UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ENGENHARIA DE SISTEMAS

HEURÍSTICAS E METAHEURÍSTICAS PARA OTIMIZAÇÃO COMBINATÓRIA MULTIOBJETIVO

José Elias Claudio Arroyo

Orientador: Prof. Dr. Vinícius Amaral Armentano

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas - UNICAMP, como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica.

Área de concentração: Automação.

Banca Examinadora:

Débora Pretti Ronconi – EP/USP
Luis Gimeno Latre – FEEC/UNICAMP
Luiz Satoru Ochi – DCC/IC/UFF
Paulo A. Valente Ferreira – FEEC/UNICAMP
Paulo Morelato França – FEEC/UNICAMP
Reinaldo Morabito Neto – DEP/UFSCar

Campinas, SP, Fevereiro de 2002

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Ar69h

Arroyo, José Elias Cláudio
Heurísticas e metaheurísticas para otimização combinatória multiobjetivo / José Elias Cláudio Arroyo. --Campinas, SP: [s.n.], 2002.

Orientador: Vinícius Amaral Armentano.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Otimização combinatória. 2. Heurística. 3. Algoritmos genéticos. I. Armentano, Vinícius Amaral. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

Neste trabalho apresentam-se contribuições de desenvolvimentos de métodos heurísticos para problemas de otimização combinatória multiobjetivo. O objetivo dos métodos propostos é gerar, em tempo computacional aceitável, um conjunto de soluções dominantes próximo ao conjunto Pareto-ótimo, permitindo ao decisor (*decision maker*) a escolha de uma solução que satisfaça seus interesses. Os métodos são testados no problema de programação de tarefas em um *flowshop* e no problema da mochila. Inicialmente, desenvolve-se uma heurística construtiva para gerar um conjunto de soluções dominantes do problema de flowshop multiobjetivo. Para obter conjuntos de soluções dominantes próximos aos conjuntos Pareto-ótimos, desenvolve-se uma heurística de busca local e duas metaheurísticas. A primeira é baseada em conceitos de Algoritmos Genéticos no qual usa-se fortemente o conceito de dominância de Pareto e combinam-se estratégias de elitismo, preservação de diversidade na população e busca local. A seguir, é proposto um método baseado em Busca Tabu na qual explora-se um conjunto de soluções em paralelo com o propósito de encontrar uma variedade de soluções distribuídas sobre toda a fronteira dominante. O desempenho dos métodos propostos é testado sobre uma variedade de instâncias do problema de *flowshop* multiobjetivo e do problema da mochila multiobjetivo.

Abstract

This work presents contributions to the development of heuristic methods for multiobjective combinatorial problems. The goal of the proposed methods is to generate in a reazonable time a set of approximately Pareto-optimal solutions, allowing the decision maker to choose a solution of interest. The methods are tested on the flowshop scheduling problem and the knapsack problem. First, we develop a constructive heuristic to generate a set of dominant solutions for the multiobjective flowshop scheduling problem. In order to find sets of dominant solutions which are closer to the Pareto-optimal sets, we develop a local search heuristic and two metaheuristics. The first one is based on Genetic Algorithms which strongly use the concept of Pareto dominance and combine elitism, population diversity and local search strategies. Then, we propose a Tabu Search-based method which explores a set of solutions in parallel to find a variety of solutions distributed along the Pareto frontier. The performance of the proposed methods is tested in a number of instances for the multiobjective flowshop scheduling problem and the multiobjective knapsack problem.

É com muito orgulho que dedico este trabalho à minha mãe, minha avó e meus irmãos, tenho a certeza de que mesmo distantes estamos sempre unidos pela força inexplicável que denominamos de amor.

Agradecimentos

Em primeiro lugar gostaria de expressar meus sinceros agradecimentos ao professor Vinícius Armentano pela orientação cuidadosa, paciência, amizade e pelas valiosas sugestões.

Agradecimentos a todos que me deram apoio para a realização deste trabalho, em especial:

- à Eva, pelo amor e pelo estímulo;
- à minha família, pelo carinho e solidariedade.
- aos colegas do DENSIS: Edílson, Luiz Fernando, Jeanne, Moacir, Denise, Rodrigo, Luciano, Alexandre, Cris, Luciana, Vinícius, Angela, Leandro, Marcos e a todos os que de alguma forma me auxiliaram no desenvolvimento da tese;
- a todos os professores do DENSIS;
- aos funcionários do DENSIS e da CPG em especial à Márcia, Noêmia e Ma Zé;
- aos meus amigos Adriano, M. Antonio, M. Siqueira, Richard, Herbert, Flor e a todos que compartilharam comigo, direta ou indiretamente, neste período de muito trabalho, alegrias e tristezas;
- ao CNPq pela ajuda financeira sem a qual não teria sido possível realizar este trabalho.

Sumário

Introdução

Otimização Combinatória é uma disciplina de tomada de decisões no caso de problemas discretos que pode ser encontrada em diversas áreas, tais como, problemas de planejamento e programação (*scheduling*) da produção, problemas de corte e empacotamento, roteamento de veículos, redes de telecomunicação, sistemas de distribuição de energia elétrica, problemas de localização, dentre outras. Em muitos destes problemas surgem freqüentemente vários critérios de desempenho (funções objetivos), em geral, conflitantes entre si. Por exemplo, o controle de chão de fábrica envolve a execução do plano de produção e para tal é necessário programar a produção em cada centro de trabalho de forma a satisfazer objetivos globais da empresa. Como destacado por Volmann et al. (1988), existem três objetivos básicos na programação de tarefas (*jobs*) na produção. O primeiro objetivo está relacionado com datas de entrega das tarefas: basicamente, não se deseja atraso em relação a essas datas, e quando o custo de estoque é relevante, tenta-se evitar que as tarefas sejam finalizadas muito antes dessas datas. O segundo objetivo está relacionado com o tempo de fluxo de tarefas no chão de fábrica: deseja-se que esse tempo seja curto, ou equivalentemente, que o estoque em processamento seja baixo. O terceiro objetivo envolve a utilização dos centros de trabalho: deseja-se maximizar a utilização de equipamentos e de mão de obra. No entanto, estes objetivos são, em geral, conflitantes e o analista de produção (decisor) deve optar por uma solução que pondere os objetivos globais da empresa. Objetivos conflitantes são mais a regra do que a exceção em diversos problemas reais e a otimização multiobjetivo é utilizada para tratar com essas situações.

Em otimização multiobjetivo, ao contrário de otimização mono-objetivo, em geral, não existem soluções ótimas no sentido de minimizarem (ou maximizarem) individualmente todos os objetivos. A característica principal de otimização multiobjetivo (quando todos os objetivos são de igual importância) é a existência de um conjunto grande de soluções aceitáveis que são superiores às demais. Estas soluções aceitáveis são denominadas soluções Pareto-ótimas ou

eficientes. A escolha de uma solução eficiente particular depende das caraterísticas próprias do problema e é atribuída ao decisor (*decision maker*).

Até a década de 80, a maioria dos métodos de otimização foram propostos para a resolução de problemas de programação linear e não linear. Métodos exatos propostos para resolver problemas de programação linear multiobjetivo, geralmente, usam métodos exatos de otimização mono-objetivo. As soluções Pareto-ótimas são obtidas resolvendo alguns problemas particulares, derivados do original, cujos ótimos globais correspondem às soluções Pareto-ótimas. Um exemplo típico é o método de escalarização das funções objetivos (ou das somas ponderadas) definido sobre o espaço das soluções factíveis do problema multiobjetivo original (Wierzbick, 1986). Métodos deste tipo não são facilmente adaptados para resolver problemas de otimização combinatória multiobjetivo, pois estes problemas possuem um elevado grau de complexidade. Sabe-se que o problema de decisão associado a muitos problemas de otimização combinatória mono-objetivo são NP-completos, i.e., eles não podem ser resolvidos através de algoritmos de tempo polinomial. A questão da complexidade computacional em problemas de otimização combinatória multiobjetivos envolve uma outra componente relacionada com a contagem do número de soluções do problema de decisão associado (Ehrgott, 2000). Este fator, obviamente, eleva o grau de intratabilidade de diversos problemas combinatórios.

Em otimização, a escolha do método de resolução a ser utilizado depende principalmente da razão entre a qualidade da solução gerada pelo método e o tempo gasto para encontrar essa solução. Nesse nível, a maioria dos problemas é intratável, ou seja, são problemas para os quais é improvável que se consiga desenvolver um algoritmo exato que possa ser executado em tempo razoável. Para viabilizar a obtenção de soluções é preciso lançar mão de métodos heurísticos. Esses métodos, quando bem desenvolvidos e adaptados aos problemas que se deseja resolver, são capazes de apresentar soluções de boa qualidade em tempo compatível com a necessidade de rapidez presente nos problemas. O desenvolvimento e sucesso dos métodos heurísticos, em especial as metaheurísicas, fomentou o interesse dos pesquisadores na década de 90 na aplicação desses métodos em problemas de otimização combinatória multiobjetivo, considerados difíceis computacionalmente (Ehrgott e Gandibleux, 2000).

Os métodos heurísticos, aqui considerados, podem ser divididos em três classes que diferem basicamente na forma como exploram o espaço de soluções dos problemas.

A primeira classe de heurísticas são as chamadas de construtivas. Estas heurísticas são especializadas para um dado problema e constróem uma solução pela adição de componentes da mesma através de regras específicas associadas com a estrutura do problema.

A segunda classe de heurísticas são as chamadas de Busca Local ou Busca em Vizinhança. Estas heurísticas iniciam com uma solução completa do problema, e constróem uma vizinhança desta solução que contém todas as soluções alcançáveis através de uma regra de movimento que modifica a solução inicial. Dessa vizinhança, escolhe-se uma solução que possua uma avaliação melhor que a solução inicial. A solução escolhida torna-se a nova solução inicial e o processo continua até encontrar um ótimo local. Claramente, a eficiência das heurísticas de busca local depende da escolha da solução inicial e da definição de uma vizinhança que estabelece uma relação entre as soluções no espaço de decisões. Uma vez tendo chegado ao ótimo local, essas heurísticas param e não são capazes de escapar da otimalidade local e explorar novas regiões do espaço de busca.

A terceira classe de heurísticas são chamadas de metaheurísticas, que são métodos inteligentes flexíveis, pois possuem uma estrutura com componentes genéricos que são adaptados ao problema que se quer resolver. Estes métodos possuem uma certa facilidade em incorporar novas situações e exploram o espaço de soluções permitindo a escolha estratégica de soluções piores que as já encontradas, na tentativa de superar a otimalidade local. Mesmo não garantindo otimalidade global, as metaheurísticas podem encontrar uma grande quantidade de ótimos locais. Existem várias metaheurísticas que apresentam princípios e estratégias distintas, dentre elas destacam-se as seguintes.

As metaheurísticas Busca Tabu e Simulated Annealing exploram uma vizinhança a cada iteração de acordo com suas estratégias e escolhem apenas um elemento dessa vizinhança a cada passo. Esse tipo de varredura do espaço de busca gera uma trajetória de soluções obtida pela transição de uma solução para outra de acordo com os movimentos permitidos pelo método. A metaheurística GRASP é um método de múltiplos reinícios. A cada reinício gera-se uma solução inicial através de uma heurística construtiva gulosa com aleatoriedade controlada

na escolha dos componentes da solução. A solução inicial é usada como ponto de partida para uma busca local convencional.

As metaheurísticas baseadas em Algoritmos Genéticos e Scatter Search, exploram uma população de soluções a cada iteração. As estratégias de busca destes métodos permitem explorar várias regiões do espaço de soluções de cada vez. Dessa forma, ao longo das iterações não se constrói uma trajetória única de busca pois novas soluções sempre são obtidas através de combinações de soluções anteriores.

Ultimamente, alguns conjuntos de estratégias básicas de metaheurísticas diferentes vêm sendo combinados gerando métodos híbridos, por exemplo, métodos que mesclam características de busca dos Algoritmos Genéticos com técnicas de busca local.

Metaheurísticas têm sido aplicadas com muito sucesso para resolver problemas de otimização mono-objetivo (Osman e Laporte, 1996). Em otimização multiobjetivo, para gerar o conjunto das soluções Pareto-ótimas, vários problemas requerem algoritmos de tempos exponenciais, mesmo que a otimização isolada de alguns objetivos seja fácil. Assim, os métodos heurísticos resultam ser os mais convenientes para tratar com esses problemas. Recentemente, muitos pesquisadores tem proposto extensões de metaheurísticas para resolver problemas multiobjetivos (Ehrgott e Gandibleux, 2000; Coello, 2000; Van Veldhuizen e Lamont, 2000a; Jones et al. 2002). Os métodos metaheurísticos podem ser implementados com muita flexibilidade para resolver problemas multiobjetivos de otimização combinatória e problemas de otimização não linear. Atualmente, estes métodos constituem uma das ferramentas mais ativas na pesquisa em otimização multiobjetivo.

O objetivo principal deste trabalho é propor e analisar métodos heurísticos na resolução de problemas de otimização combinatória multiobjetivo. Inicialmente, desenvolve-se uma heurística construtiva para obter um conjunto de soluções dominantes para o problema de programação de tarefas em um *flowshop* com dois objetivos. Com o propósito de gerar conjuntos de soluções dominantes próximas aos conjuntos Pareto-ótimos de um problema de otimização multiobjetivo, desenvolve-se uma heurística de busca local e duas metaheurísticas: Algoritmos Genéticos e Busca Tabu. As metaheurísticas são aplicadas a dois tipos de problemas: o problema de programação de tarefas em um *flowshop* e o problema da mochila.

Este trabalho está dividido em 7 capítulos com o seguinte conteúdo. No Capítulo 1 são apresentados os conceitos básicos usados em otimização multiobjetivo. São apresentados também alguns métodos clássicos usados na solução de problemas de otimização multiobjetivo e alguns métodos de avaliação de algoritmos heurísticos. No Capítulo 2 apresentam-se as descrições da implementação de uma heurística construtiva para um problema de *flowshop* multiobjetivo e de uma heurística de busca local para gerar o conjunto de soluções dominantes de um problema de otimização multiobjetivo. No Capítulo 3 são descritas as principais estratégias usadas nos algoritmos genéticos multiobjetivos da literatura. No Capítulo 4 apresenta-se a proposta de um novo algoritmo genético híbrido para resolver problemas de otimização combinatória multiobjetivo. Também é descrita a implementação do algoritmo genético proposto para resolver dois problemas combinatoriais: o problema de *flowshop* e o problema da mochila. Para cada aplicação, é apresentada a análise dos resultados computacionais. No Capítulo 5 descreve-se a implementação de um novo algoritmo de busca tabu para gerar o conjunto de soluções dominantes de um problema de otimização combinatória multiobjetivo. São apresentados também os resultados computacionais obtidos na resolução do problema de *flowshop* e do problema da mochila. No Capítulo 6 são comparados os desempenhos das metaheurísticas propostas (algoritmo genético e busca tabu). Finalmente no Capítulo 7, apresentam-se as principais conclusões do trabalho.

Capítulo 1

Otimização Multiobjetivo

Um problema de otimização multiobjetivo, geralmente, consiste em minimizar (ou maximizar) simultaneamente um conjunto de critérios (objetivos) satisfazendo um conjunto de restrições. Em otimização multiobjetivo, não existe uma única solução que otimize cada um dos objetivos, mas sim um conjunto de soluções eficientes no qual nenhuma solução é melhor que outra solução para todos os objetivos. O decisor (*decision maker*) é o responsável pela escolha de uma solução eficiente particular que pondere os objetivos globais do problema.

Neste capítulo apresenta-se a formulação matemática de um problema de otimização multiobjetivo e os conceitos básicos usados neste tipo de problema. São descritos também alguns métodos clássicos para obter soluções Pareto-ótimas e alguns métodos de avaliação de algoritmos heurísticos.

1.1. Problema de otimização multiobjetivo

Um problema geral de otimização multiobjetivo consiste em encontrar um vetor de variáveis de decisão (solução) que satisfaça restrições e otimize uma função vetorial cujos elementos representam as funções objetivos. Estas funções representam os critérios de otimalidade que, usualmente, são conflitantes. Portanto, o termo "otimizar" significa encontrar soluções com todos os valores dos objetivos que não podem ser melhorados simultaneamente.

Formalmente, isto pode ser definido da seguinte maneira:

(P1)

Minimizar (ou maximizar) $\quad \boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}),..., f_r(\boldsymbol{x}))$

Sujeito a $\quad \boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}),..., g_p(\boldsymbol{x})) \leq \boldsymbol{b}$

$\boldsymbol{x} = (x_1, x_2,..., x_n) \in \boldsymbol{X}$

$\boldsymbol{z} = (z_1, z_2,..., z_r) \in \boldsymbol{Z}$

onde, $\boldsymbol{x}$ é o vetor decisão, $\boldsymbol{z}$ é o vetor objetivo, $\boldsymbol{X}$ denota o *espaço de decisões*, e $\boldsymbol{Z} = \boldsymbol{f}(\boldsymbol{X})$ é a imagem de $\boldsymbol{X}$ denominada *espaço objetivo*.

As restrições $\boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{b}$, $\boldsymbol{b} \in \Re^p$ e o espaço $\boldsymbol{X}$ determinam o *conjunto das soluções factíveis*: $\boldsymbol{X}^* = \{ \boldsymbol{x} \in \boldsymbol{X} : \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{b} \}$. Portanto, o problema (P1) pode ser escrito como:

(P2)

Minimizar (ou maximizar) $\quad \boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}),..., f_r(\boldsymbol{x}))$

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

A imagem de $\boldsymbol{X}^*$, é denominado *espaço objetivo factível* e é denotada por $\boldsymbol{Z}^* = \boldsymbol{f}(\boldsymbol{X}^*) = \{ \boldsymbol{f}(\boldsymbol{x}) : \boldsymbol{x} \in \boldsymbol{X}^* \}$.

Note que, a imagem de uma solução $\boldsymbol{x} = (x_1, x_2,..., x_n) \in \boldsymbol{X}^*$ no espaço objetivo é um ponto $\boldsymbol{z} = (z_1, z_2,..., z_r) = \boldsymbol{f}(\boldsymbol{x})$, tal que $z_j = f_j(\boldsymbol{x})$, $j = 1,...,r$. Na Figura 1.1 (a) mostra-se o espaço objetivo factível do problema de minimização (P2) com dois objetivos.

Na otimização de um único objetivo f, o espaço objetivo factível é *completamente ordenado*, o significa que, dados quaisquer dois elementos $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{X}^*$ é sempre verdade que $f(\boldsymbol{x}) \geq f(\boldsymbol{y})$ ou $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$. O objetivo é encontrar a solução (ou soluções) que forneça o mínimo (ou máximo) valor de f. No entanto, quando são considerados vários objetivos conflitantes em otimização multiobjetivo, não existe uma única solução que seja ótima com respeito a todos os objetivos. Por exemplo, em um problema de minimização, minimizar um dos objetivos pode causar o acréscimo de outros objetivos. O espaço objetivo, em geral, não é completamente ordenado, mas é *parcialmente ordenado* (Pareto, 1896). A ordenação parcial dos vetores objetivos é responsável pela distinção básica entre problemas de otimização multiobjetivos. Dados quaisquer dois vetores de decisão $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{X}^*$, de acordo com a relação de preferência "$\leq$", existem três possibilidades para seus correspondentes vetores objetivos:

$$\boldsymbol{f}(\boldsymbol{x}) \leq \boldsymbol{f}(\boldsymbol{y}),\ \boldsymbol{f}(\boldsymbol{y}) \leq \boldsymbol{f}(\boldsymbol{x}) \text{ ou } (\ \boldsymbol{f}(\boldsymbol{x}) \nleq \boldsymbol{f}(\boldsymbol{y})\ \text{ e } \boldsymbol{f}(\boldsymbol{y}) \nleq \boldsymbol{f}(\boldsymbol{x})\ ).$$

Exemplo:

Sejam $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{X}^*$ tal que:

1) Se $\boldsymbol{f}(\boldsymbol{x}) = (7, 3)$ e $\boldsymbol{f}(\boldsymbol{y}) = (9, 4)$ então $\boldsymbol{f}(\boldsymbol{x}) \leq \boldsymbol{f}(\boldsymbol{y})$;

2) Se $\boldsymbol{f}(\boldsymbol{x}) = (7, 3)$ e $\boldsymbol{f}(\boldsymbol{y}) = (9, 2)$ então $\boldsymbol{f}(\boldsymbol{x}) \nleq \boldsymbol{f}(\boldsymbol{y})$ e $\boldsymbol{f}(\boldsymbol{y}) \nleq \boldsymbol{f}(\boldsymbol{x})$.

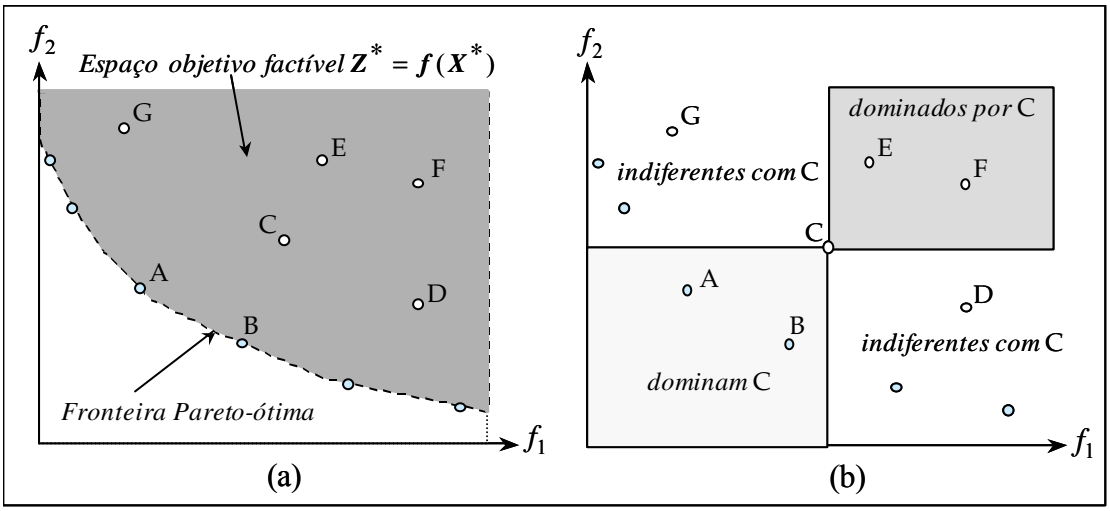


Figura 1.1. Dominância de Pareto no espaço objetivo.

A seguir, são apresentadas definições para problemas de minimização; definições para problemas de maximização são análogas.

Definição 1 (Dominância de Pareto no Espaço Objetivo Factível $\boldsymbol{Z}^*$)

Para quaisquer dois vetores objetivos $\boldsymbol{z}^1 = (z_1^1, ..., z_r^1)$ e $\boldsymbol{z}^2 = (z_1^2, ..., z_r^2)$ (pontos em $\boldsymbol{Z}^*$) diz-se que

1. $\boldsymbol{z}^1$ *domina* $\boldsymbol{z}^2$ se $\boldsymbol{z}^1 \leq \boldsymbol{z}^2$ e $\boldsymbol{z}^1 \neq \boldsymbol{z}^2$, isto é, $\forall j,\ z_j^1 \leq z_j^2$ e para algum j, $z_j^1 < z_j^2$.
2. $\boldsymbol{z}^1$ e $\boldsymbol{z}^2$ são *indiferentes* (ou possuem o mesmo grau de dominância) se $\boldsymbol{z}^1$ não domina $\boldsymbol{z}^2$ nem $\boldsymbol{z}^2$ domina $\boldsymbol{z}^1$.

Na Figura 1 (b), o ponto C domina os pontos pertencentes ao retângulo superior direito (subconjunto do espaço objetivo). Os pontos pertencentes ao retângulo inferior esquerdo dominam o ponto C. Os pontos G, C e D são indiferentes.

Definição 2 (Dominância de Pareto no Conjunto de Soluções Factíveis X^*)

Para quaisquer duas soluções $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{X}^*$ diz-se que

1. $\boldsymbol{x}$ *domina* $\boldsymbol{y}$ se a imagem de $\boldsymbol{x}$ domina a imagem de $\boldsymbol{y}$, isto é, $\boldsymbol{f}(\boldsymbol{x}) \leq \boldsymbol{f}(\boldsymbol{y})$ e $\boldsymbol{f}(\boldsymbol{x}) \neq \boldsymbol{f}(\boldsymbol{y})$.
2. $\boldsymbol{x}$ é *indiferente* com $\boldsymbol{y}$ se $\boldsymbol{f}(\boldsymbol{x}) \not\leq \boldsymbol{f}(\boldsymbol{y})$ e $\boldsymbol{f}(\boldsymbol{y}) \not\leq \boldsymbol{f}(\boldsymbol{x})$.

Definição 3 (Otimalidade de Pareto)

1. Diz-se que $\boldsymbol{x}^* \in \boldsymbol{X}^*$ é uma *solução eficiente* (ou *Pareto-ótima*) se não existe qualquer outra solução $\boldsymbol{x} \in \boldsymbol{X}^*$ tal que $\boldsymbol{x}$ domine $\boldsymbol{x}^*$; $\boldsymbol{z}^* = \boldsymbol{f}(\boldsymbol{x}^*)$ é chamado de *ponto eficiente* ou ponto Pareto-ótimo.
2. O conjunto de todas as soluções eficientes é denominado *conjunto eficiente* (ou *conjunto Pareto-ótimo*).
3. A imagem em $\boldsymbol{Z}^*$ do conjunto Pareto-ótimo é denominada *fronteira Pareto-ótima*.

Na Figura 1.1 (a), mostra-se um exemplo da *fronteira Pareto-ótima*. Os pontos pertencentes a esta fronteira são os pontos Pareto-ótimos. Note que estes pontos são indiferentes uns aos outros.

Definição 4

Um ponto $\boldsymbol{z}^{\mathrm{o}} = (z_1^{\mathrm{o}}, \ldots, z_r^{\mathrm{o}}) \in \boldsymbol{Z}^*$ tal que

$$z_j^{\mathrm{o}} = min\{f_j(\boldsymbol{x}) : \boldsymbol{x} \in \boldsymbol{X}^*\}, j = 1, \ldots, r$$

é chamado *ponto ideal* (ou *ponto utópico*).

Note que, se existe o ponto ideal, então o problema estaria resolvido. Obviamente esta situação é extremamente improvável se o problema envolve objetivos conflitantes.

Definição 5

Uma solução $\boldsymbol{x} \in \boldsymbol{X}^*$ é dominada por um subconjunto $\boldsymbol{A} \subseteq \boldsymbol{X}^*$ se existe uma solução $\boldsymbol{y} \in \boldsymbol{A}$ tal que $\boldsymbol{y}$ domina $\boldsymbol{x}$.

1.2. Classificação de Métodos Multiobjetivos

Na solução de problemas multiobjetivos, dois aspectos importantes podem ser identificados: busca de soluções e tomada de decisões. O primeiro aspecto refere-se ao processo de otimização na qual a região factível é direcionada para soluções Pareto-ótimas. Como no caso de otimização mono-objetivo, a busca pode tornar-se difícil devido ao tamanho e complexidade do espaço de busca, podendo inviabilizar o uso de métodos exatos. A tomada de decisões envolve a seleção de um critério adequado para a escolha de uma solução do conjunto Pareto-ótimo. É necessário que o decisor faça uma ponderação (*trade-off*) dos objetivos conflitantes. A partir do ponto de vista do *decisor*, os métodos de otimização multiobjetivos podem ser classificados em três categorias, descritos a seguir:

Métodos a-priori

Estes métodos são caraterizados pela participação do decisor antes do processo de busca de soluções, i.e., antes de resolver o problema. Apresentamos a seguir dois tipos de métodos a-priori:

1) Os objetivos do problema são combinados em um único objetivo. Isto requer a determinação explícita de pesos para refletir a preferência de cada objetivo. A vantagem deste método é que podem ser aplicadas estratégias clássicas de otimização mono-objetivo sem nenhuma modificação.
2) Os objetivos são classificados em ordem decrescente de prioridade. Feito isto, o problema é resolvido para o primeiro objetivo sem considerar os demais. A seguir, o problema é resolvido para o segundo objetivo sujeito ao valor ótimo encontrado para o primeiro objetivo. Este processo é continuado até que o problema seja resolvido para o ultimo

objetivo sujeito aos valores ótimos dos outros objetivos. Para o caso de dois objetivos temos o seguinte problema:

$$\text{Minimizar } f_2(\boldsymbol{x}) \text{ sujeito a } f_1(\boldsymbol{x}) = f_1^*, \quad \boldsymbol{x} \in \boldsymbol{X}^*$$

onde f_1^* é a solução do problema

$$\text{Minimizar } f_1(\boldsymbol{x}) \text{ sujeito a } \boldsymbol{x} \in \boldsymbol{X}^*.$$

Métodos a-posteriori

Nestes métodos, o processo de decisão é feito logo após a realização da busca de soluções. A busca é feita, considerando-se que todos os objetivos são de igual importância. Ao final do processo da busca tem-se um conjunto de soluções aproximadas ou Pareto-ótimas. A partir deste conjunto, o responsável pelas decisões deve selecionar uma solução que representa a solução adequada do problema.

Métodos iterativos

Nestes métodos, o responsável pela decisão intervém durante o processo de otimização (busca de soluções) articulando preferências e guiando a busca para regiões onde exista soluções de interesse.

1.3. Métodos Tradicionais de Otimização Multiobjetivo

A maior dificuldade em otimização multiobjetivo é a existência de objetivos conflitantes, isto é, nenhuma das soluções factíveis otimiza simultaneamente todos os objetivos. As soluções ótimas para cada objetivo são, geralmente, diferentes e não satisfazem as necessidades do decisor. O decisor pode precisar de soluções que satisfaçam certas prioridades associadas com os objetivos. Para encontrar tais soluções, os métodos clássicos (Cohon, 1978; Steuer, 1986) escalarizam os objetivos formando um único objetivo. Estes métodos definem um problema substituto, reduzindo a otimização vetorial a um problema de

otimização escalar. Este problema possui algumas restrições adicionais e para sua solução requer a definição sistemática de parâmetros.

A seguir, são apresentados dois métodos clássicos que têm sido muito aplicados para resolver problemas multiobjetivos de diversas áreas.

1.3.1. Método da Soma Ponderada

Este método é, provavelmente, o mais simples dos métodos clássicos que consiste em transformar o problema multiobjetivo original em um problema escalar mono-objetivo. Usando pesos diferentes para cada objetivo, forma-se uma função *f* que é a combinação linear dos objetivos. O problema escalar resultante é:

(P3)

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

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

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

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

O teorema a seguir fornece condições suficientes para que uma solução do problema ponderado (P3) seja Pareto-ótima (Chankong e Haimes, 1983).

Teorema

Dado um vetor de pesos $\boldsymbol{w} = (w_1,...,w_r)$, uma solução $\boldsymbol{x}^*$ de (P3) é solução Pareto-ótima se

a) $\boldsymbol{x}^*$ é a solução única de (P3), ou

b) $w_i > 0, \forall i = 1,...,r$.

Para tentar obter soluções Pareto-ótimas, deve-se resolver iterativamente o problema (P3) considerando diferentes vetores de pesos positivos. Neste caso, o decisor ou o otimizador é encarregado pela definição dos pesos apropriados de acordo com a importância dos objetivos. Para que os pesos w_i reflitam aproximadamente a importância dos objetivos, as funções objetivos devem ser normalizadas expressando aproximadamente os mesmos valores.

A principal desvantagem deste método é que ele não consegue gerar todas as soluções Pareto-ótimas quando o espaço objetivo é não convexo. Isto é ilustrado na Figura 1.2, para o caso de dois objetivos. Considere os pesos w_1 e w_2 para minimizar a seguinte função:

$$y = w_1.f_1(\boldsymbol{x}) + w_2.f_2(\boldsymbol{x}),\ \boldsymbol{x} \in \boldsymbol{X}^*.$$

Esta equação pode ser escrita como:

$$f_2(\boldsymbol{x}) = -\frac{w_1}{w_2} f_1(\boldsymbol{x}) + \frac{y}{w_2}.$$

Esta ultima equação define uma reta L cuja inclinação é $-\frac{w_1}{w_2}$ e intersecta o eixo do f_2 em $\frac{y}{w_2}$. Esta reta é tangente (ou suporte) ao espaço objetivo factível $\boldsymbol{Z}^*$ em um ponto Pareto-ótimo. De forma geral, o método da soma ponderada consiste em gerar diferentes retas suportes, definidas por valores de w_1 e w_2. Em geral, nem todos os pontos Pareto-ótimos admitem retas suportes. Na Figura 1.2, os pontos C e D não possuem retas suportes, i.e., estes pontos não podem ser encontrados pela minimização da função f do problema (P3).

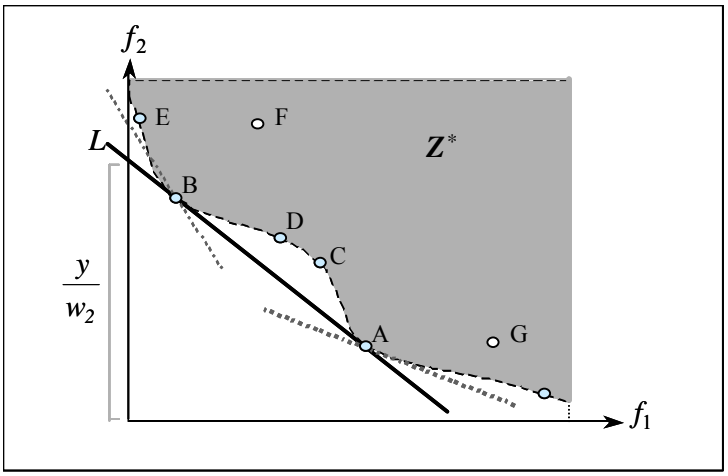


Figura 1.2. Interpretação gráfica do método da soma ponderada.

1.3.2. Método ε-restrito

Este método é baseado na minimização do objetivo de maior prioridade sujeito à

limitação dos outros objetivos. Sendo f_1 o objetivo de maior importância, o problema pode ser formulado da seguinte maneira:

$$\textbf{(P4)} \qquad \begin{aligned} &\text{Minimizar} && f_1(\boldsymbol{x}) \\ &\text{Sujeito a} && f_i(\boldsymbol{x}) \le \varepsilon_i\ ,\ i = 2,..., r \\ & && \boldsymbol{x} \in \boldsymbol{X}^* \end{aligned}$$

onde ε_i são limitantes superiores dos objetivos f_i , $i = 2,..., r$.

Variando convenientemente os limitantes ε_i, é possível gerar o conjunto Pareto-ótimo, mesmo quando o espaço objetivo é não convexo. Quando as funções objetivos e as funções restrições são lineares, então (P4) é um problema de programação linear.

Na Figura 1.3, mostra-se um exemplo deste método para o caso de um problema bi-objetivo. A reta $\varepsilon_2 = k$ limita o espaço de soluções, os pontos A, B, C, D e G correspondem a soluções factíveis do problema. A Figura 1.3 também mostra um exemplo quanto ocorre problemas com este método. Se o limitante superior não é selecionado adequadamente ($\varepsilon_2 = k'$), o sub-espaço obtido pelas restrições pode ser vazio, i.e. o problema (P4) não possui solução. Para evitar esta situação, inicialmente deve-se gerar um conjunto de valores apropriados para ε_i . Cohon (1978) desenvolveu um algoritmo para obter valores adequados dos limitantes.

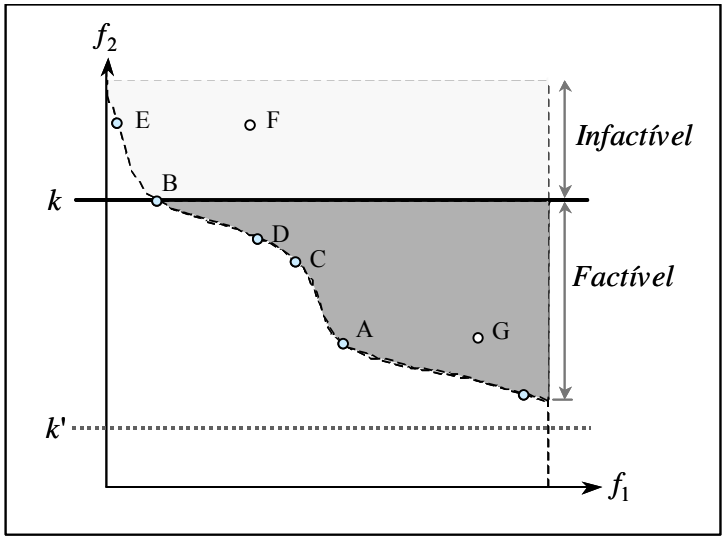


Figura 1.3. Interpretação gráfica do método ε-restríto.

1.4. Métodos de Avaliação de Heurísticas Multiobjetivo

Algoritmos heurísticos (ou simplesmente heurísticas) de otimização são métodos que obtêm soluções aproximadas para problemas de otimização. O desenvolvimento de heurísticas surge em resposta à impossibilidade de se resolver satisfatoriamente diversos problemas de otimização NP-difíceis.

Em otimização mono-objetivo, a qualidade de uma solução aproximada é avaliada de uma maneira simples, fazendo-se a diferença relativa entre os valores da solução heurística e da solução ótima. No entanto, em otimização multiobjetivo, a avaliação de aproximações do conjunto Pareto-ótimo não é tão trivial. Não existe uma medida simples e natural que seja capaz de capturar informação sobre a qualidade de um conjunto aproximado $\boldsymbol{H}$ em relação ao conjunto Pareto-ótimo (conjunto de referência $\boldsymbol{R}$).

Apresenta-se a seguir algumas técnicas de avaliação de heurísticas multiobjetivos que foram propostas na literatura.

1.4.1. Avaliação Analítica de Daniels

Daniels (1992) apresenta um algoritmo polinomial (apresentado no Apêndice A) para determinar a qualidade da aproximação de um conjunto de pontos heurísticos em relação ao conjunto dos pontos Pareto-ótimos. Na técnica de avaliação do Daniels, cada ponto eficiente (heurístico) pode ser convertido num valor que depende somente da importância associada com cada objetivo, e para o caso de problemas de minimização, assume-se que a preferência por um objetivo diminui com o incremento do valor deste objetivo. Daniels define a seguinte função linear ponderada:

$$u(\boldsymbol{z}) = u(z_1,...,z_r) = \sum_{j=1}^{r} \lambda_j \, u_j(z_j) \tag{1.1}$$

onde u_j é a função utilidade (preferência) associada com o j-ésimo objetivo e $\sum_{j=1}^{r} \lambda_j = 1$.

Para o caso bi-objetivo,

$$u(\boldsymbol{z}) = u(z_1, z_2) = \lambda \, u_1(z_1) + (1-\lambda) u_2(z_2) \tag{1.2}$$

Sejam $\boldsymbol{R} = \{ z_1^e, ..., z_{Ne}^e \}$ um conjunto de *Ne* pontos eficientes (ou de referência) e $\boldsymbol{H} = \{ z_1^h, ..., z_{Nh}^h \}$ um conjunto de *Nh* pontos heurísticos. De (1.2), um ponto z_k^e representa a melhor alternativa dentre os pontos de $\boldsymbol{R}$ se

$$u(z_k^e) \geq u(z_{k'}^e), \ \forall\, k' = 1,..., Ne. \tag{1.3}$$

Analogamente, um ponto z_l^h representa a melhor alternativa dentre os pontos de $\boldsymbol{H}$ se

$$u(z_l^h) \geq u(z_{l'}^h), \ \forall\, l' = 1,..., Ne. \tag{1.4}$$

Dado um valor do peso λ para os quais os pontos z_k^e e z_l^h representam, respectivamente, o melhor ponto eficiente e melhor ponto heurístico, então o erro relativo de aproximação *E* pode ser expressado como:

$$E = \frac{u(z_k^e) - u(z_l^h)}{u(z_k^e)} \tag{1.5}$$

Como a preferência por um objetivo varia, a identificação dos melhores pontos eficiente e heurístico e o correspondente erro relativo de aproximação, também variam. Daniels (1992) determina os valores médio e máximo de *E* sobre todos os valores do peso λ.

Um ponto eficiente z_k^e (ponto heurístico z_l^h) pode ser eliminado se para algum valor de λ entre 0 e 1 existe um ponto $z_{k'}^e \in \boldsymbol{R}$ ($z_{l'}^h \in \boldsymbol{H}$) tal que

$$u(z_{k'}^e) \geq u(z_k^e) \ \ (\, u(z_{l'}^h) \geq u(z_l^h) \,).$$

Os pontos eficientes e heurísticos que são inferiores com respeito à função $u(z)$ não são considerados no cálculo do erro de aproximação. Em outras palavras, os pontos eficientes (heurísticos) que não possuem hiperplano suporte são descartados. Esta é a principal desvantagem da ponderação linear do Daniels, contudo ela é uma medida muito útil e natural para avaliar a qualidade dos pontos heurísticos suportados por um hiperplano. A Figura 1.4 mostra um exemplo para um problema bi-objetivo. A faixa do parâmetro λ sobre a qual cada ponto eficiente permanece ótimo (melhor alternativa) é determinada e o valor objetivo correspondente comparado ao melhor ponto heurístico sobre a mesma faixa de λ para obter os

erros médio e máximo (E_{med} e E_{max}, respectivamente). Daniels (1992) sugere uma classe geral de funções utilidade que produzem preferências, as quais decrescem monotonamente com o incremento de cada objetivo:

$$u_j(z_j) = \left(\frac{\bar{z}_j - z_j}{\bar{z}_j - \underline{z}_j} \right)^{b_j}, \tag{1.6}$$

onde $\bar{z}_j$ e $\underline{z}_j$ denotam os valores máximo e mínimo do j-ésimo objetivo dentre todos os pontos heurísticos e os pontos de eficientes, e $b_j > 0$, que garante um valor escalar entre 0 e 1. Assume-se $b_j = 1$, i.e., uma preferência linear para cada objetivo.

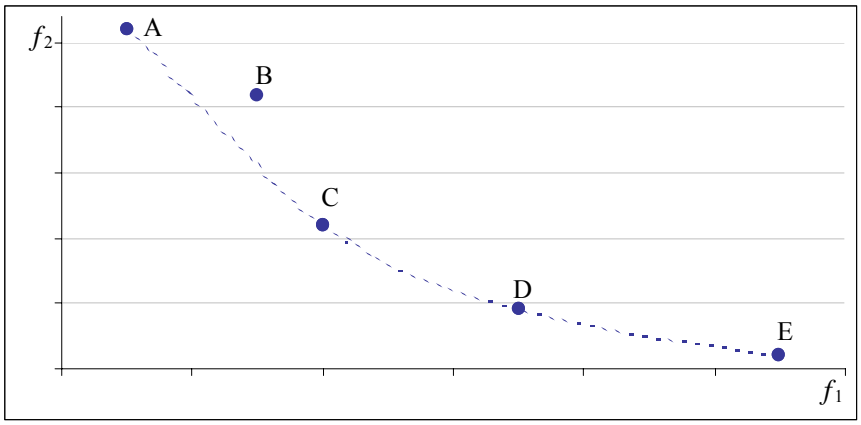


Figura 1.4. O ponto eficiente B é descartado, não possui hiperplano suporte

1.4.2. Medidas de Cardinalidade

Se for conhecido o conjunto de todos os pontos Pareto-ótimos (conjunto de referência ***R***), poderia parecer natural usar, como medida de qualidade, a porcentagem de pontos de referências encontrados pelo método heurístico:

$$C_1(\boldsymbol{H}) = \frac{|\boldsymbol{H} \cap \boldsymbol{R}|}{|\boldsymbol{H}|} \times 100\%. \tag{1.7}$$

No caso de problemas de tamanhos reais, pode ser impossível obter, em um tempo razoável, uma porcentagem significativa de pontos Pareto-ótimos. É mais importante obter pontos aproximados e distribuídos por toda a fronteira Pareto-ótima. Neste caso a medida C_1

não é apropriada para medir a qualidade de um conjunto aproximado $\boldsymbol{H}$. Note que C_1 pode ser zero mesmo que $\boldsymbol{H}$ seja uma boa aproximação.

