \frac{k_1 \times med_2}{(k_1 \times med_1) - (k_2 \times med_2)}$; o peso dado ao segundo objetivo é encontrado fazendo $\lambda_2 = 1 - \lambda_1$.

A seguir é calculado para uma instância exemplo os fatores de influência e definido os pesos que devem nortear a direção da busca no espaço de soluções factíveis.

Tabela 3: Relação fator de influência e Pesos.

Direção	Fator de Influência		Pesos	
	$K_{C\max}$	$K_{T\max}$	$\lambda_{C\max}$	$\lambda_{T\max}$
1	0,00000	1,00000	0,00000	1,00000
2	0,07143	0,92857	0,19983	0,80017
3	0,14286	0,85714	0,35112	0,64888
4	0,21429	0,78571	0,46962	0,53038
5	0,28571	0,71429	0,56496	0,43504
6	0,35714	0,64286	0,64333	0,35667
7	0,42857	0,57143	0,70888	0,29112
8	0,50000	0,50000	0,76452	0,23548
9	0,57143	0,42857	0,81234	0,18766
10	0,64286	0,35714	0,85389	0,14611
11	0,71429	0,28571	0,89031	0,10969
12	0,78571	0,21429	0,92251	0,07749
13	0,85714	0,14286	0,95117	0,04883
14	0,92857	0,07143	0,97686	0,02314
15	1,00000	0,00000	1,00000	0,00000

Dado um PFSP bi-objetivo, visando minimizar *makespan* (C_{max}) e Atraso Total (T_{max}) o primeiro passo é definir a busca apenas na direção de C_{max}, tratando o problema como se fosse monoobjetivo, após a busca é encontrado o seguinte resultado: C_{max} = 2298 e T_{max} = 969. No segundo passo a direção da busca é definida ao inverso do primeiro passo, buscando a minimização de T_{max} como se fosse um problema Monoobjetivo, o resultado alcançado é C_{max} = 2546 e T_{max} = 523. De conhecimento dos limitantes inferiores e superiores de cada objetivo já é possível calcular a média, (med_{Cmax} = 2422 e med_{Tmax} = 746). Para definição do fator de influência é necessário definir o número de direções (iterações) que serão utilizadas em toda busca. No exemplo apresentado a seguir o numero de direções foi definido

como 15. É apresentada a Tabela 3 os fatores de influência para cada direção e os pesos que devem nortear a busca. Na Tabela 4 é apresentada a distribuição das direções segundo o método linear comumente utilizado.

Tabela 4: Relação dos pesos. Método Linear.

Direção	Pesos	
	$\lambda_{C\max}$	$\lambda_{T\max}$
1	0,00000	1,00000
2	0,07143	0,92857
3	0,14286	0,85714
4	0,21429	0,78571
5	0,28571	0,71429
6	0,35714	0,64286
7	0,42857	0,57143
8	0,50000	0,50000
9	0,57143	0,42857
10	0,64286	0,35714
11	0,71429	0,28571
12	0,78571	0,21429
13	0,85714	0,14286
14	0,92857	0,07143
15	1,00000	0,00000

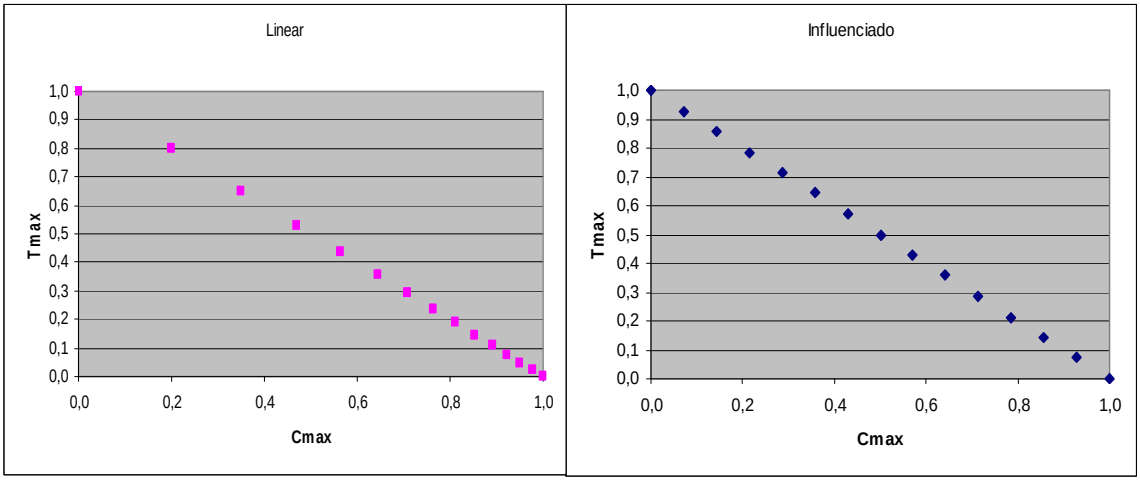


Figura 5. Fator de Influência: Método Linear x Método Influenciado.

A Figura 5 apresenta o fator de influência de cada objetivo, onde é possível observar que o método Influenciado apresenta uma distribuição mais uniforme, pois se utiliza de informações da própria instância para adaptar os *steps* e em conseqüência os pesos. Já o método Linear prioriza algumas áreas de busca em detrimento a outras.

Os testes computacionais realizados com a Heurística Construtiva NEH, assim como todos os testes realizados com outras heurísticas apresentadas nesse trabalho, utilizam-se desse conceito de distribuição influenciada, dos pesos para gerar o conjunto de soluções *Pareto*.

4.2.3 Resultados computacionais

Nessa Seção são apresentados os testes realizados com a heurística construtiva NEH. Em um problema mono-objetivo a busca por soluções se dá em um espaço de soluções factíveis completamente ordenado, aonde a relação entre duas soluções se dá somente pela avaliação da função objetivo que se quer otimizar. Conforme descrito na Seção 1.3 existem diversos métodos para avaliar se uma solução x é melhor ou pior em relação a uma solução x' em um problema de otimização multiobjetivo.

Estaremos utilizando o método da Soma Ponderada, descrito na Seção 1.3.1, com o objetivo de minimizar dois critérios: *Makespan* (C_{max}) e Atraso Máximo (T_{max}) para avaliar a qualidade de uma solução em relação a outra. Essa técnica também será utilizada em todas as heurísticas implementadas nesse trabalho.

4.2.4 Conjunto de Instâncias utilizadas

Nesse trabalho foram usadas instâncias geradas por Armentano e Ronconni (1999) e adaptado por Arroyo (2002) para o problema multiobjetivo, aonde foram gerados problemas para 20, 50 e 80 tarefas e para 5, 10 e 20 máquinas. Para cada problema são gerados quatro cenários diferentes levando em consideração a data de entrega de cada tarefa. A Tabela 5 demonstra a distribuição dos problemas.

O algoritmo NEH para o teste das regras de despacho LPT (NEH_LPT) e TLB (NEH_TLB) foi implementado na linguagem C++ e executado em uma processador Pentium D 1.4Ghz. Foi utilizada uma única iteração do algoritmo para cada problema, visto que o número de soluções factível é igual para qualquer das regras de despacho utilizado. O desempenho das duas regras de despacho utilizadas é apresentado na Tabela 6.

Tabela 5: Distribuição dos problemas nos conjuntos de instâncias.

Numero De Tarefas	Numero De Máquina	Numero de Problemas	Número de Cenários	Total de Problemas
20	5	10	4	40
20	10	10	4	40
20	20	10	4	40
50	5	10	4	40
50	10	10	4	40
50	20	10	4	40
80	5	10	4	40
80	10	10	4	40
80	20	10	4	40
Total				360

Para medir a desempenho do conjunto dominante *A* gerado por uma heurística em relação ao conjunto de referência *R* são usadas duas metodologias de avaliação: medidas de cardinalidade, e a medidas de distância Czyak e Jaszkiewics, descritas na Seção 1.4, essa mesma metodologia é utilizada por Arroyo (2002). A Tabela 6 mostra o número de soluções no conjunto referencia *R*, o número de soluções geradas por uma heurística (NSH), é o número de soluções dominantes geradas por uma heurística (NSDH). A Tabela 7 é demonstrada a medida de distância entre as heurísticas testadas.