Outra medida envolvendo cardinalidade é definida como a porcentagem de pontos heurísticos não dominados por pontos de referência:

$$C_2(\boldsymbol{H}) = \frac{|\{z \in \boldsymbol{H} : \text{não existe } z' \in \boldsymbol{R} \text{ tal que } z' \text{ domina } z\}|}{|\boldsymbol{H}|} \times 100\% . \qquad (1.8)$$

Esta medida também não é apropriada para medir a qualidade de um conjunto aproximado $\boldsymbol{H}$, pois ela não considera a distribuição dos pontos heurísticos e a distância em relação aos pontos de referencia.

1.4.3. Medidas de Distâncias de Czyzak e Jaszkiewicz

Para evitar as desvantagens das medidas de cardinalidade, Czyzak e Jaszkiewicz (1998) e Ulungu et al. (1998) propõem duas medidas de distância para medir a proximidade de um conjunto $\boldsymbol{H}$ em relação ao conjunto de referência $\boldsymbol{R}$. Eles assumem que $\boldsymbol{H}$ é uma boa aproximação de $\boldsymbol{R}$ se $\boldsymbol{H}$ fornece informação importante de todas as regiões do conjunto $\boldsymbol{R}$, em outras palavras, se para cada ponto $z \in \boldsymbol{R}$ existe um ponto $z' \in \boldsymbol{H}$ tal que a distância entre z' e z é pequena. As medidas de distância proposta por Czyzak e Jaszkiewicz (1998) são definidas da seguinte maneira:

$$D_{med} = \frac{1}{|\boldsymbol{R}|} \sum_{z \in \boldsymbol{R}} \min_{z' \in \boldsymbol{H}} d(z', z) \quad \text{e} \quad D_{max} = \max_{z \in \boldsymbol{R}} \{\min_{z' \in \boldsymbol{H}} d(z', z)\} \qquad (1.9)$$

onde $|\boldsymbol{R}|$ é a cardinalidade do conjunto $\boldsymbol{R}$ e d é definido por

$$d(z', z) = \max_{j=1,\ldots,r} \left\{ \frac{1}{\Delta_j} \left(z'_j - z_j \right) \right\}, \; z' = (z'_1, \ldots, z'_r) \in \boldsymbol{H}, z = (z_1, \ldots, z_r) \in \boldsymbol{R} \qquad (1.10)$$

onde $\Delta_j = \max f_j - \min f_j$ com $\max f_j = \max \{z_j = f_j(\boldsymbol{x}), \; z = \boldsymbol{f}(\boldsymbol{x}) \in \boldsymbol{R}\}$ e $\min f_j = \min \{z_j = f_j(\boldsymbol{x}), \; z = \boldsymbol{f}(\boldsymbol{x}) \in \boldsymbol{R}\}$.

Note que D_{med} é a média das distancias de um ponto $\boldsymbol{z} \in \boldsymbol{R}$ ao ponto mais próximo em $\boldsymbol{H}$, enquanto D_{max} fornece o máximo das distancias mínimas de um ponto $\boldsymbol{z} \in \boldsymbol{R}$ a algum ponto em $\boldsymbol{H}$.

D_{med} e D_{max} foram usadas por Viana e Pinho de Sousa (2000), que também sugerem o uso da medida de distância ao ponto ideal $\boldsymbol{z}^{o}$ dada por:

$$d(\boldsymbol{H},\ \boldsymbol{z}^{o}) = \min_{\boldsymbol{z}' \in \boldsymbol{H}} \{d(\boldsymbol{z}', \boldsymbol{z}^{o})\}$$

1.4.4. Funções de Utilidade de Hansen e Jaszkiewicz

Hansen e Jaszkiewicz (1998) analisam varias medidas e propõem um esquema geral para comparar e avaliar conjuntos aproximados. Medidas de distância não expressam a preferência do decisor, usualmente representado por funções utilidade. Hansen e Jaszkiewicz (1998) propõem um conjunto de funções utilidade baseadas nas normas ponderadas L_p dadas por

$$u_p = -\left(\sum_{j=1}^{r} \lambda_j \left(z_j - z_j^{o} \right)^p \right)^{1/p} \quad p \in \{1, 2, ...\} + \{\infty\} \tag{1.11}$$

onde $\boldsymbol{z}^{o} = (z_1^{o}, ..., z_r^{o})$ com $z_j^{o} = \min_{\boldsymbol{x} \in \boldsymbol{X}^*} \left\{ z_j = f_j(\boldsymbol{x}) \right\},\ j = 1, ..., r$ é o ponto ideal e $\lambda_j \geq 0$ são pesos.

Para $p = \infty$, obtém-se a função utilidade ponderada de Tchebycheff,

$$u_\infty = -\max_j \left\{ \lambda_j \left(z_j - z_j^{o} \right) \right\} \tag{1.12}$$

que Hansen e Jaszkiewicz sugerem, caso a preferência do decisor não seja conhecida. Esta medida pode ser facilmente avaliada, se é conhecido uma aproximação do ponto ideal identificado por uma heurística. Contudo, a função de Tchebycheff não fornece muita informação para o decisor sobre a qualidade dos pontos.

1.4.5. Métricas de Van Veldhuizen e Lamont

Van Veldhuizen e Lamont (2000b) propõem uma métrica para calcular a proximidade de um conjunto de pontos heurísticos $\boldsymbol{H}$ ao conjunto Pareto-ótimo (conjunto de referência $\boldsymbol{R}$). Esta métrica é definida como

$$G = \frac{1}{|\boldsymbol{H}|}\left(\sum_{z' \in \boldsymbol{H}} \left(\min_{z \in \boldsymbol{R}} \{d(z', z)\}\right)^2\right)^{1/2} \tag{1.13}$$

onde $d(z', z)$ é a distância Euclidiana entre os pontos z' e z.

Comparando com a medida D_{med} de Czyzak e Jaszkiewicz (1998), G é a média das distancias de um ponto $z' \in \boldsymbol{H}$ ao ponto mais próximo em $\boldsymbol{R}$, por tanto, G mede a proximidade do conjunto $\boldsymbol{H}$ ao conjunto $\boldsymbol{R}$ somente em algumas regiões de $\boldsymbol{R}$ (suponha que $|\boldsymbol{H}| = 1$ e $|\boldsymbol{R}| = 5$ e que o único ponto de $\boldsymbol{H}$ coincida com um ponto de $\boldsymbol{R}$, então $G = 0$).

Van Veldhuizen e Lamont (2000b) propõem também uma métrica adicional que mede a distribuição dos pontos no conjunto $\boldsymbol{H}$:

$$S = \sqrt{\frac{1}{|\boldsymbol{H}| - 1} \sum_{z \in \boldsymbol{H}} \left(\overline{d} - d_z\right)^2} \tag{1.14}$$

onde $d_z = \min_{z' \in \boldsymbol{H}} \{\sum_{j=1}^{r} | z_j - z'_j |\}$ $\quad z = (z_1, ..., z_r)$, $z' = (z'_1, ..., z'_r)$ e $\overline{d} = \frac{1}{|\boldsymbol{H}|} \sum_{z \in \boldsymbol{H}} d_z$

1.5. Escolha da Metodologia de Avaliação de Heurísticas

Neste trabalho para medir o desempenho do conjunto $\boldsymbol{H}$ gerado por uma heurística (ou metaheurística) em relação ao conjunto de referência $\boldsymbol{R}$, optamos pelas seguintes metodologias de avaliação:

1. *Medida de cardinalidade*: Determina-se o número de pontos Pareto-ótimos (ou referência) encontrados por uma heurística.
2. *Medida de distância*: Utiliza-se as medidas de distância D_{med} e D_{max} (1.9) propostas por Czyzak e Jaszkiewicz (1998), sendo diferente apenas a forma de calcular o intervalo de

variação de cada objetivo $\Delta_j = \max f_j - \min f_j$. Para calcular $\max f_j$ e $\min f_j$, além de considerar os pontos de referência, considera-se também todos os pontos heurísticos a serem avaliados:

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

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

Desta maneira as distâncias são normalizadas de forma que D_{med} e D_{max} estejam no intervalo [0,1].

3. *Avaliação analítica de Daniels*: As medidas de distância não expressam a preferência do decisor (*decision maker*), usualmente representado por funções utilidade. Para avaliar a qualidade dos pontos heurísticos em relação aos pontos de referência, implementa-se o algoritmo polinomial proposto por Daniels (1992) (para o caso de dois objetivos). Este algoritmo, que é baseado na maximização de uma função linear ponderada, determina os erros relativos médio (E_{med}) e máximo (E_{max}) correspondentes à qualidade dos pontos heurísticos no conjunto $\boldsymbol{H}$.

Quando o conjunto Pareto-ótimo não é conhecido e $\boldsymbol{H'}$ é o conjunto de pontos dominantes gerada por outra heurística, o conjunto de referência $\boldsymbol{R}$ é formado pelos pontos dominantes de $\boldsymbol{H} \cup \boldsymbol{H'}$ e para obter a qualidade dos conjuntos $\boldsymbol{H}$ e $\boldsymbol{H'}$ relativo a $\boldsymbol{R}$ são usadas as mesmas metodologias definidas acima.

Capítulo 2

Heurística Construtiva e de Melhoria para Problemas de Otimização Combinatória Multiobjetivo

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. Neste capítulo, é proposta uma nova heurística construtiva para resolver o problema de *flowshop* com dois objetivos. Apresenta-se também uma técnica de busca local para resolver problemas de otimização combinatória multiobjetivos.

2.1. Uma Heurística Construtiva para o Problema de Flowshop Multiobjetivo

Em muitos problemas reais de programação de tarefas, os analistas de produção defrontam-se com critérios de otimalidade conflitantes (Panwalkar et al. 1973). A maioria das pesquisas sobre problemas de programação de tarefas até a década de 70 concentrou-se na otimização de uma única medida de desempenho, tais como makespan (C_{max}), fluxo total (F), atraso máximo (T_{max}), atraso total (T) e número de tarefas atrasadas (n_T). C_{max} e F estão relacionadas, respectivamente, com a utilização máxima de recursos e com a minimização do

estoque em processamento, e os outros critérios estão relacionados com a data de entrega das tarefas. Nos anos 80, foram publicados alguns trabalhos sobre programação de tarefas multiobjetivo e resenhas destes foram escritas por Dileepan e Sen (1988) e Fry e Armstrong (1989).

Desde 1990, as pesquisas sobre programação de tarefas multiobjetivo, e mas geralmente, sobre otimização combinatória multiobjetivo, tem recebido maior atenção (Ehrgott e Gandibleux, 2000). O artigo mais recente sobre programação de tarefas multiobjetivo, publicado por Nagar et al. (1995a), inclui poucos problemas considerando múltiplas máquinas. O problema de programação flowshop multiobjetivo tem sido atacado de três maneiras. Alguns métodos *a-priori* foram apresentados por Rajendran (1995), Nagar et al. (1995b) e Şerifoğlu e Ulusoy (1998) onde os objetivos do problema são combinados em um único objetivo, e por Rajendran (1992), Gupta et al. (1999), Gupta et al. (2000a) e Gupta et al. (2000b) onde os objetivos são classificados em ordem decrescente de prioridade. Métodos *a-posteriori* foram propostos por Daniels e Chambers (1990), Liao et al. (1997), Sayin e Karabati (1999), Murata et al. (1996) e Ishibuchi et al. (1998). Métodos iterativos são apresentados por Hapke et al. (1998) e Teghem et al. (2000).

Nesta seção adota-se uma abordagem a-posteriori para o problema de programação flowshop multiobjetivo e desenvolve-se um procedimento heurístico para obter uma aproximação do conjunto de soluções Pareto-ótimas (eficientes).

A literatura sobre este tipo de problema é ainda escassa. Alguns algoritmos Branch-and-Bound para problemas com duas máquinas tem sido propostos. Daniels e Chambers (1990) consideram o par de objetivos (C_{max}, T_{max}), Liao et al. (1997) os pares (C_{max}, T) e (C_{max}, n_T), enquanto Sayin e Karabati (1999) consideram (C_{max}, F). Estes artigos apresentam resultados para instâncias até 30 tarefas. No lado dos métodos heurísticos, segundo nosso conhecimento, Daniels e Chambers propõem a única heurística construtiva multiobjetivo para duas e mais máquinas. Murata et al. (1996) desenvolvem um algoritmo genético multiobjetivo. Eles resolvem uma instância com 20 tarefas e 10 máquinas minimizando três objetivos (C_{max}, T, F) e uma instância para minimizar (C_{max}, T). Ishibuchi et al. (1998) estendem este algoritmo e incluem uma busca local. Eles também resolvem uma instância com 20 tarefas e 10 máquinas para minimizar (C_{max}, T_{max}) e a mesma instância para minimizar (C_{max}, T_{max}, F). Uma instância com 10 tarefas e 5 máquinas é resolvido para (C_{max}, T_{max}). Em ambos trabalhos

os autores mostram que seus resultados são melhores quando comparado com o algoritmo genético VEGA (Vector Evaluated Genetic Algorithm) proposto por Schaffer (1985). No entanto, os experimentos computacionais são tão limitados que não se pode concluir que os algoritmos genéticos propostos são realmente superiores ao VEGA.

Nesta seção apresenta-se um método heurístico que é especifico para o problema de flowshop e que constrói uma aproximação razoável do conjunto Pareto-ótimo com baixo tempo computacional. As heurísticas construtivas são necessárias para obter rapidamente uma primeira aproximação ao conjunto Pareto-ótimo (Ehrgott e Gandibleuz, 2000).

As soluções obtidas pela heurística aqui proposta podem ser utilizadas como soluções iniciais para alguma metaheurística. É proposta uma heurística de enumeração parcial baseada na inserção de tarefas e no conceito de dominância de Pareto quando comparam-se seqüências parciais e completas. Esta heurística é inspirada na heurística de NEH proposto por Nawaz et al. (1983), que é considerada como uma das melhores heurísticas construtivas para minimizar C_{max}. Este mesmo tipo de enumeração foi também usado com sucesso por Armentano e Ronconi (1999) para a minimização de T. A heurística é testada sobre o problema considerado para os seguintes pares de objetivos: i) (C_{max}, T_{max}) e ii) (C_{max}, T). Para o caso de duas máquinas, a heurística é comparada com os algoritmos Branch-and-Bound propostos por Daniels e Chambers (1990) e Liao et al. (1997), respectivamente, e para o caso de mais de duas máquinas, a heurística é comparada com enumeração completa. Para o par de critérios i) a heurística é também é comparada com a heurística desenvolvida por Daniels e Chambers (1990).

O problema de programação flowshop permutacional pode ser descrito da seguinte maneira. Considere n tarefas disponíveis para processamento no tempo $t = 0$ e requerendo uma única operação em cada uma das m máquinas ordenadas e disponíveis continuamente. A cada tarefa i são associados m tempos de processamentos p_{ij} que correspondem a cada máquina j, e uma data de entrega d_i. A seqüência de processamento das tarefas é a mesma em todas as máquinas. Para uma dada seqüência, seja C_i o tempo de finalização do processamento da tarefa i, e $T_i = \max\{C_i - d_i, 0\}$, o atraso da tarefa i. Os critérios de minimização de interesse neste trabalho são o makespan $C_{max} = \max_i\{C_i\}$, o atraso máximo $T_{max} = \max_i\{T_i\}$ e o atraso total $T = \sum_i T_i$.

2.1.1. Heurística Proposta

Nesta subseção descreve-se a heurística multiobjetivo de enumeração parcial (HMEP) proposta para o problema de programação flowshop com os seguintes pares de objetivos: i) (C_{max}, T_{max}) e ii) (C_{max}, T). A heurística necessita de duas seqüências de tarefas obtidas pelas seguintes regras de prioridade ou despacho:

- LPT (*longest processing time*): as tarefas são ordenadas de forma decrescente com o tempo total de processamento sobre as máquinas, $\sum_j p_{ij}$.
- TLB (*tardiness lower bound*): as tarefas são ordenadas de forma não-decrescente com o limitante inferior do atraso da tarefa i, $d_i - \sum_j p_{ij}$.

Nawaz et al. (1983) propõem uma heurística de enumeração parcial baseada na inserção de tarefas para minimizar o makespan usando a regra LPT, enquanto Armentano e Ronconi (1999) propõem a regra TLB para o critério do atraso total.

A heurística HMEP possui três fases. Na Fase 1, a partir da seqüência obtida pela regra LPT, constrói-se um conjunto de seqüências (soluções) que minimizam o makespan, e na Fase 2 é usada a seqüência gerada pela regra TLB para gerar boas soluções em termos do atraso máximo (ou total). A partir das soluções geradas nas Fases 1 e 2, a Fase 3 determina um conjunto de soluções dominantes que constitui uma aproximação do conjunto Pareto-ótimo.

Algoritmo 2.1. HMEP

Fase 1: LPT

1) Ordene as tarefas de acordo com a regra LPT obtendo a seqüência $\boldsymbol{s} = (j_1, j_2, ..., j_n)$.
2) Considere as duas primeiras tarefas j_1 e j_2 e forme as seqüências parciais: (j_1, j_2) e (j_2, j_1). Para cada seqüência parcial, calcule o valor da função $\boldsymbol{f} = (C_{max}, T_{max})$, ($\boldsymbol{f} = (C_{max}, T)$) e selecione a seqüência parcial dominante. Caso as seqüências sejam indiferentes, selecione as duas. Obtém-se então um conjunto $\boldsymbol{S}_2$ de seqüências parciais dominantes formado por duas tarefas.
3) Para $k = 3$ até n faça

   Para cada seqüência $\boldsymbol{s}_j \in \boldsymbol{S}_{k-1}$ ($\boldsymbol{S}_{k-1}$: conjunto de seqüências parciais com k-1 tarefas) faça

Para $i = 1$ até k faça

Insira a k-ésima tarefa de $\boldsymbol{s}$ na posição i de $\boldsymbol{s}_j$ obtendo a seqüência $\boldsymbol{s}_i$ com k tarefas. Calcule $\boldsymbol{f}(\boldsymbol{s}_i) = (C_{max}(\boldsymbol{s}_i), T_{max}(\boldsymbol{s}_i))$, ($\boldsymbol{f}(\boldsymbol{s}_i) = (C_{max}(\boldsymbol{s}_i), T(\boldsymbol{s}_i))$).

A partir das $k.|\boldsymbol{S}_{k-1}|$ seqüências parciais, construa o conjunto $\boldsymbol{S}_k$ das seqüências dominantes.

4) Fim da Fase 1. O conjunto $\boldsymbol{S}_n$ contém seqüências completas dominantes. Vá para Fase 2.

Fase 2: TLB

1)' Ordene as tarefas de acordo com a regra TLB obtendo a seqüência $\boldsymbol{s} = (j_1, j_2, ..., j_n)$.

Os passos 2)' e 3)' são identicos aos passos 2) e 3) da Fase 1, bastando substituir o conjunto $\boldsymbol{S}_k$ por $\boldsymbol{S'}_k$.

Ao finalizar esta fase obtém-se um conjunto $\boldsymbol{S'}_n$ com seqüências completas dominantes.

Fase 3:

A partir do conjunto $\boldsymbol{S}_n \cup \boldsymbol{S'}_n$, identifique o conjunto $\boldsymbol{D}$ das soluções dominantes que constitui uma aproximação do conjunto Pareto-ótimo.

Exemplo

Na Figura 2.1 mostra-se uma árvore gerada pela Fase 1 da heurística HMEP. Seja $\boldsymbol{s}$ = (1 2 3 4) uma seqüência de $n = 4$ tarefas ordenadas segundo a regra LPT. A partir das tarefas 1 e 2 são geradas as seqüências parciais (1 2) e (2 1), e a seqüência dominante é selecionada. Supondo que (1 2) seja dominante, então $\boldsymbol{S}_2$ = { (1 2) }. Inserindo a tarefa 3 em cada uma das posições das seqüências em $\boldsymbol{S}_2$, obtém-se o conjunto de seqüências parciais dominantes $\boldsymbol{S}_3$ = {(3 1 2), (1 2 3)}. Este processo é continuado até encontrar o conjunto de seqüências completas dominantes $\boldsymbol{S}_4$ = {(4 3 1 2), (3 1 4 2), (3 1 2 4), (1 2 4 3)}. Ordenando as tarefas segundo a regra TLB, a Fase 2 da heurística gera uma árvore similar à árvore mostrada na Figura 2.1 com o conjunto de seqüências completas dominantes $\boldsymbol{S'}_n$.

A Figura 2.2 ilustra os conjuntos $\boldsymbol{S}_n$, $\boldsymbol{S'}_n$ e $\boldsymbol{D}$ gerados pela heurística HMEP para uma instância com $n = 20$ e $m = 10$. Este exemplo mostra que as Fases 1 e 2 geram seqüências dominantes com valores otimizados de C_{max} e T_{max}, respectivamente . Portanto, ambas as fases são responsáveis pela geração de seqüências dominantes com valores intermediários de C_{max} e T_{max}.

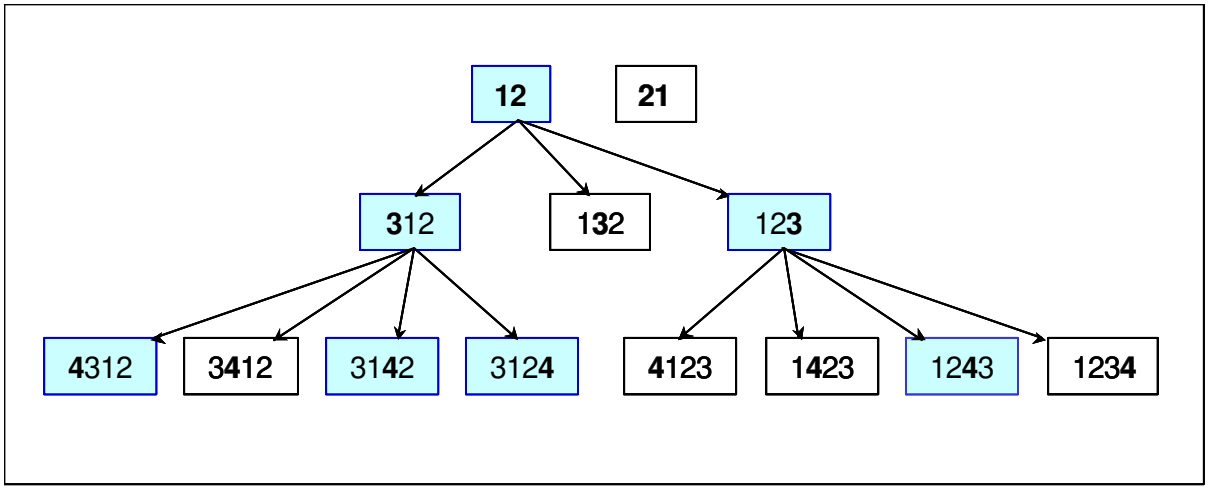


Figura 2.1. Estrutura da árvore gerada pela heurística HMEP.

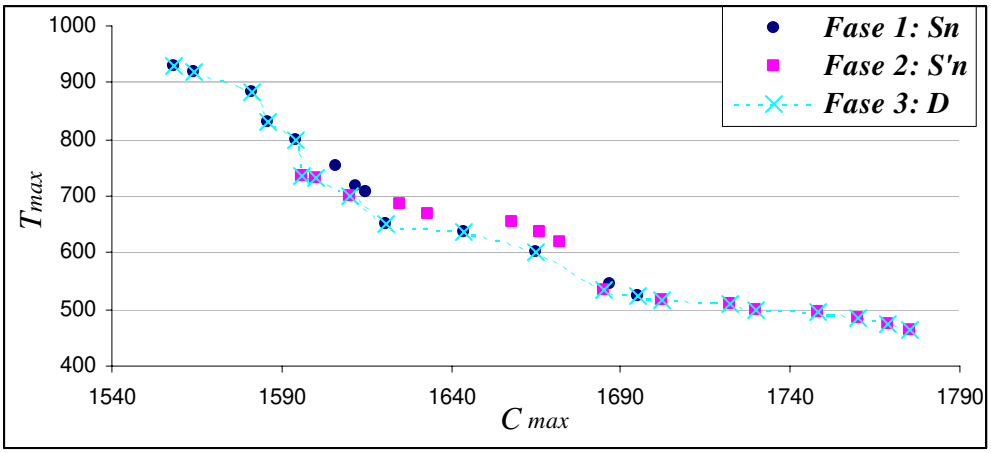


Figura 2.2. Soluções dominantes geradas pela heurística HMEP.

Na Fase 2, foram também testadas outras regras de despacho associados com a minimização do atraso: MDD (*modified due date*) (Baker e Kanet 1983) e EDD (*earliest due date*). No entanto, os melhores resultados foram obtidos com a regra TLB. Existem muitas regras de despacho para programação de tarefas (veja, por exemplo, Panwalkar e Iskander, 1977), e portanto não se pode afirmar que as regras usadas sejam as melhores. Outra combinação de duas ou mais regras pode gerar resultados melhores para uma determinada instância do problema. Nosso objetivo aqui é mostrar que o uso apropriado de regras acopladas a um esquema de enumeração parcial e a aplicação do conceito de dominância de

Pareto, conduzem a bons resultados. Outro ponto que deve ser enfatizado é o esquema da enumeração. Nawaz (1983) e Armentano e Ronconi (1999) num determinado nível da árvore selecionam um nó com o melhor valor do objetivo e expandem este nó fazendo inserções de uma nova tarefa. Neste caso o número de seqüências avaliadas é $-1 + n(n + 1)/2$. Na heurística HMEP pode-se expandir mais de um nó em cada nível, pois não teria sentido escolher só uma única seqüência parcial dominante. Então é possível que cada fase da heurística, no pior caso, avalie $2! + 3! + \ldots + n!$ seqüências, o que resulta em uma complexidade exponencial. No entanto, os experimentos computacionais têm mostrado que, para as instâncias testadas neste trabalho, o número de seqüências geradas tem complexidade polinomial $O(n^2)$.

2.1.2. Resultados Computacionais

Nesta seção apresentam-se a forma de geração das instâncias para o problema de flowshop e a análise dos resultados obtidos pela heurística HMEP. A heurística é testada sobre o problema considerado para os seguintes pares de objetivos: i) (C_{max}, T_{max}) e ii) (C_{max}, T). Para o caso de duas máquinas a heurística é comparada com os algoritmos Branch-and-Bound proposto, respectivamente, por Daniels e Chambers (1990) e Liao et al. (1997) (veja Apêndice B), e para o caso de mais de duas máquinas a heurística é comparada com enumeração completa. Para o par de critérios i) e para instâncias de grande porte a heurística é comparada com a heurística desenvolvida por Daniels e Chambers (1990) denotada por HDC (veja Apêndice C).

Para medir a performance do conjunto dominante $\boldsymbol{A}$ gerado por uma heurística em relação ao conjunto de referência $\boldsymbol{R}$, são usadas três metodologias de avaliação: medidas de cardinalidade, medidas de distância e a avaliação analítica de Daniels, descritas no Capítulo 1 (seção 1.5).

Os métodos heurísticos e exatos foram implementados na linguagem de programação C++ e os testes computacionais foram executadas numa estação de trabalho SUN Ultra 60.

2.1.2.1. Geração das Instâncias para o Problema de Flowshop

A geração das instâncias (problemas testes) para o problema de flowshop segue o mesmo esquema sugerido por Taillard (1993). Os tempos de processamento das tarefas são uniformemente distribuídos entre 1 e 99. São considerados quatro cenários para as datas de entrega, que são uniformemente distribuídos sobre o intervalo $[P(1 - a - b/2), P(1 - a + b/2)]$ (Potts e Van Wassenhove, 1982), onde a e b representam, respectivamente, o fator de atraso das tarefas e a faixa de dispersão das datas de entrega, e P é um limitante inferior do *makespan* (Taillard,1993) definido por

$$P = \max\left\{\max_{1\le j\le m}\left\{\sum_{i=1}^{n} p_{ij} + \min_{i}\sum_{l=1}^{j-1} p_{il} + \min_{i}\sum_{l=j+1}^{m} p_{il}\right\}, \max_{i}\sum_{j=1}^{m} p_{ij}\right\}.$$

Cada cenário é definido por uma combinação de valores para a e b:

Cenário 1: baixo fator de atraso (a = 0,2) e pequena faixa de datas de entrega (b = 0,6).

Cenário 2: baixo fator de atraso (a = 0,2) e larga faixa de datas de entrega (b = 1,2).

Cenário 3: alto fator de atraso (a = 0,4) e pequena faixa de datas de entrega (b = 0,6).

Cenário 4: alto fator de atraso (a = 0,4) e larga faixa de datas de entrega (b = 1,2).

Para cada combinação do número de tarefas n, número de máquinas m e uma matriz de tempos de processamento das tarefas foram geradas 4 instâncias, de acordo com os cenários de datas de entrega definidos acima. Foram considerados os seguintes conjuntos de instancias:

- ***Conjunto 1***: n = 10, 11; m = 5, 10; 20 matrizes de tempo de processamento; totalizando 4×20×4 = 320 instâncias.
- ***Conjunto 2***: n = 10, 15, 20, 25; m = 2; 10 matrizes de tempo de processamento; totalizando 4×10×4 =160 instâncias.
- ***Conjunto 3***: n = 20, 50, 80; m = 5, 10, 20; 10 matrizes de tempo de processamento; totalizando 9×10×4 = 360 instâncias.

Na Tabela 2.1, para cada cenário, mostra-se a variação das datas de entrega para alguns conjuntos de instâncias. Note que, as instâncias correspondentes aos cenários 2 e 4 possuem largas faixas de datas de entrega e, os cenários 1 e 3 geram datas de entrega variando em faixas menores. Também se observa que, as menores datas de entrega são geradas no cenário 4, seguido pelo cenário 2; e as datas de entrega mais altas são geradas no cenário 2, seguido pelo cenário 4.

Tabela 2.1. Variação das datas de entrega

Cenário	Intervalo de datas de entrega		
	$n = 10,\ m = 5$	$n = 20,\ m = 10$	$n = 50,\ m = 20$
1	[318, 850]	[644, 1419]	[1728, 3802]
2	[120, 1090]	[250, 1806]	[620, 4839]
3	[209, 690]	[434, 1161]	[1037, 3111]
4	[1, 945]	[0, 1548]	[0, 4148]

Aplicando o método de enumeração completa ao Conjunto 1 de instâncias, foi possível observar as seguintes características:

- Independente do par de critérios (C_{max}, T_{max}) ou (C_{max}, T), as instâncias com cenários 2 e 4 de data de entrega, geralmente, possuem maior número de soluções Pareto-ótimas ou eficientes.
- A minimização do par de objetivos (C_{max}, T) sempre gera maior quantidade de pontos eficientes que o par (C_{max}, T_{max}).

Para uma instância com $n = 10$ e $m = 5$, na Figura 2.3 mostram-se os conjuntos de pontos eficientes obtidos na minimização de (C_{max}, T_{max}) e (C_{max}, T) e considerando os diferentes cenários de datas de entrega. Observe que, tanto para (C_{max}, T_{max}) e (C_{max}, T) os cenários 2 e 4 geram maior quantidade de pontos eficientes.

Não é tão simples fazer uma análise sobre a distribuição das soluções eficientes em relação aos objetivos otimizados. Na Figura 2.3, observa-se que para os cenários 2 e 4 os pontos estão concentrados (próximos uns aos outros), mas este comportamento não é sempre verdadeiro para todas as instâncias.

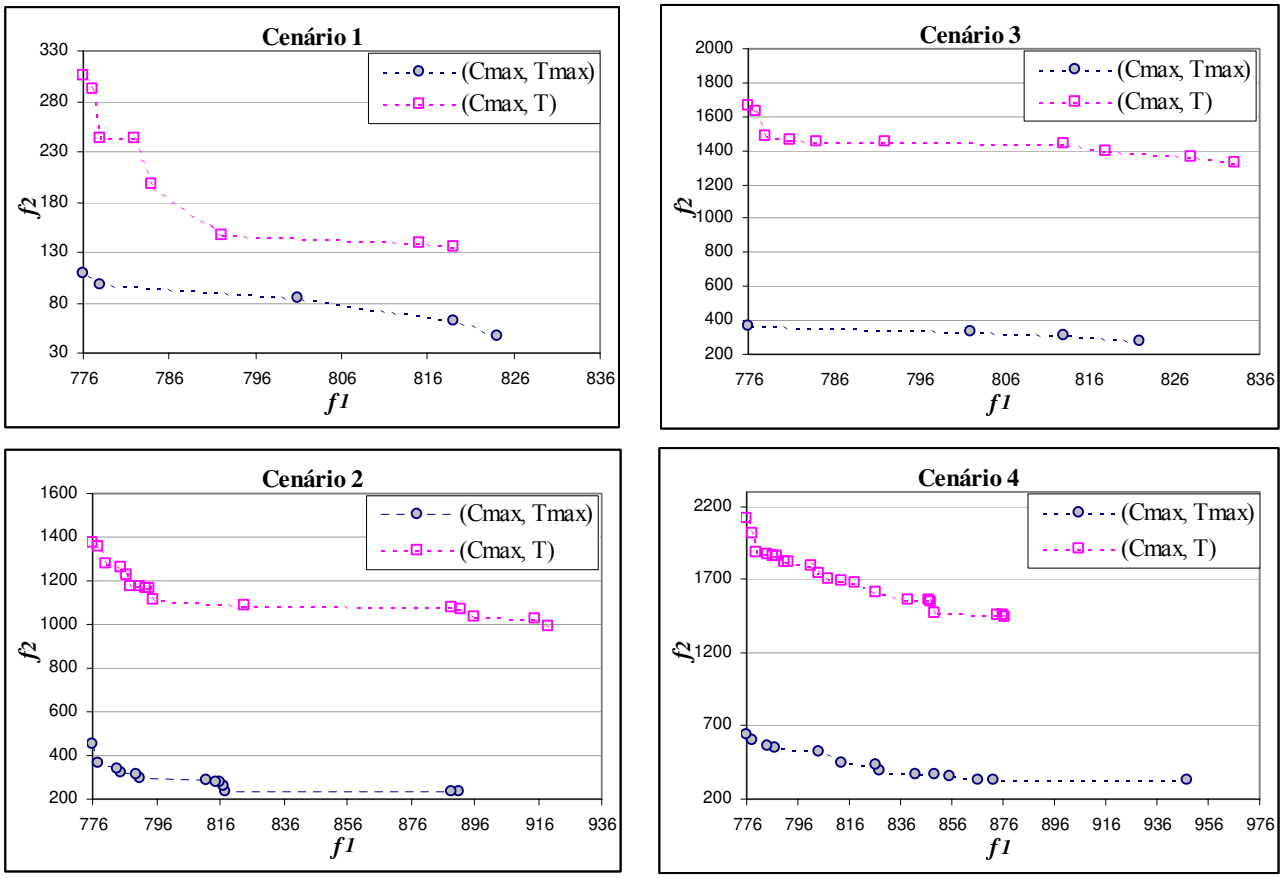


Figura 2.3. Pontos eficientes por cenário de data de entrega.

2.1.2.2. Resultados para $f = (C_{max}, T_{max})$

Resultados para o Conjunto 1 de Instâncias

Com o propósito de analisar as características dos problemas, aplicou-se a técnica de enumeração completa às instâncias deste conjunto para obter as soluções eficientes. Na Tabela 2.2 apresenta-se, para cada tamanho de problema, o número de soluções eficientes e o número de soluções obtidas pelas heurísticas HMEP e HDC associadas com 80 instâncias. Para as 320 instâncias testadas, 3004 soluções eficientes foram obtidas pela enumeração completa. As heurísticas HMEP e HDC identificaram 771 (25,67%) e 146 (4,86%) soluções eficientes, respectivamente. Além disso, a heurística HMEP encontrou o conjunto Pareto-ótimo para somente 5 instâncias, enquanto a heurística HDC não o encontrou para nenhuma instância.

Tabela 2.2. Número de soluções encontradas por Enumeração Completa e pelas heurísticas

Problema $n \times m$	Enumeração Completa	HMEP		HDC	
	NSE	NSH	NSEH	NSH	NSEH
10×5	582	435	197	342	41
10×10	786	495	195	378	38
11×5	687	464	192	359	44
11×10	949	614	187	447	23
Total	3004	2008	771	1526	146

NSE: número de soluções eficientes ou Pareto-ótimas.
NSH: número de soluções encontradas por um método heurístico.
NSEH: número de soluções eficientes encontradas por um método heurístico.

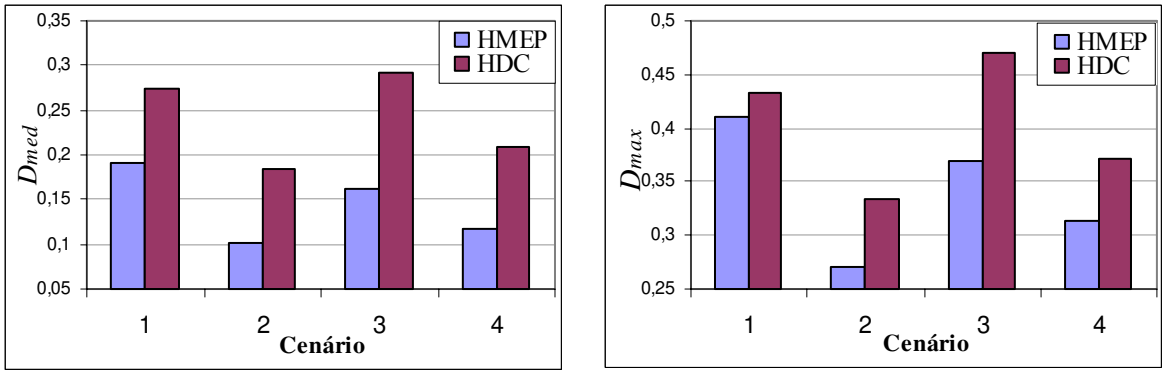
Tabela 2.3. Desempenho das heurísticas

Problema $n \times m$	Medida de distância				Erro de utilidade (%)			
	HMEP		HDC		HMEP		HDC	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×5	0,131	0,328	0,248	0,401	6,857	16,274	17,834	31,452
10×10	0,151	0,353	0,244	0420	8,941	22,833	17,895	33,093
11×5	0,149	0,333	0,237	0,404	8,769	22,425	15,667	30,084
11×10	0,138	0,348	0,230	0,383	7,773	18,902	16,802	29,360
Média total	0,142	0,340	0,240	0,402	8,085	20,109	17,050	30,997

A Tabela 2.3 mostra o desempenho das heurísticas de acordo com a medida de distância e o erro associado com a função utilidade linear proposta por Daniels (1992). Note que a heurística proposta HMEP apresenta melhor desempenho em ambas medidas.

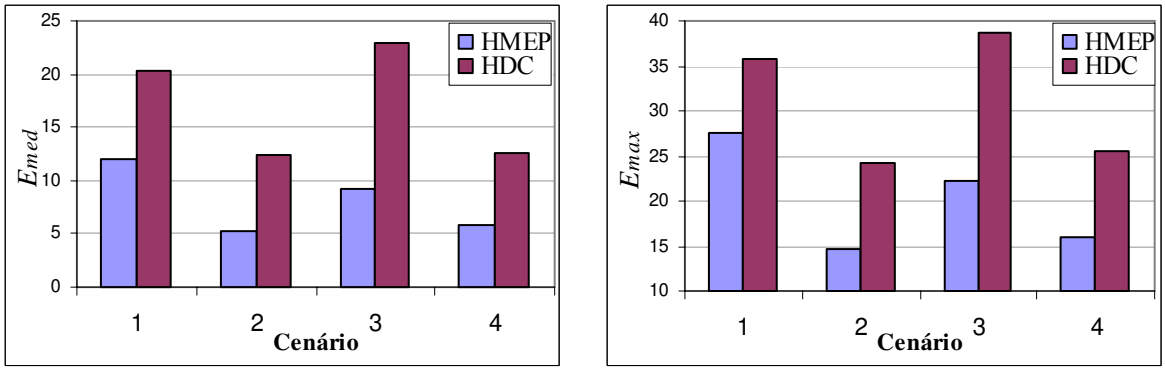
Para cada cenário de datas de entrega, as Figuras 2.4 (a) e (b) ilustram os valores médios (sobre 80 instâncias) de D_{med} e D_{max}, enquanto as Figuras 2.5 (a) e (b) ilustram os valores médios de E_{med} e E_{max}. Observe que a heurística HMEP possui melhor desempenho em todos os cenários e, ambas heurísticas apresentam valores mais altos de distâncias e erros de utilidade para os cenários 1 e 3 que estão associados com pequenas faixas de datas de entrega.

A Figura 2.6 mostra que, a média de número de soluções eficientes encontradas pela enumeração completa (EC) é maior para os cenários 2 e 4, e a heurística HMEP identifica maior quantidade de soluções eficientes em todos os cenários, quando comparado a HDC.



(a) (b)

Figura 2.4. Distância média e máxima por cenário.



(a) (b)

Figura 2.5. Erro de utilidade médio e máximo por cenário.

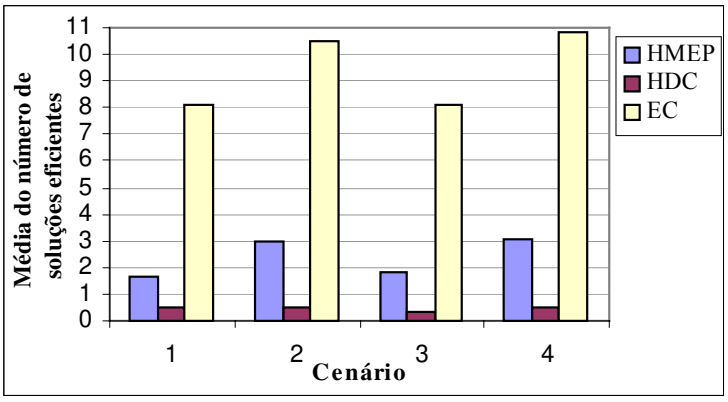


Figura 2.6. Média do número de soluções.

As médias dos tempos computacionais gastos pela enumeração completa e pelas heurísticas são apresentados na Tabela 2.4. Note que o tempo da enumeração completa cresce exponencialmente com o número de tarefas.

Tabela 2.4. Média de tempos computacionais (em segundos)

n×*m*:	10×5	10×10	11×5	11×10
Enumeração Completa	60,420	71,661	877,19	904,001
HMEP	0,0018	0,0026	0,0068	0,0079
HDC	0,00015	0,0005	0,0006	0,0009

Resultados para o Conjunto 2 de Instâncias

As instâncias deste conjunto foram resolvidas otimamente pelo algoritmo Branch and Bound (B&B) desenvolvido por Daniels e Chamber (1990) (veja Apêndice B). O número máximo de nós explorados pelo algoritmo B&B foi limitado a 10^8, gastando aproximadamente 6800 segundos.

Para 160 instâncias testadas, 507 soluções eficientes foram obtidas pelo algoritmo B&B. A Tabela 2.5 exibe, para cada tamanho de problema, o número de soluções obtidas pelas heurísticas. As heurísticas HMEP e HDC encontraram 353 (69,63%) e 357 (70,41%) soluções eficientes e identificaram exatamente o conjunto Pareto-ótimo para 86 e 97 instâncias, respectivamente.

Tabela 2.5. Número de soluções encontradas pelo algoritmo B&B e pelas heurísticas

Problema	B&B	HMEP		HDC	
n×*m*	NSE	NSH	NSEH	NSH	NSEH
10×2	122	125	100	109	97
15×2	140	142	98	121	87
20×2	144	147	81	127	94
25×2	101	116	74	91	79
Total	507	530	353	448	357

A Tabela 2.6 mostra a média da distância e do erro de utilidade das soluções heurísticas em relação ao conjunto Pareto-ótimo obtido pelo método B&B. A heurística HDC apresenta desempenho ligeiramente superior, em relação à medida de distância, para instâncias com 20 e 25 tarefas. Considerando o erro de utilidade, esta heurística é muito melhor para instâncias com 15, 20 e 25 tarefas. Esta inconsistência das medidas pode ser atribuída à eliminação de pontos que não possuem hiperplano suporte quando é calculado o erro de utilidade. As Figuras 2.6 (a) e (b) ilustram esta inconsistência através de um exemplo com um conjunto de referência contendo 9 pontos (A, B,...,I) dos quais as heurísticas HMEP e HDC identificaram 7 pontos (B, C,...,H) e 4 pontos (A, B, C, F), respectivamente. O algoritmo da avaliação analítica proposto por Daniels (1992) descarta os pontos de referência C, D, E, G e H. Na Figura 2.7 (b) são mostrados os pontos que não foram descartados (pontos que possuem hiperplano suporte). Após a eliminação de pontos, os erros de utilidade das heurísticas são calculados considerando 4 pontos para cada heurística. As heurísticas HMEP e HDC geram 2 (B e F) e 3 (A, B e F) pontos referência, respectivamente. Note que o valor do erro de utilidade da heurística HMEP é calculado a partir de dois pontos de suporte que não coincidem com os pontos de referência, enquanto o valor do erro de utilidade da heurística HDC é calculado a partir de um único ponto de suporte. A heurística HMEP apresenta maior erro de utilidade devido a que os pontos de suporte (que não coincidem com os de referência) estão mais distantes dos pontos de referência A e I.

Tabela 2.6. Desempenho das heurísticas

Problema n×m	Medida de distância				Erro de utilidade (%)			
	HMEP		HDC		HMEP		HDC	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×2	0,036	0,094	0,041	0,125	0,702	2,502	0,964	3,547
15×2	0,082	0,131	0,087	0,150	1,264	3,140	0,907	2,721
20×2	0,104	0,194	0,081	0,182	4,395	8,543	3,176	7,696
25×2	0,055	0,127	0,040	0,095	3,944	7,070	2,055	5,567
Média total	0,069	0,136	0,062	0,138	2,576	5,314	1,776	4,883

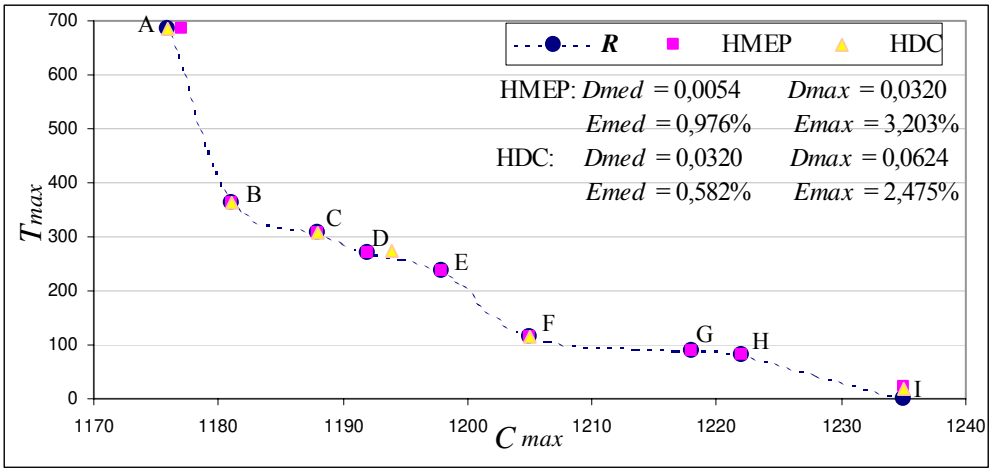


Figura 2.7.(a). Inconsistência das medidas.

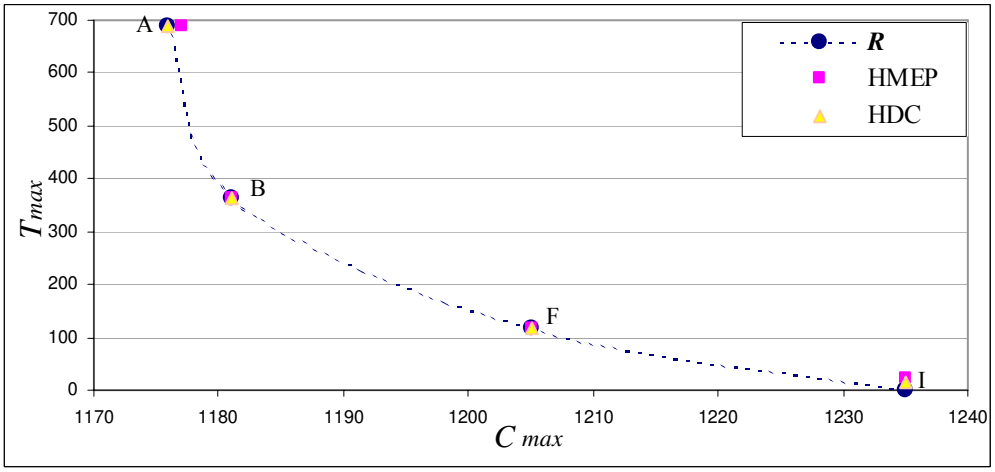
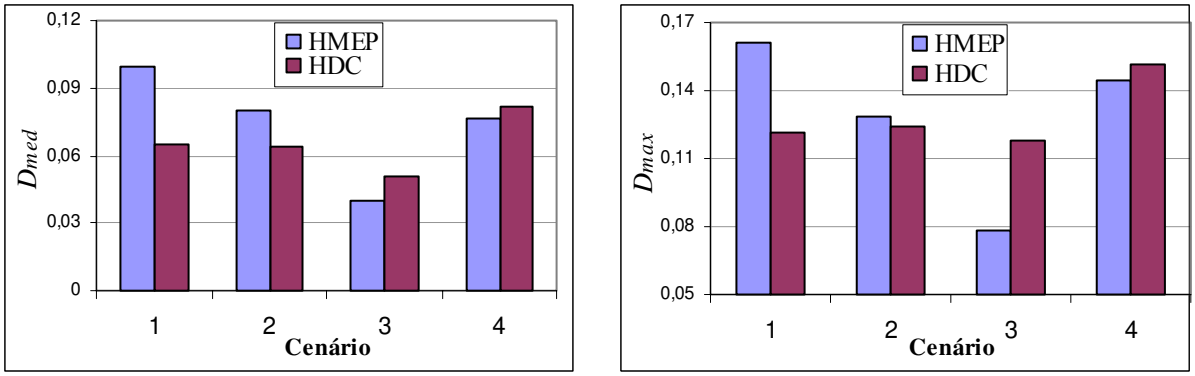


Figura 2.7.(b) Inconsistência das medidas - pontos que possuem hiperplano suporte.

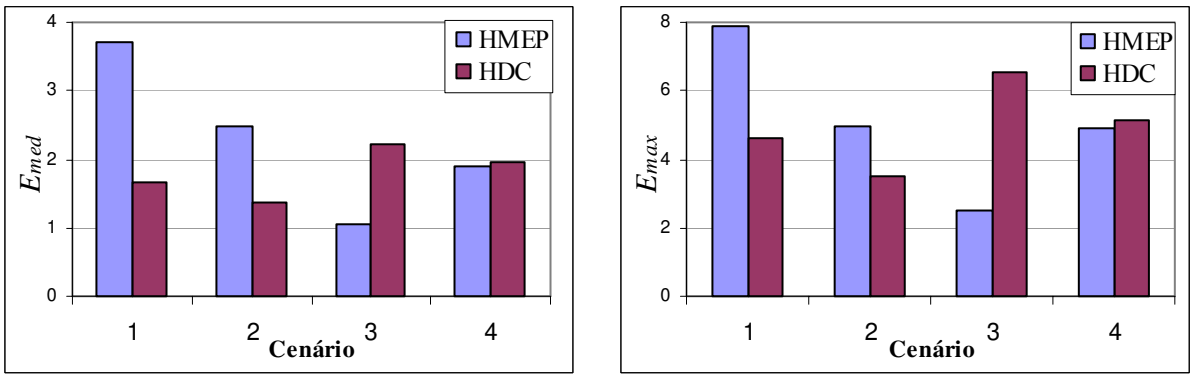
Para cada cenário de datas de entrega, as Figuras 2.8 (a) e (b) ilustram os valores médios (sobre 40 instâncias) de D_{med} e D_{max}, enquanto as Figuras 2.9 (a) e (b) ilustram os valores médios de E_{med} e E_{max}. Para ambas medidas, a heurística HDC apresenta um melhor desempenho para os cenários 1 e 2, enquanto a heurística HMEP é superior para o cenário 3.

A Figura 2.10 ilustra que as instâncias com cenários 2 e 4 de datas de entrega possuem maior número de soluções eficientes e que as heurísticas apresentam um comportamento relativamente similar com relação às medidas de distância e erro de utilidade, como mencionado acima.



(a) (b)

Figura 2.8. Distância média e máxima por cenário.



(a) (b)

Figura 2.9. Erro de utilidade médio e máximo por cenário.

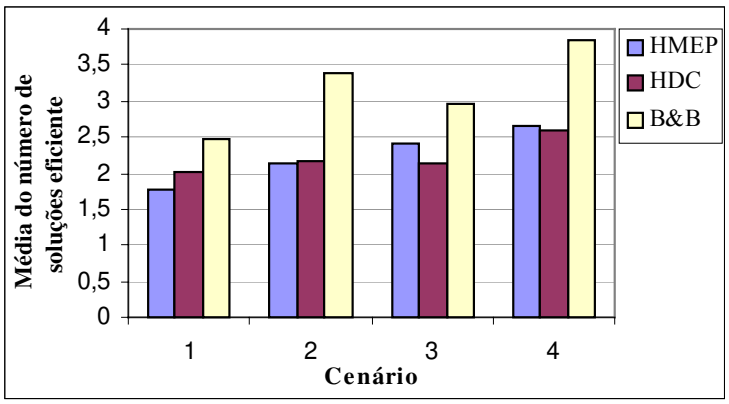


Figura 2.10. Média do número de soluções.

Dos resultados apresentados acima, pode-se concluir que a heurística HDC proposta por Daniels e Chambers (1990) apresenta bom comportamento para instâncias do problema de flowshop com duas máquinas. Isto se deve ao fato de que, a heurística HDC inicia com uma seqüência (solução) obtida através do algoritmo de Johnson (1954) que é um método ótimo para minimizar C_{max} para o caso de duas máquinas (veja apêndice C).

A média dos tempos computacionais gastos pelo algoritmo B&B e pelas heurísticas são apresentados na Tabela 2.7. Note que o algoritmo B&B requer maior esforço computacional para os cenários 2 e 4 que estão associados com largas faixas de datas de entrega.

Tabela 2.7. Tempo computacional (em segundos)

Problema	Branch and Bound				HMEP	HDC
$n \times m$	Cenário 1	Cenário 2	Cenário 3	Cenário 4		
10×2	0,014	0,026	0,001	0,012	0,00023	0,00018
15×2	0,124	0,229	0,183	0,474	0,0037	0,0037
20×2	0,436	51,957	0,845	499,171	0,0120	0,0058
25×2	0,268	720,605	0,414	3695,813	0,0220	0,020

Resultados para o Conjunto 3 de Instâncias

O desempenho das heurísticas HMEP e HDC foi avaliado em instâncias de grande porte pertencentes a este conjunto. A Tabela 2.8 mostra o número de soluções no conjunto de referência ***R***, o número de soluções heurísticas e o número de soluções dominantes geradas por cada heurística. Claramente, a heurística HMEP gera um número grande de soluções dominantes para todos os tamanhos de problemas. De um total de 360 instâncias testadas, 7082 soluções de referência foram geradas, das quais 6774 soluções foram obtidas pela heurística HMEP e 325 soluções pela heurística HDC. A Tabela 2.9 mostra que a heurística proposta possui melhor desempenho segundo as duas medidas consideradas. As médias dos tempos computacionais gastos pelas heurísticas são mostrados na Tabela 2.10.

Tabela 2.8. Número de soluções heurísticas dominantes

Problema $n \times m$	$\|\boldsymbol{R}\|$	HMEP		HDC	
		NSH	NSDH	NSH	NSDH
20×5	374	368	340	282	41
20×10	491	473	455	382	38
20×20	553	544	525	413	28
50×5	566	545	516	538	51
50×10	864	848	838	780	26
50×20	1042	1036	1028	804	14
80×5	565	576	534	666	36
80×10	1059	1014	1007	1059	53
80×20	1568	1537	1531	1309	38
Total	7082	6941	6774	6233	325

$|\boldsymbol{R}|$: número de soluções de referência
NSH: número de soluções obtidas por uma heurística.
NSDH: número de soluções dominantes obtidas por uma heurística

Tabela 2.9. Desempenho das heurísticas

Problema $n \times m$	Medida de distância				Erro de utilidade (%)			
	HMEP		HDC		HMEP		HDC	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
20×5	0,028	0,126	0,206	0,354	1,231	5,137	16,225	29,990
20×10	0,017	0,162	0,188	0,309	0,401	2,662	15,790	27,234
20×20	0,008	0,079	0,195	0,314	0,309	2,025	16,013	28,194
50×5	0,032	0,194	0,190	0,311	1,094	5,016	14,409	27,192
50×10	0,008	0,089	0,192	0,299	0,215	1,833	16,352	27,704
50×20	0,003	0,061	0,222	0,324	0,066	0,657	18,751	30,018
80×5	0,008	0,064	0,198	0,328	0,118	1,042	15,191	27,954
80×10	0,0201	0,179	0,181	0,287	0,346	2,809	14,681	26,561
80×20	0,004	0,086	0,198	0,288	0,206	1,758	16,055	27,705
Média total	0,014	0,116	0,197	0,313	0,844	2,549	15,941	28,061

Tabela 2.10. Tempo computacional (em segundos)

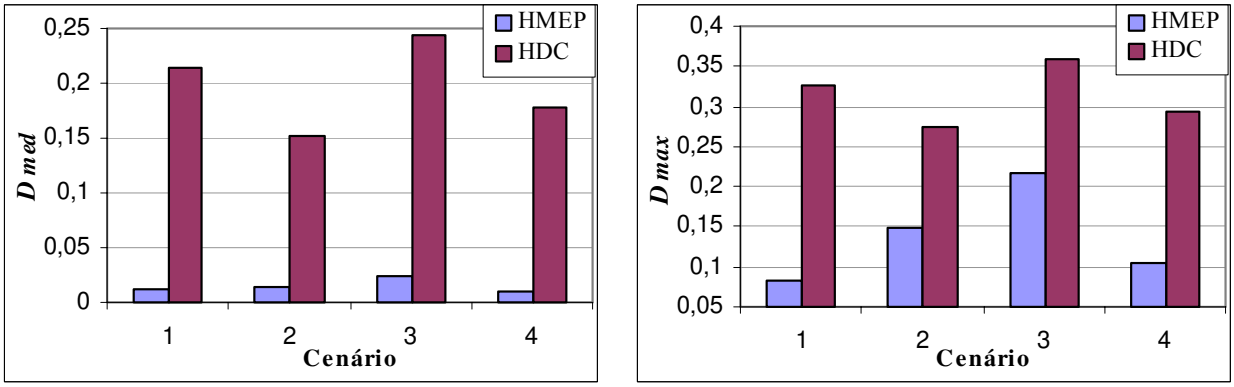
n:	20			50			80		
m:	5	10	20	5	10	20	5	10	20
HMEP	0,034	0,066	0,098	0,541	1,163	2,620	3,782	5,310	10,352
HDC	0,011	0,014	0,032	0,052	0,057	0,854	2,644	4,430	5,780

Na Tabela 2.11 mostra-se, para cada valor de n (número de tarefas), o número máximo de seqüências geradas pela heurística HMEP, e a correspondente complexidade computacional de ordem quadrática.

Tabela 2.11. Complexidade computacional

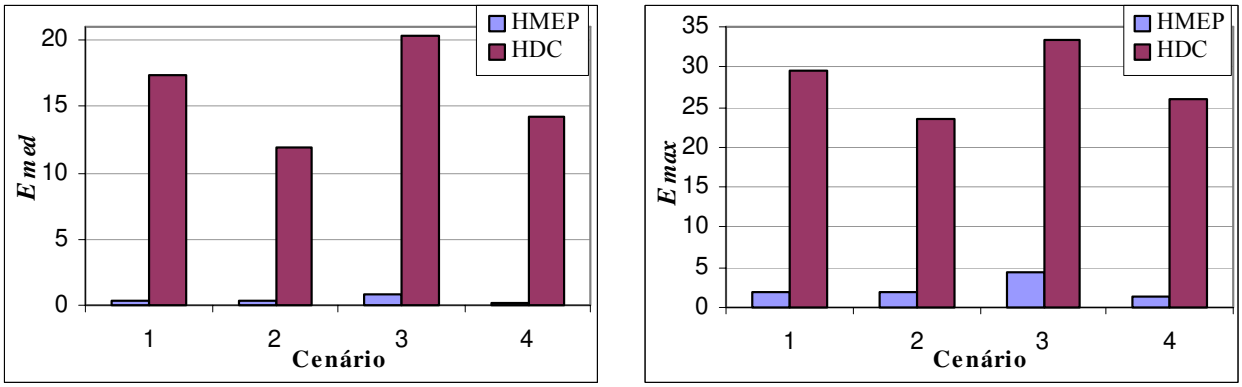
$n = 20$	$n = 50$	$n = 80$
4040	40421	118177
$10{,}1n^2$	$16{,}2n^2$	$18{,}5n^2$

Para cada cenário, as Figuras 2.11 (a) e (b) mostram os valores médios (sobre 40 instâncias) das distâncias D_{med} e D_{max}, e as Figuras 2.12 (a) e (b) mostram os valores médios de E_{med} e E_{max}. A heurística HMEP é mais eficaz para todos os cenários apresentando maiores valores da distância e erro de utilidade para o cenário 3.



(a) (b)

Figura 2.11. Distância média e máxima por cenário



(a) (b)

Figura 2.12. Erro de utilidade médio e máximo por cenário

Para cada tamanho de problema e cada cenário, na Figura 2.13 ilustra-se a média do número de soluções dominantes obtidas pela heurística HMEP. Observe que, um número alto de soluções dominantes correspondem a instâncias com cenários 2 e 4 de datas de entrega. O número de soluções dominantes cresce à medida a razão *n*/*m* decresce.

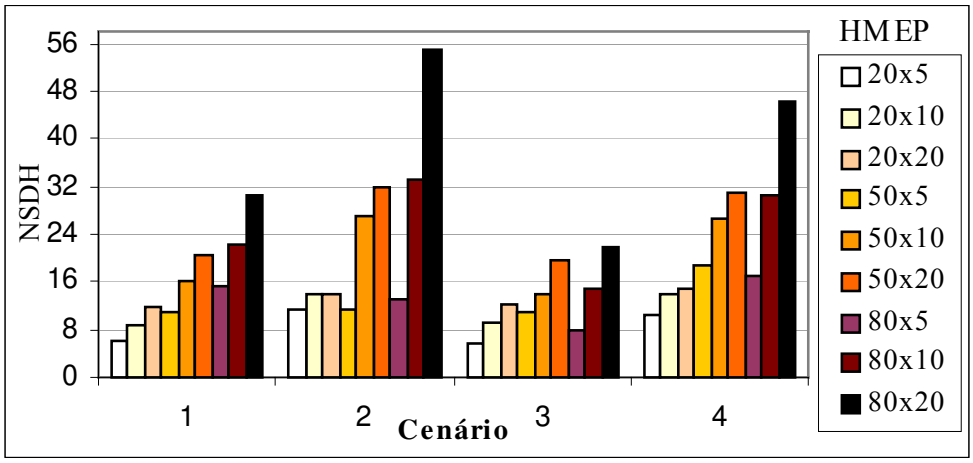


Figura 2.13. Número de soluções dominantes por tamanho de instância e por cenário.

2.1.2.3. Resultados para $f = (C_{max}, T)$

Para este par de objetivos, não temos conhecimento da existência de alguma heurística construtiva. Por este motivo, a heurística HMEP é comparada com uma variante, denotada por HMEP-1, na qual substitui-se a regra TLB pela regra MDD (modified due date) proposta por Baker e Kanet (1983). A regra MDD avalia, para cada instante de tempo *t* e para cada tarefa *i*, a medida $\max(d_i, C_i(\sigma))$, onde σ é a seqüência de tarefas já estabelecidas no instante $t = \sum_{i \in \sigma} p_{i1}$, $C_i(\sigma)$ é o instante de finalização do processamento da tarefa i, $i \notin \sigma$, se esta tarefa for seqüênciada logo após da seqüência σ. A tarefa com a menor medida é selecionada para ser processada no tempo *t*.

Apresenta-se a seguir os resultados computacionais das heurísticas comparadas HMEP e HMEP-1 para (C_{max} , T) para os três conjuntos de instancias.

Resultados para o Conjunto 1 de Instâncias

Na Tabela 2.12 mostra-se o número de soluções eficientes e o número de soluções obtidas pelas heurísticas HMEP e HMEP-1 associadas com 80 instâncias de cada tamanho. Para as 320 instâncias testadas, 3442 soluções eficientes foram obtidas pela enumeração completa. As heurísticas HMEP e HMPE-1 identificaram 755 (21,93%) e 485 (14,09%) soluções eficientes, respectivamente. A heurística HMEP encontrou o conjunto Pareto-ótimo em 7 instâncias, e a heurística HMEP-1 em 3 instâncias.

A Tabela 2.13 mostra que, em termos do erro de utilidade, o desempenho da heurística HMEP é superior à heurística HMEP-1, mas ambas heurísticas apresentam um desempenho similar em relação à medida de distância.

Tabela 2.12. Número de soluções encontradas pela enumeração completa e pelas heurísticas

Problema $n\times m$	Enumeração Completa	HMEP		HMEP-1	
	NSE	NSH	NSEH	NSH	NSEH
10×5	755	502	183	493	154
10×10	820	500	203	474	123
11×5	825	555	175	512	112
11×10	1042	595	194	592	96
Total	3442	2152	755	2071	485

Tabela 2.13. Desempenho das heurísticas

Problema $n\times m$	Medida de distância				Erro de utilidade (%)			
	HMEP		HMEP-1		HMEP		HMEP-1	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×5	0,131	0,269	0,140	0,270	6,541	16,690	9,205	18,303
10×10	0,158	0,339	0,157	0,341	9,798	19,801	12,127	20,421
11×5	0,140	0,293	0,147	0,301	8,232	17,138	10,835	21,621
11×10	0,157	0,319	0,172	0,290	7,526	15,545	11,630	17,819
Média total	0,147	0,305	0,154	0,301	8,024	17,294	10,949	19,541

Na Figura 2.14 mostra-se um exemplo no qual a enumeração completa gerou 13 pontos eficientes (A, B,...,M). As heurísticas HMEP e HMEP-1 geraram 8 e 6 pontos, respectivamente, das quais HMEP encontrou 4 pontos eficientes (G, H, I, M) e HMEP-1 encontrou somente uma solução eficiente (H). Note que os pontos dominados obtidas por ambas heurísticas estão relativamente próximos e possuem uma qualidade similar em termos de distância e erro de utilidade.

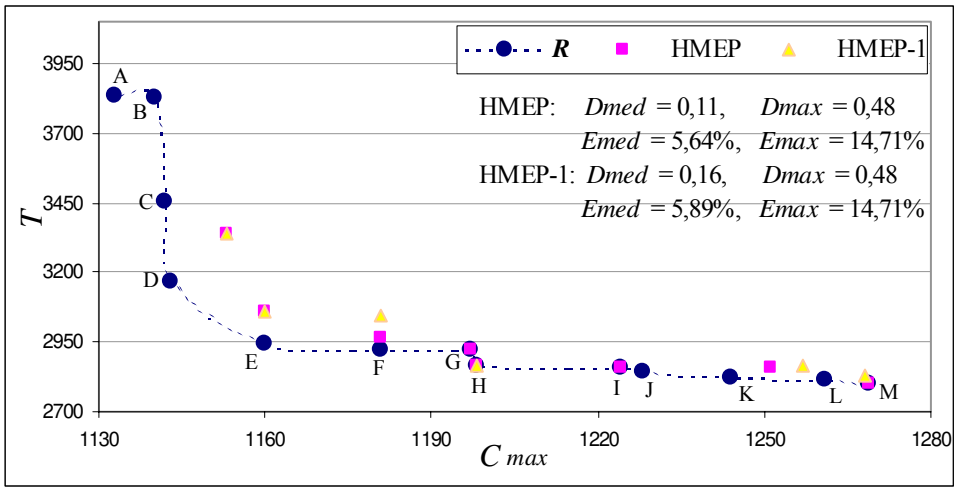


Figura 2.14. Pontos gerados por HMEP e HMEP-1.

Na Tabela 2.14 são mostradas as médias dos tempos computacionais gastos pelas pela enumeração completa e pelas heurísticas HMEP e HMEP-1.

Tabela 2.14. Média de tempos computacionais (em segundos)

n×*m*:	10×5	10×10	11×5	11×10
Enumeração Completa	60,087	71,874	879,534	907,625
HMEP	0,00725	0,00337	0,00520	0,00862
HMEP-1	0,00662	0,0045	0,00735	0,0104

Resultados para o Conjunto 2 de Instâncias

As instâncias deste conjunto foram resolvidas otimamente pelo algoritmo B&B desenvolvido por Liao et al. (1997) (veja Apêndice B). O número máximo de nós explorados

por B&B foi limitado a 10^8, requerendo aproximadamente 6800 segundos.

Na Tabela 2.15 mostra-se o número de soluções eficientes e o número de soluções obtidas pelas heurísticas. Para 160 instâncias testadas, 474 soluções eficientes foram obtidas pelo algoritmo B&B. As heurísticas HMEP e HMEP-1 identificaram 214 (45,15%) e 153 (32,28%) soluções eficientes e encontraram exatamente o conjunto Pareto-ótimo para 45 e 28 instâncias, respectivamente. A Tabela 2.16 mostra os desempenhos das heurísticas, e observa-se que HMEP supera HMEP-1.

Tabela 2.15. Número de soluções obtidas pelo algoritmo B&B pela heurística HMPE

Problema	B&B	HMEP		HMEP-1	
$n\times m$	NES	NSH	NESH	NSH	NESH
10×2	158	144	88	150	79
15×2	172	159	73	159	48
20×2	144	164	53	179	26
Total	474	467	214	488	153

Tabela 2.16. Desempenho das heurísticas

Problema $n\times m$	Medida de distância				Erro de utilidade (%)			
	HMEP		HMEP-1		HMEP		HMEP-1	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×2	0,115	0,242	0,166	0,276	2,036	6,579	3,563	7,954
15×2	0,090	0,174	0,163	0,272	4,814	11,103	9,183	17,695
20×2	0,219	0,317	0,311	0,418	8,602	16,603	25,817	38,922
Média total	0,141	0,245	0,213	0,322	5,151	11,428	12,854	21,524

A média dos tempos computacionais requeridos pelo algoritmo B&B e as heurísticas são mostradas na Tabela 2.17. Como no caso de (C_{max}, T_{max}), o algoritmo B&B de Liao et al. (1997) requer também maior tempo computacional para os cenários 2 e 4.

Tabela 2.17. Tempo computacional (em segundos)

Problema	Branch and Bound				HMEP	HMEP-1
$n\times m$	Cenário 1	Cenário 2	Cenário 3	Cenário 4		
10×2	0,021	0,123	0,128	0,272	0,00035	0,00043
15×2	0,137	27,33	4,298	116,175	0,0042	0,0039
20×2	0,746	2331,38	87,317	6011,88	0,0204	0,0224

Resultados para o Conjunto 3 de Instâncias

O comportamento das heurísticas HMEP e HMEP-1 foi avaliado em instâncias de grande porte pertencentes a este conjunto. Na Tabela 2.18 mostra-se o número de soluções no conjunto de referência $\boldsymbol{R}$, o número de soluções heurísticas e o número de soluções dominantes obtidas por cada heurística. A heurística HMEP gera maior quantidade de soluções dominantes para todos os tamanhos de problemas. De um total de 360 instâncias testadas, 9645 soluções de referência foram geradas, das quais 7064 soluções foram obtidas por HMEP e 3971 soluções por HMEP-1. 1570 soluções foram encontradas por ambas heurísticas. Considerando as medidas de distância e erro de utilidade, a Tabela 2.19 mostra que a heurística HMEP possui melhor desempenho que HMEP-1. Na Tabela 2.20 apresentam-se as médias dos tempos computacionais, em segundos, requeridas por ambas heurísticas.

Na Tabela 2.21 mostra-se, para cada valore de n (número de tarefas), o número máximo de seqüências geradas pelas heurísticas HMEP e HMEP-1, e a correspondente complexidade computacional de ordem quadrática.

Tabela 2.18. Soluções heurísticas dominantes

Problema $n\times m$	$\lvert\boldsymbol{R}\rvert$	HMEP		HMEP-1	
		NSH	NSDH	NSH	NSDH
20×5	467	424	325	447	265
20×10	521	493	402	501	244
20×20	537	502	430	480	315
50×5	698	728	507	723	320
50×10	1203	1204	831	1104	551
50×20	1323	1317	1141	1237	429
80×5	845	834	601	879	393
80×10	1674	1597	1242	1595	630
80×20	2197	2139	1585	2155	824
Total	9465	9238	7064	9121	3971

Tabela 2.19. Desempenho das heurísticas

Problema $n \times m$	Medida de distância				Erro de utilidade (%)			
	HMEP		HMEP-1		HMEP		HMEP-1	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
20×5	0,034	0,151	0,071	0,199	1,070	4,400	5,008	11,545
20×10	0,033	0,153	0,075	0,226	2,368	6,585	6,381	14,129
20×20	0,061	0,259	0,069	0,219	3,209	11,405	4,053	11,687
50×5	0,018	0,097	0,065	0,193	1,311	3,926	5,555	11,283
50×10	0,022	0,113	0,055	0,167	1,702	5,199	5,026	11,772
50×20	0,006	0,037	0,092	0,244	0,584	2,135	8,427	18,093
80×5	0,021	0,090	0,050	0,156	1,628	4,503	4,135	8,905
80×10	0,011	0,092	0,048	0,150	0,633	2,505	4,251	9,032
80×20	0,011	0,081	0,053	0,163	0,996	3,432	4,766	11,140
Média total	0,024	0,119	0,064	0,191	1,500	4,900	5,289	11,954

Tabela 2.20. Tempo computacional (em segundos)

n:	20			50			80		
m:	5	10	20	5	10	20	5	10	20
HMEP	0,048	0,064	0,094	0,635	1,457	3,081	4,037	7,182	15,171
HMEP-1	0,046	0,067	0,096	0,656	1,569	3,032	4,193	7,019	17,577

Tabela 2.21. Complexidade computacional

Heurística	$n = 20$	$n = 50$	$n = 80$
HMEP	3930 $9{,}8n^2$	50820 $20{,}32n^2$	170949 $26{,}7n^2$
HMEP-1	4087 $10{,}22n^2$	50393 $20{,}16n^2$	213895 $33{,}42n^2$

2.2. Busca Local Multiobjetivo

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 $\boldsymbol{x}$, e gera um conjunto $\boldsymbol{N}(\boldsymbol{x})$ de soluções vizinhas de $\boldsymbol{x}$. Cada solução em $\boldsymbol{N}(\boldsymbol{x})$ é gerada através de uma operação, denominada movimento, sobre $\boldsymbol{x}$. Um método de busca local, conhecido como método de descida, busca dentre as soluções de $\boldsymbol{N}(\boldsymbol{x})$, uma solução $\boldsymbol{x}'$ com valor de função objetivo menor que $\boldsymbol{x}$, no caso de minimização. A partir de $\boldsymbol{x}'$ o processo é repetido até a obtenção de uma solução sem nenhum vizinho melhor, denominada ótimo local em relação à vizinhança. Um algoritmo genérico do método de descida é apresentado a seguir.

Algoritmo 2.2. Busca Local mono-objetivo

1) Gere uma solução inicial $\boldsymbol{x}$ factível para iniciar o processo.
2) Gere a vizinhança de $\boldsymbol{x}$, $\boldsymbol{N}(\boldsymbol{x})$.
3) Escolha uma solução vizinha $\boldsymbol{x}' \in \boldsymbol{N}(\boldsymbol{x})$ tal que $f(\boldsymbol{x}') < f(\boldsymbol{x})$.
4) Se não é possível encontrar $\boldsymbol{x}'$ então pare, $\boldsymbol{x}$ é um ótimo local.
5) Senão faça $\boldsymbol{x} = \boldsymbol{x}'$ e volte ao passo 2.

Uma 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.

Nesta seção é proposto um método de busca local multiobjetivo (BLM) com o objetivo de melhorar um conjunto de soluções dominantes gerada por uma heurística multiobjetivo. Este método é baseado no conceito de dominância de Pareto e ele realiza uma busca em paralelo, explorando várias regiões do espaço de soluções fatíveis. O método BLM parte de um conjunto inicial $\boldsymbol{S}$ de soluções fatíveis, e a partir de cada solução $\boldsymbol{x}$ deste conjunto gera-se

uma vizinhança $\boldsymbol{N}(\boldsymbol{x})$. Constrói-se então o próximo conjunto $\boldsymbol{S}'$ das soluções vizinhas não dominadas por $\boldsymbol{S}$ onde $\boldsymbol{S}'$ é um subconjunto de $\bigcup_{\boldsymbol{x}\in\boldsymbol{S}} \boldsymbol{N}(\boldsymbol{x})$. Caso $\boldsymbol{S}' \neq \varnothing$, o processo se repete a partir deste novo conjunto até que se verifique algum critério de parada. Quando os conjuntos $\boldsymbol{S}$ e $\boldsymbol{S}'$ contém um número grande de soluções, é necessário reduzir estes conjuntos considerando somente N_{max} de soluções. No algoritmo 2.3 apresentado a seguir, descreve-se os passos da busca local multiobjetivo BLM.

Algoritmo 2.3. Busca Local Multiobjetivo (BLM)

Entrada: $\boldsymbol{S}$ (um conjunto de soluções dominantes).
N_{max} (número máximo de caminhos a explorar).
Niter (número máximo de iterações)
Saída: $\boldsymbol{D}_{BL}$ (conjunto de soluções dominantes melhoradas).

0) *Inicialização*:

Faça $t = 1$ (contador de iterações) e $\boldsymbol{D}_{BL} = \boldsymbol{S}$.

1) *Redução do conjunto* $\boldsymbol{S}$:

Se $|\boldsymbol{S}| > N_{max}$ então

Reduza o conjunto $\boldsymbol{S}$ selecionando somente N_{max} soluções.

Seja $\boldsymbol{LC}$ o conjunto das soluções não selecionadas.

2) *Construção de um conjunto de soluções vizinhas* $\boldsymbol{S}'$:

Faça $\boldsymbol{S}' = \varnothing$ (conjunto das soluções vizinhas dominantes).

Para cada $\boldsymbol{x}_k \in \boldsymbol{S}$ faça

Gere a vizinhança de $\boldsymbol{x}_k$: $\boldsymbol{N}(\boldsymbol{x}_k)$.

Para cada $\boldsymbol{y} \in \boldsymbol{N}(\boldsymbol{x}_k)$ faça

Se $\boldsymbol{y} \notin \boldsymbol{D}_{BL}$ ou $\boldsymbol{y}$ não é dominado por $\boldsymbol{D}_{BL}$ então

Faça $\boldsymbol{S}'$ = soluções dominantes de $(\boldsymbol{S}' \cup \{\boldsymbol{y}\})$.

3) *Atualização do conjunto* $\boldsymbol{D}_{BL}$ *com as novas soluções vizinhas em* $\boldsymbol{S}'$

$\boldsymbol{D}_{BL}$ = soluções dominantes de $(\boldsymbol{D}_{BL} \cup \boldsymbol{S}')$.

4) Faça $\boldsymbol{S}'$ = soluções dominantes de $(\boldsymbol{S}' \cup \boldsymbol{LC})$.

Se $(\boldsymbol{S}' \neq \varnothing$ e $t \leq Niter)$ então

Faça $\boldsymbol{S} = \boldsymbol{S}'$, $t = t + 1$ e volte ao passo 1.

Senão pare.

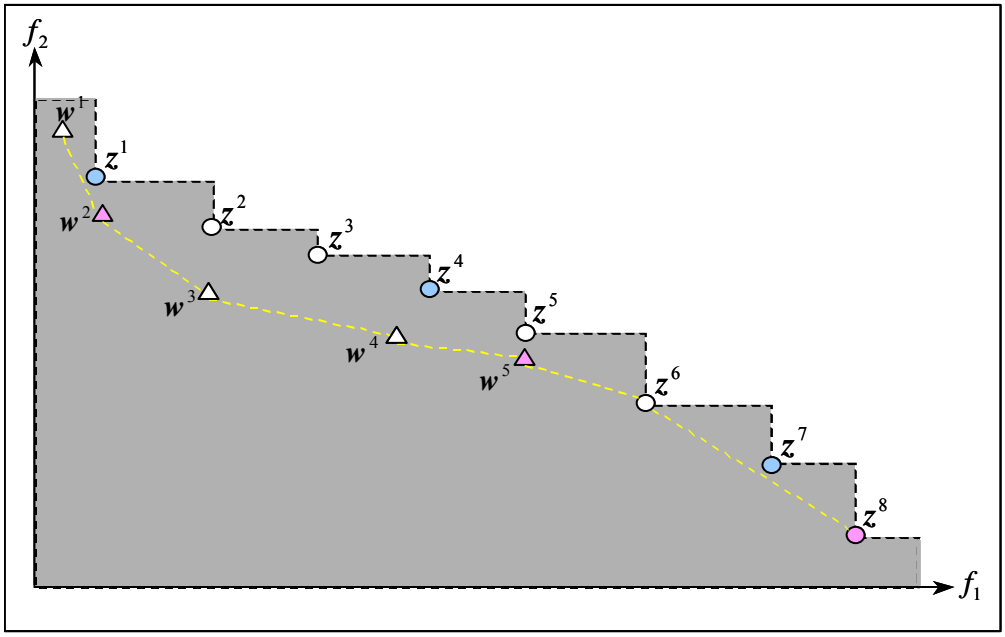


Figura 2.15. Busca Local Multiobjetivo.

Exemplo:

Considere $\boldsymbol{S} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^8\}$ o conjunto de partida da busca local e $N_{max} = 3$ o número máximo de caminhos a serem explorados. Na Figura 2.15 ilustra-se a fronteira $\boldsymbol{f}(\boldsymbol{S})$ formada pelos pontos $\boldsymbol{z}^i = \boldsymbol{f}(\boldsymbol{x}^i)$, $i = 1, \ldots, 8$. Inicialmente, o conjunto de soluções dominantes $\boldsymbol{D}_{BL} = \boldsymbol{S}$. Segundo o passo 1) do Algoritmo 2.3, o conjunto $\boldsymbol{S}$ deve ser reduzido a 3 soluções. Suponha que $\boldsymbol{x}^1$, $\boldsymbol{x}^4$ e $\boldsymbol{x}^7$ sejam as 3 soluções selecionadas para serem exploradas pela busca local. Então o conjunto inicial da busca local é $\boldsymbol{S} = \{ \boldsymbol{x}^1, \boldsymbol{x}^4, \boldsymbol{x}^7\}$, e as soluções não selecionadas são armazenadas no conjunto $\boldsymbol{LC} = \{ \boldsymbol{x}^2, \boldsymbol{x}^3, \boldsymbol{x}^5, \boldsymbol{x}^6, \boldsymbol{x}^8 \}$. A partir do conjunto $\boldsymbol{S}$ constrói-se o conjunto $\boldsymbol{S}'$ com 5 soluções vizinhas, $\boldsymbol{S}' = \{ \boldsymbol{y}^1, \boldsymbol{y}^2, \boldsymbol{y}^3, \boldsymbol{y}^4, \boldsymbol{y}^5 \}$ (veja na Figura 2.15 os pontos imagens $\boldsymbol{w}^i = \boldsymbol{f}(\boldsymbol{y}^i)$, $i = 1, \ldots, 5$). Note que, $\boldsymbol{S}'$ sempre conterá soluções não dominadas pelos conjuntos $\boldsymbol{S}$ e $\boldsymbol{D}_{BL}$. Atualizando o conjunto $\boldsymbol{D}_{BL}$ tem-se, $\boldsymbol{D}_{BL}$ = soluções dominantes de $(\boldsymbol{D}_{BL} \cup \boldsymbol{S}') = \{ \boldsymbol{y}^1, \boldsymbol{y}^2, \boldsymbol{y}^3, \boldsymbol{y}^4, \boldsymbol{y}^5, \boldsymbol{x}^6, \boldsymbol{x}^7, \boldsymbol{x}^8 \}$ (passo 3 do Algoritmo 2.3). Segundo o passo 4 do Algoritmo BLM, as soluções que não foram selecionadas para busca local, podem ser consideradas na próxima iteração se estas não são dominadas por $\boldsymbol{S}'$, então $\boldsymbol{S}'$ = soluções dominantes de $(\boldsymbol{S}' \cup \boldsymbol{LC}) = \{ \boldsymbol{y}^1, \boldsymbol{y}^2, \boldsymbol{y}^3, \boldsymbol{y}^4, \boldsymbol{y}^5, \boldsymbol{x}^6, \boldsymbol{x}^8\}$. Como o conjunto inicial $\boldsymbol{S}$ foi

melhorado pelo conjunto $\boldsymbol{S}' \neq \varnothing$ ("$\boldsymbol{f}(\boldsymbol{S}') \leq \boldsymbol{f}(\boldsymbol{S})$"), o processo é reiniciado numa nova iteração fazendo $\boldsymbol{S} = \boldsymbol{S}'$. No passo 1 desta nova iteração, os pontos selecionados para serem explorados podem ser $\boldsymbol{y}^2$, $\boldsymbol{y}^5$ e $\boldsymbol{x}^8$ que correspondem aos pontos $\boldsymbol{w}^2$, $\boldsymbol{w}^5$ e $\boldsymbol{z}^8$ (veja Figura 2.15).

Observe que, a busca local BLM termina pela execução de um número de iterações ou quando não são encontradas novas soluções dominantes, isto é, o conjunto $\boldsymbol{D}_{BL}$ não é alterado. Se no passo 2 nenhuma solução é adicionada a $\boldsymbol{S}'$ o algoritmo tentará encontrar novas soluções dominantes a partir das soluções armazenadas em $\boldsymbol{LC}$, finalizando quando $\boldsymbol{LC} = \varnothing$.

A busca local BLM explora vários caminhos simultaneamente, e o conjunto final obtido por este método contém várias soluções dominantes que constituem uma aproximação da fronteira das soluções Pareto-ótimas, cuja qualidade depende da definição da vizinhança e do conjunto inicial de soluções dominantes. Para evitar que o método BLM gaste alto tempo computacional, sugere-se explorar um número conveniente (N_{max}) de caminhos por um dado número (*Niter*) de iterações.

No passo 1 do algoritmo BLM, para reduzir o tamanho do conjunto $\boldsymbol{S}$ podem ser utilizados dos procedimentos de redução. O primeiro procedimento (válido somente para problemas bi-objetivos) divide o conjunto $\boldsymbol{S}$ em N_{max} subconjuntos $\boldsymbol{S}_i$ ordenados de acordo com o primeiro objetivo. A partir de cada subconjunto $\boldsymbol{S}_i$ (i = 1,...,N_{max}) seleciona-se aleatoriamente uma solução. Este procedimento é descrito a seguir:

Algoritmo 2.4. Redução Aleatória

Entrada: $\boldsymbol{S} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^{|S|}\}$ (um conjunto de $|\boldsymbol{S}| > N_{max}$ soluções dominantes tal que $f_1(\boldsymbol{x}^i) < f_1(\boldsymbol{x}^{i+1})$).
N_{max} (número máximo de soluções).