Tabela 6: Resultado heurística NEH com as regras de despacho TLB e LPT.

Problema *N* x *m*	|***R***|	NEH_TLB		NEH_LPT	
		NSH	NSDH	NSH	NSDH
20 x 5	238	185	74	186	165
20 x 10	287	216	105	235	184
20 x 20	318	247	142	249	176
50 x 5	257	216	48	231	210
50 x 10	327	269	99	266	99
50 x 20	382	318	184	288	199
80 x 5	256	236	91	213	165
80 x 10	358	298	116	316	242
80 x 20	450	372	172	344	279
Total		2357	1031	2328	1719

Através da Tabela 6 é possível afirmar que a regra de despacho LPT (*longest processing time*) é mais eficaz, visto que encontrou 1719 soluções dominantes contra 1031 soluções dominantes encontradas pela regra de despacho TLB (*tardiness lower bound*). A medida de distância confirma a posição de eficácia da regra LPT na heurística construtiva NEH para o conjunto de instâncias testadas nesse trabalho.

Tabela 7: Medida de distância entre as regras TLB e LPT.

Problema	NEH_TLB		NEH_LPT	
n x *m*	D_{med}	D_{max}	D_{med}	D_{max}
20 x 5	0.1563	0.2975	0.0327	0.0993
20 x 10	0.1151	0.2589	0.0292	0.1020
20 x 20	0.0686	0.1939	0.0394	0.1183
50 x 5	0.1754	0.3143	0.0105	0.0437
50 x 10	0.0981	0.2119	0.0204	0.0838
50 x 20	0.0503	0.1543	0.0355	0.1114
80 x 5	0.0229	0.0717	0.0913	0.1801
80 x 10	0.0231	0.0881	0.0887	0.1911
80 x 20	0.0195	0.0799	0.0659	0.1709
Total	0.0810	0.1856	0.0459	0,1222

4.3 Heurística de Busca Local

Métodos de busca local (ou busca em vizinhança) são procedimentos utilizados para melhorar uma solução, em geral, obtida por uma heurística construtiva. Estes métodos dependem da definição de uma vizinhança que estabelece uma relação entre as soluções no espaço de decisões. O método de busca local para otimização mono-objetivo parte de uma solução inicial factível *x*, e gera um conjunto N(*x*) de soluções vizinhas de *x*. Cada solução em N(*x*) é gerada através de uma operação, denominada movimento, sobre *x*. Um método de busca local, conhecido como método de descida, busca dentre as soluções de N(*x*), uma solução *x'* com valor de função objetivo menor que *x*, no caso de minimização. A partir de *x'*, o processo é repetido até a obtenção de uma solução sem nenhum vizinho melhor, denominada ótimo local em relação à vizinhança.

A grande deficiência do método de busca local de descida é que ele termina em um ótimo local que na maioria dos casos não é um ótimo global. Uma primeira alternativa para reparar esta deficiência consiste em reiniciar a busca a partir de outra solução inicial e confiar que, nesta ocasião, a busca siga um outro caminho que leve a uma solução melhor. Um algoritmo genérico do método de descida (ARROYO, 2002) é apresentado a seguir.

Procedimento Busca Local

Início:

Melhora = *Verdade*;
Enquanto *Melhora* faça
Gere a vizinhança de *x*: *N*(*x*).
Determine o melhor vizinho $x' \in N(x)$
Se $f(x') < f(x)$ então
Faça $x = x'$;
Melhora = *Verdade*;
Senão *Melhora* = *Falso*;
Fim Enquanto;
Retorne *x*;

Fim;

Foram testados nesse trabalho duas estratégias de geração de vizinhanças, inserção de tarefas (*insert*) e troca de tarefas por pares (*all pairs*).

Na vizinhança de inserção, as soluções (seqüências) vizinhas de uma solução são geradas inserindo-se uma tarefa *i*, localizada na posição *k* da seqüência, em todas as outras posições da seqüência; esta estratégia gera uma vizinhança de tamanho $(n\text{-}1)^2$.

Na vizinhança de troca, as soluções vizinhas de uma solução são geradas através da troca de posições de duas tarefas *i* e *j* ($\forall$ *i*, *j* com $i \neq j$). Neste caso, o tamanho da vizinhança é $\frac{n}{2}(n-1)$. As duas estratégias de geração de vizinhanças

para o Problema de Flowshop Permutacional têm sido apresentadas em diversos trabalhos na literatura (ARMENTATO e RONCONI, 1999; GUPTA *et al.*, 2000a; TAILLARD, 1990). É apresentado um exemplo das vizinhanças estudas nesse trabalho na Tabela 8, onde dado uma solução *S* é gerado para as vizinhanças todos os vizinhos possíveis.

Tabela 8: Exemplo das vizinhanças Inserção e Troca de Pares.

Solução Inicial	**Inserção**	**Troca**
S = {3,2,1,4}	(2, **3**, 1, 4)	(**2**, **3**, 1, 4)
	(2, 1, **3**, 4)	(**1**, 2, **3**, 4)
	(2, 1, 4, **3**)	(**4**, 2, 1, **3**)
	(3, 1, **2**, 4)	(3, **1**, **2**, 4)
	(3, 1, 4, **2**)	(3, **4**, 1, **2**)
	(**1**, 3, 2, 4)	(3, 2, **4**, **1**)
	(3, 2, 4, **1**)	
	(**4**, 3, 2, 1)	
	(3, **4**, 2, 1)	

São realizados testes com o objetivo de apontar a melhor estrutura de vizinhança para o Problema do FlowShop Permutacional (PFSP). Os testes utilizam a metaheurística da fase de busca local da heurística GRASP, que será descrita na Seção 4.2. Essa abordagem é válida, uma vez que a fase de Busca Local da metaheurística GRASP é de extrema importância para a qualidade das soluções geradas, sabido que na fase construtiva a metaheurística utiliza-se de um fator de aleatoriedade para definir diferentes pontos dentro do espaço de soluções factível, e na fase de melhoria o algoritmo refina a busca. No Capítulo 4 são apresentadas as metaheurísticas GRASP e Busca Tabu implementadas nesse trabalho. Os testes computacionais utilizando as heurísticas de Busca Local e as metaheurísticas são apresentadas no Capítulo 5.

5 Métodos Metaheurísticos Implementados

5.1 Introdução

As metaheurísticas representam um conjunto de técnicas de otimização adaptadas para lidar com problemas complexos e que apresentam como característica a explosão combinatória. A idéia fundamental de uma metaheurísticas consiste em analisar ou visitar apenas um conjunto reduzido do espaço de soluções factíveis, considerando que o espaço de busca é excessivamente grande. Esse processo de busca deve ser realizado de uma forma eficiente para que, na medida do possível, seja encontrada (visitada) a solução que representa o ótimo global ou uma solução mais próxima possível do ótimo global. Portanto, uma metaheurísticas é uma estratégia que especifica a forma em que deve ser realizada a busca de forma inteligente, isto é, a forma em que devem ser realizadas as transições através do espaço de soluções partindo de um ponto inicial gerado por uma heurística construtiva. Assim, a diferença entre as metaheurísticas é a estratégia de busca utilizada por cada uma delas. Essa estratégia deve geralmente responder a cinco pontos para que a metaheurísticas possa ter bons resultados: (1) especificar uma forma de identificar ou representar um elemento do espaço de busca, isto é, uma proposta de solução do problema; (2) especificar a forma de encontrar a função objetivo ou seu equivalente para cada proposta de solução; (3) especificar a vizinhança da solução corrente; (4) especificar se a forma de realizar as transições deve ser realizada a partir de um único ponto ou de um conjunto de pontos (ou propostas de solução) e (5) se o processo de busca deve ser realizado através de

soluções factíveis ou podem ser consideradas também soluções infactíveis no processo de busca. Nesse trabalho foram utilizadas duas diferentes metaheurísticas: GRASP e Busca Tabu. As duas foram adaptadas para serem utilizadas para o Problema do Flowshop Permutacional Multiobjetivo. Nas seções abaixo são apresentadas as metaheurísticas implementadas e detalhes de implementação.