Saída: $\boldsymbol{S}$ (conjunto de N_{max} soluções dominantes).

1) Faça np = N_{max} e $p_1 = 1$.
2) Para $k = 2$ até N_{max} faça
   $p_k = p_{k-1} + \lfloor (|\boldsymbol{S}| - p_{k-1})/np \rfloor$, ($\lfloor a \rfloor$: maior inteiro $\leq a$).
   np = np -1.
3) Divida o conjunto $\boldsymbol{S}$ em N_{max} subconjuntos da seguinte forma:

$\boldsymbol{S}_1 = \{ \boldsymbol{x}^1, ..., \boldsymbol{x}^{p_1} \}$, $\boldsymbol{S}_2 = \{ \boldsymbol{x}^{p_2+1}, ..., \boldsymbol{x}^{p_3} \}$, ..., $\boldsymbol{S}_{Nmax} = \{ \boldsymbol{x}^{p_{Nmax}+1}, ..., \boldsymbol{x}^{|\boldsymbol{S}|} \}$

4) De cada subconjunto $\boldsymbol{S}_i$ (i = 1, ..., N_{max}) selecione aleatoriamente uma solução e descarte as outras soluções. O novo conjunto $\boldsymbol{S}$ é formado pelas soluções selecionadas.

Observe que, o método descrito acima requer que os pontos dominantes estejam ordenados ($f_1(\boldsymbol{x}^i) < f_1(\boldsymbol{x}^{i+1})$), por isto o método é aplicável somente a problemas bi-objetivos. Na Figura 2.15 mostra-se um exemplo da aplicação do Algoritmo 2.3 sobre o espaço bi-objetivo. Neste exemplo, um conjunto $\boldsymbol{S}$ com $|\boldsymbol{S}| = 8$ soluções é reduzido a $N_{max} = 3$ soluções. A Figura 2.16, ilustra as imagens dos subconjuntos $\boldsymbol{S}_i$ (i = 1,...,3) e os correspondentes pontos $\boldsymbol{z}^i = \boldsymbol{f}(\boldsymbol{x}^i)$. Note que, $p_2 = 3$ e $p_3 = 5$; e $\boldsymbol{S}_1 = \{ \boldsymbol{x}^j \in \boldsymbol{S} : 1 \le j \le p_2 \}$, $\boldsymbol{S}_2 = \{ \boldsymbol{x}^j \in \boldsymbol{S} : p_2+1 \le j \le p_3 \}$ e $\boldsymbol{S}_3 = \{ \boldsymbol{x}^j \in \boldsymbol{S} : p_3+1 \le j \le 8 \}$.

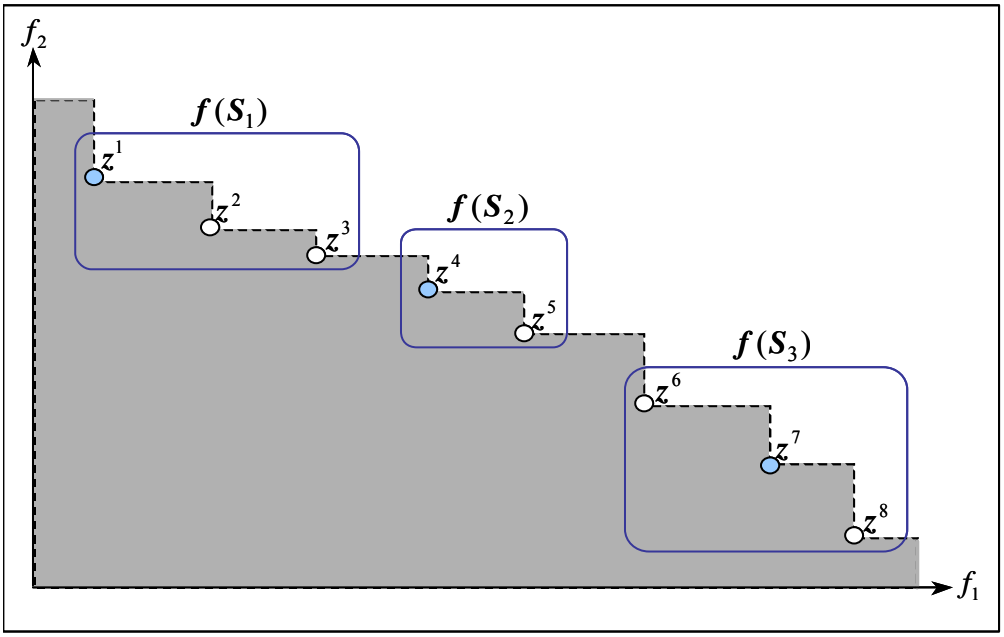


Figura 2.16. Redução Aleatória.

O segundo procedimento de redução que pode ser utilizado é a técnica denominada *clustering* proposta por Morse (1980), válida para dois ou mais objetivos. Esta técnica particiona o conjunto $\boldsymbol{S}$ de soluções dominantes em N_{max} grupos (clusters) de acordo com a

proximidade das soluções ($N_{max} < |\boldsymbol{S}|$). Para cada cluster, seleciona-se uma solução representativa (*centróide*), e as soluções restantes são descartadas. A seguir, são apresentados os passos da técnica de redução *clustering*.

Algoritmo 2.5. Redução por Clustering

Entrada: $\boldsymbol{S}$ (um conjunto de $|\boldsymbol{S}| > N_{max}$ soluções dominantes).
N_{max} (número máximo de soluções).

Saída: $\boldsymbol{S}$ (conjunto de N_{max} soluções dominantes).

1) Inicialize o conjunto de clusters: $\mathbf{C} = \bigcup_{i=1}^{|S|} \mathrm{C}_i$, onde $\mathrm{C}_i = \{\boldsymbol{x}_i\}$, $i = 1, ..., |\boldsymbol{S}|$.
2) Se $|\mathbf{C}| \leq N_{max}$ então vá para o passo 5, senão vá para o passo 3.
3) Calcule a distância de todos os pares possíveis de clusters. A distância d_c entre dois clusters C_1 e C_2 é calculada da seguinte maneira:

$$d_c(\mathrm{C}_1, \mathrm{C}_2) = \frac{1}{|C_1| \times |C_2|} \times \sum_{\boldsymbol{x} \in \mathrm{C}_1 \text{ e } \boldsymbol{y} \in \mathrm{C}_2} d(\boldsymbol{x}, \boldsymbol{y})$$

onde d é uma distância entre duas soluções (no espaço objetivo).
4) Selecione, dentre todos os clusters, dois clusters C_1 e C_2 com distância d_c mínima. Junte estes dois clusters em um único: $\mathrm{C}_{1,2} = \mathrm{C}_1 \cup \mathrm{C}_2$. Faça $\mathbf{C} = \mathbf{C} - \{\mathrm{C}_1, \mathrm{C}_2\} \cup \{\mathrm{C}_{1,2}\}$. Volte ao passo 2.
5) De cada cluster, selecione uma solução representativa (centroide). A solução $\boldsymbol{x}_c$ centroide do cluster C é aquela com $d_{\boldsymbol{x}_c} = \sum_{\boldsymbol{x} \in \mathrm{C}} d(\boldsymbol{x}_c, \boldsymbol{x})$ mínimo. O novo conjunto $\boldsymbol{S}$ será formado pelas N_{max} soluções centroides.

Na Figura 2.17, ilustra-se um exemplo da redução por *clustering*, onde o conjunto inicial $\boldsymbol{S}$ possui $|\boldsymbol{S}| = 19$ soluções e ele é reduzido a $N_{max} = 6$ soluções.

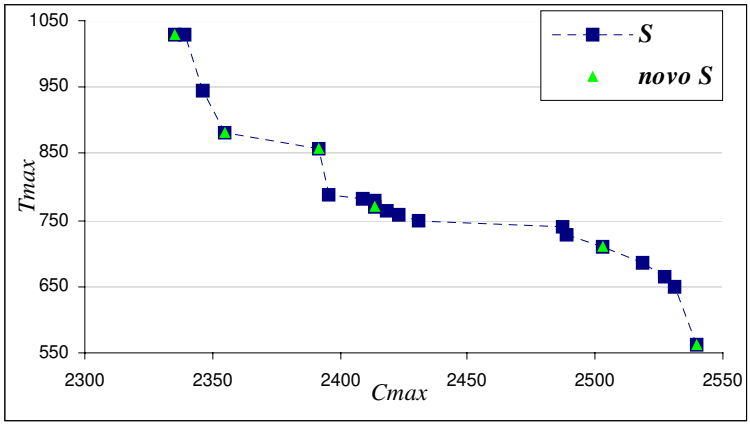


Figura 2.17. Redução por *Clustering*.

2.2.1. Aplicação da Busca Local Multiobjetivo ao Problema de Flowshop

Nesta seção apresentam-se os resultados computacionais do método proposto BLM quando aplicado ao problema de flowshop multiobjetivo. O método é testado para os seguintes pares de objetivos: i) (C_{max}, T_{max}) e ii) (C_{max}, T). A busca local BLM é iniciada a partir dos conjuntos dominantes obtidos pela heurística construtiva HMEP apresentada na seção 2.1. O desempenho da busca local BLM é avaliado sobre os três conjuntos de instancias como descrito na seção 2.1.2.1. Para o caso de duas máquinas, o método BLM é comparado com os algoritmos Branch-and-Bound propostos, respectivamente, por Daniels e Chambers (1990) e Liao et al. (1997), e para o caso de mais de duas máquinas a heurística é comparada com enumeração completa. Para instâncias de grande porte, determina-se a melhoria do método BLM em relação à heurística HMEP. Algumas estratégias e parâmetros usados no algoritmo BLM são descritos a seguir:

- *Estratégia de geração de vizinhança*: Foram testadas duas estratégias de geração de vizinhanças, inserção de tarefas e troca de tarefas. 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 $\frac{n}{2}(n-1)$. Estas duas estratégias de geração de vizinhanças para o problema de flowshop são bastante usadas em métodos de busca em vizinhança (Armentano e Ronconi, 1999; Gupta et al., 2000a; Taillard, 1990). Vale ressaltar que, os melhores resultados foram obtidos usando a vizinhança de inserção, portanto os resultados apresentados nas próximas seções correspondem ao uso desta vizinhança.

- *Número máximo de caminhos (ou soluções) explorados*: Como a busca local BLM explora várias soluções em cada iteração, é necessário determinar o número máximo de soluções a serem exploradas N_{max}. Se este número é demasiado grande, a busca local realizará uma busca exaustiva em toda a fronteira dominante gastando alto tempo computacional em cada iteração. Foram testados vários valores para N_{max} e bons resultados foram obtidos com $N_{max} = 6$.

- *Critério de parada*: Nos testes preliminares realizados observou-se que, a busca local BLM, geralmente, finaliza após executar I iterações ($I < 200$). No entanto, para algumas instâncias a busca local executa um número muito maior de iterações. Portanto para ter uma uniformidade, definiou-se um número máximo de iterações: *Niter* = 200. Isto significa que a busca local é executada até que $\boldsymbol{S}' = \varnothing$ (conjunto de soluções vizinhas dominantes) ou até atingir o número máximo de iterações.

- *Técnica de redução*: A técnica utilizada para reduzir o conjunto de soluções dominantes $\boldsymbol{S}$ foi *clustering* (Algoritmo 2.5). Os pontos representativos selecionados pela técnica *clustering* sempre estão distribuídos sobre a fronteira dominante correspondente ao conjunto $\boldsymbol{S}$. Além disso, esta técnica é computacionalmente eficiente quando aplicada para reduzir um conjunto $\boldsymbol{S}$ não muito grande.

2.2.1.1. Resultados para $f = (C_{max}, T_{max})$

Resultados para o Conjunto 1 de Instâncias

Para as instâncias deste conjunto (n = 10, 11 e m = 5, 10) a busca local BLM é comparada com o método de enumeração completa e com a heurística HMEP. Na Tabela 2.22 apresenta-se, para cada tamanho de problema, o número de soluções eficientes e o número de soluções obtidas pelos métodos BLM e HMEP associadas com 80 instâncias. Para as 320 instâncias testadas, os métodos BLM e HMEP identificaram 1897 (63,15%) e 771 (25,67%) soluções eficientes, respectivamente. Além disso, os métodos BLM e HMEP encontraram o conjunto Pareto-ótimo para 39 e 5 instâncias respectivamente.

Tabela 2.22. Número de soluções encontradas pela busca local BLM comparado com Enumeração Completa e HMEP

Problema $n \times m$	Enumeração Completa	BLM		HMEP	
	NSE	NS-BLM	NSE-BLM	NS-HMEP	NSE-HMEP
10×5	582	491	411	435	197
10×10	786	650	522	495	195
11×5	687	562	431	464	192
11×10	949	742	533	614	187
Total	3004	2445	1897	2008	771

NSE: número de soluções eficientes encontradas pelo método exato.
NS-BLM (NS-HMEP): número de soluções encontradas pelo método BLM (HMEP).
NSE-BLM (NSE-HMEP): número de soluções eficientes encontradas pelo método BLM (HMEP).

Na Tabela 2.23 comparam-se os desempenhos dos métodos BLM e HMEP de acordo com a medida de distância e o erro associado com a função utilidade linear proposta por Daniels (1992). Segundo estas medidas, observa-se que o método BLM apresenta uma melhoria considerável com relação à heurística HMEP.

Na Tabela 2.24 para cada tamanho do problema mostra-se, a média dos tempos computacionais gasto pela busca local BLM e a média do número de iterações realizadas por este método. Para todas as instâncias deste conjunto, a busca local BLM não precisou executar o número máximo de iterações N_{max} = 200.

Tabela 2.23. Desempenho da busca Local BLM comparado com a heurística HMEP

Problema $n\times m$	Medida de distância				Erro de utilidade (%)			
	BLM		HMEP		BLM		HMEP	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×5	0,040	0,105	0,131	0,328	1,719	6,769	6,857	16,274
10×10	0,026	0,098	0,151	0,353	2,176	7,975	8,941	22,833
11×5	0,062	0,138	0,149	0,333	3,477	10,359	8,769	22,425
11×10	0,034	0,105	0,138	0,348	2,265	8,003	7,773	18,902
Média	0,040	0,111	0,142	0,340	2,409	8,276	8,085	20,109

Tabela 2.24. Média de tempos computacionais (em segundos) e do número de iterações

$n\times m$:	10×5	10×10	11×5	11×10
Tempo - BLM	0,011	0,015	0,013	0,027
Nº iterações - BLM	3,2	3,5	4	4,3

Resultados para o Conjunto 2 de Instâncias

A busca local BLM é comparada com o método Branch-and-Bound e com a heurística HMEP. De um total de 507 soluções eficientes, 464 (91,52%) soluções eficientes foram obtidas pela busca local BLM. Isto significa que, BLM conseguiu encontrar 111 novas soluções eficientes a mais que a heurística HMEP. A Tabela 2.25 exibe, para cada tamanho de problema, o número de soluções eficientes e o número de soluções obtidas por cada método. Vale ressaltar que para este conjunto de instâncias, não foi necessário aplicar a técnica de redução no processo da busca local, pois N_{max} sempre é menor que 4.

Na Tabela 2.26 compara-se o desempenho dos métodos BLM e HMEP de acordo com a medida de distância e o erro associado com a função utilidade linear proposta por Daniels (1992). Segundo estas medidas, observa-se que o método BLM apresenta uma melhoria considerável com relação à heurística HMEP.

Na Tabela 2.27 para cada tamanho do problema mostra-se, a média dos tempos computacionais gasto pela busca local BLM e a média do número de iterações realizadas por este método.

Tabela 2.25. Número de soluções encontradas pela busca local BLM comparados com B&B e HMEP

Problema	B&B	BLM		HMEP	
$n \times m$	NSE	NS-BLM	NSE-BLM	NS-HMEP	NSE-HMEP
10×2	122	120	116	125	100
15×2	140	137	130	142	98
20×2	144	134	125	147	81
25×2	101	101	93	116	74
Total	507	492	464	530	353

Tabela 2.26. Desempenho da busca local BLM comparado com a heurística HMEP

Problema	Medida de distância				Erro de utilidade (%)			
	BLM		HMEP		BLM		HMEP	
$n \times m$	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×2	0,003	0,120	0,036	0,094	0,128	0,605	0,702	2,502
15×2	0,033	0,047	0,082	0,131	0,823	2,012	1,264	3,140
20×2	0,024	0,059	0,104	0,194	1,221	4,045	4,395	8,543
25×2	0,031	0,043	0,055	0,127	0,431	1,398	3,944	7,070
Média	0,023	0,067	0,069	0,136	0,651	2,015	2,576	5,314

Tabela 2.27. Média de tempos computacionais (em segundos) e do número de iterações

$n \times m$:	10×2	15×2	20×2	25×2
Tempo - BLM	0,0057	0,0082	0,025	0,023
Nº iterações - BLM	1,7	1,93	2,6	2,2

Resultados para o Conjunto 3 de Instâncias

O comportamento da busca local BLM foi avaliado em instâncias de grande porte pertencentes a este conjunto (n = 20, 50, 80 e m = 5, 10, 20). Das 6941 soluções encontradas pela heurística HMEP, 6864 soluções foram melhoradas pela busca local BLM gerando um total de 7590 soluções dominantes. É importante ressaltar que, 77 soluções geradas pela heurística HMEP não foram melhoradas pela busca local, ou seja, estas 77 soluções continuam dominantes após a aplicação da busca local BLM. Na Tabela 2.28 apresentam-se os erros de

distância e de utilidade da heurística HMEP em relação à busca local BLM. Note que o método BLM teve um "ganho" considerável para todos os tamanhos de problemas. A Figura 2.18 ilustra um exemplo do conjunto de pontos dominantes da busca local BLM geradas a partir do conjunto de pontos da heurística HMEP.

Na Tabela 2.29 apresenta-se a média dos tempos computacionais e a média do número de iterações da busca local BLM. Vale ressaltar que a busca local BLM atingiu o número máximo de iterações em 9 instâncias com $n = 80$ e $m = 20$.

Tabela 2.28. Erro da heurística HMEP em relação à busca local BLM

Problema	Medida de distância		Erro de utilidade (%)	
$n \times m$	D_{med}	D_{max}	E_{med}	E_{max}
20×5	0,155	0,251	9,767	19,884
20×10	0,141	0,236	12,988	21,821
20×20	0,126	0,215	11,345	20,132
50×5	0,138	0,209	9,708	16,522
50×10	0,173	0,253	15,992	23,812
50×20	0,170	0,254	16,061	24,103
80×5	0,125	0,202	8,194	14,350
80×10	0,167	0,240	15,052	22,548
80×20	0,183	0,252	17,229	25,106
Média	0,153	0,235	12,926	20,920

Tabela 2.29. Média de tempos computacionais (em segundos) e do número de iterações

Problema n×m	Tempo - BLM	Nº iterações - BLM
20×5	0,162	9,0
20×10	0,444	14,07
20×20	0,886	14,5
50×5	5,12	20,3
50×10	29,84	55,6
50×20	78,31	75,0
80×5	20,15	20,6
80×10	192,11	79,05
80×20	637,18	148,8

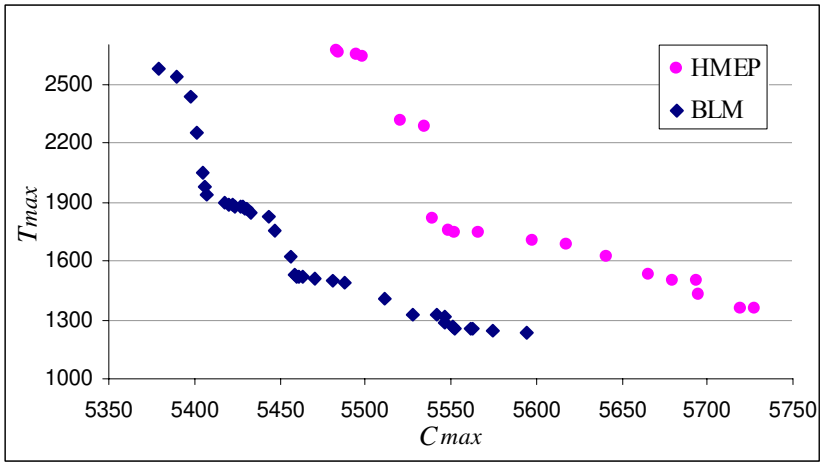


Figura 2.18. Conjuntos de pontos dominantes gerados pelos métodos HMEP e BLM para uma instância de tamanho 80×20.

2.2.1.2. Resultados para $f = (C_{max}, T)$

- **Resultados para o Conjunto 1 de Instâncias**

Para as instâncias deste conjunto a busca local BLM é comparada com o método de enumeração completa e com a heurística HMEP. Para cada grupo de 80 instâncias, na Tabela 2.30 apresenta-se o número de soluções eficientes e o número de soluções obtidas pelos métodos BLM e HMEP. Para o total de 320 instâncias testadas, os métodos HMEP e BLM identificaram 755 (21,93%) e 2239 (65,05%) soluções eficientes, respectivamente. A heurística HMEP identificou o conjunto Pareto-ótimo para 7 instâncias e a busca local BLM para 40 instâncias.

Na Tabela 2.31 comparam-se os desempenhos dos métodos BLM e HMEP de acordo com a medida de distância e o erro de utilidade. Segundo estas medidas, observa-se que o método BLM apresenta uma melhoria considerável em relação à heurística HMEP.

Na Tabela 2.32 para cada tamanho do problema mostra-se, a média dos tempos computacionais gasto pela busca local BLM e a média do número de iterações realizadas por este método.

Tabela 2.30. Número de soluções encontrados pela busca local BLM comparados com Enumeração Completa e HMEP

Problema $n\times m$	Enumeração Completa	BLM		HMEP	
	NSE	NS-BLM	NSE-BLM	NS-HMEP	NSE-HMEP
10×5	755	628	507	502	183
10×10	820	704	598	500	203
11×5	825	690	521	555	175
11×10	1042	808	613	595	194
Total	3442	2830	2239	2152	755

NSE: número de soluções eficientes encontradas pelo método exato.
NS-BLM (NS-HMEP): número de soluções encontradas pelo método BLM (HMEP).
NSE-BLM (NSE-HMEP): número de soluções eficientes encontradas pelo método BLM (HMEP).

Tabela 2.31. Desempenho da busca Local BLM comparado com a heurística HMEP

Problema $n\times m$	Medida de distância				Erro de utilidade (%)			
	BLM		HMEP		BLM		HMEP	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×5	0,045	0,128	0,131	0,269	3,164	10,455	6,541	16,690
10×10	0,025	0,093	0,158	0,339	2,221	7,893	9,798	19,801
11×5	0,041	0,131	0,140	0,293	3,220	10,320	8,232	17,138
11×10	0,036	0,121	0,157	0,319	2,951	10,562	7,526	15,545
Média	0,037	0,118	0,147	0,305	2,889	9,808	8,024	17,294

Tabela 2.32. Média de tempos computacionais (em segundos) e do número de iterações da busca local BLM

$n\times m$:	10×5	10×10	11×5	11×10
Tempo – BLM	0,010	0,020	0,018	0,0033
Nº iterações – BLM	3,97	4,31	4,48	5,4

Resultados para o Conjunto 2 de Instâncias

A busca local BLM é comparada com o método Branch-and-Bound proposto por Liao et al. (1997) e com a heurística HMEP. De um total de 474 soluções eficientes, 357 (75,32%) soluções eficientes foram obtidas pela busca local BLM. Isto significa que, BLM conseguiu encontrar 143 novas soluções eficientes a mais que a heurística HMEP. A Tabela 2.33 exibe,

para cada grupo de 40 problemas, o número de soluções eficientes e o número de soluções obtidas por cada método.

Tabela 2.33. Número de soluções encontradas pela busca local BLM comparados com B&B e HMEP

Problema	B&B	BLM		HMEP	
$n \times m$	NSE	NS-BLM	NSE-BLM	NS-HMEP	NSE-HMEP
10×2	158	154	139	144	88
15×2	172	166	128	159	73
20×2	144	139	90	164	53
Total	474	459	357	467	214

Na Tabela 2.34 compara-se o desempenho dos métodos BLM e HMEP de acordo com a medida de distância e o erro de utilidade. Observe que a busca local BLM apresenta uma melhoria razoável em relação à heurística HMEP.

Na Tabela 2.35 para cada tamanho do problema mostra-se, a média dos tempos computacionais gasto pela busca local BLM e a média do número de iterações realizadas por este método.

Tabela 2.34. Desempenho da busca local BLM comparado com a heurística HMEP

Problema	Medida de distância				Erro de utilidade (%)			
	BLM		HMEP		BLM		HMEP	
$n \times m$	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×2	0,034	0,057	0,115	0,242	0,926	4,812	2,036	6,579
15×2	0,020	0,044	0,090	0,174	1,760	4,539	4,814	11,103
20×2	0,101	0,147	0,219	0,317	2,231	8,020	8,602	16,603
Média	0,052	0,083	0,141	0,245	1,639	5,790	5,151	11,428

Tabela 2.35. Média de tempos computacionais (em segundos) e do número de iterações da busca local BLM

$n \times m$:	10×2	15×2	20×2
Tempo – BLM	0,007	0,012	0,041
Nº iterações – BLM	2,35	3,17	4,3

Resultados para o Conjunto 3 de Instâncias

O comportamento da busca local BLM, para (C_{max} , T), foi avaliado em 360 instâncias de grande porte pertencentes a este conjunto. Das 9238 soluções encontradas pela heurística HMEP, 9216 soluções foram melhoradas pela busca local BLM gerando 13206 soluções dominantes. 22 soluções obtidas pela heurística HMEP não foram melhoradas pela busca local, ou seja, estas soluções permanecem dominantes após a execução da busca local BLM. Na Tabela 2.36 apresentam-se os erros de distância e de utilidade da heurística HMEP em relação à busca local BLM. Note que o método BLM teve um "ganho" relevante para todos os tamanhos de problemas.

Tabela 2.36. Erro da heurística HMEP em relação à busca local BLM

Problema	Medida de distância		Erro de utilidade (%)	
$n \times m$	D_{med}	D_{max}	E_{med}	E_{max}
20×5	0,177	0,282	15,271	25,768
20×10	0,206	0,317	18,92	29,932
20×20	0,208	0,328	21,455	31,951
50×5	0,199	0,278	14,909	22,731
50×10	0,233	0,315	22,137	30,818
50×20	0,229	0,294	23,895	30,74
80×5	0,207	0,246	32,83	38,806
80×10	0,213	0,279	18,691	26,208
80×20	0,229	0,291	23,31	29,937
Média	0,211	0,292	21,269	29,655

Na Tabela 2.37 apresenta-se a média dos tempos computacionais gastos pela busca local BLM e a média do número de iterações realizadas por este método. A busca local BLM atingiu o número máximo de iterações (*Niter* = 200) em 13 instâncias de tamanho 80×10 e em 24 instâncias de tamanho 80×20.

Tabela 2.47. Média de tempos computacionais (em segundos) e do número de iterações da busca local BLM

Problema $n \times m$	Tempo - BLM	Nº iterações - BLM
20×5	0,233	13,3
20×10	0,516	15,95
20×20	0,869	14,85
50×5	9,574	39,3
50×10	49,517	94
50×20	92,07	92,8
80×5	52,57	49,7
80×10	303,37	145,72
80×20	863,69	182,27

Capítulo 3

Algoritmos Genéticos Para Otimização Multiobjetivo

Algoritmos Genéticos (AGs) foram introduzidos por John Holland (1975) e fazem parte da área de Computação Evolutiva, que constitui uma família de métodos computacionais inspirados na evolução natural das espécies. Os AGs são métodos flexíveis e têm a capacidade de produzir soluções de boa qualidade em problemas complexos e de grande porte, em tempo computacional viável. Por esta razão têm sido aplicados com enorme sucesso em uma grande variedade de problemas em otimização combinatória NP-Completos e NP-Difíceis (Goldberg 1989, Michalewicz 1996).

Quando um algoritmo genético (AG) é aplicado a um problema de otimização, cada solução do problema deve ser codificada ou representada na forma de uma estrutura finita (vetor, matriz, etc.). Em seguida, devem ser definidos os operadores genéticos de seleção, recombinação, mutação e estratégias de elitismo. Estes operadores devem ser escolhidos de acordo com as características intrínsecas do problema. Por exemplo, operadores apropriados para soluções de permutação são, em geral, distintos de operadores para soluções com representação binária. Antes de aplicar o AG para resolver um problema de otimização, vários parâmetros devem ser especificados, tais como tamanho da população, probabilidade de recombinação e probabilidade de mutação. De uma maneira geral, os AGs "puros" têm mostrado um desempenho inferior aos métodos baseados em busca em vizinhança, tais como busca tabu e simulated annealing (Glass et al., 1992; Ishibuchi et al., 1994). A hibridização dos AGs com outros métodos, em especial a busca em vizinhança, foi muito bem sucedida e estes AGs híbridos são extremamente competitivos com as demais metaheurísticas.

Muitos problemas reais de otimização combinatória admitem diferentes funções objetivos, em geral conflitantes entre si. Na maioria dos casos é impossível melhorar um objetivo sem deteriorar algum outro. Os problemas de otimização multiobjetivo, em geral, não possuem soluções ótimas que otimizem individualmente todos os objetivos considerados. Estes problemas possuem como solução um conjunto de soluções denominadas eficientes. A dificuldade em resolver problemas combinatórios multiobjetivos não somente é dada pela complexidade combinatorial, como no caso de problemas mono-objetivo, mas também pela busca de todas as soluções eficientes que crescem com o número de objetivos do problema. Para resolver problemas de otimização multiobjetivos, os métodos baseados em metaheurísticas parecem ser a melhor alternativa, pois estes métodos são flexíveis, eficientes e de fácil implementação. Os objetivos principais de toda metaheurística de otimização multiobjetivo são:

- Minimizar a distância do conjunto dominante encontrado ao conjunto Pareto-ótimo
- Obter uma boa distribuição das soluções no conjunto dominante gerado.

A maioria esmagadora das publicações de metaheurísticas para problemas de otimização multiobjetivos são baseadas em algoritmos genéticos (Ehrgott e Gandibleux, 2000; Coello, 2000; Van Veldhuizen e Lamount, 2000a; Jones et al., 2002). Esta preferência se deve ao argumento questionável, que os AGs trabalham com uma população de soluções que podem conter informação sobre várias regiões do espaço de busca, e portanto, estes oferecem maiores possibilidades para encontrar o conjunto Pareto-ótimo ou uma aproximação dele. Depois do primeiro algoritmo genético multiobjetivo, (Vector Evaluated Genetic Algorithm - VEGA), proposto por Schaffer (1985), várias extensões de AGs, para problemas multiobjetivos, foram sugeridas em diferentes maneiras (Horn et al. 1993; Fonseca e Fleming 1993; Srinivas e Deb 1995; Ishibuchi e Murata 1998; Zitzler e Thiele 1999; Jaskiewicz 2002).

A diferença principal entre um AG multiobjetivo e um AG mono-objetivo reside na forma como é atribuído o nível de aptidão (fitness) às soluções. Devido à indiferença existente entre as soluções dominantes de um problema multiobjetivo, é necessário definir algumas estratégias para calcular o fitness das soluções e para selecionar as soluções com maior probabilidade de reprodução.

Neste capítulo apresentamos um estudo sobre extensões de algoritmos genéticos para otimização multiobjetivo. Na seção 3.1 apresentam-se algumas estratégias usadas em AGs multiobjetivos. Os principais AGs multiobjetivos da literatura são apresentados na seção 3.2.

3.1. Estratégias usadas em Algoritmos Genéticos Multiobjetivos

No desenvolvimento de algoritmos genéticos para otimização multiobjetivo, existem duas questões importantes:

1. Para guiar a busca nas direções das soluções Pareto-ótimas, como calcular o nível de aptidão (*fitness*) das soluções da população, e como fazer a escolha de soluções (seleção) que participarão na reprodução de novas soluções.
2. Como preservar a diversidade na população, evitando uma convergência prematura e obtendo um conjunto de soluções dominantes bem distribuídas.

   A seguir, são apresentadas algumas estratégias que tratam sobre estas questões.

3.1.1. Cálculo do Fitness e Técnicas de Seleção de Soluções

Em um AG convencional, a seleção das soluções pais, que se reproduzirão, é feita de acordo com o *fitness* das soluções na população corrente. Em otimização mono-objetivo a função fitness e a função objetivo de uma solução, geralmente coincidem, ou a primeira depende da segunda. No entanto, em otimização multiobjetivo, os objetivos são considerados separadamente, sendo necessário definir critérios adequados para calcular o fitness e a probabilidade de reprodução das soluções.

Na literatura de algoritmos genéticos multiobjetivos, existem técnicas diferentes para determinar os fitness das soluções e/ou para selecionar as soluções pais.

Schaffer (1985) implementa um método de seleção que consiste na separação dos objetivos. A cada iteração geram-se r sub-populações $\boldsymbol{P}_j$, associadas aos r objetivos e de tamanho N / r, onde N é o tamanho da população, e cada sub-população contém as melhores soluções, segundo um dado objetivo. Portanto, este método busca pontos segundo direções dos

eixos das funções objetivos e, consequentemente, dá maior preferência a pontos dominantes localizados nos extremos da fronteira. A Figura 3.1 ilustra a divisão da população no método de Schaffer.

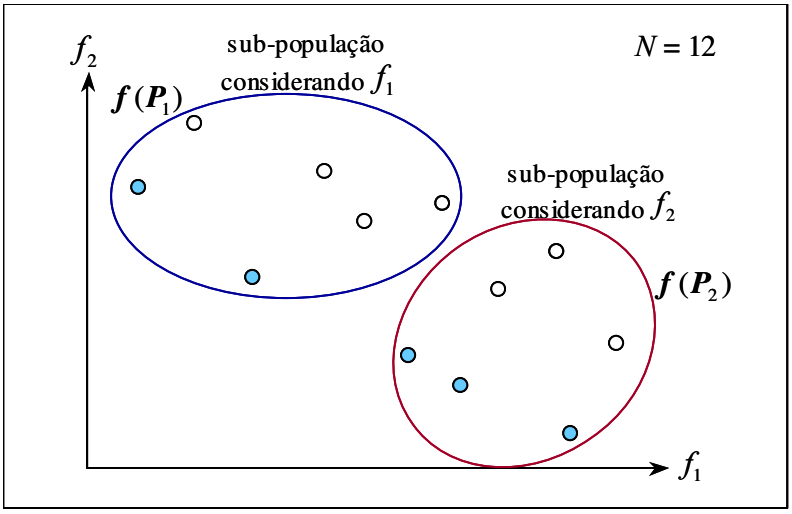


Figura 3.1. Divisão da população no algoritmo genético de Schaffer.

Hajela e Lin (1992), Murata (1996), Ishibushi e Murata (1998), atribuem um valor de fitness para cada solução através do método tradicional da somas ponderadas dos objetivos (veja Seção 1.3). Neste caso, o fitness de uma solução $\boldsymbol{x}$ é expresso como a combinação linear dos objetivos:

$$fitness(\boldsymbol{x}) = \sum_{k=1}^{r} w_k . f_k(\boldsymbol{x})$$

onde w_k é o peso associado ao objetivo f_k e $\sum_{k=1}^{r} w_k = 1$. Para cada solução, os pesos são gerados aleatoriamente (Ishibushi e Murata, 1998) ou de forma pré-determinada (Hajela e Lin, 1992) para definir várias direções de busca, evitando assim a limitação do método de Schaffer. Já que a magnitude dos objetivos pode ser bastante diferente, Hajela e Lin (1992) sugerem normalizar os objetivos da seguinte maneira:

$$fitness(\boldsymbol{x}) = \sum_{k=1}^{r} w_k . \frac{f_k(\boldsymbol{x})}{f_k^*}$$

onde f_k^* é um parâmetro de escalamento correspondente ao objetivo *k*. Note que, pela maneira como é definido o *fitness*, o problema multiobjetivo é transformado em um problema mono-objetivo no qual se otimiza a função *fitness*($\boldsymbol{x}$). A desvantagem deste enfoque é a impossibilidade de encontrar soluções eficientes nas partes não-convexas da fronteira Pareto-ótima, além da falta de controle na geração de direções de busca. Um método similar é proposto por Jaszkiewicz (2002), onde a seleção das soluções pais depende dos valores uma função utilidade ponderada. No método de Jaszkiewicz, atribui-se a cada solução $\boldsymbol{x}$ um valor de escalar calculado pela seguinte função utilidade de Tchebycheff:

$$u_\infty(\boldsymbol{x}) = \max_j \left\{ \lambda_j . (f_j(\boldsymbol{x}) - z_j^{\mathrm{o}}) \right\}$$

onde, $z^{\mathrm{o}} = (z_1^{\mathrm{o}}, ..., z_r^{\mathrm{o}})$ é uma aproximação do ponto ideal e $\lambda_j \geq 0$ são pesos gerados aleatoriamente tal que, $\sum_{i=1}^{r} \lambda_i = 1$.

Goldberg (1989) sugere o uso do conceito de dominância de Pareto para calcular o fitness das soluções e apresenta um esboço de um procedimento iterativo denominado "*ranking*". Neste procedimento, inicialmente, atribui-se às soluções dominantes da população um valor *rank* = 1. Desconsiderando temporariamente estas soluções, determinam-se as novas soluções dominantes da população restante para atribuir-lhes *rank* = 2. Este procedimento continua até que toda a população seja classificada. O *rank* de uma solução determina o potencial de reprodução (*fitness*). Note que, as melhores soluções possuirão menor *rank* e o fitness de uma solução particular depende da relação de dominância entre as outras soluções. Esta idéia de utilização do conceito de dominância de Pareto para o cálculo de fitness tem sido muito utilizada em vários algoritmos genéticos multiobjetivos (Horn et al. 1993; Fonseca e Fleming 1993; Srinivas e Deb 1995; Zitzler e Thiele 1999; Deb et al. 2000).

3.1.2. Preservação da Diversidade na População

Em um AG mono-objetivo, a população de soluções, em geral, tende a convergir para uma única solução. Este comportamento pode ser aceitável quando se deseja encontrar uma única solução ótima ou uma aproximação a esta. Entretanto, em otimização multiobjetivo, se

não é preservada a diversidade da população, ela pode convergir a regiões pequenas do conjunto Pareto-ótimo. Portanto, manter a diversidade da população é crucial para encontrar um conjunto de soluções dominantes cujos pontos estejam bem distribuídos ao longo de toda a fronteira de Pareto. Na literatura, foram desenvolvidas algumas técnicas para manter a diversidade da população. A seguir, apresentamos as mais relevantes.

Goldberg e Richardson (1987), propõem uma técnica denominada *fitness sharing*, que consiste em dividir a população em diferentes sub-populações (*niches*) de acordo com a proximidade das soluções. A idéia desta técnica é penalizar (no caso, diminuir) o fitness de soluções que estão muito próximas numa dada sub-população. A proximidade das soluções é determinada através de uma função no espaço de decisões ($\boldsymbol{X}$) ou no espaço objetivo ($\boldsymbol{Z}$), chamada função *sharing*:

$$Sh(d(\boldsymbol{x},\boldsymbol{y})) = \begin{cases} 1-\left(\dfrac{d(\boldsymbol{x},\boldsymbol{y})}{\sigma_{sh}}\right)^{\alpha}, & \text{se } d(\boldsymbol{x},\boldsymbol{y}) < \sigma_{sh} \\ 0, & \text{caso contrário} \end{cases}$$

onde $d(\boldsymbol{x},\boldsymbol{y})$ é uma distância entre duas soluções $\boldsymbol{x}$ e $\boldsymbol{y}$ pertencente à sub-população, σ_{sh} é a distância máxima (raio) permitida entre qualquer duas soluções pertencentes à sub-população, e α é um parâmetro que normalmente é 1 ou 2. Em cada sub-população $\boldsymbol{S}$, a penalização das soluções é realizada através do seguinte algoritmo:

Algoritmo 3.1. Fitness Sharing

1) Para cada solução $\boldsymbol{x} \in \boldsymbol{S}$ atribua um fitness inicial $h(\boldsymbol{x})$ (*dummy fitness*).
2) Para cada solução $\boldsymbol{x} \in \boldsymbol{S}$ calcule a seguinte *medida niche*:

$$m_{\boldsymbol{x}} = \sum_{\boldsymbol{y} \in \boldsymbol{S}} Sh(d(\boldsymbol{x},\boldsymbol{y}))$$

($m_{\boldsymbol{x}}$ é uma estimativa da quantidade de soluções que rodeiam a solução $\boldsymbol{x}$).

3) Para cada solução $\boldsymbol{x} \in \boldsymbol{S}$ calcule o novo fitness (*shared fitness*) da forma seguinte:

$$fitness(\boldsymbol{x}) = \frac{h(\boldsymbol{x})}{m_{\boldsymbol{x}}}, \; m_{\boldsymbol{x}} \neq 0.$$

Na Figura 3.2, mostra-se um exemplo do *fitness sharing* no espaço de dois objetivos. Note que na sub-população de pontos $\boldsymbol{f}(\boldsymbol{S})$, o fitness inicial *h* das soluções representadas pelos pontos z^1 e z^2 é reduzido, pois para o raio σ_{sh} considerado na figura, $d(z^1, z^2) < \sigma_{sh}$. Esta técnica de diversificação foi usada em várias implementações de algoritmos genéticos da literatura (Hajela e Lin 1992; Horn et al. 1993; Fonseca e Fleming 1993; Srinivas e Deb 1995; Todd e Sen 1997, Zhou e Gen 1999). O valor do parâmetro σ_{sh} influencia bastante o desempenho dos AGs, e por este motivo Deb e Goldberg (1991) e Fonseca e Fleming (1993) sugerem métodos para calcular este parâmetro.

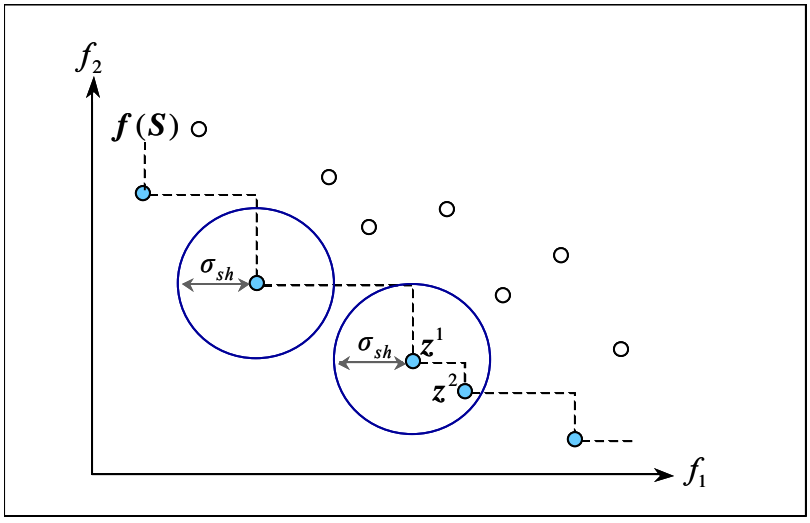


Figura 3.2. Técnica de diversificação: *fitness sharing*.