5.2 GRASP

O *Greedy Randomized Adaptive Search Procedure* (GRASP) é uma metaheurística de múltiplas partidas proposta por Feo e Resende (1995), na qual a cada iteração é gerada uma solução gulosa-aleatória (fase construtiva), que em seguida é melhorada por uma heurística de busca local (fase de melhoria). Na fase construtiva é construída iterativamente uma solução viável, inserindo na solução parcial um elemento de cada vez segundo um critério guloso – regra de despacho – e aleatório. A cada iteração da fase construtiva, são avaliados apenas elementos que podem ser adicionados à solução sem violar as restrições de viabilidade, gerando assim sempre uma solução factível. Esses elementos são chamados de elementos candidatos. A escolha do próximo elemento a ser adicionado a solução é determinada ordenando-se todos os elementos candidatos em uma lista de candidatos *C*, de acordo com uma função gulosa. Essa função mede o benefício associado à seleção de cada elemento. A heurística é adaptativa porque os benefícios associados a cada elemento são atualizados a cada iteração da fase construtiva para incorporar as mudanças causadas pela escolha do último elemento. O componente probabilístico é caracterizado pela escolha aleatória de um dos melhores candidatos da lista *C*, que não é necessariamente o melhor. A lista de melhores candidatos é denominada de lista restrita de candidatos (*LRC*).

Sumarizando, a estratégia da metaheurísticas GRASP consiste em usar diferentes soluções iniciais como pontos de partida para ser melhorada por uma busca local. Uma solução *x* é dita como pertencente ao vale de ótimo local quando, a partir de uma busca local iniciada em *x*, é possível atingir este ótimo local. Caso uma das soluções iniciais esteja no vale de um ótimo global, a busca local irá encontrar este ótimo global. Caso contrário, a solução do algoritmo será um ótimo

local. O uso de diversos pontos de partidas aleatórios permite eventualmente ao algoritmo encontrar um ponto dentro do vale de um ótimo global, porém as soluções de partida aleatórias geralmente são de baixa qualidade e acaba sendo necessário um grande número de movimentos para que se ache o ótimo local ou global. Por um outro lado, as soluções gulosas apresentam boas soluções como ponto de partida, no entanto sempre param em um mesmo ótimo local quando aplicado um procedimento de busca local. Para utilizar a diversidade das soluções aleatórias e qualidade inicial das soluções produzidas pelos algoritmos gulosos, o GRASP utiliza um fator *alfa* de aleatoriedade, aonde quanto maior esse fator maior é a aleatoriedade da solução de partida.

Em geral, os algoritmos baseados na metaheurística GRASP possuem poucos parâmetros a serem ajustados, basicamente o fator *alfa* de aleatoriedade e o número de iterações. Tal característica facilita a avaliação dos resultados e permite uma correção nos desvios de forma fácil. Por outro lado a falta de uma memória dos resultados encontrados durante a busca é uma desvantagem, pois em alguns casos a mesma solução é visitada diversas vezes, pois o fator de aleatoriedade definido não é suficiente para quebrar esse vício. É apresentado a seguir um pseudocódigo genérico da metaheurística GRASP.

Procedimento GRASP

Inicio

f(*x**) = infinito;

Sejam *α* o parâmetro de aleatoriedade;

Para *i* = 1 até *Max_Iterações* faça

x = *fase Construitiva*(*α*);

x' =*fase de Busca Local* (*x*);

Se *f*(*x'*) < *f*(*x**) então *x** = *x*;

Fim Para;

Retorne x*;

Fim.

5.2.1 Heurística Construtiva Gulosa Aleatória Proposta

Para implementação da metaheurística GRASP utilizada nesse trabalho, é proposto uma heurística gulosa aleatória baseada na heurística NEH (NAWAZ *et al.*, 1983), apresentada na Seção 3.2, denominada V_NEH. Tal heurística utiliza um fator de aleatoriedade α para alterar a seqüência na quais as tarefas são inseridas na solução, não respeitando assim a regra de despacho. As tarefas a serem inseridas na seqüência *x* são escolhidas aleatoriamente a partir de um subconjunto *LRC* de *LC*, aonde *LC* é lista de todas as tarefas previamente ordenadas de acordo com uma regra de despacho, e *LRC* é lista de candidatos restrita que é formada pelo *max*(1, α×|*LC*|) primeiras tarefas de *LC*, onde $\alpha \in [0, 1]$ é o parâmetro que determina a taxa de aleatoriedade e |*LC*| denota a cardinalidade da lista *LC*. Note que, se α = 0, as tarefas sempre serão escolhidas na ordem determinada pela regra de despacho (escolha gulosa). Se α = 1, a escolha de tarefas será completamente aleatória, ignorando completamente a regra de despacho. A seguir é apresentado o pseudocódigo da heurística construtiva gulosa aleatória V_NEH:

Procedimento V_NEH

Inicio

Seja $LC = \{J_1, J_2, ..., J_n\}$ a lista de tarefas ordenadas de acordo á regra de despacho LPT.

Escolha uma tarefa J_k aleatoriamente de $\{J_1, ..., J_t\}$onde t = max(1, α×|*LC*|).

Seja $x = (J_k)$ a seqüência parcial formada pela primeira tarefa escolhida.

Remova a tarefa J_k de *LC*.

Para i = 2 até n faça:

Escolha uma tarefa J_k aleatoriamente de *LCR*, onde *LCR* é formada pelas t = max(1, α×|*LC*|) primeiras tarefas de *LC*.

Insira a tarefa J_k em cada uma das posições de *x*, obtendo *i* seqüências parciais com *i* tarefas.

Faça *x* = a seqüência com menor valor de *f(x)*;

Remova a tarefa J_k de *LC*.

Fim Para.

Retorne a seqüência *x* com *n* tarefas e com menor valor de *f*.

Fim.

5.3 Variable Neighborhood Descent (VND)

O VND é uma técnica de melhoria baseada em Busca Local proposta por Mladenovic e Hansen (1997), que utiliza várias estruturas de vizinhança para procurar soluções de boa qualidade o “mais distante” da solução inicial, escapando assim de ótimos locais – principal deficiência das heurísticas de Busca Local. Sejam N_1, N_2,... N_p, onde p é o número de estrutura de vizinhança utilizada no VND. Normalmente a seqüência das estruturas de vizinhança se inicia com a de menor número de soluções, variando a busca até alcançar a estrutura N_p . O algoritmo é repetido até que sejam examinadas todas as estruturas de vizinhanças de uma solução x sem melhoria da melhor solução. O processo é reiniciado quando a solução x é melhorada por qualquer estrutura de vizinhança. A seguir é apresentado o procedimento VND com duas estruturas de vizinhanças.

Procedimento VND

Inicio

$f(x^*) = \infty$;

$k = 1$;

Enquanto $k \leq 2$ faça:

Se $k = 1$ então $x' =$ busca na Vizinhança $N_1(x)$;

Se $k = 2$ então $x' =$ busca na Vizinhança $N_2(x)$;

Se $f(x') < f(x^*)$ então faça:

$x^* = x'$;

$k = 1$;

Se não $k = k + 1$;

Fim Se;

Fim enquanto.

Retorne x^*;

Fim

Observando o pseudocódigo da metaheurística VND é fácil identificar que a mesma é muito semelhante ao procedimento de Busca Local padrão, alterando basicamente a variação das vizinhanças, quando a estrutura corrente de vizinhança

encontra um ótimo local, e não consegue melhorar a solução corrente.

5.4 GRASP+VND

É apresentado nesse trabalho uma estratégia GRASP híbrida que usa heurística construtiva V_NEH, apresentada na Seção 4.2.1, na fase construtiva e na heurística de Busca Local VND, apresentada na Seção 4.3, na fase de busca local. O pseudocódigo da heurística GRASP+VND é apresentado a seguir:

Procedimento GRASP+VND

Inicio

$f(x^*)$ = infinito;

Seja α o fator de aleatoriedade;

Sejam λ_1 e λ_2 os pesos para os objetivos C_{max} e $T_{max;}$

Para i = 1 até $TCPU$ faça

$x = V\_NEH\ (\alpha)$;

$x' = VND(x)$;

Se $f(x') = \lambda_1 C_{max}(x') + \lambda_2 T_{max}(x') < f(x^*) = \lambda_1 C_{max}(x^*) + \lambda_2 T_{max}(x^*)$ então faça: $x^* = x$;

Fim do se;

Fim Para;

Retorne x*;

Fim.

No pseudocódigo do procedimento GRASP+VND é utilizado como critério de parada o tempo de processamento TCPU. Após testes foi definido como 0.2 o fator de aleatoriedade α, por se mostrar mais robusto.

5.5 Busca Tabu

A Busca Tabu (BT) é uma metaheurística baseada em busca local. Foi

proposta por Glover (1989, 1990) para resolver problemas complexos de otimização combinatória aonde não é conhecido um algoritmo polinomial para solução. A estratégia da Busca Tabu consiste em realizar um processo iterativo que procura explorar o espaço de soluções do problema, movendo-se sucessivamente de uma solução *x* para outra solução *x'*, dentre sua vizinhança *N(x)*. Para o caso monoobjetivo, os movimentos são realizados com a finalidade de encontrar boas soluções, segundo a avaliação de alguma função objetivo *f(x)* a ser otimizada. No entanto, é permitido que o valor da função *f(x')* não necessariamente seja melhor que *f(x)*, isto é, a busca tabu pode aceitar uma solução *x*' que não seja melhor que a solução atual *x*. No processo de busca, a metaheurística utiliza do recurso de uma memória adaptativa que guarda os movimentos que estão sendo utilizados para que depois seja usado tanto para intensificá-la em uma região com boas soluções, quanto para *diversificar* a busca, levando a mesma para regiões pouco exploradas ou ainda que sequer tenham sido avaliadas. A memória adaptativa pode ser de curto prazo ou de longo prazo. A idéia de utilizar memória de curto prazo é de restringir a busca através da proibição de certos movimentos permitindo superar ótimos locais, prevenirem ciclagem e direcionar a busca para regiões não exploradas. É utilizada para isso uma estrutura chamada de *lista tabu* aonde são armazenados os movimentos, e aonde é feita uma consulta cada vez que o um novo movimento é realizado; caso esse movimento já tenha sido realizado o mesmo é proibido de ser executado e, portanto, é considerado um *movimento tabu*. Uma maneira de retirar a condição tabu de um movimento é pelo critério de aspiração que libera um movimento considerado suficientemente importante naquele momento da busca. A memória de longo prazo é utilizada quando a memória de curto prazo não é suficiente para produzir soluções de boa qualidade. As principais estratégias da memória de longo prazo são diversificação e intensificação; a diversificação conduz a busca para novas regiões e, em geral, é baseada em medidas de freqüência de atributos das soluções obtidas durante o processo de busca. Os movimentos que geram soluções com atributos muito freqüentes podem ser penalizados, enquanto movimentos que geram soluções com atributos pouco freqüentes podem ser incentivados tentando explorar soluções com novas características.

A idéia principal da estratégia de intensificação é retornar a busca para regiões consideradas promissoras. Para isto também é usada uma medida de

freqüência de atributos das melhores soluções encontradas durante a busca, denominadas *soluções de elite*.

A metaheurística Busca Tabu têm sido aplicada com muito sucesso a uma variedade de problemas monoobjetivo (GLOVER e LAGUNA, 1997).

No entanto, a aplicação desta metaheurística para problemas de otimização multiobjetivo é ainda escassa (EHRGOTT e GANDIBLEUX, 2000; JONES *et al*., 2002).

Neste trabalho é implementado um algoritmo Busca Tabu (BT) com memória de curto prazo e critério de aspiração. O algoritmo é baseado na minimização da função da soma ponderada $f = \lambda_1 C_{max} + \lambda_2 T_{max}$, onde os pesos são definidos de acordo com a estratégia apresentado na Seção 3.2.2 desse trabalho. A cada conjunto de pesos definido é executada a seguinte seqüência para a busca por soluções: a solução inicial é gerada pela heurística construtiva NEH_LPT descrito na Seção 3.2; a melhor solução é melhorada via uma heurística de Busca Local baseada na estrutura de troca de vizinhança por troca de pares, apresentado na Seção 3.3, e então a melhor solução é aplicada como inicial para a metaheurística BT.

Os componentes do algoritmo BT incluem: estratégia de geração de vizinhança e regras de proibição, duração tabu e critério de parada. Estes componentes determinam o desempenho do algoritmo e são dependentes do problema a ser resolvido e serão apresentados nas seções a seguir.

5.5.1 Estratégia de geração de vizinhança

Para o PFSP Multiobjetivo abordado neste trabalho foram testadas duas estruturas de geração de vizinhança no algoritmo BT, a regra de Inserção (*Insert)* e a regra de Troca de Pares (*AllPairs)*, Estas estratégias de geração de vizinhanças são muito usadas na literatura para o Problema de F*lowshop* Permutacional (TAILLARD, 1990; ARMENTANO e RONCONI, 2000; GUPTA *et al*., 2000a). É utilizado a regra de inserção, por se mostrar mais eficiente, conforme resultados da Seção 5.2.

5.5.2 Regras de Proibição

Uma vez definido a estrutura de geração de vizinhança, o próximo passo é a definição da regra de proibição. A regra de proibição usada para o movimento de inserção é: se a tarefa J_i é inserida em alguma posição da seqüência, esta tarefa é adicionada na lista tabu e, portanto, a tarefa J_i não pode ser escolhida para inserção enquanto a duração tabu esteja ativa.

5.5.3 Duração Tabu

A duração tabu (ou tamanho da lista tabu) é um parâmetro muito importante do algoritmo BT. Durações tabus muito pequenas podem causar ciclagem da busca, enquanto altas durações tabus podem restringir a exploração do espaço de busca. Neste trabalho, a duração tabu de um movimento é feita de forma dinâmica. Para cada combinação movimento-regra de proibição, a duração tabu *dt* é gerada aleatoriamente dentro de um intervalo [dt_{min}, dt_{max}] definido em função do número de tarefas *n*. Foram testados e analisados diferentes valores para dt_{min} e dt_{max}, e os valores que geraram os melhores resultados são: Para problemas de tamanho menores que 50 tarefas ($n < 50$) foi utilizado uma variação entre [$n/4$, $3n/4$], para problemas de 50 tarefas ($n \geq 50$) foi utilizado uma variação entre [$n/5$, $5n/8$] e para problemas com 80 ($n = 80$) tarefas foi utilizado a variação entre [$n/6$, $n/4$].

5.5.4 Critério de Aspiração

Nesse trabalho, é definido como critério de aspiração a possibilidade de aceitar uma solução gerada através de um movimento proibido, desde que essa solução seja a melhor solução encontrada até aquele momento da busca.

5.5.5 Critério de Parada

É utilizado como critério de parada para o Busca Tabu implementado nesse

trabalho o número de soluções avaliadas como pertencentes ou não ao conjunto de soluções *pareto-ótimas* ou conjunto eficiente. Para cada tamanho de instância é definido um número diferente de soluções avaliadas, relacionando o numero de tarefas e a complexidade de se alcançar uma solução de qualidade. Esse critério é baseado no trabalho de Armentano e Arroyo (2005).

No Capítulo 5 são mostrados os testes computacionais realizados com as heurísticas Busca Tabu e uma implementação hibrida GRASP+VND. Os resultados são comparados entre as heurísticas e o melhor método é comparado com uma referência da literatura.

6 Testes e Resultados Computacionais

6.1 Introdução

Nesse capítulo são apresentados os resultados para os testes realizados com as metaheurísticas GRASP+VND e Busca Tabu (BT). Na Seção 5.2 é apresentado um estudo comparativo para as estruturas de vizinhanças de inserção *(insert)* e troca de pares (*all pairs)* para o Problema do Flowshop Permutacional Multiobjetivo. O teste é realizado aplicando as estruturas de vizinhança na fase de melhoria da metaheurística GRASP; os resultados encontrados por cada estrutura de vizinhança e comparado entre eles e também com uma implementação da técnica VND.