Deb et al. (2000) propõem uma outra técnica para preservar a diversidade na população. Inicialmente a população é classificada em sub-populações, da mesma forma sugerida por Goldberg (1989). As soluções pertencentes a uma dada sub-população $\boldsymbol{S}$ possuem o mesmo nível de dominância, ou seja, elas são indiferentes. Para atribuir um fitness a estas soluções, Deb et al. (2000) propõem uma estratégia baseada na proximidade das soluções. Para cada solução $\boldsymbol{x} \in \boldsymbol{S}$, determina-se uma estimativa do número de soluções que rodeiam a imagem da solução $\boldsymbol{x}$ no espaço objetivo $\boldsymbol{Z}$. Inicialmente os pontos, correspondentes às soluções em $\boldsymbol{S}$, são ordenados em forma crescente do primeiro objetivo; suponha que a ordenação seja $z^1, ..., z^{np}$, onde *np* é o número de soluções na sub-população $\boldsymbol{S}$. Para cada ponto z^i calcula-se a distância

$$d_1(z^i) = f_1(z^{i+1}) - f_1(z^{i-1})$$

Ordenando os pontos de acordo com os outros objetivos, calcula-se de maneira similar $d_2(z^i)$, ..., $d_r(z^i)$. Então, a estimativa do número de pontos situados em redor de um ponto z^i é dada por $dist(z^i) = \sum_{j=1}^{r} d_j$. Esta medida é chamada de *distância crowding*, que representa uma estimativa do tamanho do maior "cubóide" cobrindo o ponto z^i, sem incluir nenhum outro ponto. Aos pontos z^1 e z^{np} são associados valores da distância *crowding* arbitrariamente grande. Para calcular a distância *crowding* das soluções na sub-população $\boldsymbol{S}$ é usado o seguinte algoritmo:

Algoritmo 3.2. Atribuição da distância Crowding

Entrada $\boldsymbol{S}$, $|\boldsymbol{S}| = np$ (*sub-população de tamanho np*)

Saída $dist(z)$, $\forall z \in \boldsymbol{f}(\boldsymbol{S})$ (*Distância crowding no espaço objetivo*)

1) Para cada ponto $z^i \in \boldsymbol{f}(\boldsymbol{S})$ faça $dist(z^i) = 0$.
2) Para cada objetivo j faça
   2.1) Ordene os pontos em $\boldsymbol{f}(\boldsymbol{S})$ de acordo com o objetivo j obtendo $z^1, ..., z^{np}$.
   2.2) $dist(z^1) = v$, $dist(z^{np}) = v$ (v é um valor suficientemente grande)
   2.3) Para $i = 2$ até (np-1) faça
      $dist(z^i) = dist(z^i) + (f_j(z^{i+1}) - f_j(z^{i-1}))$, onde $f_j(z^i)$ é o valor da j-ésima função objetivo do ponto z^i.

Na Figura 3.3 mostra-se o cubóide (retângulo) e a distância *crowding* do ponto z^i, para o caso de dois objetivos. Observe que os pontos cobertos por cuboides grandes correspondem às soluções que terão maiores probabilidades de reprodução.

Capítulo 4

Proposta de um Algoritmo Genético Multiobjetivo

O objetivo deste Capítulo é apresentar um Algoritmo Genético Híbrido Multiobjetivo (AGHM), desenvolvido para encontrar um conjunto de soluções dominantes de um problema de otimização combinatória envolvendo dois ou mais objetivos. Na seção a seguir são descritas as principais estratégias usadas na implementação da metaheurística proposta. Apresenta-se na seção 4.2 o resumo do AGHM na forma de um pseudocódigo. Nas seções 4.3 e 4.4 são apresentados os componentes do AGHM aplicado ao problema de flowshop multiobjetivo e a análise dos testes computacionais realizados, respectivamente. Nas seções 4.5 e 4.6 apresenta-se a descrição da implementação do AGHM para resolver o problema da mochila multiobjetivo e os testes computacionais realizados, respectivamente.

4.1. Estratégias Usadas no Algoritmo Genético Híbrido Multiobjetivo (AGHM) proposto

No AGHM proposto, combinam-se as seguintes estratégias a fim de realizar uma busca eficaz e encontrar várias soluções em paralelo.

- Elitismo: é mantido um subconjunto de soluções dominantes encontradas até o momento.
- Classificação da população de acordo com a dominância das soluções para calcular o fitness de cada solução.
- Preservação de diversidade na população.
- Uso de busca local multiobjetivo baseada na relação de dominância de Pareto.

4.1.1. Estratégia de Elitismo

A incorporação da estratégia de elitismo em AGs multiobjetivos é mais complexa que em AGs mono-objetivo. Em AGs mono-objetivo, geralmente, a melhor solução é mantida na população. No caso multiobjetivo, em lugar da melhor solução, existe um conjunto de soluções dominantes (melhor aproximação do conjunto Pareto-ótimo) cujo tamanho pode crescer de acordo com o número de objetivos do problema ou com o tamanho da instância do problema. Por isto, é necessário fixar critérios para a escolha das soluções de elite que serão adicionados na população seguinte. Na metaheurística AGHM proposta, é usado uma estratégia de elitismo que combina as estratégias de Zitzler e Thiele (1999) e Ishibuchi e Murata (1998). Selecionam-se N_{elite} soluções dominantes encontradas até a iteração atual t para serem adicionadas na população seguinte $\boldsymbol{P}_{t+1}$. Então, na iteração t+1 constrói-se uma população combinada $\boldsymbol{P}_{t+1} \cup \boldsymbol{E}_t$, onde $\boldsymbol{E}_t$ é o conjunto das N_{elite} soluções de elite selecionadas a partir do conjunto de soluções dominantes $\boldsymbol{D}_t$ encontradas até a iteração t. A seleção das N_{elite} soluções de elite pode ser baseada em um critério que garanta a escolha de soluções dominantes dispersas, isto é, pontos no espaço objetivo distribuídos sobre toda a fronteira dominante atual. Na nossa implementação, estas soluções de elite são selecionadas aleatoriamente com uma distribuição uniforme. A cada iteração, escolhe-se aleatoriamente N_{elite} soluções de elite do conjunto dominante. A razão para a não utilização de uma técnica que garanta a distribuição das soluções é a seguinte: se o conjunto de soluções dominantes $\boldsymbol{D}$ não é alterado de uma iteração para outra, uma técnica deterministica, como por exemplo a técnica de *clustering* (Morse, 1980), de uma iteração para outra, sempre selecionará as mesmas N_{elite} soluções dispersas (centróides). Uma possibilidade mais custosa não explorada aqui consiste em determinar os clusters e escolher um ponto aleatório de cada cluster.

4.1.2. Classificação da População

Para atribuir um valor de fitness às soluções no AGHM, em primeiro lugar classificam-se as soluções da população combinada $\boldsymbol{P}_t \cup \boldsymbol{E}_{t\text{-}1}$, onde $\boldsymbol{P}_t$ é a população da iteração atual e $\boldsymbol{E}_{t\text{-}1}$ é o conjunto de soluções de elite encontrados até a iteração anterior. Adota-se uma técnica de ordenação por dominância (*Nondominated Sorting technique*) inicialmente proposta por

Goldberg (1989) e utilizada por Srinivas e Deb (1995) e Deb et al. (2000). Esta técnica ordena a população de acordo com a relação de dominância das soluções, construindo uma população classificada em K conjuntos dominantes $\boldsymbol{F}_1$, $\boldsymbol{F}_2$,...,$\boldsymbol{F}_K$. Em cada conjunto $\boldsymbol{F}_i$, todas as soluções são indiferentes (ou seja, ninguém domina ninguém). Na Figura 4.1 mostra-se um exemplo para o caso de um problema de minimização bi-objetivo, onde a população de pontos $\boldsymbol{f}(\boldsymbol{P}_t \cup \boldsymbol{E}_{t\text{-}1})$ no espaço objetivo é classificada em $K = 3$ fronteiras $\boldsymbol{f}(\boldsymbol{F}_1)$, $\boldsymbol{f}(\boldsymbol{F}_2)$ e $\boldsymbol{f}(\boldsymbol{F}_3)$. Observe que os pontos na fronteira $\boldsymbol{f}(\boldsymbol{F}_1)$ são os dominantes (correspondem às soluções mais aptas), e os pontos em $\boldsymbol{f}(\boldsymbol{F}_2)$ dominam os pontos em $\boldsymbol{f}(\boldsymbol{F}_3)$.

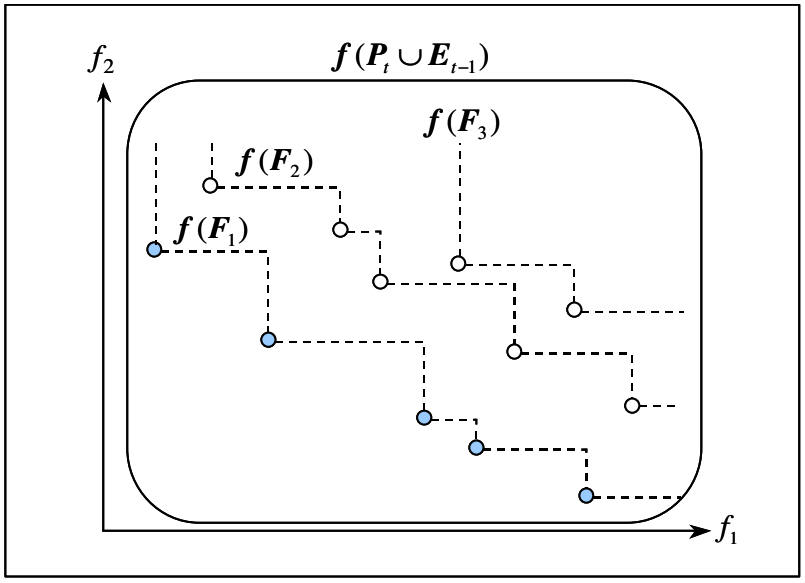


Figura 4.1. Classificação da população combinada.

Deb et al. (2000) propõem um algoritmo para classificar uma população $\boldsymbol{P}$ de soluções (considerando os vetores objetivos) que é implementado em tempo $O(r.N^2)$, onde N é o tamanho da população a ser ordenada e r é o número de objetivos. Este algoritmo é denominado *fast nondominated sort* e é uma versão melhorada da ordenação usada no algoritmo genético de Srinivas e Deb (1995) implementado em tempo $O(r.N^3)$. Neste algoritmo, em primeiro lugar, para cada solução $\boldsymbol{x}$ de uma população $\boldsymbol{P}$ determinam-se: (i) $n_{\boldsymbol{x}}$, o número de soluções que dominam $\boldsymbol{x}$, e (ii) $\boldsymbol{S}_{\boldsymbol{x}}$, o conjunto de soluções dominadas por $\boldsymbol{x}$. Em seguida, identifica-se todas as soluções com $n_{\boldsymbol{x}} = 0$ (soluções não dominadas), e estas são armazenadas em um conjunto $\boldsymbol{F}_1$, que constitui a fronteira atual. Para cada solução $\boldsymbol{y}$ dominada

por alguma solução $\boldsymbol{x}$ da fronteira atual, o valor de n_y é diminuído em uma unidade ($n_y = n_y - 1$). Então, as soluções $\boldsymbol{y}$ que tenham $n_y = 0$ são armazenadas num novo conjunto $\boldsymbol{F}_2$ (fronteira atual). Este processo continua até que todas as soluções possuam $n_y = 0$, isto é, até que todas as soluções sejam armazenadas numa fronteira. A seguir, é apresentado o pseudocódigo do algoritmo *fast nondominated sort* aplicado a uma população $\boldsymbol{P}$.

Algoritmo 4.1. Fast nondominated sort

Entrada: $\boldsymbol{P}$ (*uma população de soluções*)

Saída: $\boldsymbol{F}_1, \boldsymbol{F}_2, ..., \boldsymbol{F}_K$ (*população classificada em K sub-populações*)

1) Para cada $\boldsymbol{x} \in \boldsymbol{P}$ faça

   Para cada $\boldsymbol{y} \in \boldsymbol{P}$ faça

   Se $\boldsymbol{x}$ domina $\boldsymbol{y}$ então $\boldsymbol{S}_x = \boldsymbol{S}_x \cup \{\boldsymbol{y}\}$ ($\boldsymbol{S}_x$ é o conjunto de soluções dominadas por $\boldsymbol{x}$).

   Senão, se $\boldsymbol{y}$ domina $\boldsymbol{x}$ então $n_x = n_x + 1$.

   Se $n_x = 0$ (n_x é o número de soluções que dominam $\boldsymbol{x}$) então $\boldsymbol{F}_1 = \boldsymbol{F}_1 \cup \{\boldsymbol{x}\}$.

   Faça $j = 1$.

2) Enquanto $\boldsymbol{F}_j \neq \varnothing$ faça

   Faça $\boldsymbol{F}_{j+1} = \varnothing$.

   Para cada $\boldsymbol{x} \in \boldsymbol{F}_j$ faça

   Para cada $\boldsymbol{y} \in \boldsymbol{S}_x$ faça

   Faça $n_y = n_y - 1$. Se $n_y = 0$ então $\boldsymbol{F}_{j+1} = \boldsymbol{F}_{j+1} \cup \{\boldsymbol{y}\}$.

   Faça $j = j + 1$.

Observe que, a fase 1) do algoritmo *fast nondominated sort* requer tempo $O(r.N^2)$, pois as soluções da população são comparadas duas a duas, e para cada comparação de duas soluções, comparam-se os r objetivos. A fase de 2) do algoritmo, no pior caso (quando cada fronteira contém um único ponto), requer tempo $O(N^2)$. Portanto, a complexidade do algoritmo é $O(r.N^2) + O(N^2) = O(r.N^2)$.

Na metaheurística AGHM é usado o algoritmo *fast nondominated sort* para classificar a população combinada $\boldsymbol{P}_t \cup \boldsymbol{E}_{t\text{-}1}$.

4.1.3. Preservação de Diversidade na População e Cálculo do Fitness

Depois de classificar a população $\boldsymbol{P}_t \cup \boldsymbol{E}_{t\text{-}1}$ em K conjuntos dominantes $\boldsymbol{F}_1$, $\boldsymbol{F}_2$,...,$\boldsymbol{F}_K$, deve-se atribuir um valor de fitness às soluções de cada conjunto. De fato, as soluções de um conjunto $\boldsymbol{F}_k$ possuirão maior potencial de reprodução (fitness) que as soluções de um conjunto $\boldsymbol{F}_j$, $j > k$. Dado que as soluções em um determinado conjunto $\boldsymbol{F}_k$ são indiferentes (possuem o mesmo grau de dominância), é necessário aplicar alguma técnica para diferenciar estas soluções. Então, na metaheurística AGHM o cálculo do fitness é baseado na técnica proposta por Deb et al. (2000) (veja Capítulo 3, seção 3.1.2). Esta técnica, quando aplicada a cada conjunto $\boldsymbol{F}_k$, tenta preservar a diversidade da população $\boldsymbol{P}_t \cup \boldsymbol{E}_{t\text{-}1}$. Para cada solução $\boldsymbol{x}$ pertencente a um conjunto $\boldsymbol{F}_k$, determina-se uma estimativa do número de soluções localizadas em redor de $\boldsymbol{x}$. Esta estimativa é determinada considerando o vetor de objetivos das soluções e é dada pela *distância de crowding* (Algoritmo 3.2, Capítulo 3). Para uma determinada fronteira $\boldsymbol{f}(\boldsymbol{F}_k)$, inicialmente calcula-se a distância de *crowding* (*dist*) do seus pontos $\boldsymbol{z}^1,...,\boldsymbol{z}^{np}$, onde *np* é o número de pontos na fronteira. Os pontos extremos $\boldsymbol{z}^1$ e $\boldsymbol{z}^{np}$, geralmente, terão valores altos de distância de *crowding*. Uma forma de atribuir fitness aos pontos de uma fronteira consiste em "normalizar" as distâncias de *crowding*, considerando as distâncias máxima e mínima. O algoritmo apresentado a seguir calcula as distâncias de *crowding* "normalizadas" de todas as soluções da população classificada $\boldsymbol{P}_t \cup \boldsymbol{E}_{t\text{-}1}$.

Algoritmo 4.2. Calculo da distância Crowding no AGHM

Entrada $\boldsymbol{F}_1$, $\boldsymbol{F}_2$,...,$\boldsymbol{F}_K$, (*K sub-populações*)

Saída *dist*($\boldsymbol{z}$), $\forall \boldsymbol{z} \in \boldsymbol{f}(\boldsymbol{F}_k)$, $k = 1,...,K$ (*distância de crowding no espaço objetivo*)

1) Para cada conjunto $\boldsymbol{F}_k$ faça os seguintes passos:
2) Calcule a distância de *crowding* (*dist*) de todos os pontos em $\boldsymbol{f}(\boldsymbol{F}_k)$. Use o Algoritmo 3.2 (de Deb et al., 2000) sob $\boldsymbol{F}_k$. Note que, os pontos extremos da fronteira $\boldsymbol{f}(\boldsymbol{F}_k)$ terão como distâncias de *crowding* um valor *v* arbitrariamente grande.

3) Determine a distância máxima $Md < v$ e a distância mínima $md \neq 0$ na fronteira $\boldsymbol{f}(\boldsymbol{F}_k)$:

Se $np = |\boldsymbol{F}_k| \leq 2$ então faça $Md = 1$ e $md = 1$.

Senão $Md = \max\{dist(\boldsymbol{z}^j) < v \ : \ \forall \boldsymbol{z}^j \in \boldsymbol{f}(\boldsymbol{F}_k)\}$ e

$$md = \min\{dist(\boldsymbol{z}^j) \neq 0 \ : \forall \boldsymbol{z}^j \in \boldsymbol{f}(\boldsymbol{F}_k)\}.$$

4) Para cada ponto $\boldsymbol{z}^i \in \boldsymbol{f}(\boldsymbol{F}_k)$ faça

Se $dist(\boldsymbol{z}^i) \geq v$, então $dist(\boldsymbol{z}^i) = (dist(\boldsymbol{z}^i) - v) + (Md + md)/md$.

Senão, $dist(\boldsymbol{z}^i) = dist(\boldsymbol{z}^i)/md$.

Na Figura 4.2 mostra-se um exemplo das distâncias de *crowding* de toda a população, obtidas através do Algoritmo 3.2 proposto por Deb et al. (2000). Na Figura 4.3, estão as distâncias de *crowding* obtidas pelo Algoritmo 4.2, isto é, "normalizando" as distâncias mostradas na Figura 4.2. Note que para o caso bi-objetivo, os pontos extremos de qualquer fronteira sempre possuirão a mesma distância de *crowding*: (*Md* + *md*)/*md*. Note também que, os pontos mais dispersos (ou isolados), numa fronteira, sempre possuirão maiores valores da distância *crowding*.

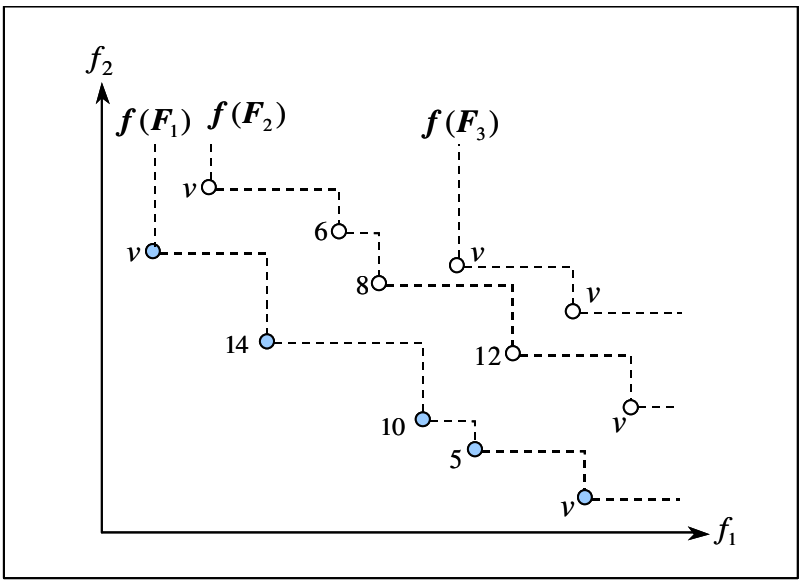


Figura 4.2. Distâncias *crowding* calculadas pelo Algoritmo 3.2.

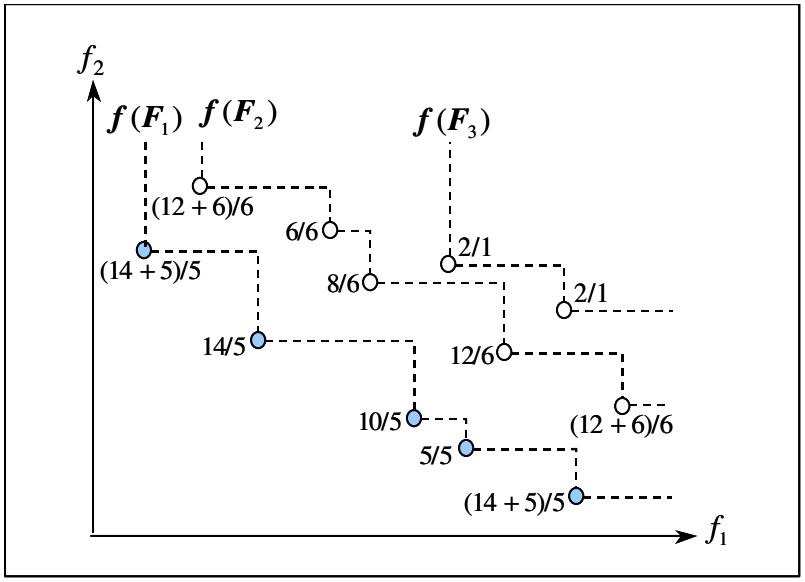


Figura 4.3. Distâncias *crowding* calculadas pelo Algoritmo 4.2.

Após o cálculo da distância *crowding* de todos os pontos da população $\boldsymbol{P}_t \cup \boldsymbol{E}_{t\text{-}1}$, calcula-se o valor do fitness de cada solução. Isto é feito da seguinte forma:

Algoritmo 4.3. Cálculo do fitness no AGHM

1) Faça $maxfitness_{K+1} = 0$, (máximo valor de fitness na fronteira K+1).

2) Para $j = K$ diminuindo até 1 faça

   2.1) Para cada solução $\boldsymbol{x} \in \boldsymbol{F}_j$ faça

   $$fitness(\boldsymbol{x}) = maxfitness_{j+1} + dist(\boldsymbol{f}(\boldsymbol{x})).$$

   2.2) $maxfitness_j = \max\{fitness(\boldsymbol{y}): \boldsymbol{y} \in \boldsymbol{F}_j\}$ é o máximo valor de fitness de $\boldsymbol{F}_j$.

A Figura 4.4 mostra os valores de fitness calculados a partir das distâncias *crowding* mostradas na Figura 4.3. O valor do fitness de uma solução depende da fronteira à qual ela pertence e da sua distância *crowding*. As soluções em $\boldsymbol{F}_k$ sempre possuirão maior potencial de reprodução que qualquer solução em $\boldsymbol{F}_{k+1}$. Note que, para problemas de maximização e minimização o fitness é maximizado, isto é, altos valores de fitness correspondem a altas probabilidades de reprodução.

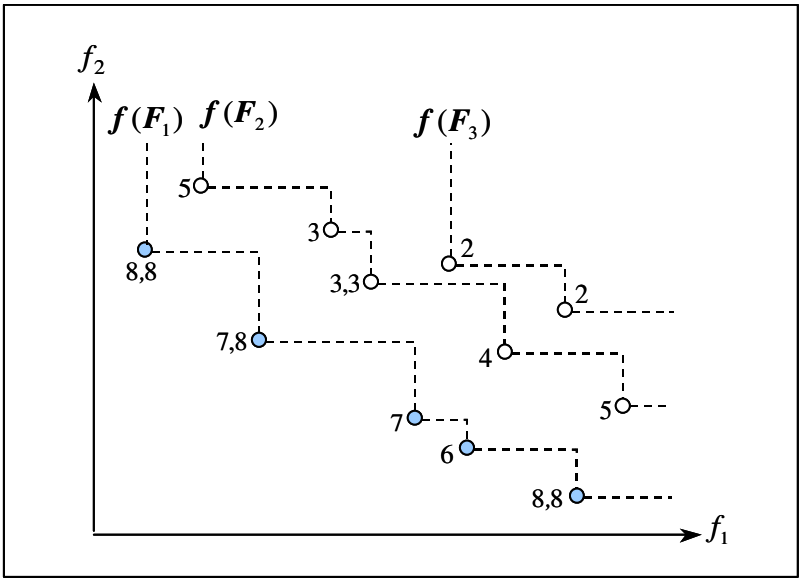


Figura 4.4. Valores de fitness no AGHM.

Após calcular os valores de fitness de todas as soluções, a população $\boldsymbol{P}_t \cup \boldsymbol{E}_{t\text{-}1}$ é reproduzida como em um algoritmo genético convencional.

4.1.4. Seleção e Reprodução

Depois de classificar a população combinada $\boldsymbol{P}_t \cup \boldsymbol{E}_{t\text{-}1}$ e calcular os valores dos fitness das soluções, seleciona-se através de um método de seleção, *N* soluções "aptas" que formarão a população de soluções pais (*mating pool*) $\boldsymbol{SP}_t$. Aplica-se então os operadores genéticos recombinação (ou crossover) e mutação, à população $\boldsymbol{SP}_t$, construindo uma população de *N* novas soluções (filhos) $\boldsymbol{P}_t'$.

4.1.5. Busca Local Multiobjetivo

No algoritmo genético AGHM, aplica-se um procedimento de busca local para melhorar as soluções dominantes da população $\boldsymbol{P}_t'$ gerada pelos operadores genéticos. O procedimento da busca local multiobjetivo BLM descrito no Algoritmo 2.3 do Capítulo 2, é executado a partir do conjunto $\boldsymbol{S}$ de soluções dominantes da população $\boldsymbol{P}_t'$. Para evitar que a

busca local consuma a maior parte do tempo computacional do algoritmo genético, sugere-se acionar a busca local a cada β iterações do algoritmo genético. Como a busca local explora vários caminhos em paralelo, é necessário definir, de forma apropriada, os parâmetros N_{max} (número máximo de caminhos a explorar) e *Niter* (número máximo de iterações da busca local).

Ao finalizar a busca local BLM, deve-se atualizar a população $\boldsymbol{P}_t'$ com o conjunto $\boldsymbol{D}_{BL}$ de soluções obtido por BLM. Em outras palavras, as soluções de $\boldsymbol{D}_{BL}$ devem ser inseridas na população e $|\boldsymbol{D}_{BL}|$ soluções da população são removidas. Como as soluções de $\boldsymbol{D}_{BL}$ foram obtidas a partir de $\boldsymbol{S}$, uma alternativa é substituir $\boldsymbol{S}$ por $\boldsymbol{D}_{BL}$. No entanto, existem três casos para os tamanhos destes conjuntos, a serem analisados:

$$|\boldsymbol{D}_{BL}| = |\boldsymbol{S}|,\ |\boldsymbol{D}_{BL}| > |\boldsymbol{S}| \text{ ou } |\boldsymbol{D}_{BL}| < |\boldsymbol{S}|.$$

Apresenta-se a seguir um algoritmo para atualizar a população $\boldsymbol{P}_t'$ com o conjunto $\boldsymbol{D}_{BL}$ obtido pela busca local.

Algoritmo 4.4. Atualização da população com o conjunto dominante $\boldsymbol{D}_{BL}$

Sejam $\boldsymbol{S} = \{\boldsymbol{x}^i : i = 1, ..., |\boldsymbol{S}|\} \subset \boldsymbol{P}_t'$ o conjunto de soluções dominantes da população $\boldsymbol{P}_t'$ antes da busca local e $\boldsymbol{D}_{BL} = \{\boldsymbol{y}^i : i = 1, ..., |\boldsymbol{D}_{BL}|\}$ o conjunto de soluções dominantes obtido pela busca local.

1) Se $|\boldsymbol{D}_{BL}| = |\boldsymbol{S}|$ e $\boldsymbol{D}_{BL} \neq \boldsymbol{S}$ então faça $\boldsymbol{S} = \boldsymbol{D}_{BL}$.
   Senão
2) Se $|\boldsymbol{D}_{BL}| > |\boldsymbol{S}|$, então
   Para $i = 1$ até $|\boldsymbol{S}|$ faça $\boldsymbol{x}^i = \boldsymbol{y}^i$ (insira $\boldsymbol{y}^i$ em $\boldsymbol{P}_t'$ e descarte $\boldsymbol{x}^i$).
   Para $i = |\boldsymbol{S}| + 1$ até $|\boldsymbol{D}_{BL}|$ faça
   Selecione aleatoriamente uma solução $\boldsymbol{x}'$ de ($\boldsymbol{P}_t' - \boldsymbol{S}$) e faça $\boldsymbol{x}' = \boldsymbol{y}_i$.
   Faça $\boldsymbol{S} = \boldsymbol{S} \cup \{\boldsymbol{x}'\}$ (para que $\boldsymbol{x}'$ não seja selecionada novamente).
3) Senão
   Ordene aleatoriamente as soluções em $\boldsymbol{S}$.
   Para $i = 1$ até $|\boldsymbol{D}_{BL}|$ faça $\boldsymbol{x}^i = \boldsymbol{y}^i$ (note que esta substituição é aleatória).

A Figura 4.5 (a) ilustra um exemplo de atualização da população P_t' no caso onde $|D_{BL}| > |S|$. Note que, $f(S)$ e $f(D_{BL})$ são respectivamente as imagens dos conjuntos S e D_{BL} no espaço objetivo, e $z^i = f(x^i)$, $\forall i$. No exemplo da Figura 4.5 (a), todas as soluções de S são substituídas por soluções de D_{BL}, e as soluções restantes de D_{BL} substituem soluções selecionadas aleatoriamente de ($P_t' - S$). A Figura 4.5 (b) ilustra um exemplo onde $|D_{BL}| < |S|$. Neste caso, $|D_{BL}|$ soluções de S são substituídas pelas soluções de D_{BL}, e as outras soluções de S, permanecem intactas na população.

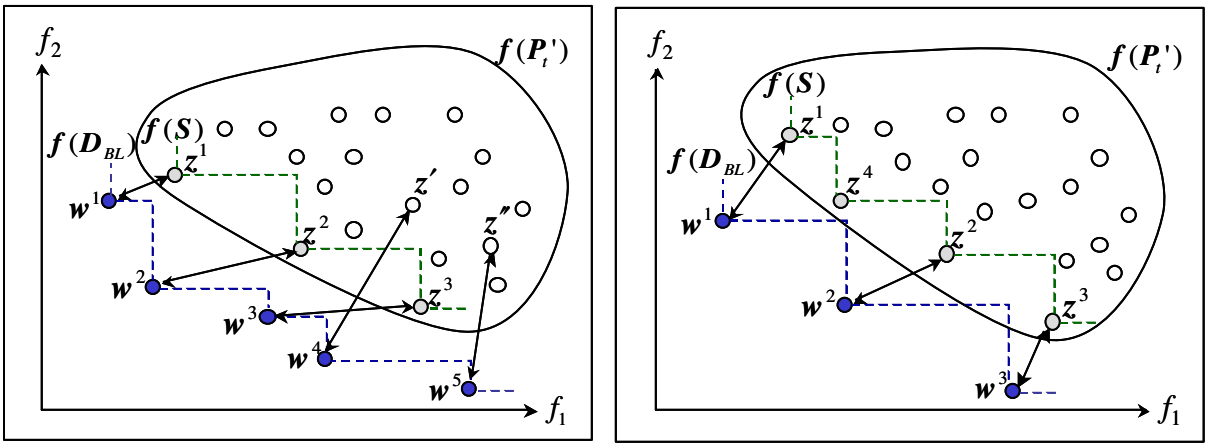


Figura 4.5. Atualização da população com as soluções obtidas pela busca local.

Vale ressaltar que, todos os pontos z^i explorados pela busca local são armazenados numa lista L, para evitar que a busca local explore novamente estes pontos. Já que a lista L pode crescer rapidamente, então ela armazena somente os pontos dominantes daqueles já explorados. Portanto, se numa dada iteração um ponto $z \in f(S)$ (z ponto dominante da população) pertence também à lista L, então z deve ser descartado do conjunto $f(S)$, pois ele já foi explorado em alguma iteração anterior. Note que S é o conjunto de soluções de partida da busca local.

4.2. Pseudocódigo do Algoritmo AGHM

O AGHM pode ser resumido no seguinte algoritmo:

Algoritmo 4.5. AGHM

Entrada: N (*tamanho da população corrente*)
p_R, p_M (*probabilidade de recombinação e mutação*)
MaxE (*número máximo de soluções de elite*)
N_{max} (*número máximo de caminhos a serem exploradas pela busca local*)
Niter (*número máximo de iterações da busca local*)

Saída: $\boldsymbol{D}$ (*conjunto de soluções dominantes*)

1) *Inicialização*: Gere uma população inicial $\boldsymbol{P}_1$ de N soluções (as soluções em $\boldsymbol{P}_1$ podem ser geradas aleatoriamente ou usando alguma heurística construtiva).
   Para cada solução em $\boldsymbol{P}_1$, calcule os valores dos r objetivos.
   Inicialize o conjunto de soluções dominantes: $\boldsymbol{D}_1 = \varnothing$.
   Inicialize o conjunto de soluções de elite: $\boldsymbol{E}_0 = \varnothing$.
   Seja $\boldsymbol{L}$ uma lista contendo as soluções já exploradas pela busca local. Faça $\boldsymbol{L} = \varnothing$ e $t = 1$.
2) *Classificação da População*: Classifique a população $\boldsymbol{P}_t \cup \boldsymbol{E}_{t-1}$ de acordo com a relação de dominância das soluções obtendo uma lista de K conjuntos dominantes $\boldsymbol{F}_1, \boldsymbol{F}_2, ..., \boldsymbol{F}_K$.
3) *Atualização do conjunto dominante*: Faça $\boldsymbol{D}_t$ = soluções dominantes de $(\boldsymbol{D}_t \cup \boldsymbol{F}_1)$.
4) *Calculo dos Fitness*: Para cada $\boldsymbol{x} \in \boldsymbol{F}_i$ ($i = 1, 2, ..., K$) calcule *fitness*($\boldsymbol{x}$).
5) *Seleção*: Seja $\boldsymbol{SP}_t = \varnothing$. Usando um método de seleção, escolha N soluções de $\boldsymbol{P}_t \cup \boldsymbol{E}_{t-1}$ e armazene-os em $\boldsymbol{SP}_t$.
6) *Atualização de conjunto de soluções de Elite*:
   Se $|\boldsymbol{D}_t| > MaxE$, então faça $N_{elite} = MaxE$, senão faça $N_{elite} = |\boldsymbol{D}_t|$.
   Selecione aleatoriamente N_{elite} soluções de $\boldsymbol{D}_t$ e armazene-as em $\boldsymbol{E}_t$.
7) *Recombinação e Mutação*:
   Seja $\boldsymbol{P}'_t = \varnothing$ uma nova população.
   Repita $N/2$ vezes

•Escolha duas soluções (pais) $\boldsymbol{x}^1, \boldsymbol{x}^2 \in \boldsymbol{SP}_t$ e faça $\boldsymbol{SP}_t = \boldsymbol{SP}_t - \{\boldsymbol{x}^1, \boldsymbol{x}^2\}$. Recombine $\boldsymbol{x}^1$ e $\boldsymbol{x}^2$ aplicando um operador de recombinação.

•Adicione a $\boldsymbol{P}'_t$ as novas soluções (filhos) $\boldsymbol{y}^1$ e $\boldsymbol{y}^2$ com probabilidade p_R. Caso contrário, adicione $\boldsymbol{x}^1$ e $\boldsymbol{x}^2$ a $\boldsymbol{P}'_t$.

Para cada solução $\boldsymbol{x} \in \boldsymbol{P}'_t$ faça: Aplique a $\boldsymbol{x}$ um operador de mutação com probabilidade p_M.

8) *Avaliação*: Para cada solução em $\boldsymbol{P}'_t$, calcule os valores dos r objetivos.

9) Se $t = 1$ ou t é um múltiplo de β, então vá para o passo 10.

Senão, faça $t = t + 1$, $\boldsymbol{P}_t = \boldsymbol{P}'_{t-1}$ vá para o passo 12.

10) *Determinação das soluções dominantes de $\boldsymbol{P}'_t$ a serem exploradas pela busca local*:

$\boldsymbol{S}$ = soluções dominantes de $\boldsymbol{P}'_t$.

Para cada solução $\boldsymbol{x} \in \boldsymbol{S}$ faça

Se $\boldsymbol{x} \in \boldsymbol{L}$ (i.e., $\boldsymbol{x}$ já foi explorado pela busca local) então remova $\boldsymbol{x}$ de $\boldsymbol{S}$.

Atualize $\boldsymbol{L}$ com os pontos a serem explorados: $\boldsymbol{L}$ = soluções dominantes de $(\boldsymbol{L} \cup \boldsymbol{S})$ (note que, $\boldsymbol{L}$ armazena somente os pontos dominantes de todos os pontos já explorados pela busca local).

11) *Busca Local Multiobjetivo*:

Execute a busca local multiobjetivo BLM (Algoritmo 2.3) a partir do conjunto $\boldsymbol{S}$ (utilize os parâmetros N_{max} e *Niter*). Seja $\boldsymbol{D}_{BL}$ o conjunto de soluções dominantes obtidos ao final da busca local.

Atualize a população $\boldsymbol{P}'_t$ com $\boldsymbol{D}_{BL}$ (Algoritmo 4.4)

12) *Teste de Parada*:

Se a condição de parada é satisfeita, então faça $\boldsymbol{D}$ = soluções dominantes de $(\boldsymbol{D}_t \cup \boldsymbol{P}'_t)$ e pare. Senão, faça $t = t + 1$, $\boldsymbol{P}_t = \boldsymbol{P}'_{t-1}$ e retorne ao passo 2.

Na Figura 4.6 ilustra-se os passos e etapas do algoritmo genético AGHM.

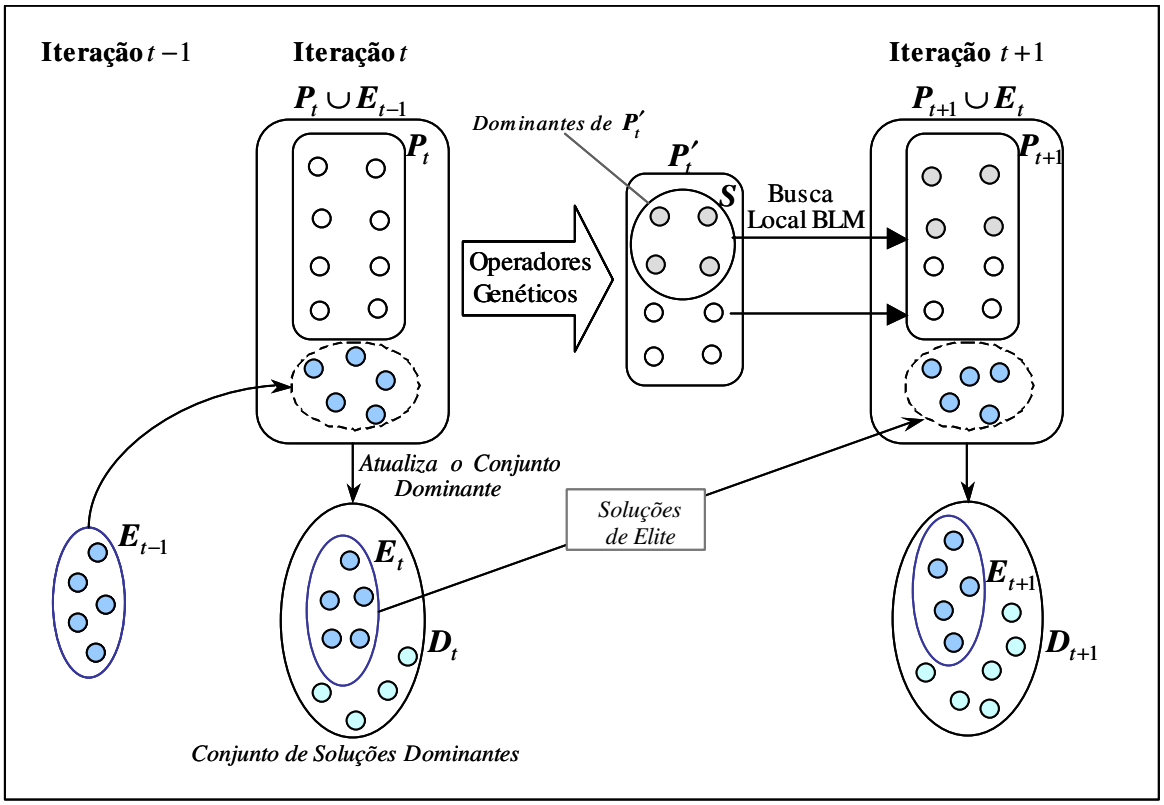


Figura 4.6. Etapas e procedimentos básicos no AGHM.

No algoritmo genético AGHM, para selecionar as soluções reprodutoras (pais), pode ser usado o método *roulette wheel* (Goldberg, 1989) que consiste no sorteio de soluções baseado nos valores dos fitness, isto é, soluções com altos valores de fitness possuem maiores chances de serem escolhidos. Cabe destacar que, na implementação da metaheurística AGHM o fitness de uma solução é obtido a partir do valor normalizado da distância *crowding*, como aqui proposto. A contribuição mais importante apresentada na metaheurística AGHM é a combinação das estratégias de algoritmos genéticos com técnicas de busca local multiobjetivo. Vale lembrar que, na literatura de otimização multiobjetivo, a maioria dos algoritmos genéticos propostos não utilizam estratégias de busca local (Ehrgott and Gandibleux, 2000; Coello, 2000; Van Veldhuizen e Lamount, 2000a; Jones et al., 2002).

4.3. Aplicação do AGHM para Resolver o Problema de Flowshop Multiobjetivo

Nesta seção descreve-se a implementação dos componentes do algoritmo AGHM aplicado ao problema de flowshop multiobjetivo. Considera-se a minimização dos seguintes pares de objetivos: i) (C_{max}, T_{max}) e ii) (C_{max}, T).

De acordo com o pseudocódigo apresentado na seção 4.2, vários componentes devem ser definidos antes da implementação do algoritmo genético. Estes componentes incluem a representação de uma solução (cromossomo), população inicial, avaliação de soluções, seleção, operadores genéticos e critério de parada. Em geral, estes componentes afetam o desempenho do algoritmo e são dependentes do problema a ser resolvido. Apresenta-se a seguir, a descrição de cada um dos componentes do algoritmo AGHM.

4.3.1. Representação de Soluções e População Inicial

Uma seqüência de n tarefas (solução) para o problema de flowshop pode ser representada por um vetor de n elementos.

A população inicial de tamanho N é formada por soluções dominantes obtidas pela heurística multiobjetivo de enumeração parcial (HMEP), descrita no Capítulo 2 (Arroyo e Armentano 2000), e por soluções construídas através do método de geração de soluções diversas para problemas de permutação, sugerido por Glover (1998). Se a heurística HMEP gera $Nh < N$ soluções dominantes, então o método de geração de soluções diversas gera $N - Nh$ soluções para completar a população.

4.3.2. Avaliação de Soluções e Função Fitness

Seja $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$ a função vetorial objetivo de uma solução $\boldsymbol{x}$. O tempo de finalização do processamento da tarefa i sobre a máquina j, $C_{i,j}$, é determinado através da seguinte fórmula recursiva:

$$C_{i,j} = \max\{C_{i,j-1}, C_{i-1,j}\} + p_{ij},\ \ i = 1,...,n,\ \ j = 1,...,m,$$
$$\text{onde}\ \ C_{0,j} = 0,\ \ j = 1,...,m;\ \ C_{i,0} = 0,\ \ i = 1,...,n.$$

Portanto, uma solução $\boldsymbol{x}$ do problema de flowshop bi-objetivo é avaliada pelos valores dos objetivos $f_1(\boldsymbol{x}) = C_{max}(\boldsymbol{x}) = C_{n,m}$ e $f_2(\boldsymbol{x}) = T_{max}(\boldsymbol{x}) = \max_i\{T_i\}$ (ou $f_2(\boldsymbol{x}) = \sum_{i=1}^{n} T_i$), onde $T_i = \max_i\{d_i - C_{i,m}, 0\}$ é o atraso da tarefa i.

A função fitness $fitness(\boldsymbol{x})$ da solução $\boldsymbol{x}$ é calculada através do Algoritmo 4.3 descrito na seção 4.1.3.

4.3.3. Seleção

Segundo os conceitos básicos de algoritmos genéticos, a probabilidade de seleção de uma solução está relacionada com o valor do fitness e reflete a eficácia da solução em resolver o problema. A estratégia adotada para a seleção de soluções é baseada no princípio da roleta (*roulette wheel*) implementada da seguinte forma:

Dada uma população $\boldsymbol{P}$ de N soluções $\boldsymbol{x}^i$, $i = 1,..., N$, seja $fitness(\boldsymbol{x}^i)$ o valor do fitness da solução $\boldsymbol{x}^i$, $i = 1,...,N$. Seja $I = (I_1,I_2,...,I_N)$ um vetor que armazena os índices das soluções ordenadas aleatoriamente. Para cada solução $\boldsymbol{x}^{I_i}$ calcula-se o valor do fitness cumulativo, $q(\boldsymbol{x}^{I_i}) = \sum_{j=I_1}^{I_i} fitness(\boldsymbol{x}^j)$. A cada seleção de uma solução da população $\boldsymbol{P}$, gera-se um número aleatório $rand \in [1,\ sum]$, onde $sum = \sum_{j=1}^{N} fitness(\boldsymbol{x}^j)$. Se $rand \le q(\boldsymbol{x}^{I_1})$, a solução $\boldsymbol{x}^{I_1}$ é selecionada. Caso contrário, seleciona-se uma solução $\boldsymbol{x}^{I_i}$ tal que $q(\boldsymbol{x}^{I_{i-1}}) < rand \le q(\boldsymbol{x}^{I_i})$, $i = 2,...,N$.

Exemplo:

Dada uma população com $N = 6$ soluções. Os valores dos fitness são $fitness(\boldsymbol{x}^1) = 8$, $fitness(\boldsymbol{x}^2) = 6$, $fitness(\boldsymbol{x}^3) = 1$, $fitness(\boldsymbol{x}^4) = 10$, $fitness(\boldsymbol{x}^5) = 3$ e $fitness(\boldsymbol{x}^6) = 14$. $Sum = 42$. $I = (5, 1, 3, 6, 2, 4)$ é o vetor de índices ordenado aleatoriamente. Os valores de fitness cumulativos são $q(\boldsymbol{x}^5) = 3$, $q(\boldsymbol{x}^1) = 11$, $q(\boldsymbol{x}^3) = 12$, $q(\boldsymbol{x}^6) = 26$, $q(\boldsymbol{x}^2) = 32$ e $q(\boldsymbol{x}^4) = 42$. Seja

rand = 19 um número aleatório ∈ [1, 42], onde $q(\boldsymbol{x}^3) < rand \leq q(\boldsymbol{x}^6)$, portanto, a solução x_6 é selecionada.

4.3.4. Operadores Genéticos

Os operadores genéticos são fatores fundamentais no processo evolutivo. Os operadores incorporados na implementação do AGHM aplicado ao problema de flowshop foram *recombinação e mutação*.

Operador de Recombinação

O operador de recombinação é o elemento principal de um algoritmo genético para a geração de novas soluções de boa qualidade. Selecionam-se duas soluções da população de soluções pais. A partir destas duas soluções pais, são geradas duas soluções filhos herdando qualidades de ambos pais.

Três operadores diferentes de recombinação foram testados para o problema de flowshop.

1. *Order crossover* (OX) (Goldberg, 1989): Após selecionar as duas soluções pais, um fragmento do cromossomo de um dos pais é copiado no filho 1 e um fragmento do cromossomo do outro pai é copiado no filho 2. A seleção do fragmento é feita aleatoriamente. As posições vazias do filho 1 e 2 são preenchidas seqüencialmente de acordo com o cromossomo do outro pai e seguindo a seqüência das tarefas. Um exemplo deste operador é mostrado na Figura 4.7.

2. *Crossover dois pontos* (2PX) (Goldberg, 1989): Para gerar duas soluções filhos a partir das duas soluções pais, inicialmente, seleciona-se arbitrariamente dois pontos p_1 e p_2 ($1 \leq p_1 < p_2 \leq n$). As primeiras p_1 tarefas e as ultimas p_2 tarefas do pai 1 (pai 2) são copiados no filho 1 (filho 2). Seguindo a seqüência das tarefas, o fragmento $<p_1, p_2>$ do filho 1 e 2 são preenchidas

seqüencialmente de acordo com o cromossomo do pai 2 e 1, respectivamente. Um exemplo do operador 2PX é mostrado na Figura 4.8.

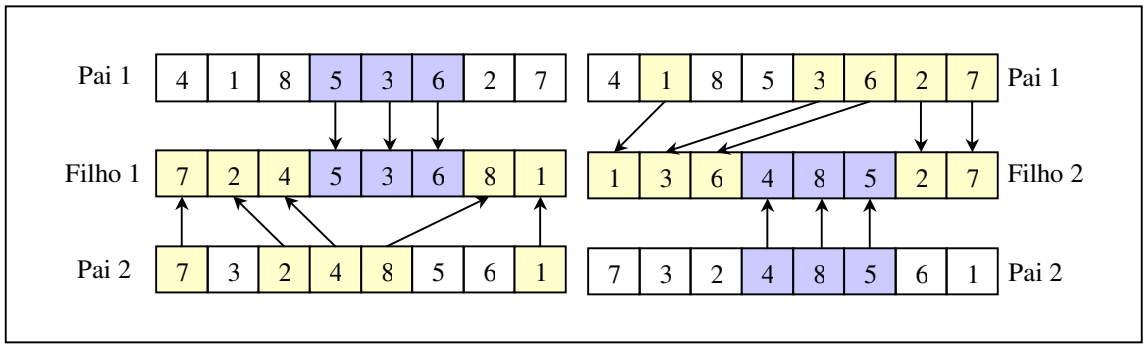


Figura 4.7. Order Crossover.

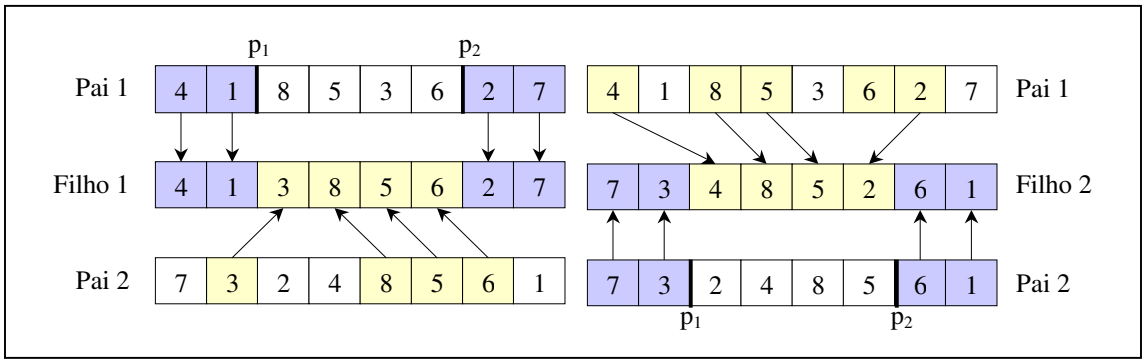


Figura 4.8. Crossover dois pontos.

3. *Recombinação por precedência de tarefas* (RPT) (Michalewicz, 1996): Este operador pode ser mais bem explicado mediante um exemplo ilustrativo. Considere $\boldsymbol{x}$ = (4, 1, 8, 5, 3, 6, 2, 7) e $\boldsymbol{y}$ = (7, 3, 2, 4, 8, 5, 6, 1) as duas soluções pais a serem recombinadas. A Tabela 4.1 mostra a precedência de cada tarefa na solução $\boldsymbol{x}$ e $\boldsymbol{y}$.

Tabela 4.1. Precedência de cada tarefa nas soluções *x* e *y* a serem recombinadas

Tarefa→	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Solução $\boldsymbol{x}$	4	6	5	-	8	3	2	1
Solução $\boldsymbol{y}$	6	3	7	2	8	5	-	4

A combinação é feita da seguinte maneira. A partir da Tabela 4.1, são definidos os seguintes conjuntos: Pred(1) = {4, 6}, Pred(2) = {6, 3}, Pred(3) = {5, 7}, Pred(4) = {2}, Pred(5) = {8}, Pred(6) = {3, 5}, Pred(7) = {2} e Pred(8) = {1, 4}. Seja J = {1, 2, 3, 4, 5, 6, 7, 8} o conjunto de tarefas não selecionadas. Para gerar o primeiro filho f_1, inicialmente escolhe-se aleatoriamente uma tarefa de J para ocupar a ultima posição em f_1. Suponha que a tarefa 3 foi escolhida, então f_1 = (_ , ... , _ , 3). Seja *last* = 3. Excluir a tarefa *last* de J (J = J−{*last*}). Para cada j∈J, se *last*∈Pred(j), faça Pred(j) = Pred(j)−{*last*}. Então, J = {1, 2, 4, 5, 6, 7, 8}, Pred(1) = {4, 6}, Pred(2) = {6}, Pred(3) = {5, 7}, Pred(4) = {2}, Pred(5) = {8}, Pred(6) = {5}, Pred(7) = {2} e Pred(8) = {1, 4}.

A próxima tarefa a ser inserido em f_1 é escolhida entre os predecessores da tarefa *last*. Selecione aleatoriamente uma tarefa de Pred(*last*) (Se Pred(*last*) = ∅, escolha aleatoriamente uma tarefa de J). Suponha que a tarefa 5 foi escolhida, então f_1 = (_ , ... , _ , 5, 3), e *last* = 5. Este processo prossegue até que f_1 seja completado (i.e., até que J = ∅). Uma possível nova solução gerada pode ser f_1 = (1, 7, 6, 2, 4, 8, 5, 3). O outro filho f_2 é gerada de maneira similar, selecionando inicialmente uma tarefa aleatória para ocupar a ultima posição.

Operador de Mutação

O operador de mutação é um procedimento dedicado à manutenção da diversidade da população. Na aplicação ao problema de flowshop, foi implementada a mutação por inserção (Ishibuchi e Murata, 1998). Este operador gera uma nova solução, selecionando aleatoriamente uma tarefa da solução corrente e inserindo-a em uma posição selecionada também aleatoriamente. Um exemplo deste operador é mostrado na Figura 4.9.

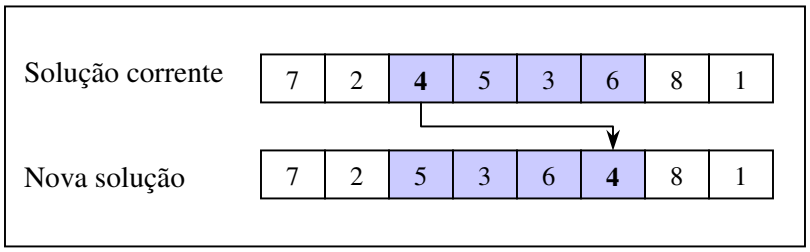


Figura 4.9. Mutação por inserção.

4.3.5. Busca Local e Estratégia de Geração de Vizinhança

Após aplicar os operadores genéticos, executa-se o processo de busca local multiobjetivo descrito no Capítulo 2. Com o objetivo de gerar novas soluções dominantes, a busca local parte do conjunto (***S***) das soluções dominantes da população. As soluções (seqüências) vizinhas de uma solução ***x***, são geradas através da "inserção de tarefas". Uma tarefa *i*, localizada na posição *k* da seqüência, é inserida em todas as outras posições da seqüência; esta estratégia gera uma vizinhança de tamanho $(n\text{-}1)^2$. É claro que, a avaliação de todas as soluções geradas pela vizinhança de inserção é custosa. Com o propósito de evitar que a busca local consuma a maior parte do tempo computacional do algoritmo genético, aplica-se uma estratégia simples de redução de vizinhança similar ao proposto por Armentano e Ronconi (1999). Através de testes computacionais, observou-se que nas primeiras iterações do algoritmo genético e da busca local, as inserções de uma tarefa *i* são realizadas em posições mais distantes, e quando se aproxima de ótimos locais, as inserções são realizadas em posições próximas da tarefa *i*. Devido a este comportamento, definiu-se uma expressão para determinar a distância máxima de inserção, que apresentou bons resultados:

$$Distância = \lceil (n-1)/(\, iter - 1 + I\,) \rceil, \quad (\lceil a \rceil : \text{menor inteiro} \geq a),$$

onde *n* é o tamanho da seqüência, *iter* é a iteração atual da busca local, e

$$I = \begin{cases} 1, & \text{se } \gamma = 0 \\ 0, & \text{caso contrário} \end{cases}$$

onde γ o resto da divisão do número máximo de iterações da busca local (*Niter*) por *iter*: γ = *Niter* módulo *iter*.

Por exemplo, se *Niter* é fixado em 12, a distância de inserção na primeira iteração da busca local é *n*-1, ou seja, são realizadas inserções de uma tarefa em toda as posições da seqüência, avaliando-se toda a vizinhança. Nas iterações 2 e 3 a distância máxima de inserção é diminuída para $\lceil (n\text{-}1)/2 \rceil$ e $\lceil (n\text{-}1)/3 \rceil$, respectivamente. Para as iterações 4 e 5 esta distância é $\lceil (n\text{-}1)/4 \rceil$, e para as iterações subseqüentes a distância máxima de inserção é $\lceil (n\text{-}1)/5 \rceil$.

4.3.6. Critério de Parada

O critério de parada escolhido no algoritmo AGHM aplicado ao problema de flowshop é a avaliação de um número máximo de soluções. No problema de flowshop multiobjetivo observou-se que, quando se incrementa o número de tarefas, cresce o número de soluções Pareto-ótimas, tornando-se o problema mais difícil de resolver. Depois de realizar vários testes computacionais, o número máximo de soluções a serem avaliados pelo algoritmo AGHM foi fixado em $(800+10n)n^2$, onde n é o número de tarefas. Esta expressão é motivada pelo tamanho da vizinhança de inserção, de ordem quadrática. Por exemplo, para uma instância com $n = 20$ tarefas são avaliadas no máximo $1000n^2$ soluções, e para uma instância com $n = 80$ tarefas são avaliadas no máximo $1600n^2$ soluções. O número total de soluções para uma instância com $n = 20$, é dado por $n! = 2{,}43\times10^{18}$.

4.3.7. Determinação de Parâmetros

Um conjunto de parâmetros foi testado para o algoritmo AGHM aplicado ao problema de flowshop multiobjetivo. Na Tabela 2 mostra-se a combinação de parâmetros que geraram os melhores resultados. OX, 2PX e RPT são os três operadores de recombinação testados para o problema. Note que quando é usado o operador de recombinação RPT, a probabilidade da mutação é muito pequena. Isto ocorre porque o RPT gera soluções diversificadas mesmo quando são combinadas duas soluções pais iguais. Quando se combina duas soluções pais parecidas (ou iguais) através dos operadores OX ou 2PX, as soluções filhos, geralmente, são iguais aos pais e portanto, é necessário que estas soluções filhos sofram mutações com probabilidades maiores.

Tabela 4.2. Parâmetros da implementação

N	Probabilidades de Recombinação e Mutação			$MaxE$	β	N_{max}	$Niter$
	OX	2PX	RPT				
100	$P_R = 0{,}9$ $p_M = 0{,}3$	$p_R = 0{,}9$ $p_M = 0{,}3$	$p_R = 0{,}8$ $p_M = 0{,}05$	20	5	6	12

N = Tamanho da população.
$MaxE$ = Número máximo de soluções de elite.
β = Ativação da busca local (a cada $\beta = 5$ iterações).
N_{max} = número máximo de caminhos a serem exploradas pela busca local.
$Niter$ = número máximo de iterações da busca local.

É importante ressaltar que, o desempenho da metaheurística AGHM foi semelhante para os três operadores de recombinação. O AGHM com o operador OX teve um comportamento ligeiramente melhor. Portanto, na seção seguinte apresentam-se os resultados computacionais da metaheurística AGHM implementado com o operador de recombinação OX.

4.4. Resultados Computacionais sobre o Problema de Flowshop Multiobjetivo

Nesta seção apresenta-se a análise dos resultados obtidos pela metaheurística AGHM aplicada ao problema de flowshop, minimizando os seguintes pares de objetivos: i) (C_{max}, T_{max}) e ii) (C_{max}, T). A metaheurística AGHM é testada nos três conjuntos de instâncias que foram usados para testar a heurística construtiva de enumeração parcial HMEP (veja Capítulo 2, Seção 2.1.2.1). Os métodos usados para avaliar o desempenho da metaheurística AGHM são os mesmos usados na avaliação da heurística HMEP. Estes métodos são descritos no Capítulo 1.4.6.

Para o caso de duas máquinas, o algoritmo genético AGHM é comparado com os algoritmos Branch-and-Bound propostos, respectivamente, por Daniels e Chambers (1990) e Liao et al. (1997) (veja Apêndice B), e para o caso de mais de duas máquinas a metaheurística é comparada com enumeração completa.

Para instâncias de grande porte a metaheurística proposta é comparada com dois algoritmos genéticos multiobjetivos da literatura. O primeiro algoritmo genético considerado é o SPEA proposto por Zitzler e Thiele (1999) (Algoritmo 3.7). Zitzler e Thiele mostraram que o desempenho do SPEA é superior ao outros quatro algoritmos genéticos, quando aplicados aos problemas multiobjetivos da mochila 0/1 e do caixeiro viajante. A segunda metaheurística considerada é o algoritmo genético híbrido proposto por Ishibushi e Murata (1998) denotado por AGIM (Algoritmo 3.8). O AGIM é uma das primeiras metaheurísticas que combina características dos algoritmos genéticos com técnicas de busca local. Além disso, o AGIM é uma metaheurística não baseada no conceito de dominância de Pareto, onde o fitness das soluções é calculado pela soma ponderada dos objetivos.

Na implementação das metaheurísticas SPEA e AGIM, a população inicial é gerada de forma análoga ao algoritmo AGHM, e utilizou-se os operadores de recombinação OX e de mutação por inserção. Os parâmetros usados em SPEA e AGIM foram as seguintes:

Tamanho da população $N = 100$;

Probabilidade de recombinação $p_R = 0{,}8$;

Probabilidade de mutação $p_M = 0{,}3$.

Na implementação da metaheurística SPEA, não é considerado o parâmetro $\overline{N}$ (tamanho máximo da população de soluções dominantes), ou seja, guarda-se todas as soluções dominantes encontradas pelo método.

Na implementação da metaheurística AGIM, além dos parâmetros N, p_R e p_M, precisa-se definir mais dois parâmetros: o número máximo de soluções de elite (soluções dominantes) N_{elite} a serem adicionadas à população antes de executar a busca local e, o número de soluções vizinhas avaliadas numa iteração da busca local, h. Ishibushi e Murata (1998), na aplicação da metaheurística AGIM ao problema de flowshop para a minimização de (C_{max}, T_{max}), consideram $N_{elite} = 3$ e $h = 2$ para uma instância por eles testadas com 5 máquinas e 10 tarefas. Na nossa implementação do AGIM considerou-se $N_{elite} = \min\{20, |\boldsymbol{D}|\}$ e $h = n/4$, onde $\boldsymbol{D}$ é conjunto de soluções dominantes encontrado até o momento e n é o número de tarefas. Vale lembrar que, na implementação da busca local da metaheurística AGIM usou-se a vizinhança de inserção.

As metaheurísticas SPEA e AGIM, para o problema de flowshop multiobjetivo, foram implementadas na linguajem de programação C++. Procedimentos gerais da implementação do SPEA são fornecidos na seguinte página de internet: http://www.tik.ee.ethz.ch/~zitzler. Os testes computacionais foram executados em uma estação de trabalho SUN Ultra 60.

4.4.1. Resultados Computacionais para $f = (C_{max}, T_{max})$

Nesta seção apresentam-se os resultados computacionais obtidos na solução do problema de flowshop, considerando o par de objetivos (C_{max}, T_{max}). O desempenho das metaheurísticas implementada foi testado sobre os seguintes conjuntos de instancias, geradas como descrito no Capítulo 2, seção 2.1.2.1.

- ***Conjunto 1***: n = 10, 11; m = 5, 10; 20 matrizes de tempo de processamento; 4 cenários de datas de entrega ; totalizando 4×20×4 = 320 instâncias.
- ***Conjunto 2***: n = 10, 15, 20, 25; m = 2; 10 matrizes de tempo de processamento; 4 cenários de datas de entrega; totalizando 4×10×4 =160 instâncias.
- ***Conjunto 3***: n = 20, 50, 80; m = 5, 10, 20; 10 matrizes de tempo de processamento; 4 cenários de datas de entrega; totalizando 9×10×4 = 360 instâncias.

Inicialmente, o comportamento do algoritmo genético híbrido AGHM foi comparado com uma versão na qual não é considerada a estratégia de busca local, ou seja, o algoritmo genético "puro" (AGM). Para tal, as metaheurísticas AGHM e AGM foram aplicadas para resolver as 360 instâncias (pertencentes ao Conjunto 3) do problema de flowshop considerando a minimização de C_{max} e T_{max}. As metaheurísticas AGHM e AGM geraram um total de 9012 e 6648 soluções, respectivamente. Foram obtidas 8167 soluções de referências (soluções dominantes dentre todas as soluções encontradas por ambas metaheurísticas) das quais 6480 foram obtidas por AGHM e 2714 por AGM. Comparando os menores valores do makespan (C_{max}) e atraso máximo (T_{max}) obtidos pelas metaheurísticas resulta que, AGHM e AGM obtiveram o menor valor de C_{max} em 304 e 146 instâncias e o menor valor de T_{max} em 319 e 220 instâncias, respectivamente. É notório que a estratégia da busca local multiobjetivo influi muito na eficiência e no comportamento do algoritmo genético.

4.4.1.1. Resultados Para o Conjunto 1 de Instâncias

Os resultados do algoritmo genético proposto AGHM e dos algoritmos da literatura SPEA e AGIM são comparados com os resultados ótimos obtidos pela técnica de enumeração completa. Na Tabela 4.3 apresentam-se, para cada tamanho de problema, o número de soluções eficientes e o número de soluções obtidas pelas metaheurísticas AGHM, SPEA e AGIM associadas com 80 instâncias. Nas 320 instâncias, a enumeração completa obteve 3004 soluções eficientes. As metaheurísticas AGHM, AGIM e SPEA identificaram 2932 (97,60%), 2769 (92,18%) e 2557 (85,12%) soluções eficientes, respectivamente. Além disso, estas metaheurísticas encontraram o conjunto Pareto-ótimo para 264, 193 e 137 instâncias,

respectivamente. Observe que, em cada grupo de instâncias, a metaheurística AGHM foi superior, seguida por AGIM.

Tabela 4.3. Número de soluções encontradas pela Enumeração Completa e pelas metaheurísticas comparadas

Problema	EC	AGHM		AGIM		SPEA	
n×*m*	NSE	NTS	NSEM	NTS	NSEM	NTS	NSEM
10×5	582	572	571	569	563	548	524
10×10	786	775	771	765	753	732	687
11×5	687	681	673	666	631	651	610
11×10	949	929	917	881	822	822	736
Total	3004	2957	2932	2881	2769	2753	2557

NSE: número de soluções eficientes encontradas pelo método exato.
NTS: número total de soluções encontradas pela metaheurística.
NSEM: número de soluções eficientes encontradas pela metaheurística.

Na Figura 4.10 mostra-se a média de número de soluções eficientes, para cada cenário de datas de entrega, obtidas pelos métodos. Observa-se que, a média de número de soluções eficientes encontradas pela enumeração completa (EC) é maior para os cenários 2 e 4, associados a faixas largas de datas de entrega. Além disso, a metaheurística AGHM identificou maior quantidade de soluções eficientes em todos os cenários.

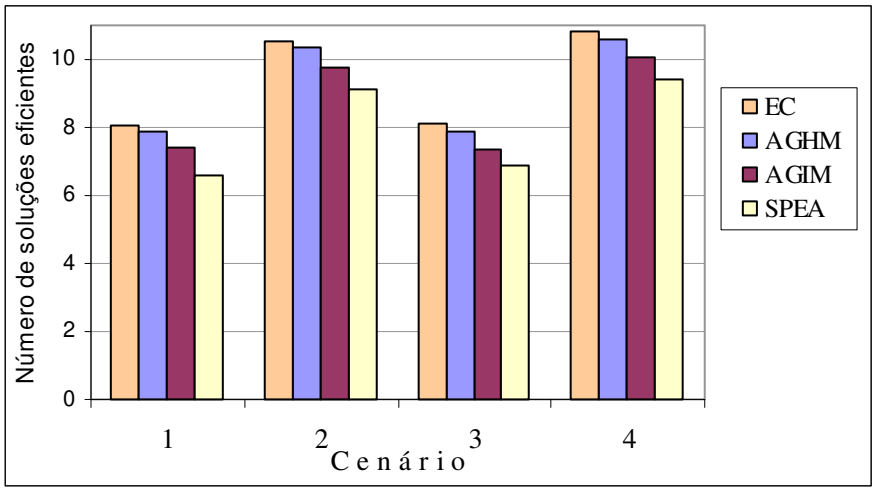


Figura 4.10. Média de número de soluções eficientes por cenário.

Para cada tamanho de problema, a Figura 4.11 ilustra a média do número de soluções eficientes obtidas pela enumeração completa (EC) e pelas metaheurísticas AGHM, AGIM e SPEA. Nota-se que o número de soluções cresce quando o número de tarefas e o número máquinas são incrementados. Nota-se também que a metaheurística AGHM sempre apresenta a maior média de soluções eficientes para cada tamanho de problema.

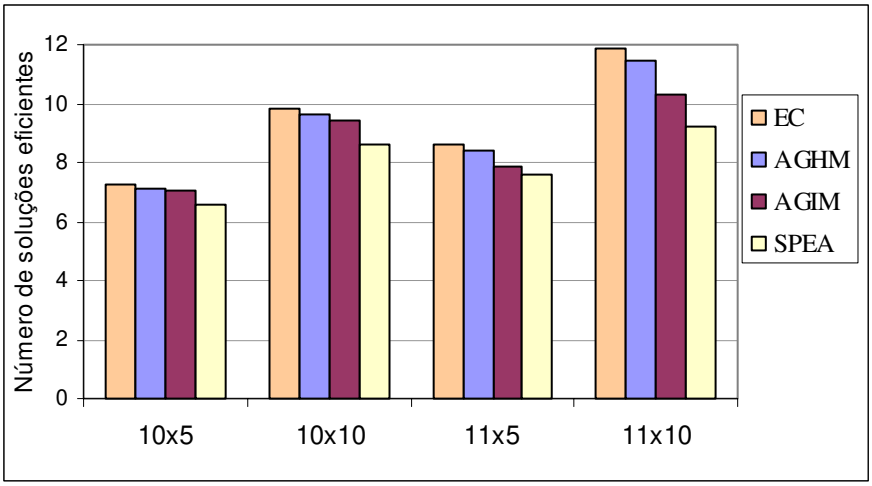


Figura 4.11. Média do número de soluções eficientes por tamanho de problema.

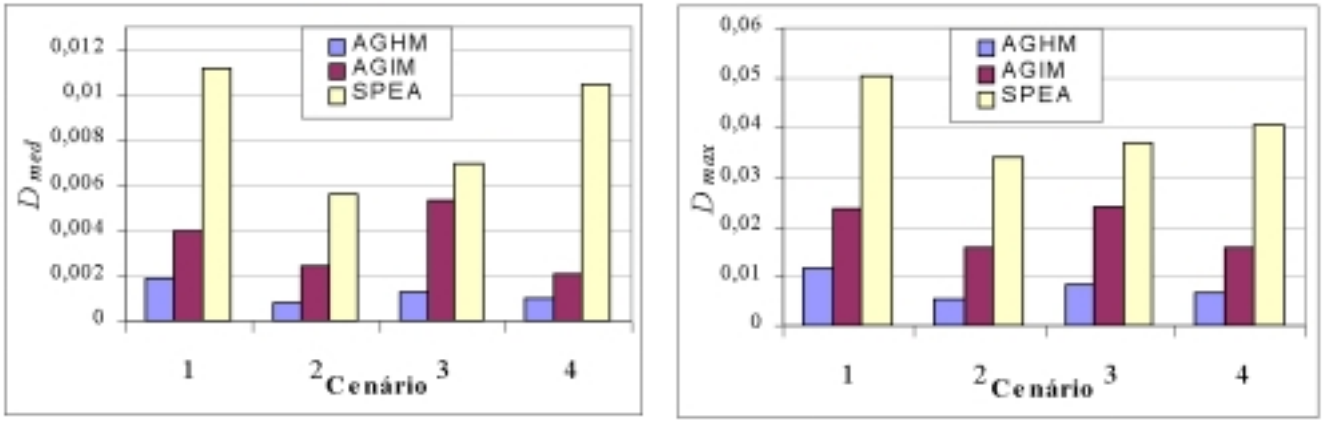
A Tabela 4.4 mostra o desempenho das metaheurísticas de acordo com a medida de distância. Na Tabela 4.5 estão os erros associados com a função utilidade linear proposta por Daniels (1992). Note que o AGHM apresenta melhor desempenho segundo as duas medidas.

Tabela 4.4. Desempenho das metaheurísticas: Medida de distância

Problema	AGHM		AGIM		SPEA	
$n \times m$	D_{med}	D_{max}	D_{med}	D_{max}	D_{med}	D_{max}
10×5	0,0017	0,0077	0,0022	0,0126	0,0062	0,0315
10×10	0,0005	0,0039	0,0014	0,0116	0,0050	0,0331
11×5	0,0012	0,0079	0,0033	0,0182	0,0073	0,0311
11×10	0,0013	0,0122	0,0067	0,0366	0,0154	0,0661
Média	0,0012	0,0079	0,0034	0,0198	0,0085	0,0405

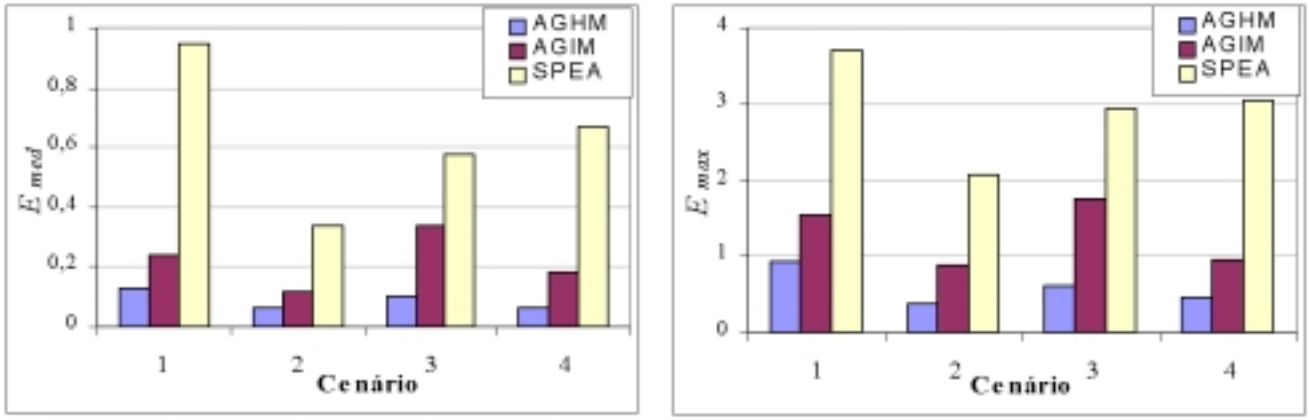
Tabela 4.5. Desempenho das metaheurísticas: Erro de utilidade(%)

Problema	AGHM		AGIM		SPEA	
$n \times m$	E_{med}	E_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×5	0,104	0,754	0,154	0,961	0,426	2,184
10×10	0,021	0,186	0,111	0,638	0,345	2,094
11×5	0,099	0,401	0,137	0,862	0,576	2,227
11×10	0,122	0,992	0,464	2,658	1,177	5,243
Média	0,087	0,583	0,217	1,280	0,631	2,937



(a) (b)

Figura 4.12. Distância média e máxima por cenário de data de entrega.

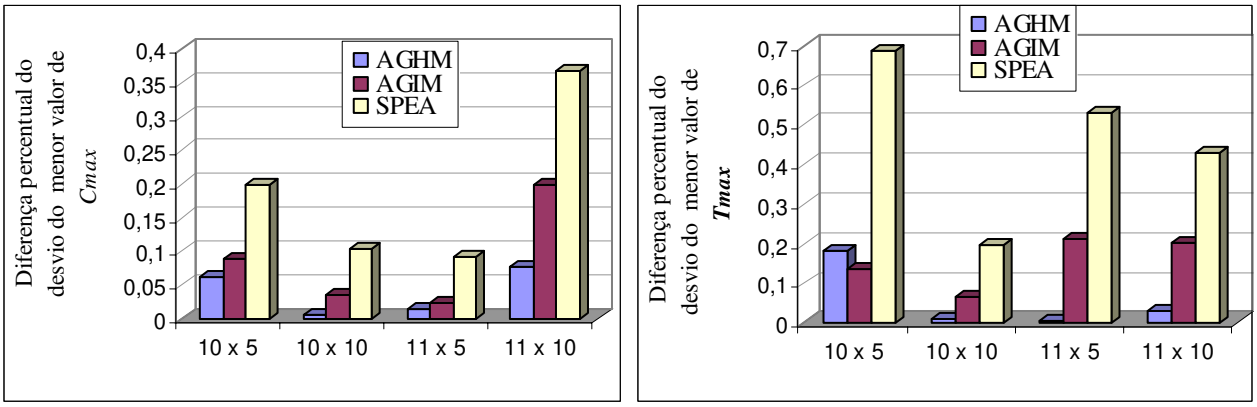


(a) (b)

Figura 4.13. Erro de utilidade médio e máximo por cenário de data de entrega.

Para cada cenário de datas de entrega, as Figuras 4.12 (a) e (b) ilustram os valores médios (sobre 80 instâncias) de D_{med} e D_{max}, enquanto as Figuras 4.13 (a) e (b) ilustram os

valores médios de E_{med} e E_{max}. Observe que a metaheurística AGHM (SPEA) possui melhor (pior) desempenho em todos os cenários. As metaheurística AGHM e AGIM apresentam valores mais altos de distâncias e erros de utilidade para os cenários 1 e 3 que estão associados com pequenas faixas de datas de entrega.



(a) (b)

Figura 4.14. Diferença percentual média do desvio dos menores valores de C_{max} e T_{max} obtidas pelas metaheurísticas em relação aos valores ótimos.

Avaliou-se também a capacidade das metaheurísticas em obter os valores mínimos dos objetivos (pontos extremos da fronteira Pareto-ótima). Para tal, comparam-se os menores valores do makespan (C_{max}) e atraso máximo (T_{max}), obtidos por uma metaheurística, com os respectivos valores ótimos obtidos por enumeração completa. Do total de 320 instâncias, as metaheurísticas AGHM, AGIM e SPEA obtiveram valores ótimos de C_{max} em 295, 273 e 237 instâncias e valores ótimos de T_{max} em 316, 301 e 287 instâncias, respectivamente. Para cada tamanho de problema, as Figuras 4.14 (a) e (b) mostram a diferença percentual média do desvio dos menores valores de C_{max} e T_{max}, obtidas pelas metaheurísticas, em relação aos valores ótimos obtidos pela enumeração completa. Observe que, para cada de tamanho de problema, a metaheurística AGHM obteve, em média, os menores valores de C_{max}. Em problemas com 10 tarefas e 5 máquinas, a metaheurística AGIM obteve, em média, menores valores de T_{max}.

Na Tabela 4.6 são mostrados os tempos computacionais gastos pelas metaheurísticas AGHM, AGIM e SPEA.

Tabela 4.6. Média de tempos computacionais (em segundos)

n×*m*:	10×5	10×10	11×5	11×10
AGHM	0,923	1,134	1,152	1,632
AGIM	0,874	1,127	1,102	1,464
SPEA	0,920	1,143	1,056	1,526

4.4.1.2. Resultados Para o Conjunto 2 de Instâncias

As 160 instâncias deste conjunto foram resolvidas otimamente pelo algoritmo Branch-and-Bound (B&B) desenvolvido por Daniels e Chamber (1990), obtendo 507 soluções eficientes. A Tabela 4.7 mostra, para cada tamanho de problema, o número de soluções obtidas pelo algoritmo B&B e pelas metaheurísticas. As metaheurísticas AGHM, AGIM e SPEA encontraram 504 (99,41%), 491 (96,84%) e 489 (96,45%) soluções eficientes e identificaram exatamente o conjunto Pareto-ótimo para 158, 153 e 152 instâncias, respectivamente. Observe que as metaheurísticas AGHM e SPEA encontraram o conjunto Pareto-ótimo em todos os problemas com 10 e 15 tarefas.

Tabela 4.7. Número de soluções encontradas pelo algoritmo B&B e pelas metaheurísticas comparadas

Problema	B&B	AGHM		AGIM		SPEA	
n×*m*	NSE	NTS	NSEM	NTS	NSEM	NTS	NSEM
10×2	122	122	122	121	120	122	122
15×2	140	140	140	139	138	140	140
20×2	144	142	142	138	133	136	131
25×2	101	101	100	100	100	100	96
Total	507	505	504	498	491	498	489

O desempenho das metaheurísticas de acordo com a medida de distância e erro de utilidade é mostrado nas Tabelas 4.8 e 4.9, respectivamente. Segundo as duas medidas, o AGHM apresenta melhor desempenho. Note que, segundo a medida de erro de utilidade proposta por Daniels (1992), a superioridade da metaheurística AGHM é ainda acentuada. Esta inconsistência das medidas pode ser atribuída à eliminação de pontos que não possuem hiperplano suporte quando é calculado o erro de utilidade.

Tabela 4.8. Desempenho das metaheurísticas: Medida de distância

Problema	AGHM		AGIM		SPEA	
$n \times m$	D_{med}	D_{max}	D_{med}	D_{max}	D_{med}	D_{max}
10×2	0,0	0,0	0,0009	0,0045	0,0	0,0
15×2	0,0	0,0	0,0028	0,005	0,0	0,0
20×2	0,0003	0,0022	0,0078	0,0233	0,0129	0,038
25×2	0,00006	0,0003	0,0005	0,002	0,238	0,353
Média	0,0001	0,0006	0,0030	0,0087	0,0627	0,0978

Tabela 4.9. Desempenho das metaheurísticas: Erro de utilidade (%)

Problema	AGHM		AGIM		SPEA	
$n \times m$	E_{med}	E_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×2	0,0	0,0	0,070	0,309	0,0	0,0
15×2	0,0	0,0	0,098	0,417	0,0	0,0
20×2	0,0	0,0	0,564	1,911	0,680	2,611
25×2	0,00011	0,0068	0,023	0,200	1,376	3,362
Média	0,00003	0,0017	0,189	0,709	0,514	1,493

Nas 160 instâncias testadas, as três metaheurísticas obtiveram os valores ótimos do C_{max}. O AGHM identificou os valores ótimos de T_{max} para todas as instâncias, enquanto AGIM e SPEA identificaram tais valores em 157 e 155 instâncias, respectivamente.

Na Tabela 4.10 são mostrados os tempos computacionais gastos pelo algoritmo Branch and Bound (B&B) e pelas metaheurísticas AGHM, AGIM e SPEA. Note que o tempo do algoritmo B&B é muito mais alto para os problemas com 20 e 25 tarefas e com cenários 2 e 4 de datas de entrega.

Tabela 4.10. Média de tempos computacionais (em segundos)

Problema	Branch and Bound				AGHM	AGIM	SPEA
$n \times m$	Cenário 1	Cenário 2	Cenário 3	Cenário 4			
10×2	0,014	0,026	0,001	0,012	0,918	0,766	0,824
15×2	0,124	0,229	0,183	0,474	1,854	1,624	1,728
20×2	0,436	51,957	0,845	499,171	3,626	3,524	3,482
25×2	0,268	720,605	0,414	3695,813	6,254	6,052	6,125

4.4.1.3. Resultados Para o Conjunto 3 de Instâncias

O desempenho das metaheurísticas AGHM, AGIM e SPEA foi avaliado em instâncias de grande porte (n = 20, 50, 80 e m = 5, 10, 20) pertencentes a este conjunto. Para obter um conjunto de soluções de referência $\boldsymbol{R}$, determinam-se as soluções dominantes dentre todas as soluções encontradas pelas metaheurísticas AGHM, AGIM e SPEA.

Na Tabela 4.11, mostra-se o número de soluções de dominantes no conjunto $\boldsymbol{R}$, o número total de soluções e o numero de soluções dominantes obtidas por cada metaheurística. Claramente, a metaheurística AGHM gerou maior quantidade de soluções dominantes. Para as 360 instâncias testadas, foram encontradas 9291 soluções dominantes que são as soluções de referência. As metaheurísticas AGHM, AGIM, e SPEA obtiveram, respectivamente, 6966, 2021 e 2267 soluções dominantes.

Tabela 4.11. Número de soluções encontradas pelas metaheurísticas comparadas

Problema	$\lvert\boldsymbol{R}\rvert$	AGHM		AGIM		SPEA	
$n\times m$		NTS	NSD	NTS	NSD	NTS	NSD
20×5	511	491	430	447	312	418	257
20×10	892	867	710	674	282	644	218
20×20	1023	1017	810	841	356	779	304
50×5	512	488	388	446	230	395	215
50×10	1197	1137	838	887	156	746	217
50×20	1664	1626	1300	1250	118	1131	249
80×5	380	368	297	328	190	315	217
80×10	953	922	651	740	162	696	187
80×20	2159	2096	1542	1518	215	1500	403
Total	9291	9012	6966	7131	2021	6624	2267

$\lvert\boldsymbol{R}\rvert$: número de soluções de referência.
NTS: número total de soluções obtidas por uma metaheurística..
NSD: número de soluções dominantes obtidas por uma metaheurística.

Na Figura 4.15 mostra-se a média de número de soluções dominantes, para cada cenário de datas de entrega, obtida pelas metaheurísticas. Observa-se que, a média de número de soluções de referência é maior para os cenários 2 e 4, e a metaheurística AGHM identificou maior quantidade de soluções dominantes em todos os cenários. Comparando as metaheurísticas AGIM e SPEA, a primeira obteve maior número de soluções dominantes nos

cenários 1 e 3, no entanto, a segunda metaheurística obteve maior número de soluções dominantes nos cenários 2 e 4.

Para cada tamanho de problema e cada cenário, na Figura 4.16 ilustra-se a média do número de soluções dominantes obtidas pela metaheurística AGHM. Observe que, um número alto de soluções dominantes correspondem a instâncias com cenários 2 e 4 de datas de entrega. O número de soluções dominantes cresce à medida a razão *n*/*m* decresce.

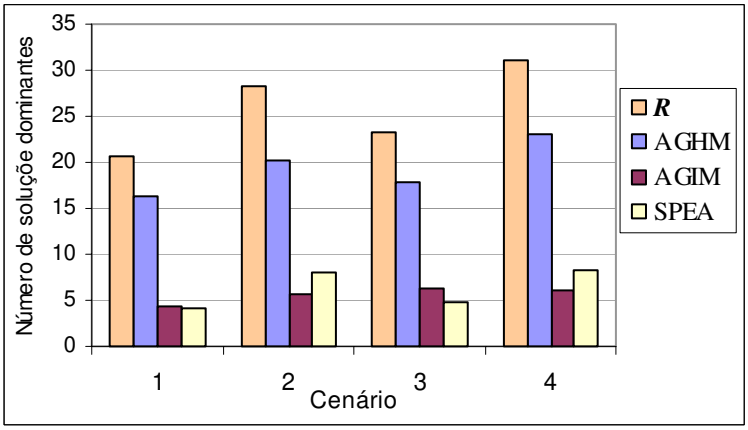


Figura 4.15. Média de número de soluções dominantes por cenário.