A metaheurística BT tem seu resultado comparado com a implementação da heurística GRASP+VND na Seção 5.3; Na seção é comparado os resultados encontrados na literatura - heurística MOGLS (ARROYO, 2005) - com a Busca Tabu implementada nesse trabalho.

Os objetivos a serem minimizados nos testes foram C_{max} e T_{max} , as instâncias utilizadas foram as mesmas descritas na Seção 3.2.4, totalizando 360 problemas divididos em 9 conjuntos, de acordo com o número de tarefas e máquinas apresentado na Tabela 6.

A estratégia de definição dos pesos e direções de exploração no espaço de soluções factíveis é a estratégia propostas na Seção 3.2.2; para todos os testes foram utilizadas 10 direções diferentes.

Como critério de parada foi utilizada para os testes o critério de tempo total de

processamento (TCPU). Somente para o teste apresentado na Seção 5.4 que foi utilizado o numero de soluções avaliadas como critério de parada.

Os critérios de avaliação utilizados para comparar os métodos implementados são a Medida de Cardinalidade e de Distância, ambos descritos na Seção 1.4. O conjunto dominante gerado por cada metaheurística é comparado com o respectivo conjunto de referência *R* formado pelas soluções dominantes dentre todas as soluções encontradas por ambas as metaheurísticas. Para cada instância do problema é mostrado o número de soluções do conjunto referência (|*R*|), o número total de soluções geradas por uma metaheurística (NSH) e o número de soluções dominantes obtidas (NSDH).

Todos os algoritmos foram implementados na linguagem de programação C++ (ANSI) e executados sobre um processador Pentium D Dual Core 1.4Mhz.

6.2 Resultados para Busca Local e GRASP

É apresentado nessa seção os resultados computacionais envolvendo a heurística GRASP com as estruturas de vizinhanças troca de pares (GRASP+*AllPairs*) e de inserção de tarefas (GRASP+*Insert)* e o procedimento híbrido GRASP+VND. É apresentado na Tabela 9 o tempo total de processamento (TCPU*)* usado para cada conjunto de instâncias.

Tabela 9: TCPU utilizado como critério de parada para Busca Local e GRASP.

n×*m*:	20×5	20×10	20×20	50×5	50×10	50x20	80x5	80x10	80×20
TCPU:	3	3.3	3.8	17	25	40	52	140	246

Na Tabela 10 é demonstrado a medida de cardinalidade e a Tabela 11 demonstra a medida de distância.

É possível observar na Tabela 10, que a metaheurística híbrida GRASP+VND alcança melhores resultados em todos os conjuntos de instâncias testados,

encontrando um total de 6.016 soluções dominantes (NSDH), enquanto o GRASP com Busca Local por Inserção (GRASP+*Insert*) encontrou 3.699 soluções dominantes e o GRASP com a Busca Local por troca de tarefas (GRASP+*All Pairs)* encontrou 1.584.

Tabela 10: Resultados entre as implementações GRASP.

Problema	|***R***|	GRASP+*AllPairs*		GRASP+*Insert*		GRASP+*VND*	
n x *m*		NSH	NSDH	NSH	NSDH	NSH	NSDH
20 x 5	523	496	192	503	340	489	411
20 x 10	802	694	151	751	363	753	630
20 x 20	854	669	146	805	525	846	620
50 x 5	481	446	93	473	226	459	372
50 x 10	905	142	682	856	402	865	729
50 x 20	1226	895	115	1165	703	1189	1102
80 x 5	293	314	28	295	182	293	286
80 x 10	802	601	81	825	351	771	719
80 x 20	1243	921	96	1189	607	1209	1147
Total	7129	5178	1584	6862	3699	6874	6016

Na Tabela 11, é apresentado as medidas de distância entre as implementações. É observado que a implementação GRASP+VND sempre apresenta uma menor medida de distância para todos os conjuntos,

Na Figura 6 é demonstrado através de um problema da instância 50x20 o ganho de qualidade alcançado pela heurística híbrida GRASP+VND, quando comparado com o GRASP implementado com uma estrutura de vizinhança.

Tabela 11: Medida de Distância entre as implementações GRASP.

Problema	GRASP+*AllPairs*		GRASP+*Insert*		GRASP+*VND*	
n x *m*	D_{med}	D_{med}	D_{med}	D_{med}	D_{med}	D_{med}
20 x 5	0.032	0.087	0.011	0.496	0.004	0.030
20 x 10	0.332	0.902	0.018	0.696	0.004	0.035
20 x 20	0.044	0.121	0.012	0.604	0.011	0.062
50 x 5	0.067	0.131	0.020	0.062	0.011	0.047
50 x 10	0.071	0.151	0.021	0.084	0.007	0.047
50 x 20	0.076	0.157	0.013	0.069	0.002	0.031
80 x 5	0.304	0.404	0.020	0.055	0.001	0.005
80 x 10	0.125	0.229	0.023	0.081	0.003	0.026
80 x 20	0.106	0.204	0.016	0.078	0.002	0.029
Média Total	0.128	0.265	0.017	0.247	0.005	0.034

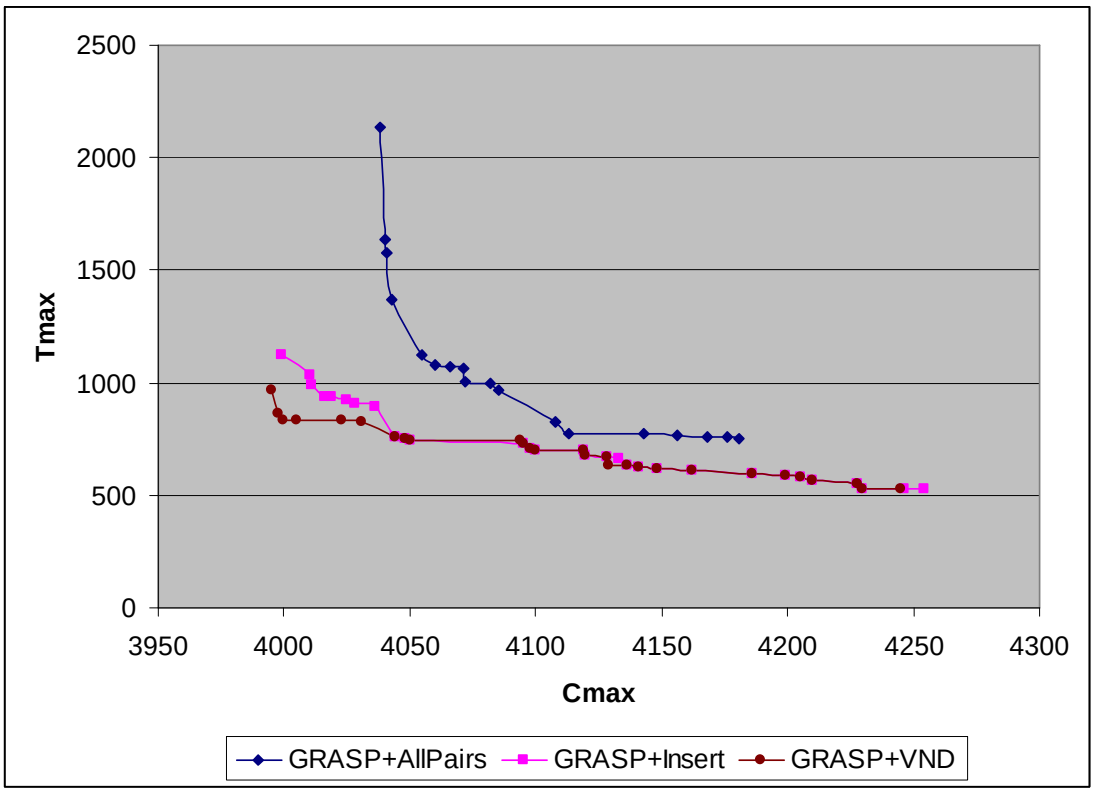


Figura 6: Resultados para a instância 50x20, Problema 1 – Cenário 1.