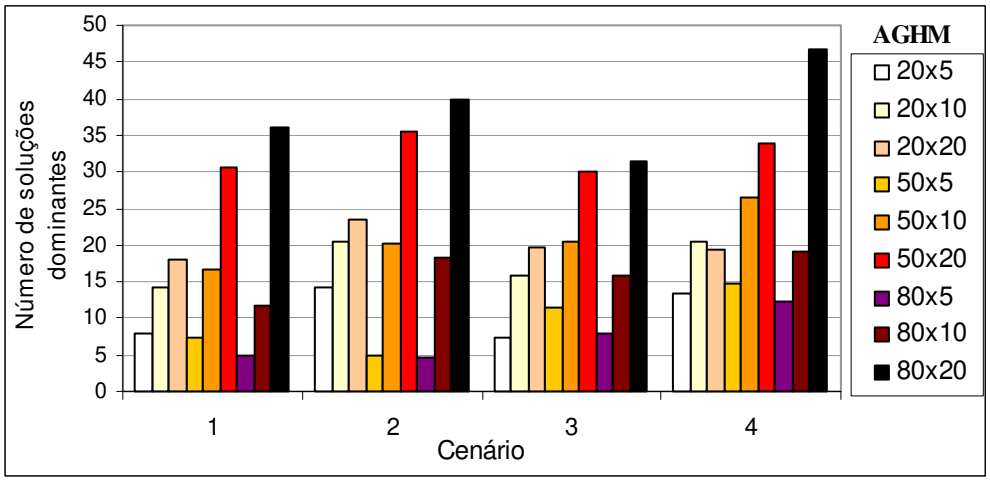


Figura 4.16. Média de número de soluções dominantes por tamanho de instância e por cenário correspondente à metaheurística AGHM.

As Tabelas 4.12 e 4.13 mostram que a metaheurística AGHM possui um melhor desempenho em relação às medidas de distância e erro de utilidade percentual.

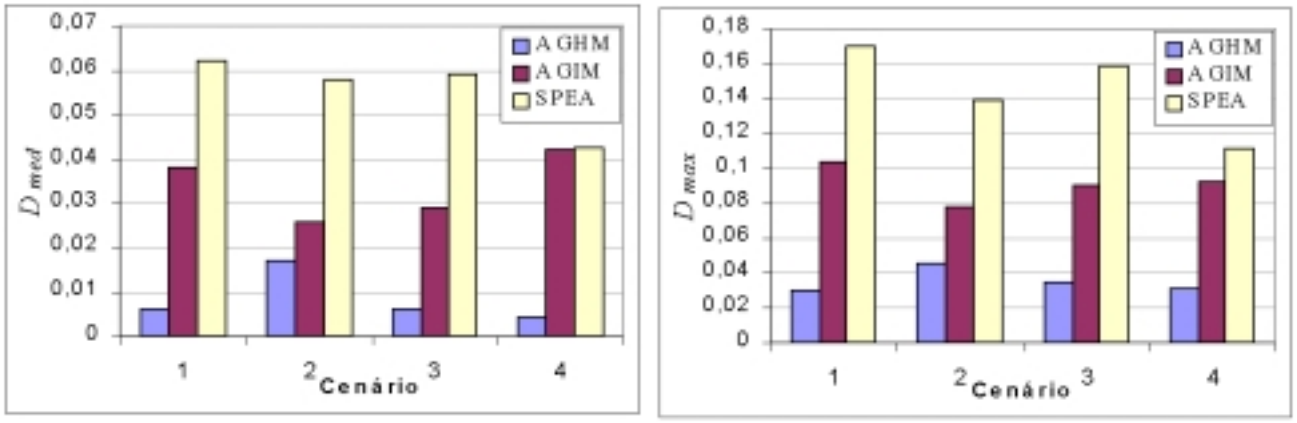
Tabela 4.12. Desempenho das metaheurísticas: Medida de distância

Problema	AGHM		AGIM		SPEA	
$m \times m$	D_{med}	D_{max}	D_{med}	D_{max}	D_{med}	D_{max}
20×5	0,007	0,037	0,016	0,061	0,042	0,134
20×10	0,004	0,036	0,024	0,089	0,039	0,121
20×20	0,006	0,045	0,021	0,079	0,026	0,103
50×5	0,029	0,041	0,012	0,041	0,095	0,176
50×10	0,008	0,043	0,043	0,115	0,058	0,172
50×20	0,006	0,033	0,050	0,145	0,044	0,141
80×5	0,002	0,016	0,052	0,070	0,089	0,158
80×10	0,006	0,033	0,037	0,099	0,055	0,148
80×20	0,005	0,033	0,048	0,135	0,049	0,150
Média	0,007	0,031	0,032	0,086	0,051	0,130

Tabela 4.13. Desempenho das metaheurísticas: Erro de utilidade (%)

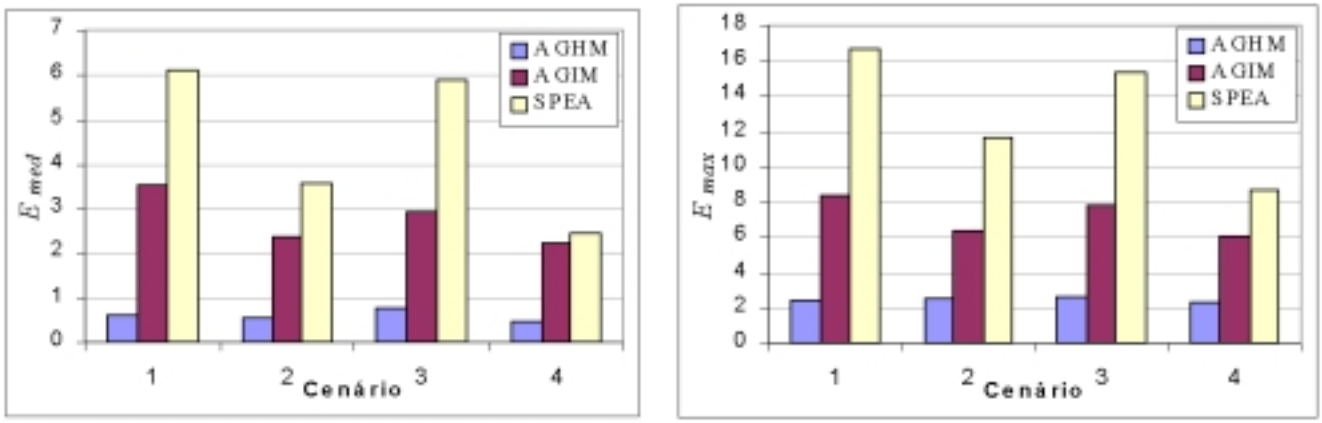
Problema	AGHM		AGIM		SPEA	
$n \times m$	E_{med}	E_{max}	E_{med}	E_{max}	E_{med}	E_{max}
20×5	0,652	3,188	1,073	3,873	3,858	12,560
20×10	0,369	1,731	2,297	6,634	3,901	11,576
20×20	0,873	3,715	1,907	6,476	2,571	9,066
50×5	0,336	1,522	1,113	3,183	5,283	13,481
50×10	1,043	3,667	4,238	10,163	5,840	17,265
50×20	0,453	1,754	5,468	14,884	5,181	14,524
80×5	0,305	1,357	0,670	2,322	2,832	10,181
80×10	0,963	3,376	2,828	7,512	5,278	14,594
80×20	0,455	2,014	5,298	12,301	5,844	14,538
Média	0,605	2,480	2,766	7,483	4,510	13,087

Para cada cenário, as Figuras 4.17 (a) e (b) mostram os valores médios (sobre 90 instâncias) das distâncias D_{med} e D_{max}, e as Figuras 4.18 (a) e (b) mostram os valores médios de E_{med} e E_{max}. A metaheurística AGHM apresenta menores valores da distância e erro de utilidade para todos os cenários.



(a) (b)

Figura 4.17. Distância média e máxima por cenário de data de entrega.



(a) (b)

Figura 4.18. Erro de utilidade percentual médio e máximo por cenário de data de entrega.

Para este conjunto de instâncias, também se comparam os menores valores do makespan (C_{max}) e atraso máximo (T_{max}) obtidos pelas metaheurísticas. Nas 360 instâncias testadas, as metaheurísticas AGHM, AGIM e SPEA obtiveram o menor valor de C_{max} em 279, 160 e 92 instâncias e o menor valor de T_{max} em 327, 202 e 185 instâncias, respectivamente.

Nas avaliações realizadas acima pode ser observado que, a metaheurística SPEA obteve maior número de soluções dominantes que a metaheurística AGIM. No entanto, AGIM possui melhor desempenho em relação às medidas de distância e erro de utilidade. Isto se deve aos seguintes comportamentos destas metaheurísticas: os pontos dominantes obtidas pela metaheurística SPEA, geralmente, estão localizadas numa pequena faixa da fronteira

dominante (***R***) e, os poucos pontos dominantes obtidos pela metaheurística AGIM estão melhor distribuídos sobre a fronteira dominante. Na Figura 4.19 ilustra-se este comportamento para uma instância com 80 tarefas e 10 máquinas. Nesta instância, a fronteira dominante (***R***) é formada por 24 pontos dominantes, dos quais as metaheurísticas SPEA e AGIM encontraram 11 e 6 pontos, respectivamente. Na Tabela 4.14 estão as medidas de distância e erro de utilidade percentual correspondentes às metaheurísticas AGIM e SPEA. Note que, existe uma diferencia considerável entre os erros de utilidade apresentada pelas metaheurísticas AGIM e SPEA. Na Figura 4.20 ilustram-se os pontos que possuem hiperplano suporte (pontos restantes após da eliminação de pontos através da técnica de avaliação de Daniels (1990)). Observe que, AGIM possui 6 pontos de suporte dos quais 2 pontos são de referência, enquanto SPEA possui 8 pontos de suporte dos quais somente um ponto é de referência. Além disso, os pontos de suporte de AGIM estão mais próximos dos pontos de referência.

Tabela 4.14. Medidas de distância e erro de utilidade para o exemplo da Figura 4.19

AGIM				SPEA			
D_{med}	D_{max}	E_{med}	E_{max}	D_{med}	D_{max}	E_{med}	E_{max}
0,0276	0,0602	1,603	6,021	0.0289	0,0874	3,737	8,290

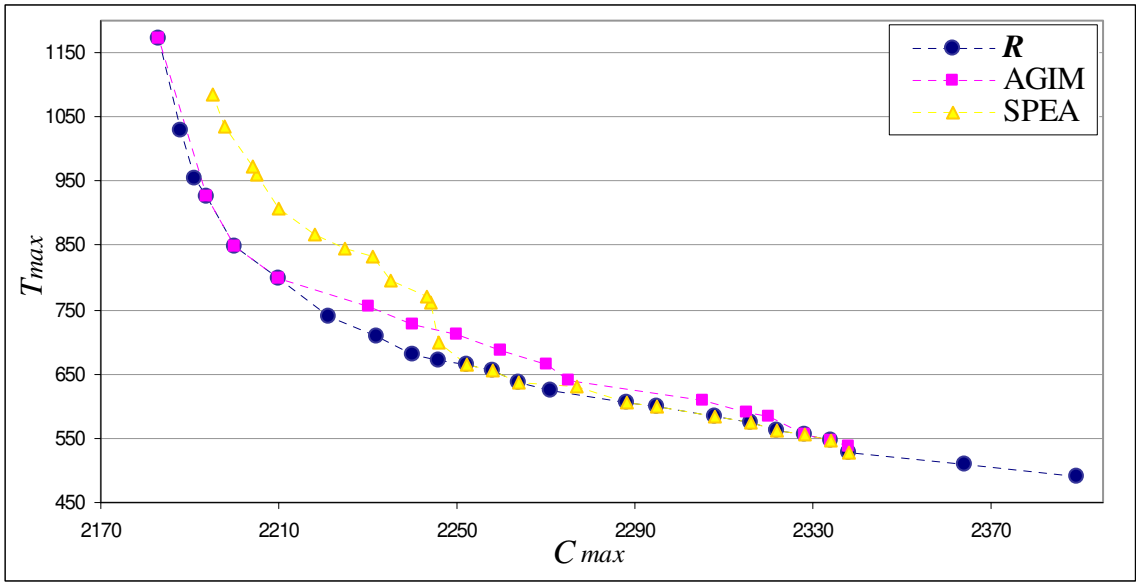


Figura 4.19. Distribuição dos pontos dominantes obtidos pelas metaheurísticas.

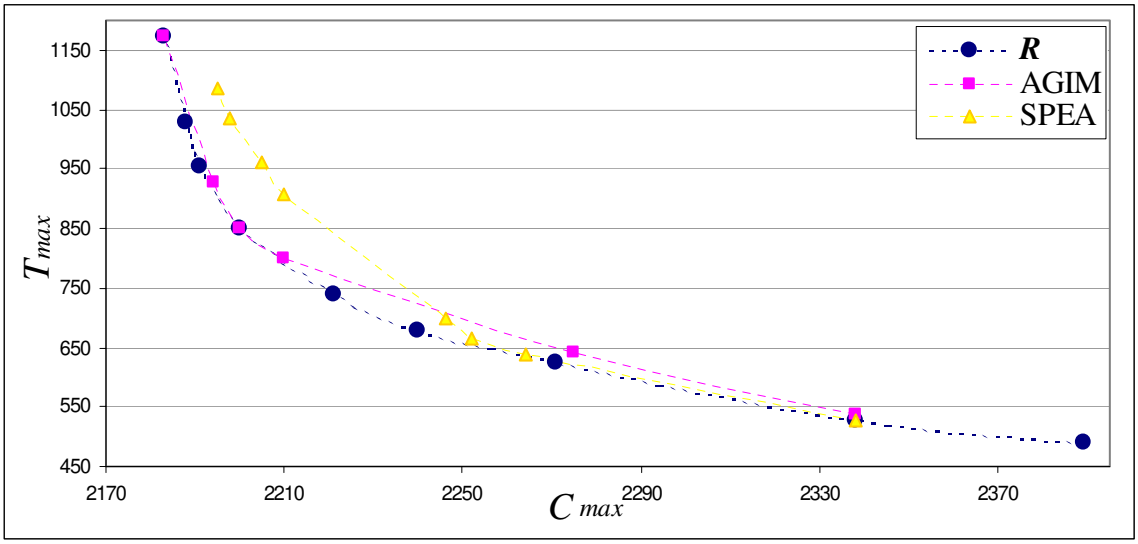


Figura 4.20. Pontos de suporte.

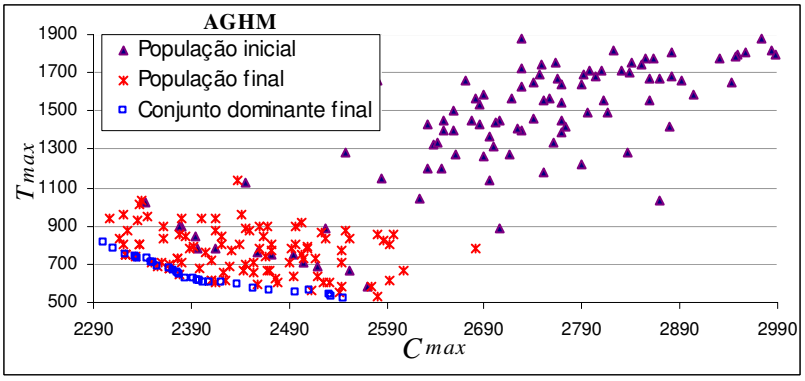
Nas Figuras 4.21 (a), (b) e (c), ilustra-se, para cada metaheurística, a população inicial, a população final e o conjunto de soluções dominantes obtido para uma instância com 20 tarefas e 20 máquinas. Observe que, a população inicial é a mesma para cada metaheurística. A população inicial é formada pelas soluções dominantes obtidas pela heurística construtiva de enumeração parcial HMEP (apresentada no Capítulo 2) e por soluções construídas através do método de geração de soluções diversas para problemas de permutação, sugerida por Glover (1998). Geralmente, os pontos obtidos pela heurística HMEP constituem os pontos dominantes da população inicial. A Figura 4.21 (d) ilustra a comparação dos conjuntos dominantes obtidas pelas metaheurísticas para esta instância com 20 tarefas e 20 máquinas.

Para cada tamanho de instância, as médias dos tempos computacionais gastos pelas metaheurísticas são mostrados na Tabela 4.15.

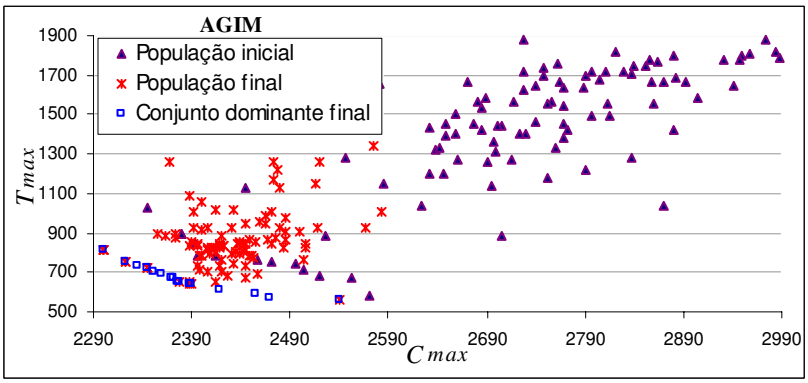
Tabela 4.15. Média de tempos computacionais (em segundos)

n:	20			50			80		
m:	5	10	20	5	10	20	5	10	20
AGHM	5,75	8,16	12,25	80,19	152,42	247,26	351,03	625,82	1106.2
AGIM	5,62	8,36	12,34	86,05	170,22	264,56	354,33	643,62	1143,3
SPEA	5,66	8,22	13,04	82,32	174,47	258,43	348,26	629,27	1154,1

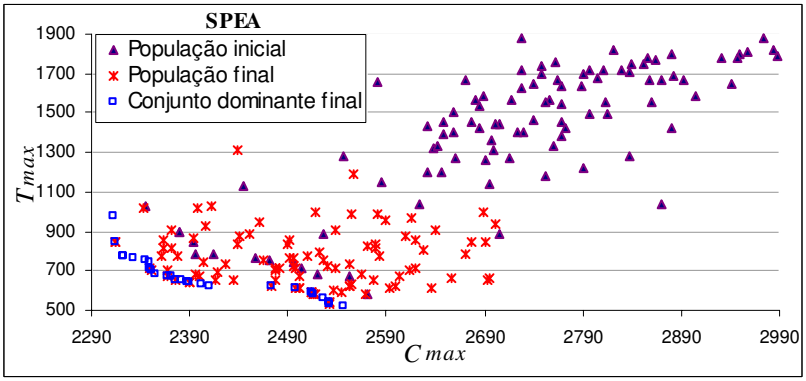
(a)



(b)



(c)



(d)

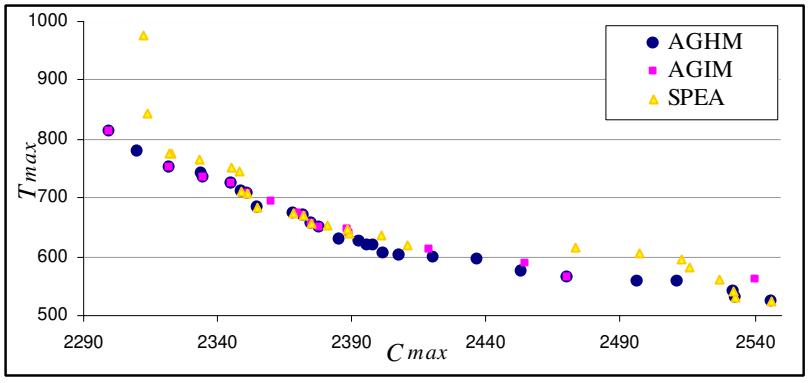


Figura 4.21. População inicial, final e conjunto dominante obtidas pelas metaheurísticas.

4.4.2. Resultados Computacionais para $f = (C_{max}, T)$

Nesta seção apresentam-se os resultados computacionais obtidos na solução do problema de flowshop para o par de objetivos (C_{max}, T). O desempenho das metaheurísticas implementadas foi testado nos mesmos três conjuntos de instâncias usados na seção anterior.

4.4.2.1. Resultados Para o Conjunto 1 de Instâncias

As soluções obtidas pelas metaheurísticas AGHM, AGIM e SPEA são comparadas com as soluções Pareto-ótimas obtidas por enumeração completa. Na Tabela 4.16 mostra-se, para cada tamanho de problema, o número de soluções eficientes e o número de soluções obtidas pelas metaheurísticas, associadas com 80 instâncias. Nas 320 instâncias testadas, a enumeração completa obteve 3442 soluções eficientes. As metaheurísticas AGHM, AGIM e SPEA encontraram 3335 (96,89%), 3173 (92,18%) e 2870 (83,38%) soluções eficientes e identificaram exatamente o conjunto Pareto-ótimo para 246, 201 e 114 instâncias, respectivamente.

Tabela 4.16. Número de soluções encontrados pela enumeração completa e pelas metaheurísticas

Problema	EC	AGHM		AGIM		SPEA	
$n \times m$	NSE	NTS	NSEM	NTS	NSEM	NTS	NSEM
10×5	755	738	725	747	724	713	666
10×10	820	805	803	795	783	779	712
11×5	825	809	803	782	745	748	706
11×10	1042	1016	1004	971	921	856	786
Total	3442	3368	3335	3295	3173	3096	2870

NSE: número de soluções eficientes encontradas pelo método exato.
NTS: número total de soluções encontradas pela metaheurística.
NSEM: número de soluções eficientes encontradas pela metaheurística.

Na Figura 4.22 mostra-se graficamente a média do número de soluções eficientes, para cada cenário de datas de entrega, obtida pela enumeração completa (EC) e pelas metaheurísticas AGHM, AGIM e SPEA. Como para o par de objetivos (C_{max}, T_{max}), observa-se que, a média de número de soluções eficientes encontradas pela enumeração completa é

maior para os cenários 2 e 4 e, a metaheurística AGHM sempre identifica a maior quantidade de soluções eficientes em todos os cenários. Note que, o número de soluções eficientes é maior para o par de objetivos (C_{max}, T).

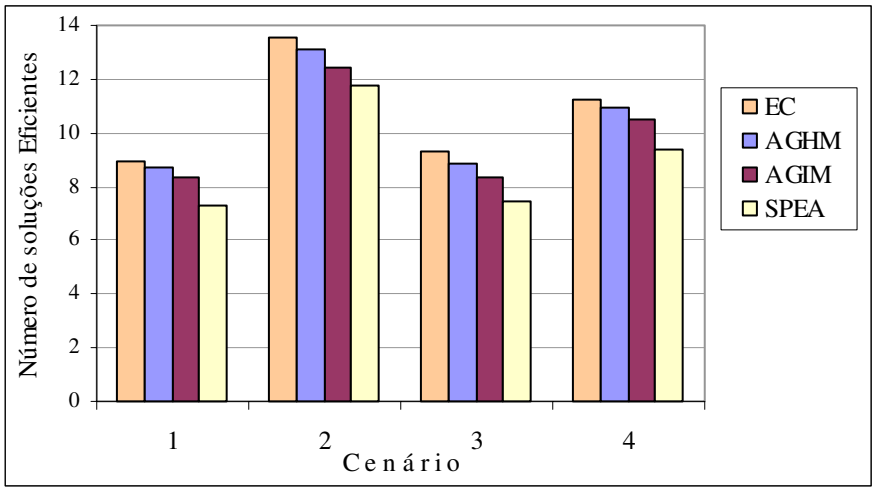


Figura 4.22. Média do número de soluções eficientes por cenário.

A Figura 4.23 ilustra a média do número de soluções eficientes, para cada tamanho de problema, obtida pela enumeração completa (EC) e pelas metaheurísticas AGHM, AGIM e SPEA. Observa-se que a metaheurística AGHM sempre apresenta a maior média de soluções eficientes para cada tamanho de problema.

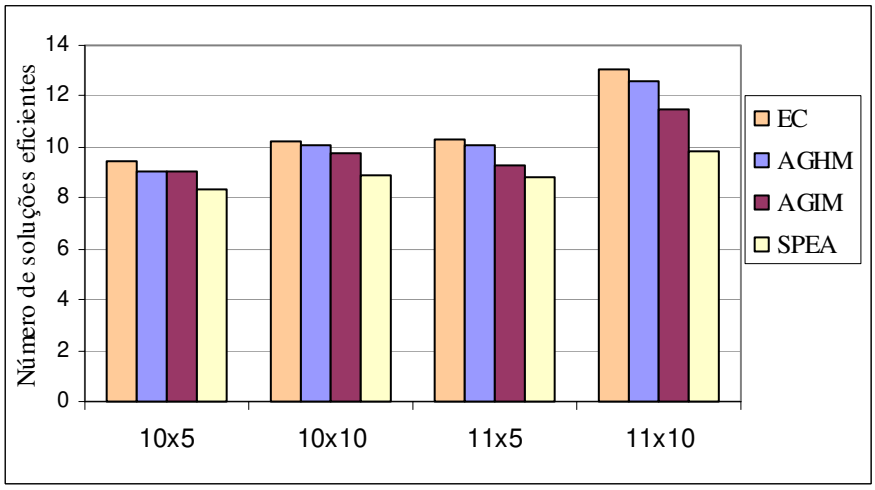


Figura 4.23. Média do número de soluções eficientes por tamanho de problema.

O desempenho das metaheurísticas em relação às medidas de distância e erro de utilidade são mostrados nas Tabelas 4.17 e 4.18, respectivamente. Com relação às duas medidas, o AGHM apresenta melhor desempenho.

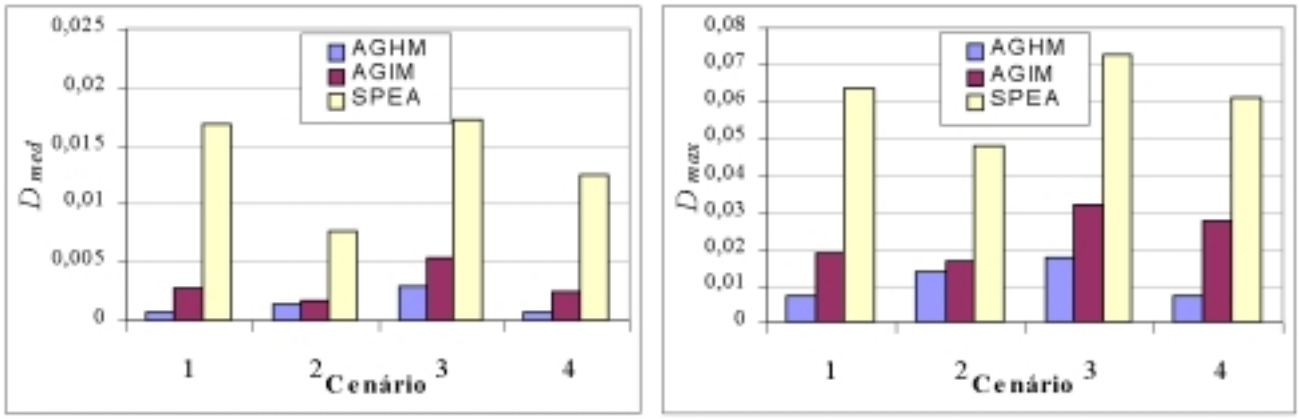
Tabela 4.17. Desempenho das metaheurísticas: Medida de distância

Problema	AGHM		AGIM		SPEA	
$n \times m$	D_{med}	D_{max}	D_{med}	D_{max}	D_{med}	D_{max}
10×5	0,0025	0,0166	0,0024	0,0152	0,0162	0,0586
10×10	0,0011	0,0080	0,0019	0,0132	0,0103	0,0538
11×5	0,0009	0,0074	0,0039	0,0282	0,0110	0,0508
11×10	0,0012	0,014	0,0039	0,0294	0,0164	0,0825
Média	0,0014	0,0115	0,0030	0,0215	0,0135	0,0614

Tabela 4.18. Desempenho das metaheurísticas: Erro de utilidade (%)

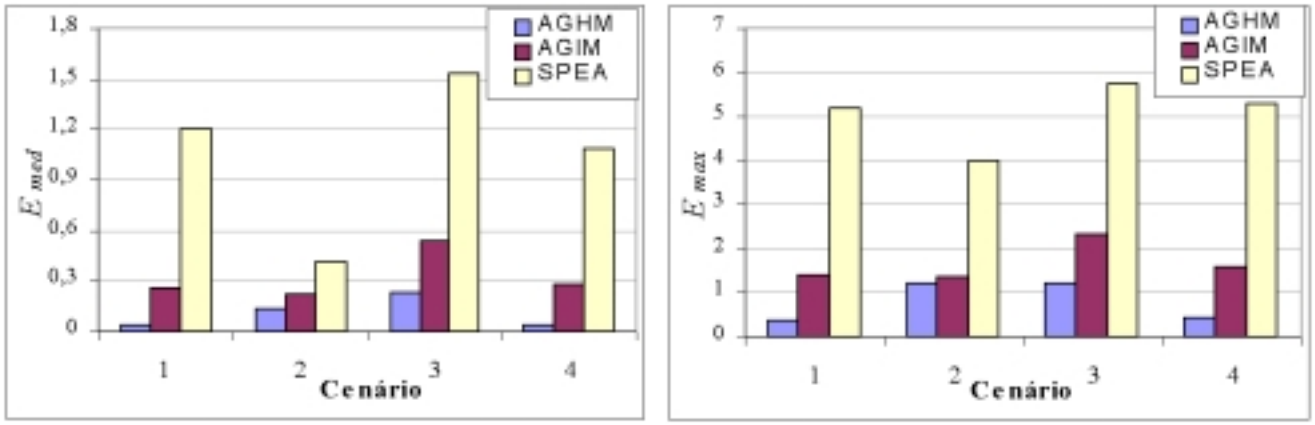
Problema	AGHM		AGIM		SPEA	
$n \times m$	E_{med}	E_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×5	0,134	1,109	0,295	1,128	1,217	4,773
10×10	0,133	0,651	0,212	1,024	0,960	4,057
11×5	0,053	0,423	0,402	2,084	0,806	4,018
11×10	0,1002	1,012	0,383	2,403	1,429	7,372
Média	0,105	0,800	0,323	1,660	1,103	5,055

Para cada cenário de datas de entrega, as Figuras 4.24 (a) e (b) ilustram os valores médios de D_{med} e D_{max}, enquanto as Figuras 4.25 (a) e (b) ilustram os valores médios de E_{med} e E_{max}. Observe que a metaheurística AGHM (SPEA) possui melhor (pior) desempenho em todos os cenários. As metaheurísticas AGIM e SPEA apresentam valores mais altos de distâncias e erros de utilidade para os cenários 1 e 3, que estão associados com pequenas faixas de datas de entrega.



(a) (b)

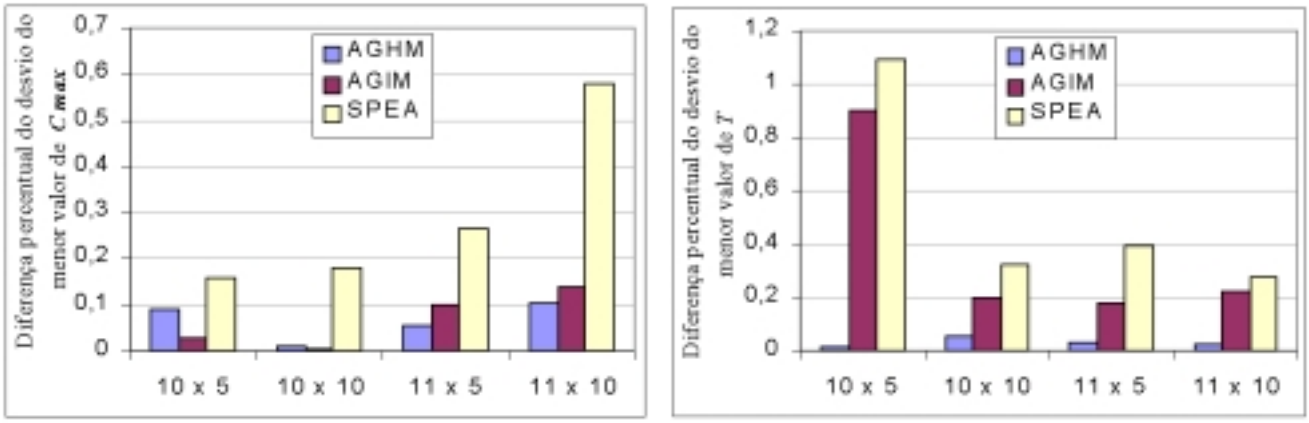
Figura 4.24. Distância média e máxima por cenário de data de entrega.



(a) (b)

Figura 4.25. Erro de utilidade percentual médio e máximo por cenário de data de entrega.

Nas 320 instâncias testadas, as metaheurísticas AGHM, AGIM e SPEA obtiveram o menor valor do makespan (C_{max}) em 287, 286 e 217 instâncias e o menor valor da soma dos atrasos (T) em 313, 284 e 273 instâncias, respectivamente. Para cada tamanho de problema, as Figuras 4.26 (a) e (b) ilustram a diferença percentual média do desvio dos menores valores de C_{max} e T, obtidas pelas metaheurísticas, em relação aos valores ótimos obtidos pela enumeração completa. Observe que, para cada de tamanho de problema, a metaheurística AGHM obteve, em média, os menores valores de T. Em problemas com 10 tarefas, a metaheurística AGIM obteve, em média, menores valores de C_{max}.



(a) (b)

Figura 4.26. Diferença percentual média do desvio dos menores valores de C_{max} e T obtida pelas metaheurísticas em relação aos valores ótimos.

Na Tabela 4.19 são mostrados os tempos computacionais gastos pelas metaheurísticas AGHM, AGIM e SPEA.

Tabela 4.19. Média de tempos computacionais (em segundos)

$n \times m$	10×5	10×10	11×5	11×10
AGHM	0,952	1,256	1,104	1,670
AGIM	0,912	1,205	1,019	1,474
SPEA	0,905	1,244	1,073	1,514

4.4.2.2. Resultados Para o Conjunto 2 de Instâncias

As 120 instâncias deste conjunto foram resolvidas otimamente pelo algoritmo Branch-and-Bound (B&B) desenvolvido por Liao et al. (1997), obtendo-se um total de 474 soluções eficientes. A Tabela 4.20 mostra, para cada tamanho de problema, o número de soluções obtidas pelo algoritmo B&B e pelas metaheurísticas. As metaheurísticas AGHM, AGIM e SPEA encontraram 461 (97,26%), 423 (89,24%) e 416 (87,76%) soluções eficientes e identificaram exatamente o conjunto Pareto-ótimo para 114, 96 e 93 instâncias, respectivamente. Observe que a metaheurística AGHM encontra o conjunto Pareto-ótimo em todos os problemas com 10 tarefas.

Tabela 4.20. Número de soluções encontradas pelo algoritmo B&B e pelas metaheurísticas.

Problema	B&B	AGHM		AGIM		SPEA	
$n \times m$	NSE	NSM	NSEM	NSM	NSEM	NSM	NSEM
10×2	158	158	158	157	153	154	148
15×2	172	174	168	171	159	168	155
20×2	144	141	135	140	111	133	113
Total	474	473	461	468	423	455	416

A média do número de soluções eficientes, para cada cenário de datas de entrega, obtida pelo algoritmo B&B e pelas metaheurísticas AGHM, AGIM e SPEA é apresentada na Figura 4.27. Observa-se que, a metaheurística AGHM identifica a maior quantidade de soluções eficientes em todos os cenários. A Figura 4.28 ilustra a média do número de soluções eficientes para cada tamanho de problema. Observa-se que problemas com 15 tarefas e 5 máquinas possuem, em média, maior número de soluções. Além disso, a metaheurística AGHM gerou maior quantidade de soluções eficientes para cada tamanho de problema.

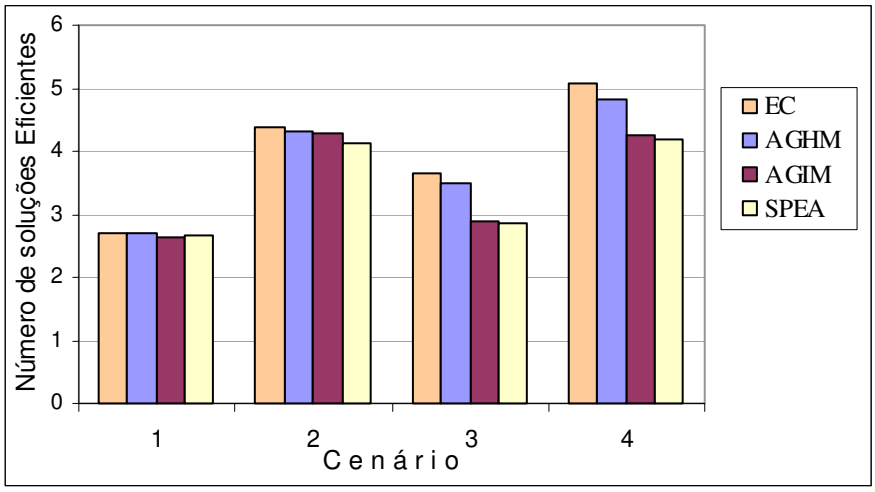


Figura 4.27. Média do número de soluções eficientes por cenário.

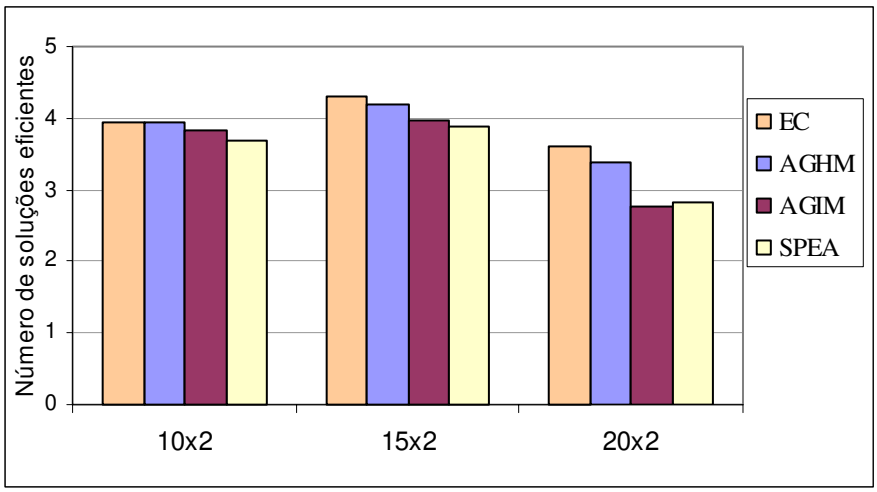


Figura 4.28. Média do número de soluções eficientes por tamanho de problema.

O desempenho das metaheurísticas de acordo com a medida de distância e erro de utilidade é mostrado nas Tabelas 4.21 e 4.22, respectivamente. Segundo as duas medidas, o AGHM apresenta melhor desempenho.

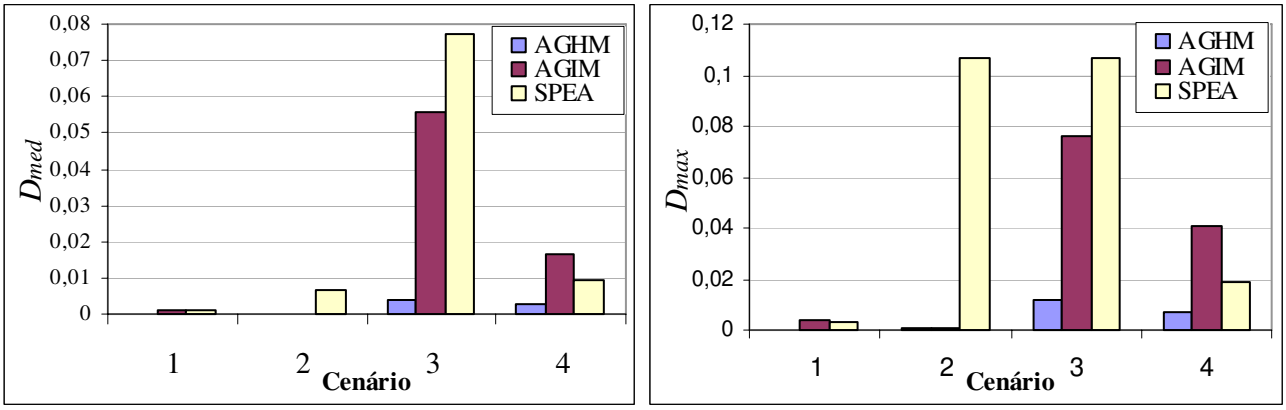
Para cada cenário de datas de entrega, as Figuras 4.29 (a) e (b) ilustram os valores médios de D_{med} e D_{max}, e, as Figuras 4.30 (a) e (b) ilustram os valores médios de E_{med} e E_{max}. Para este conjunto de instâncias, as metaheurísticas apresentam valores mais altos de distâncias e erros de utilidade para os cenários 3 e 4.

Tabela 4.21. Desempenho das metaheurísticas: Medida de distância

Problema	AGHM		AGIM		SPEA	
$n \times m$	D_{med}	D_{max}	D_{med}	D_{max}	D_{med}	D_{max}
10×2	0,0	0,0	0,009	0,019	0,0009	0,0044
15×2	0,0007	0,004	0,0021	0,0074	0,002	0,0069
20×2	0,0045	0,0108	0,044	0,064	0,067	0,102
Média	0,0017	0,0049	0,0184	0,0301	0,0233	0,0378

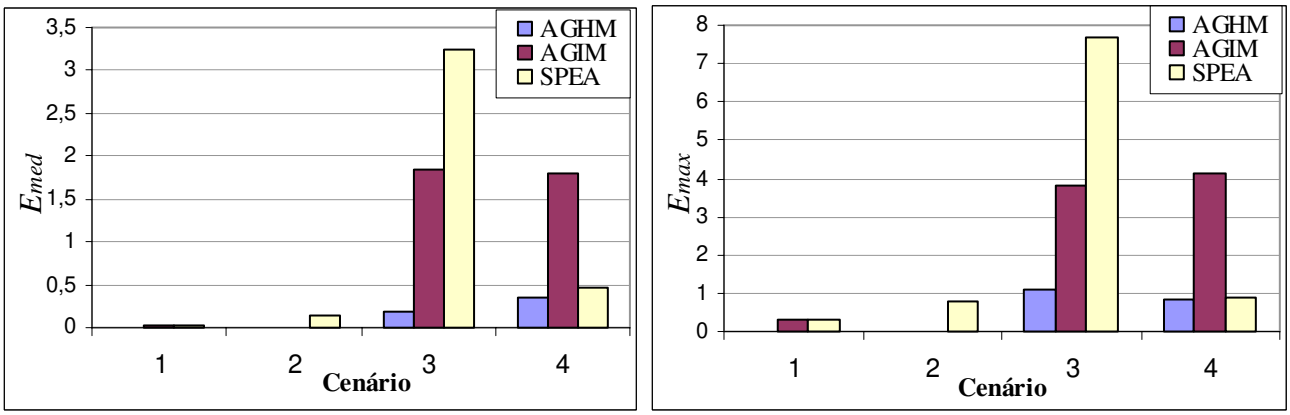
Tabela 4.22. Desempenho das metaheurísticas: Erro de utilidade (%)

Problema	AGHM		SPEA		AGIM	
$n \times m$	E_{med}	E_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×2	0,0	0,0	0,037	0,099	0,874	1,949
15×2	0,061	0,281	0,095	0,472	0,107	0,541
20×2	0,35	1,181	2,773	6,696	1,775	3,705
Média	0,13700	0,48733	0,96833	2,42233	0,91867	2,06500



(a) (b)

Figura 4.29. Distância média e máxima por cenário de data de entrega.



(a) (b)

Figura 4.30. Erro de utilidade percentual médio e máximo por cenário de data de entrega.

Nas 120 instâncias testadas, as três metaheurísticas obtiveram os valores ótimos do C_{max}. As metaheurísticas AGHM, AGIM e SPEA identificaram os valores ótimos de T em 117, 109 e 108 instâncias, respectivamente.

Para cada tamanho de instância, as médias dos tempos computacionais gastos pelo algoritmo B&B e pelas metaheurísticas são mostrados na Tabela 4.23.