6.3 Resultados para GRASP+VND e Busca Tabu

Nessa Seção é apresentado os resultados computacionais comparando a heurística híbrida GRASP+VND e a implementação da metaheurística Busca Tabu. Na Tabela 12 é apresentado o tempo de processamento total (TCPU) em segundos usado como critério de parada.

Tabela 12: TCPU utilizado como critério de parada para GRASP+VND e Busca Tabu.

n×*m*:	20×5	20×10	20×20	50×5	50×10	50x20	80x5	80x10	80×20
TCPU:	18	35	62	247	515	820	951	1168	1361

Tabela 13: Resultados comparando Busca Tabu e GRASP+VND.

Problema	$\|R\|$	Busca Tabu		GRASP+VND	
n x m		NSH	NSDH	NSH	NSDH
20 x 5	548	531	429	549	332
20 x 10	944	927	762	868	359
20 x 20	1147	1121	913	998	440
50 x 5	508	452	342	541	220
50 x 10	1209	1193	1146	1117	64
50 x 20	1674	1649	1607	1596	67
80 x 5	316	302	240	297	169
80 x 10	800	1153	507	850	619
80 x 20	1918	1902	1297	1427	410
Total	9064	9230	7243	8243	2680

Na Tabela 14, é apresentado a medida de distância, a heurística BT encontra a menor distância em todos os conjuntos demonstrando sua melhor qualidade, quando comparado com a implementação GRASP+VND.

Tabela 14: Medida de Distância entre as implementações Busca Tabu e GRASP+VND.

Problema	Busca Tabu		GRASP+*VND*	
n x *m*	D_{med}	D_{max}	D_{med}	D_{max}
20 x 5	0.008	0.039	0.024	0.072
20 x 10	0.004	0.031	0.016	0.062
20 x 20	0.003	0.294	0.016	0.068
50 x 5	0.029	0.089	0.018	0.051
50 x 10	0.001	0.015	0.059	0.122
50 x 20	0.001	0.013	0.059	0.125
80 x 5	0.009	0.035	0.107	0.148
80 x 10	0.081	0.141	0.111	0.223
80 x 20	0.007	0.133	0.124	0.341
Média Total	0.015	0.087	0.059	0.134

Na próxima seção, a heurística Busca Tabu tem seu resultado comparado com o os resultados da heurística MOGLS implementada por Arroyo e Armentano (2005).

6.4 Resultados para Busca Tabu e MOGLS

A heurística MOGLS apresentada nessa seção como referência, foi proposta por Arroyo e Armentano (2005), e é baseada na técnica de Algoritmos Genéticos acrescida de uma Busca Local Multiobjetivo implementada com uma sofisticada regra de redução de vizinhança e exploração de direções paralelas. Recentemente no trabalho de Minella *et al.* (2008), a heurística MOGLS foi testada junto com outras 23 heurísticas e seu resultado foi considerado como consistente comparando com a metaheurística MOSA_Varadharajan (VARADHARAJAN e RAJENDRAN, 2005), aclamada nesse estudo como a metaheurística de melhor performance dentre as testadas.

São utilizadas as mesmas instâncias testadas na Seção 3.2.4. O critério de parada foi o número máximo de soluções avaliadas. Para o MOGLS foi respeitado o número relatado na literatura (ARROYO, 2002), para as instâncias com n = 20 foram avaliados $1000n^2$ e para as instâncias n = 50 foram avaliados $1300n^2$ e para n = 80 foram avaliadas $1600n^2$. Na implementação do Busca Tabu foram avaliados quatro vezes mais soluções que os números propostos para o MOGLS.

A seguir são apresentados os resultados com os testes realizados entre a heurística Busca Tabu implementada nesse trabalho e os resultados da heurística MOGLS. Detalhes da implementação da heurística Busca Tabu são descritos na Seção 4.5 desse trabalho.

Analisando a Tabela 15, é observado que a heurística Busca Tabu (BT) apresentou bons resultados, encontrando resultados equivalentes ao MOGLS nas instância com 20 e 50 tarefas (n = 20 e n = 50), entretanto o BT não encontrou bons resultados no conjunto de 80 tarefas (n = 80). Detalhando, no conjunto 20x5 (n = 20 e m = 5) o Busca Tabu encontrou 404 soluções dominantes (NSDH), enquanto o MOGLS encontrou 406, um número bem próximo. Na instância 20x10 o Busca Tabu superou o MOGLS encontrando 682 soluções dominantes contra 545 soluções dominantes encontradas pelo MOGLS. No conjunto de instâncias 20x20 a heurística Busca Tabu também superou o MOGLS, encontrando 844 e 604, respectivamente. Entretanto no conjunto 50x5 e 50x10 o Busca Tabu não conseguiu os mesmos resultados encontrados no conjunto com 20 tarefas, conseguindo por fim uma melhoria no conjunto de instâncias 50x20. Nas instâncias com 80 tarefas, o Busca Tabu não conseguiu superar o MOLGS em nenhum dos conjuntos.

Dessa forma é possível afirmar que a implementação da metaheurística Busca Tabu encontrou um bom número de soluções dominante, principalmente nos conjuntos de instâncias com 20 e 50 tarefas chegando, a superar a referência da literatura. Entretanto a implementação proposta nesse trabalho não foi robusta o suficiente para superar a referência quando são avaliados problemas de maior porte. Outro ponto que deve ser levado em consideração é o número de soluções avaliadas por cada heurística, como foi dito anteriormente. A Busca Tabu avaliou uma quantidade quatro vezes maior que a MOGLS, despendendo assim um maior tempo computacional.

Tendo como ponto positivo a Busca Tabu é de fácil implementação quando

comparada a MOGLS que utiliza sofisticadas técnicas de redução de vizinhança e busca por soluções em direções paralelas.

Tabela 15: Resultados comparando Busca Tabu e MOGLS.

Problemas	$\|\boldsymbol{R}\|$	Busca Tabu		MOGLS	
N x m		NSH	NSDH	NSH	NSDH
20 x 5	555	531	404	491	406
20 x 10	989	927	682	867	545
20 x 20	1151	1121	844	1017	604
50 x 5	510	452	159	488	423
50 x 10	1325	1193	393	1137	938
50 x 20	1659	1649	1607	1320	1300
80 x 5	379	302	129	368	354
80 x 10	998	850	160	922	853
80 x 20	2352	1902	551	2096	1801
Total	9918	8907	4929	8706	7224

Tabela 16: Medidas de Distâncias entre Busca Tabu e MOGLS.

Problemas	Busca Tabu		MOGLS	
n x *m*	D_{med}	D_{max}	D_{med}	D_{max}
20 x 5	0.0133	0.5036	0.0157	0.06839
20 x 10	0.0138	0.0721	0.0063	0.0399
20 x 20	0.0182	0.0961	0.0046	0.0375
50 x 5	0.0313	0.0949	0.0116	0.0447
50 x 10	0.0347	0.0969	0.0128	0.0814
50 x 20	0.0182	0.6972	0.6749	0.9776
80 x 5	0.0891	0.1405	0.0021	0.0103
80 x 10	0.0609	0.1302	0.0070	0.0538
80 x 20	0.0445	0.1131	0.0094	0.0730
Média Total	0.0360	0.2160	0,0827	0,1540

7 Conclusões

O objetivo desse trabalho foi demonstrar a aderência de métodos heurísticos para a resolução de problemas de otimização multiobjetivo, especialmente o problema do Flowshop Permutacional Multiobjetivo (PFSP). O PFSP tem sido bastante estudado, porém na maioria dos trabalhos encontrados na literatura é otimizado apenas um objetivo, entretanto o interesse por otimizar mais de um objetivo tem crescido nos últimos anos (T'KINDT e BILLAUT, 2001).

Foi utilizado nesse trabalho o conceito de dominância de Pareto (PARETO, 1896 *apud* ARROYO, 2002) para encontrar um conjunto de soluções eficientes para o PFSP ultiobjetivo, tendo como função minimizar o par de objetivos *makespan* (C_{max}) e Atraso Máximo (T_{max}). O conceito de dominância de Pareto é utilizado, porque o espaço de soluções dos problemas multiobjetivos não é completamente ordenado, e sim parcialmente ordenado (PARETO, 1896 *apud* ARROYO, 2002).

Para realizar a otimização de problemas multiobjetivos é necessário definir o método que será usado para avaliar a qualidade de uma solução em relação à outra. Foi utilizado nesse trabalho o método tradicional da soma ponderada (ARROYO, 2002), que depende da definição de pesos atrelados aos objetivos. É proposta uma estratégia para definição dos pesos necessários para utilização do método da soma ponderada baseado nos limitantes superiores e inferiores dos objetivos a serem otimizados e um fator de influência. Foi constatado que a distribuição dos pesos utilizando essa estratégia se faz de forma mais uniforme, quando comparada com a estratégia de definir os pesos linearmente (método tradicional).

Para os testes realizados nesse trabalho foram utilizados nove conjuntos diferentes de instâncias agrupados por número de tarefas (20, 50 e 80) e número de máquinas (5, 10 e 20). Cada conjunto contém 10 problemas e cada problema possui

4 cenários diferenciados pela data de entrega. As instâncias foram as mesmas utilizadas por Arroyo (2002). Quanto a comparação entre os diversos métodos heurísticos implementados foram usados a medida de cardinalidade (ARROYO, 2002) e de distância (CZYAK e JASZKIEWICZ, 1998).

Durante o desenvolvimento do trabalho foi implementado primeiramente a heurística construtiva NEH proposta por Nawas *et al.* (1983), utilizando duas diferentes regras de despacho – LTP (NAWAS *et al.*, 1983) e TLB (ARMENTANO e RONCONNI, 1999). Testes foram realizados e os resultados demonstraram que a regra LPT mostrou-se mais eficiente. É proposta, baseado na heurística NEH, uma heurística construtiva gulosa aleatória, denominada V_NEH.

A heurística V_NEH foi utilizada na fase construtiva da meutaheurística GRASP, também implementada nesse trabalho. Quanto a fase de busca local, foram testados duas diferentes estruturas de vizinhança: vizinhanças por Inserção (*Insert*) e por Troca de Pares (*AllPairs*). Os testes realizados demonstraram que a estrutura de vizinhança por Inserção apresenta melhores resultados. Utilizando ainda as estruturas de vizinhança foi implementado a técnica VND que foi combinada com a fase de melhoria do GRASP gerando uma implementação híbrida GRASP+VND. Os resultados do GRASP+VND supera as implementações do GRASP com a busca local simples quer seja com a Troca de Pares, quer seja com a Inserção.

Com objetivo de gerar soluções mais próximas do conjunto *pareto-ótimo*, além da heurística GRASP+VND, foi implementada uma heurística Busca Tabu. Nos testes realizados, a Busca Tabu encontrou um conjunto de soluções melhores que os encontrados no GRASP+VND.

Por último, os resultados encontrados pela metaheurística Busca Tabu foram comparados com a heurística MOGLS, proposta por Arroyo e Armentano (2005). A MOGLS é uma heurística baseado na técnica de Algoritmos Genéticos (HOLLAN, 1975), aonde é acrescentada uma estratégia de *eletismo*, o que permite incluir indivíduos de menor aptidão dentro da população com o intuito de *diversificar* as soluções. Também é implementado no MOGLS uma Busca Local Multiobjetivo Paralela para intensificar a busca em determinada região do espaço de soluções. Com essa técnica várias direções dentro do espaço de soluções são pesquisadas utilizando para limitar a busca, uma redução de vizinhança denominada *clustering* (MORSE, 1980) Em recente pesquisa realizada por Minella *et al*. (2008), a heurística MOGLS usada nesse trabalho como referência, é considerada uma heurística de

desempenho consistente.

Os resultados demonstraram que a implementação da Busca Tabu encontrou resultados semelhantes para as instâncias com número de tarefas n = 20 e 50; para as instâncias mais robustas (*n* = 80), o algoritmo busca tabu proposto perde eficiência quando comparado com o MOGLS.

É importante citar ainda que o número de soluções avaliadas pela heurística Busca Tabu propostas nesse trabalho, foi quatro vezes maior que o número de soluções avaliadas pela MOGLS.

Concluindo, a heurística Busca Tabu, se mostrou eficiente quanto comparada a implementação da heurística híbrida GRASP+VND, entretanto não superou os resultados encontrados pela heurística MOGLS em todas os conjuntos de instâncias testados, mesmo tendo um número maior de soluções avaliadas.

Como contribuições deste trabalho podem-se citar: (1) a proposta de uma estratégia de distribuição de pesos baseado nos limitantes inferiores e superiores dos objetivos a serem otimizados; (2) a proposta de uma variação da heurística construtiva NEH, adaptando um fator de aleatoriedade a mesma; (3) por último é apresentada uma técnica heurística para o PFSP Multiobjetivo, baseado na metaheurística Busca Tabu, de fácil implementação e com resultados de qualidade quando comparados com resultados da literatura.

Como pontos para o desenvolvimento futuro desse trabalho podem-se citar: (1) o desenvolvimento de uma estratégia de redução de vizinhança, com o intuito de reduzir o número de soluções avaliadas, sem reduzir a qualidade das soluções; (2) implementação da memória de longo prazo para a Busca Tabu, visando intensificar a busca em espaços mais promissores dentro do espaço de soluções factíveis; (3) aplicar os métodos utilizados em outros problemas de otimização multiobjetivo e comprovar a eficácia em termos de aderência.

8 Referência Bibliográfica

1 ALLAHVERDI, A., 2001. The tricriteria two-machine flowshop scheduling problem. *International Transactions. In Operational Research*, 8, p. 403-425.

2 ALLAHVERDI, A., 2003. The two and m machines flowshop scheduling problems with bicriteria of makespan and mean flowtime. *European Journal of Operational Research* 147, p. 373-396.

3 ALLAHVERDI, A., 2004. A new heuristic for m-machine flowshop scheduling problem with bicriteria of makespan and maximum tardiness. *Computers and Operations Research*, 31, p. 157-180.

4 ALLAHVERDI, A. & ALDOWAISAN, T., 2002. New Heuristics to Minimize Total Completion Time in m-machine Flowshops. *International Journal of Production Economics* Vol. 77, p.71-83.

5 ARMENTANO, V. A., RONCONI, D.P., 1999. Tabu search for total tardiness minimization in flowshop scheduling problems. *Computers and Operations Research*, 26, p. 219-235.

6 ARMENTANO, V.A., ARROYO, J.E.C., 2004. An Application of a Multi-Objective Tabu Search Algorithm to a Bicriteria Flowshop Problem. *Journal of Heuristics*, 5, p. 463-481.

7 ARROYO, J.E.C., 2002. Heurísticas e Metaheurísticas para Otimização Combinatória Multiobjetivo. *Tese de Doutorado*, FEEC – UNICAMP, 256 ps.

8 ARROYO, J.E.C., ARMENTANO, V.A., 2004. A Partial Enumeration Heuristic for Multi-Objective Flowshop Scheduling Pr oblems. *Journal of Operational Research Society*, 55, 9, p. 1000-1007.

9 ARROYO, J.E.C., ARMENTANO, V.A., 2005. Genetic Local Search for Multi-Objective Flowshop Scheduling Problems. *European Journal of Operational Research*, 167, 3, p. 717-738.

10 BAKER, K. R., 1974. *Introduction to Sequencing and Scheduling*. Johm Wileys & Sons, Inc., New York.

11 BAKER K.R., KANET J.J., 1983. Job shop scheduling with modified due dates, *Journal of Operations Management*, vol. 4, pp. 11-22.

12 BEN-DAYA, M., AL-FAWZAN, M., 1998. A tabu search approach for the flowshop scheduling problem. *European Journal of Operational Research*, 109, p. 88–95.

13 CHAKRAVARTHY, K., RAJENDRAN, C., 1999. A heuristic for scheduling in a flowshop with the bicreteria of makespan and maximum tardiness minimization. *Production Planning and Control*, 10, p. 707-714.

14 CHEN, B., POTTS, C.N., WOEGINGER, G.J., 1998. A review of machine scheduling: Complexity, algorithms and applications. In: Du, D-Z., Pardalos, P.M. (Eds.), *Handbook of Combinatorial Optimization*. Kluwer, Dordrecht, pp. 21–169.

15 CZYZAK P. & JASZKIEWICZ A., 1998. Pareto Simulated Annealing – a metaheuristic technique for multiple objective combinatorial optimization, *Journal of Multi-Criteria Decision Analysis*, vol. 7, p. 34-47.

16 DANIELS R.L., 1992. Analytical evaluation of multi-criteria heuristics. *Management Science*, vol. 38, p. 501-513.

17 DANIELS, R.L., CHAMBERS, R.J., 1990. Multiobjective flow-shop scheduling. *Naval Research Logistics* , 37:, p. 981-995.

18 EHRGOTT, M., GANDIBLEUX, X., 2000. A survey and annotated bibliography of multicriteria combinatorial optimization. *OR* Spektrum, 22, p. 425-460.

19 EHRGOTT, M., 2000 Approximation algorithms for combinatorial multicriteria optimization problems, *International Transactions in Operational Research*, vol. 7, p. 5-31.

20 FEO, T.A., RESENDE, M.G.C., 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6, p. 109-133.

21 FRAMINAN, J.M,, LEISTEN, R., RUIZ-USANO, R., 2002. Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimization. *European Journal of Operational Research* 141, p. 559-569.

22 GAREY, M.R., JOHNSON, D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-completeness. W.H. Freeman and Company, San Francisco.

23 GELDERS, L.F., SAMBANDAM, N., 1978. Four simple heuristics for scheduling a flowshop. *International Journal of Production Research*, 16, p. 221–231.

24 GLOVER, F.W., 1989. Tabu search, Part I. *ORSA Journal on Computing*, 1, p. 190-206.

25 GLOVER, F.W., 1990. Tabu search, Part II. *ORSA Journal on Computing*, 2, p. 4-32.

26 GLOVER, F.W., Laguna M., 1997. Tabu Search, *Kluwer Academic Publishers.*

27GUPTA, J.N.D., HENNIG, K., WERNER, F., 2000a. Local Search heuristics for two-stage flow shop problems with secondary criterion. *Computers and Operations Research*, 29, p. 123-149.

28 GUPTA, J.N.D., NEPPALLI, V.R., WERNER, F., 2000b. Minimizing total flow time in a two-machine flowshop problem with minimum makespan. *International Journal of Production Economics*, 69, p. 323-338.

29 GUPTA, J.N.D., PALANIMUTHU, N., CHEN, C.L., 1999. Designing a tabu search algorithm for the two-stage flow shop problem with secondary criterion. *Production Planning and Control*, 10:, p. 251-265.

30 HO, J.C., 1995. Flowshop sequencing with mean flow time objective. *European Journal of Operational Research*, 81, p. 571–578.

31 HOLLAN, J.H., 1975. Adaptation in Natural and Artifitial System, *University of Michigan Press*, Ann Arbor.

32 ISHIBUCHI, H., MURATA, T., 1998. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transaction. on Systems, Man and Cybernetics - Part C: Applications and Reviews* , 28:, p. 392-403.

33 ISHIBUSHI, H., MISAKI, S., TANAKA, H., 1995. Modified simulated annealing algorithms for the flowshop sequencing problems. *European Journal of Operational Research* , 81, p. 388–398.

34 JOHNSON, S.M., 1954. Optimal two and three-stage production schedules with set-up times included. *Naval Research Logistics Quarterly* ,1, p. 61–68.

35 JONES, D.F., S. K. MIRRAZAVI S.K., TAMIZ, M., 2002. Multi-Objective Meta-Heuristics: An Overview of the Current State-of-the-Art. *European Journal of Operational Research*, 137, p. 1-9.

36 LAWLER, E.L., LENSTRA, J.K., RINNOOY KAN, A.H.G., SHMOYS, D.B., 1993. Sequencing and scheduling: Algorithms and complexity. In: Graves, S.C. (Ed.), Handbooks in Operations Research and Management Science, Vol. 4. Elsevier Science Publishers, Amsterdam, pp. 445–552.

37 LIAO, C.J., YU, W.C., JOE, C.B., 1997. Bicriterion scheduling in the two-machine flowshop. *Journal of Operational Research Society* , 48, p. 929-935.

38 MIYAZAKI, S., NISHIYAMA, N., HASHIMOTO, F., 1978. An adjacent pairwise approach to the mean flowtime scheduling problem. *Journal of the Operations Research Society of Japan*, 21, p. 287–299.

39 MINELLA, G., RUIZ, R., CIAVOTTA, M., 2008. A review and evaluation of multi-objective algorithms for the flowshop scheduling problem. *Accepted for publication at INFORMS Journal on Computing.*

40 MLADENOVIC, N. & HANSEN, P., 1997 - Variable Neighborhood Search. *Computers and Operations Research*, 24:1097-1100.

41 MOCCELLIN, J.V., 1994, "Comparison of Neighborhood Search Heuristics for the Flow Shop Sequencing Problem, *Fourth International Workshop on Project Management and Scheduling*, July 1994, Leuven-Belgium, Proceedings, p. 228-231.

42 MOCCELLIN, J.V. & NAGANO, M.S.: Evaluating the Performance of Tabu Search Procedures for Flow Shop Sequencing. *Journal of the Operational Research Society*, 49: 1296-1302, 1998.

43 MORSE, J.N., 1980. Reducing the size of the nondominated set: Pruning by clustering,
Computers and Operations Research, 7, 55-66.

44 MURATA, T., ISHIBUCHI, H., TANAKA, H., 1996. Multi-objective genetic algorithm and its applications to flowshop scheduling. *Computers and Industrial Engineering*, 30, p. 957-968.

45 NAWAZ, M., ENSCORE, Jr., E.E., HAM, I., 1983. A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. *OMEGA*, 11, p. 91–95.

46 PARETO, V. , 1896. *Cours D'Economie Politique*, vol. 1. Lausanne: F. Rouge.

47 PINEDO, M., 1995. Scheduling: Theory, Algorithms and Systems. *Prentice Hall international series in industrial and systems engineering*, Prentice Hall, New Jersey, 1995.

48 RAJENDRAN, C., 1993. Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. *International Journal of Production Economics*, 29, p. 65–73.

49 RAJENDRAN. C., ZIEGLER, H., 2003. Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flowshop with sequence-dependent setup times. *European Journal of Operational Research*, 149, p. 513-522.

50 RUIZ, R., MAROTO, C., ALCARAZ, J., 2005. Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. *European Journal of Operational Research*, 165, p. 34–54.

51 SAYIN, S., KARABATI, S., 1999. A bicriteria approach to the two-machine flow shop scheduling problem. *European Journal of Operational Research*, 113, p. 435-449.

52 SRIDHAR, J., RAJENDRAN, C., 1996. Scheduling in flowshop and cellular manufacturing systems with multiple objectives – a genetic algorithm approach. *Production Planning and Control*, 7, p. 374-382.

53 STAFFORD E.F., 1988. On the development of a mixed integer linear programming model for the flowshop sequencing problem., *Journal of Operational Research Society*, 39, p. 1163 - 1174.

54 TAILLARD, E., 1990. Some efficient heuristics methods for the flowshop sequencing problems. *European Journal of Operational Research*, 47, p. 65–74.

55 T'KINDT, V., BILLAUT, J.C., 2001. Multicriteria scheduling problems: a survey. *RAIRO Operational Research,* 35, p. 143-163 .

56 VARANDHARAJAN, T.K., REJENDRAN, C., 2005. A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime jobs. *European Journal of Operational Research*, 167, p. 772-795.

57 WIDMER, M., HERTZ, A., 1989. A new heuristic method for the flowshop sequencing problem. *European Journal of Operational Research*, 41, p. 186–193.

58 WOO, H.S., YIM, D.S., 1998. A heuristic algorithm for mean flowtime objective in flowshop scheduling. *Computers and Operations Research*, 25, p.175–182.