Tabela 4.23. Média de tempos computacionais (em segundos)

Problema	Branch and Bound				AGHM	AGIM	SPEA
$n \times m$	Cenário 1	Cenário 2	Cenário 3	Cenário 4			
10×2	0,021	0,123	0,128	0,272	0,933	0,857	0,772
15×2	0,137	27,33	4,298	116,175	1,852	1,862	1,763
20×2	0,746	2331,38	87,317	6011,88	3,802	3,962	3,824

4.4.2.3. Resultados Para o Conjunto 3 de Instâncias

Para o par de objetivos (C_{max}, T), o desempenho das metaheurísticas AGHM, AGIM e SPEA foi avaliado em instâncias de grande porte pertencentes a este conjunto. Na Tabela 4.24, para cada tamanho de problema, mostra-se o número de soluções de dominantes de referência $\boldsymbol{R}$ e, para cada metaheurística mostra-se o número total de soluções e o número de soluções dominantes obtidas. Para as 360 instâncias testadas, foram encontradas 14173 soluções de referência. As metaheurísticas AGHM, AGIM, e SPEA obtiveram, respectivamente, 9971, 2089 e 4396 soluções dominantes. A tabela 4.24 mostra que AGHM é responsável pela maioria dos pontos dominantes de referência para todas as dimensões, enquanto que AGIM apresenta um mau desempenho quando comparado com SPEA para dimensões envolvendo 50 e 80 tarefas.

Nas 360 instâncias testadas comparam-se os menores valores de C_{max} e T obtidos pelas metaheurísticas. AGHM, AGIM e SPEA obtiveram o menor valor de C_{max} em 290, 167 e 98 instâncias e o menor valor de T em 297, 136 e 1134 instâncias, respectivamente.

Tabela 4.24. Número de soluções encontradas pelas metaheurísticas comparadas

Problema	$\|\boldsymbol{R}\|$	AGHM		AGIM		SPEA	
$n\times m$		NTS	NSD	NTS	NSD	NTS	NSD
20×5	799	782	686	685	369	649	306
20×10	1083	1047	903	817	296	788	267
20×20	1127	1088	995	852	320	812	237
50×5	805	771	493	678	159	691	376
50×10	2026	1966	1480	1518	222	1525	384
50×20	2315	2260	1657	1730	124	1955	539
80×5	664	633	382	547	125	619	284
80×10	1997	1971	1218	1576	196	1745	846
80×20	3357	3250	2157	2542	278	2920	1157
Total	14173	13768	9971	10945	2089	11704	4396

$|\boldsymbol{R}|$: número de soluções de referência.
NTS: número total de soluções obtidas por uma metaheurística..
NSD: número de soluções dominantes obtidas por uma metaheurística.

Na Figura 4.31 mostra-se a média de número de soluções dominantes, para cada cenário de datas de entrega, obtida pelas metaheurísticas. Observa-se que, a média de número de soluções dominantes é maior para os cenários 2 e 4, e a metaheurística AGHM identificou maior quantidade de soluções dominantes em todos os cenários. Novamente, SPEA se mostrou superior a AGIM.

Para cada tamanho de problema, na Figura 4.32 ilustra-se a média do número de soluções dominantes obtidas pelas metaheurísticas. Observe que, o número de soluções dominantes cresce à medida que a razão n/m decresce, e a metaheurística AGHM é responsável pela maioria dos pontos dominantes de referência para todas as dimensões, enquanto que AGIM apresenta um bom comportamento quando comparado com SPEA para dimensões envolvendo 20 tarefas.

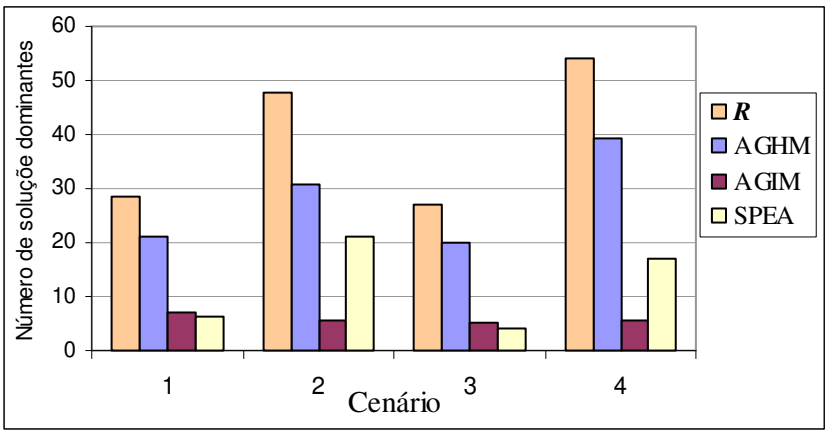


Figura 4.31. Média do número de soluções dominantes por cenário.

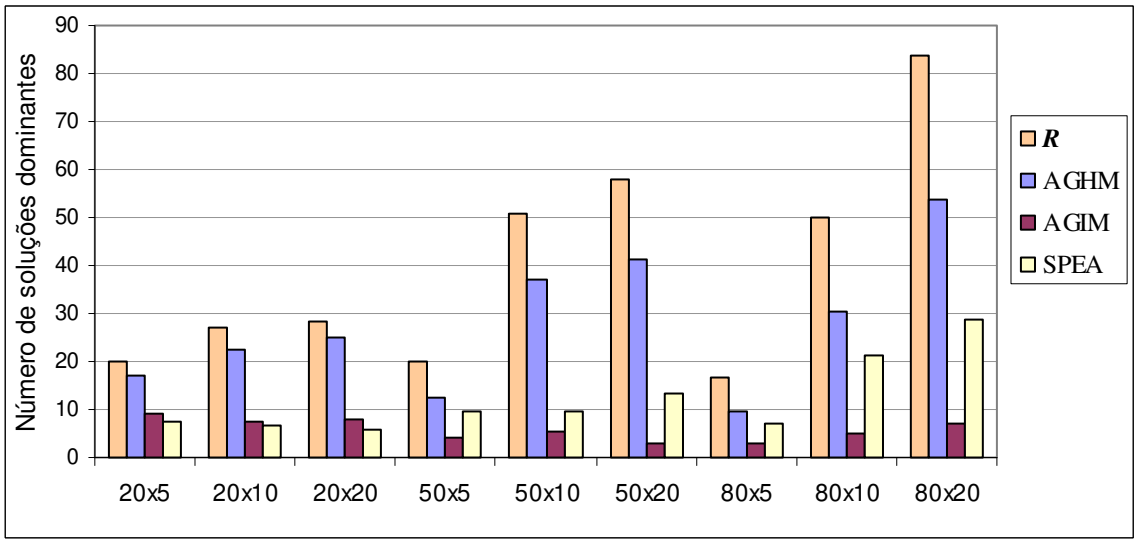


Figura 4.32. Média do número de soluções dominantes por tamanho de instância.

As Tabelas 4.25 e 4.26 mostram que a metaheurística AGHM possui um melhor desempenho em relação às medidas de distância e erro de utilidade percentual.

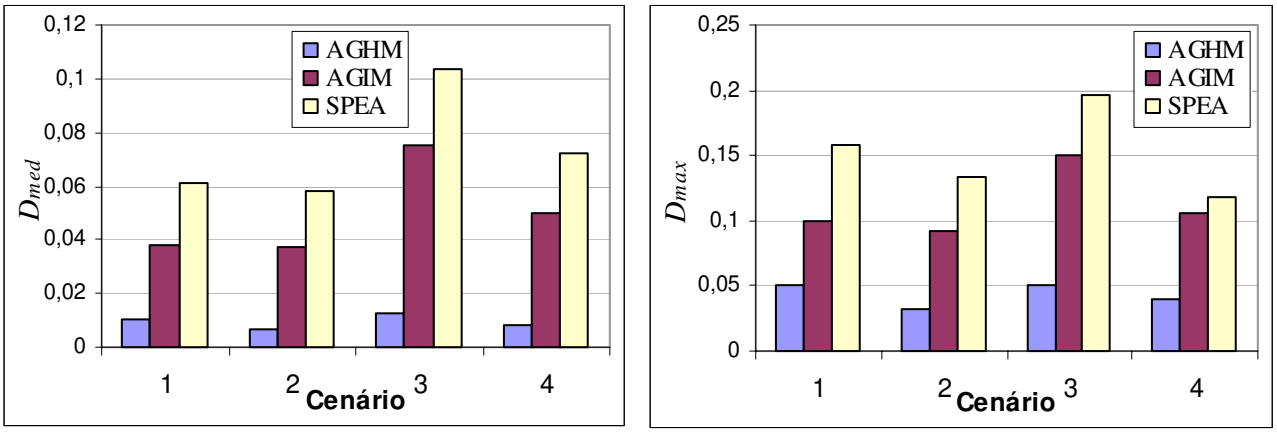
Tabela 4.25. Desempenho das metaheurísticas: Medida de distância

Problema	AGHM		AGIM		SPEA	
$m \times m$	D_{med}	D_{max}	D_{med}	D_{max}	D_{med}	D_{max}
20×5	0,0072	0,0407	0,0193	0,0646	0,0328	0,1133
20×10	0,006	0,041	0,037	0,109	0,056	0,158
20×20	0,003	0,026	0,037	0,118	0,053	0,141
50×5	0,012	0,044	0,044	0,092	0,077	0,131
50×10	0,007	0,044	0,054	0,118	0,068	0,165
50×20	0,005	0,036	0,075	0,144	0,082	0,158
80×5	0,020	0,048	0,075	0,122	0,153	0,221
80×10	0,015	0,059	0,053	0,115	0,054	0,150
80×20	0,010	0,049	0,059	0,127	0,047	0,126
Média	0,009	0,039	0,048	0,105	0,065	0,139

Tabela 4.26. Desempenho das metaheurísticas: Erro de utilidade (%)

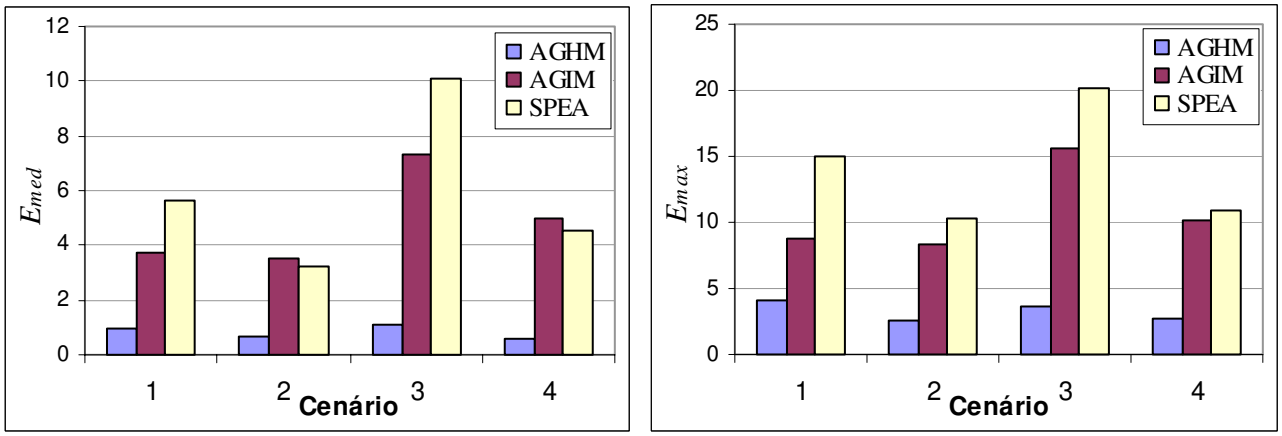
Problema	AGHM		AGIM		SPEA	
$n \times m$	E_{med}	E_{max}	E_{med}	E_{max}	E_{med}	E_{max}
20×5	0,619	2,843	1,742	5,707	3,009	9,597
20×10	0,524	2,713	3,543	10,166	6,105	15,472
20×20	0,253	1,851	4,180	11,050	5,461	13,879
50×5	1,142	3,781	3,798	8,179	4,356	9,660
50×10	0,666	3,059	5,420	11,011	6,451	16,202
50×20	0,607	3,270	8,296	15,334	8,857	17,103
80×5	1,388	3,865	5,661	12,395	6,866	16,202
80×10	1,146	3,735	4,984	10,169	6,023	15,354
80×20	1,098	4,264	6,402	13,658	5,577	13,640
Média	0,827	3,265	4,892	10,852	5,856	14,123

Para cada cenário, as Figuras 4.33 (a) e (b) mostram os valores médios (sobre 90 instâncias) das distâncias D_{med} e D_{max}, e as Figuras 4.34 (a) e (b) mostram os valores médios de E_{med} e E_{max}. A metaheurística AGHM apresenta menores valores da distância e erro de utilidade para todos os cenários.



(a) (b)

Figura 4.33. Distância média e máxima por cenário de data de entrega.



(a) (b)

Figura 4.34. Erro de utilidade percentual médio e máximo por cenário de data de entrega.

Nas avaliações realizadas para o par de objetivos (C_{max}, T) também pode ser observado que, a metaheurística SPEA obteve maior número de soluções que a metaheurística AGIM. No entanto, AGIM possui melhor desempenho em relação às medidas de distância e erro de utilidade. Como para o par de objetivos (C_{max}, T_{max}), os pontos dominantes obtidos pela metaheurística SPEA, geralmente, estão localizadas numa pequena faixa da fronteira dominante (***R***) e, os pontos dominantes obtidos pela metaheurística AGIM estão melhor distribuídos e mais próximos da fronteira dominante. Na Figura 4.35 ilustra-se este comportamento, para uma instância com 50 tarefas e 10 máquinas, no qual a fronteira

dominante (***R***) é formada por 37 pontos dominantes, das quais as metaheurísticas SPEA e AGIM encontraram 16 (B,C,...,Q) e 7 (A, I, J, L, N, P, Q) pontos dominantes, respectivamente. Observa-se que, os pontos gerados pela metaheurística AGIM estão mais próximos dos pontos de referência. Na Tabela 4.27 estão as medidas de distância e erro de utilidade percentual correspondentes às metaheurísticas AGIM e SPEA. Note que, existe uma diferencia considerável entre os erros de utilidade apresentada pelas metaheurísticas AGIM e SPEA. Na Figura 4.36 ilustram-se os pontos que possuem hiperplano suporte (pontos restantes após a eliminação de pontos através da técnica de avaliação de Daniels (1990)). Observe que, cada metaheurística (AGIM e SPEA) possui 10 pontos de suporte. Observe também que, os pontos de suporte de AGIM estão mais bem distribuídos e mais próximos dos pontos de referência.

Tabela 4.27. Medidas de distância e erro de utilidade para o exemplo da Figura 4.35

AGIM				SPEA			
D_{med}	D_{max}	E_{med}	E_{max}	D_{med}	D_{max}	E_{med}	E_{max}
0,0157	0,0794	1,491	4,404	0,0215	0,113	2,871	8,539

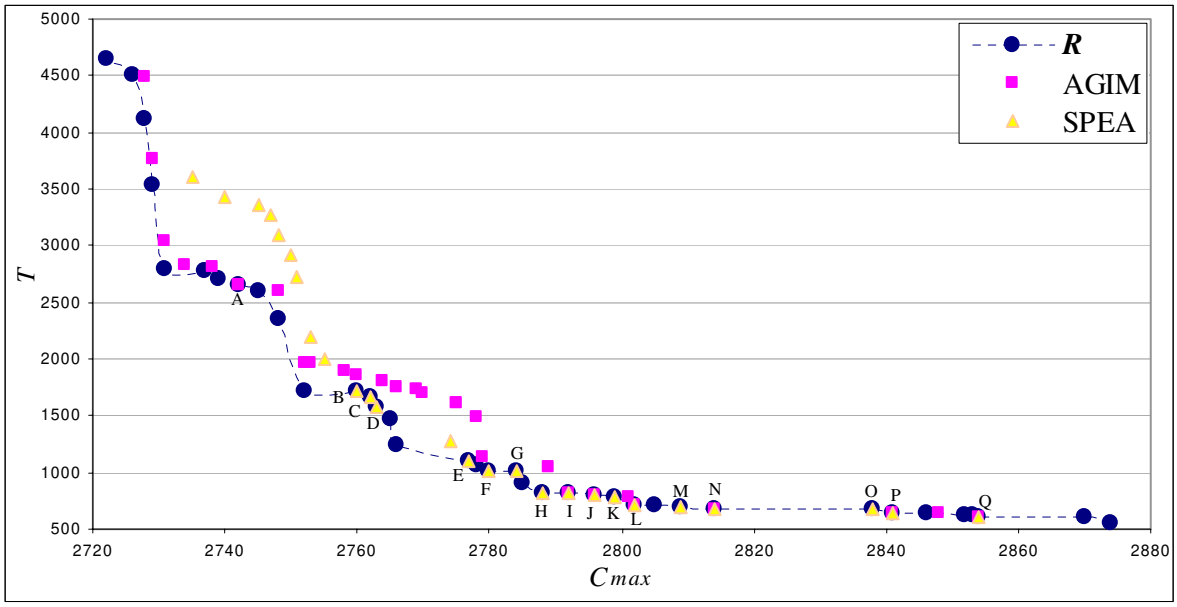


Figura 4.35. Distribuição dos pontos dominantes obtidos pelas metaheurísticas AGIM e SPEA.

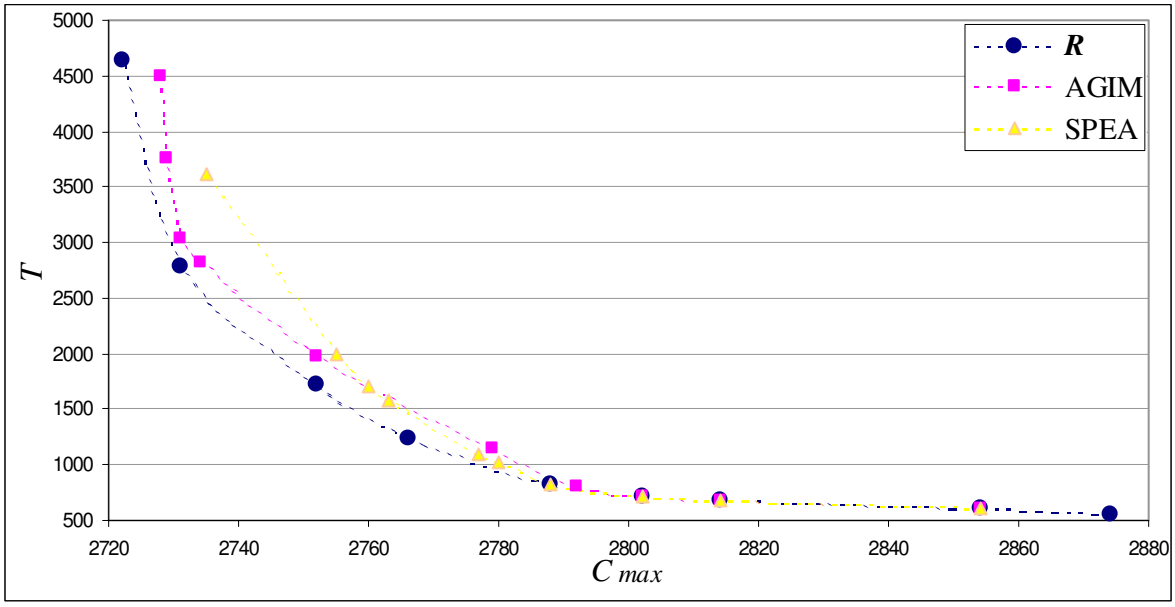


Figura 4.36. Pontos de suporte (pontos que possuem hiperplano suporte).

Para cada tamanho de instância, as médias dos tempos computacionais gastos pelas metaheurísticas são mostrados na Tabela 4.28.

Tabela 4.28. Média de tempos computacionais (em segundos)

n:	20			50			80		
m:	5	10	20	5	10	20	5	10	20
AGHM	6,42	8,82	12,78	83,92	164,35	262,24	353,24	642,81	1152,6
AGIM	6,52	9,63	13,27	91,57	182,82	274,53	363,34	662,76	1174,2
SPEA	5,63	8,72	13,34	85,37	175,53	268,24	348,62	635,42	1146,3

4.5. Aplicação do Algoritmo AGHM ao Problema da Mochila Multiobjetivo

Nesta seção, apresenta-se a implementação do algoritmo genético híbrido multiobjetivo AGHM para resolver o problema da mochila multiobjetivo 0-1. A qualidade das soluções da metaheurística AGHM é avaliada através da comparação com as soluções Pareto-ótimas obtidas pelo algoritmo Branch-and-Bound proposto por Ulungu e Teghem (1995).

Uma formulação geral do Problema da Mochila Multiobjetivo 0-1 (PMM) é a seguinte:

$$\begin{aligned} &\text{Maximizar} \quad z_j(\boldsymbol{x}) = \sum_{i=1}^{q} c_i^j x_i \ , \qquad j = 1,...,r \\ &\text{Sujeito a} \quad C_k = \sum_{i=1}^{q} w_i^k x_i \le W_k \ , \quad k = 1,...,t \\ &\qquad\qquad x_i \in \{0,1\}, \qquad\qquad i = 1,...,q \end{aligned} \qquad \textbf{(PMM)}$$

onde r é o número de objetivos, q é o número de itens (ou objetos) a serem inseridos na mochila, c_i^j é a utilidade do item i de acordo com o objetivo j, t é o número de restrições do problema, w_i^k denota um atributo tal como peso ou volume do item i segundo a restrição k, W_k é a capacidade da mochila considerando a restrição k e $\boldsymbol{x} = (x_1, ..., x_q)$ é um vetor de variáveis binárias x_i tal que:

$$x_i = \begin{cases} 1, \text{ se o item } i \text{ está na mochila} \\ 0, \text{ caso contrário} \end{cases}$$

O problema consiste em encontrar um conjunto de itens que maximize as utilidades totais $z_j(\boldsymbol{x})$.

Observe que, o problema (PMM) agrupa r problemas da mochila multidimensional mono-objetivos:

$$\begin{aligned} &\text{Maximizar } z_1(x) \text{ sujeito a } C_k \le W_k, \ k = 1,..., t \\ &\qquad\qquad \vdots \\ &\text{Maximizar } z_r(x) \text{ sujeito a } C_k \le W_k, \ k = 1,..., t. \end{aligned}$$

Na literatura, são estudados alguns casos particulares do problema (PMM). Ulungu e Teghem (1995), Teghem et al. (2000) e Gandibleux e Freville (2000) consideram o caso no qual o número de restrições $t = 1$:

$$\begin{aligned} &\text{Maximizar} \quad z_j(\boldsymbol{x}) = \sum_{i=1}^{q} c_i^j x_i \ , \qquad j = 1,...,r \\ &\text{Sujeito a} \quad C = \sum_{i=1}^{q} w_i x_i \le W \\ &\qquad\qquad x_i \in \{0,1\}, \qquad\qquad i = 1,...,q \end{aligned} \qquad \textbf{(PMM-1)}$$

Zitzeler e Thiele (1999) abordam o problema considerando o número de objetivos igual ao número de restrições:

$$\begin{aligned} &\text{Maximizar} \quad z_j(\boldsymbol{x}) = \sum_{i=1}^{q} c_i^j x_i \\ &\text{Sujeito a} \quad \sum_{i=1}^{q} w_i^j x_i \le W_j \ , \qquad j = 1,...,r \\ &\qquad\qquad x_i \in \{0,1\}, \qquad\qquad i = 1,...,q \end{aligned} \qquad \textbf{(PMM-2)}$$

Neste trabalho é abordado o problema da mochila multiobjetivo (PMM-1). Apresenta-se a seguir a descrição da implementação dos componentes do algoritmo AGHM aplicado a este problema.

4.5.1. Componentes do Algoritmo Genético

- **Representação de soluções e geração da população inicial**

Uma solução para o problema da mochila é representada por um vetor de q elementos, $\boldsymbol{x} = (x_1,...,x_q)$, no qual, $x_i = 1$ se o item i pertence à mochila e, $x_i = 0$, caso contrário.

Para gerar um conjunto inicial de soluções dominantes do problema, resolve-se através de um método heurístico o seguinte problema ponderado:

$$\text{Maximizar} \quad z(\boldsymbol{x}) = \sum_{i=1}^{r} \lambda_j z_j(\boldsymbol{x})$$

$$\text{Sujeito a} \quad \sum_{i=1}^{q} w_i x_i \le W \ , \ x_i \in \{0,1\}, \ \ i = 1,...,q \qquad \textbf{(P}\boldsymbol{\lambda}\textbf{)}$$

$$\sum_{j=1}^{r} \lambda_j = 1, \ \ 0 \le \lambda_j \le 1.$$

Note que (Pλ) é um problema mono-objetivo que maximiza a combinação linear dos objetivos do problema da mochila (PMM-1). Resolvendo o problema (Pλ), para um determinado vetor peso $\boldsymbol{\lambda} = (\lambda_1,...,\lambda_r)$, obtém-se uma solução Pareto-ótima do problema (PMM-1). As soluções do problema (Pλ) são denominadas *soluções eficientes de suporte*, pois estas soluções possuem hiperplano suporte. Soluções eficientes do problema (PMM-1) que não admitem hiperplano suporte não são geradas como soluções do problema (Pλ). Na Figura 4.37 ilustram-se os pontos eficientes de suporte, para o caso de dois objetivos. Geralmente, cada vetor peso $\boldsymbol{\lambda} = (\lambda_1,...,\lambda_r)$, determina uma direção de busca de pontos eficientes na fronteira Pareto-ótima.

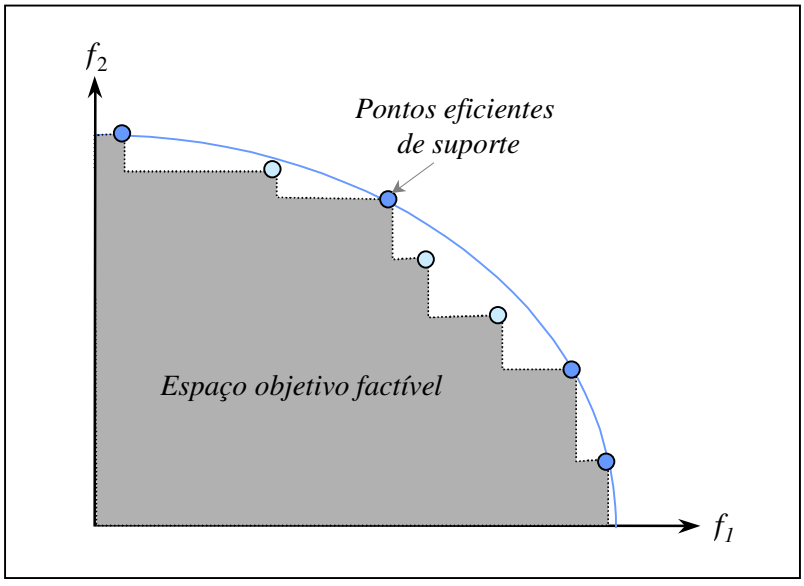


Figura 4.37. Pontos eficientes de suporte em um espaço objetivo não convexo.

Com o intuito de gerar N soluções que farão parte da população inicial do algoritmo genético AGHM, primeiramente, define-se um conjunto de vetores pesos $\mathbf{\Lambda} = \{\boldsymbol{\lambda}^d, \boldsymbol{\lambda}^d = (\lambda_1^d,...,\lambda_r^d)$, $d = 0,..., N-1\}$. Em seguida, resolve-se o problema (Pλ), para cada um dos N vetores pesos. Para resolver o problema (Pλ) é usada uma heurística gulosa simples descrita a seguir:

Algoritmo 4.6.

Entrada: $\boldsymbol{\lambda}^d$, $d = 0,..., N-1$ (*vetores pesos - direções de busca*)

Saída: $\boldsymbol{P}$ (*conjunto de soluções aproximadas*)

1) Faça $\boldsymbol{P} = \varnothing$.
2) Para $d = 0$ até $N-1$ faça
3) Faça $\boldsymbol{x}^d = (x_1,...,x_q) = (0,...,0)$.
4) Ordene os itens i em ordem decrescente de $(\sum_{j=1}^{r} \lambda_j^d c_i^j) / w_i$.
5) Faça $Soma = 0$, $i = 1$.
6) Enquanto $Soma + w_i \leq W$ faça
7) $x_i = 1$.
   $Soma = Soma + w_i$.
   $i = i + 1$.
8) Se $\boldsymbol{x}^d \notin \boldsymbol{P}$, então faça $\boldsymbol{P} = \boldsymbol{P} \cup \{\boldsymbol{x}^d\}$.
9) Fim.

Para o caso bi-objetivo, o conjunto de vetores pesos $\mathbf{\Lambda}$ é determinado da seguinte maneira:

$$\mathbf{\Lambda} = \{ \boldsymbol{\lambda}^d \mid \boldsymbol{\lambda}^d = (\lambda, 1-\lambda), \lambda = \frac{d}{N-1}, \ d = 0, ..., N-1 \}.$$

Note que, os N vetores pesos no conjunto $\mathbf{\Lambda}$ determinam diferentes direções de busca distribuídas por toda a fronteira de soluções dominantes.

Para o caso de mais de dois objetivos, os pesos podem ser determinados aleatoriamente:

$$\lambda_j = \frac{\omega_j}{\sum_{k=1}^{r} \omega_k}, \; j = 1,..., r$$

onde $\omega_1, ..., \omega_r$ são números reais não negativos gerados aleatoriamente.

Se após a execução do Algoritmo guloso 4.6, o número de soluções na população $\boldsymbol{P}$ é menor que N então, para completar a população, gera-se $N - |\boldsymbol{P}|$ soluções através do método de geração de soluções diversas para problemas 0/1, sugerida por Glover (1998).

Operadores Genéticos

A seguir descrevem-se os operadores de *recombinação e mutação* usados no problema da mochila multiobjetivo.

- *Operador de Recombinação*: Foi implementado o operador *crossover dois pontos* para problemas 0/1 (Goldberg, 1989). Para cada par de soluções pais, gera-se aleatoriamente um número real *rand* sobre o intervalo [0,1]. Se *rand* < p_R (p_R: probabilidade de recombinação), as soluções pais são submetidos a recombinação gerando duas soluções filhos. Para tal, seleciona-se arbitrariamente dois pontos p_1 e p_2 ($1 \le p_1 < p_2 \le q$) e os componentes (bits) dos pais entre estes dois pontos são trocados, produzindo dois filhos. Um exemplo do operador *crossover dois pontos* é mostrado na Figura 4.38. Se *rand* ≥ p_R, então as soluções pais são adicionadas (sem sofrer recombinação) na população seguinte.

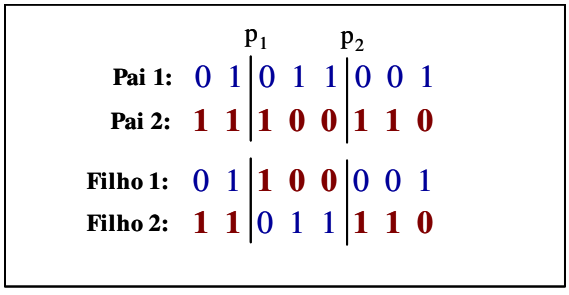


Figura 4.38. Crossover de dois pontos para problemas 0/1.

- *Operador de Mutação*: Este operador é executado em cada filho tentando alterar os bits 0/1. Para cada bit de um filho, gera-se aleatoriamente um número real *rand* sobre o intervalo [0,1]; se *rand* < p_M, altera-se o bit de 0 para 1 ou de 1 para 0 (p_M é a probabilidade de mutação). Um exemplo do operador de mutação aplicado ao Filho 1 da Figura 4.32 é mostrado na Figura 4.39.

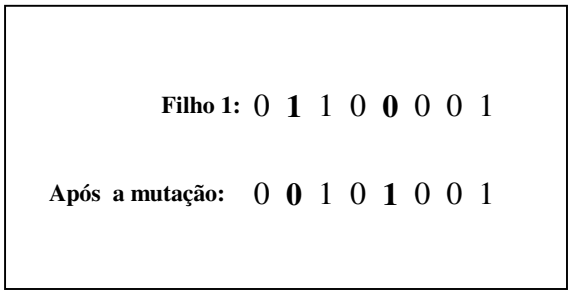


Figura 4.39. Operador de mutação.

Factibilização de soluções

Na população inicial, as soluções geradas através do método de geração de soluções diversas, sugerido por Glover (1998), podem ser infactíveis. Similarmente, os operadores genéticos podem gerar soluções infactíveis. Para factibilizar soluções, remove-se itens da mochila enquanto a restrição de capacidade seja violada ($\sum_{i=1}^{q} w_i x_i > W$). A ordem na qual os itens são removidos é determinado pela razão máxima de lucro/peso: $\gamma_i = max_{j=1}^{r}\{\frac{c_i^j}{w_i}\}$. Os itens são ordenados em ordem crescente de γ_i, i.e., os itens com valores pequenos de lucro e valores grandes de peso são removidos primeiro. A seguir é apresentado o algoritmo de factibilização.

Algoritmo 4.7. (Factibilização)

Entrada: $\boldsymbol{x} = (x_1, \ldots x_q),\ x_i \in \{0,1\}$, (*Solução infactível*)

Saída: $\boldsymbol{x} = (x_1, \ldots x_q),\ x_i \in \{0,1\}$, (*Solução factível*)

1) Ordene os elementos do conjunto $J_1 = \{i \mid x_i = 1\}$ em ordem crescente de $\gamma_i = max_{j=1}^{r}\{\frac{c_i^j}{w_i}\}$, onde r é o número de objetivos.

2) Faça $C = \sum_{i=1}^{q} w_i x_i$ e $i = 1$.

3) Enquanto $C > W$ faça

4) $x_i = 0$.

5) $C = C - w_i$.

6) $i = i + 1$.

7) Fim.

Busca Local e Definição da Vizinhança

A cada β iterações do algoritmo genético executa-se o procedimento da busca local multiobjetivo BLM (descrito no Algoritmo 2.3 do Capítulo 2) adaptado para o problema da mochila multiobjetivo. O algoritmo BLM começa a partir do conjunto $\boldsymbol{S}$ de soluções dominantes da população atual.

Para o problema da mochila multiobjetivo, foi considerada uma vizinhança baseada na remoção e inserção de itens (*drop-add*). Para uma solução $\boldsymbol{x} = (x_1, ..., x_q)$ define-se os conjuntos $J_1 = \{ j \mid x_j = 1\}$ (conjunto de itens na mochila) e $J_0 = \{ j \mid x_j = 0\}$ (conjunto de itens que não pertencem à mochila). A vizinhança de uma solução $\boldsymbol{x}$, $\boldsymbol{N}(\boldsymbol{x})$, é construída através do seguinte algoritmo:

Algoritmo 4.8. (Vizinhança drop-add)

1) Faça $\boldsymbol{N}(\boldsymbol{x}) = \varnothing$ e $C(\boldsymbol{x}) = \sum_{i=1}^{q} w_i x_i$.

3) Para cada $\nu \in J_1$ faça

4) Para cada $\mu \in J_0$ faça

5) - $\boldsymbol{y} = (y_1, ..., y_q)$ tal que, $y_i = x_i$, $i = 1,...,q$, $i \neq \nu$, $i \neq \mu$, $y_\nu = 0$, $y_\mu = 1$.

6) - Se $(C(\boldsymbol{x}) - w_\nu + w_\mu) \leq W$, então faça $\boldsymbol{N}(\boldsymbol{x}) = \boldsymbol{N}(\boldsymbol{x}) \cup \{\boldsymbol{y}\}$.

7) Fim.

Note que, as soluções vizinhas $\boldsymbol{y} \in \boldsymbol{N}(\boldsymbol{x})$ são sempre soluções factíveis, portanto não é necessário aplicar a estas soluções o algoritmo de factibilização. Para cada solução vizinha $\boldsymbol{y}$ obtida a partir de $\boldsymbol{x}$, os valores dos r objetivos são calculados da seguinte maneira:

$$f_j(\boldsymbol{y}) = f_j(\boldsymbol{x}) - c_v^j + c_\mu^j \ , \ j = 1, \ldots, r.$$

4.5.2. Resultados Computacionais Para o Problema da Mochila Multiobjetivo

Nesta seção analisa-se o comportamento e o desempenho da metaheurística proposta AGHM adaptado para resolver o problema da mochila multiobjetivo. Foram consideradas dez instâncias do problema com dois objetivos. O número de itens q varia de 50 a 500, os valores de utilidade c_i^j e do peso w_i, de cada item, são gerados aleatoriamente sobre o intervalo [1,100]. A capacidade da mochila, W, é um valor inteiro igual à metade do total de pesos: $W = 0.5\sum_{i=1}^{q} w_i$. Esta forma de determinar os valores de W gera problemas difíceis (Visée et al., 1998).

A qualidade das soluções geradas pela metaheurística AGHM é avaliada através da comparação com as soluções Pareto-ótimas obtidas pelo algoritmo Branch-and-Bound (B&B) proposto por Ulungu e Teghem (1995). Vale mencionar que, as instâncias e as respectivas soluções ótimas foram gentilmente fornecidas por Daniels Tuyttens, professor da Faculdade Politécnica de Mons, Bélgica (Tuyttens, et al., 2000).

4.5.2.1. Parâmetros

Alguns parâmetros foram testados para o algoritmo AGHM aplicado ao problema da mochila multiobjetivo. Combinações de parâmetros que geraram bons resultados são apresentados a seguir:

- Tamanho da população (N): Este parâmetro depende do tamanho do problema e é fixado da forma como apresentado na Tabela 4.29.

Tabela 4.29. Tamanho da população

q = 50, 100	q = 150, 200, 250	q = 300, 350, 400, 450, 500
N = 100	N = 150	N = 200

- Número máximo de soluções de elite: $MaxE = 30$.
- Probabilidade de recombinação: $p_R = 0.8$.
- Probabilidade de mutação: $p_M = 0.01$.
- Ativação da busca local: A cada $\beta = 4$ iterações.
- Número máximo de caminhos a serem exploradas pela busca local: $N_{max} = 6$.
- Número máximo de iterações da busca local: $Niter = 12$.
- Critério de parada: Como a busca local multiobjetivo BLM explora N_{max} caminhos em paralelo durante no máximo *Niter* iterações então, o algoritmo genético precisa de poucas iterações para encontrar soluções de boa qualidade. O algoritmo genético termina após a execução de 150 iterações, o que significa que a busca local BLM é ativada, aproximadamente, $150/\beta \approx 38$ vezes.

Os resultados computacionais para o problema da mochila multiobjetivo mostram que, o número de soluções dominantes encontradas é consideravelmente grande. Por esta razão, para reduzir o tamanho do conjunto $\boldsymbol{S}$ de soluções dominantes da população (conjunto de partida da busca local) utiliza-se o método de redução aleatória (Algoritmo 2.4). Este método divide o conjunto $\boldsymbol{S}$ em N_{max} subconjuntos distribuídos segundo os valores dos objetivos das soluções, e em cada subconjunto seleciona-se aleatoriamente uma solução. O método de redução aleatória é mais eficiente, em termos de tempo computacional, que a técnica de redução denominada *clustering* proposta por Morse (1980) (Algoritmo 2.5). A técnica *clustering* particiona o conjunto $\boldsymbol{S}$ de soluções dominantes em N_{max} grupos (clusters) de acordo com a proximidade das soluções. Para cada cluster, seleciona-se uma solução representativa (*centróide*). Esta técnica constrói um conjunto reduzido $\boldsymbol{S}$ cujos pontos estão bem distribuídos sobre a fronteira dominante. Portanto, a técnica *clustering* é usada, no lugar do método de redução aleatória, a cada 20 gerações do algoritmo genético.

4.5.2.2. Avaliação dos Resultados

Na Tabela 4.30, para cada tamanho de instância, apresenta-se o número de soluções eficientes obtidas pelo algoritmo B&B, o número total de soluções e o número de soluções eficientes obtidas pela metaheurística AGHM. Para as 10 instâncias testadas, o algoritmo B&B encontrou 7003 soluções eficientes sendo que a metaheurística AGHM identificou 3590 (51,26%) soluções eficientes.

Na Figura 4.40 ilustra-se o número de soluções eficientes obtidos pelo algoritmo B&B e pela metaheurística AGHM. Observe que o número de soluções eficientes cresce consideravelmente quando o número de itens é incrementado.

Tabela 4.30. Número de soluções obtidas pelo algoritmo B&B e pela metaheurística

Número de itens	B&B	AGHM		
q	NSE	NTS	NSEM	%NSE
50	34	34	34	100
100	172	171	170	98,84
150	244	243	234	95,90
200	439	429	413	94,08
250	629	575	474	75,36
300	713	655	520	72,93
350	871	768	417	47,88
400	1000	872	420	42,00
450	1450	1166	430	29,66
500	1451	1217	478	32,94
Total	7003	6130	3590	51,26

NSE: número de soluções eficientes obtidas pelo algoritmo B&B.
NTS: número total de soluções obtidas pela metaheurística.
NSEM: número de soluções eficientes obtidas pela metaheurística.
%NSE: porcentagem de soluções eficientes obtidas pela metaheurística.

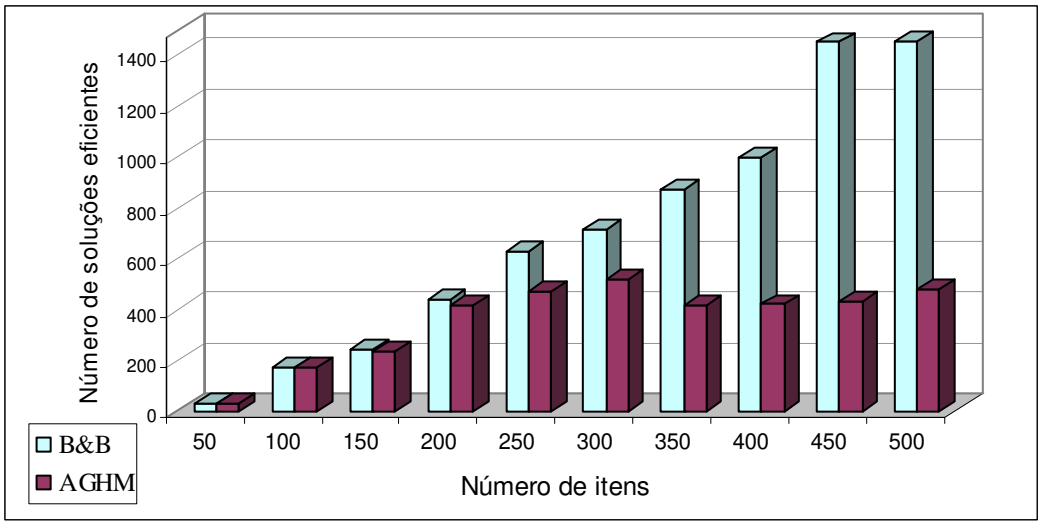


Figura 4.40. Número de soluções eficientes obtidas pelo algoritmo B&B e pela metaheurística AGHM.

O desempenho da metaheurística AGHM é avaliado através da medida de distância e erro de utilidade, ambos adaptados para problemas de maximização. Na Tabela 4.31, para cada tamanho de instância, mostra-se o desempenho da metaheurística AGHM em relação às duas medidas. Observe que, os valores das medidas de distância e erro de utilidade são muito pequenos, o que significa que as soluções encontradas pela metaheurística estão bem próximas às soluções eficientes e que são soluções de boa qualidade de acordo com a função de utilidade linear.

Tabela 4.31. Desempenho da metaheurística AGHM

Número de itens	Medida de Distância		Erro de Utilidade (%)	
	D_{med}	D_{max}	E_{med}	E_{max}
50	0	0	0	0
100	0,0000048	0,00083	0	0
150	0,000015	0,00079	0,0026	0,0441
200	0,000032	0,00233	0,00097	0,0556
250	0,000317	0,0610	0,0027	0,0784
300	0,000121	0,0217	0,00187	0,0733
350	0,000113	0,0362	0,0108	0,0649
400	0,000007	0,00159	0,0072	0,0653
450	0,000053	0,00091	0,0107	0,0613
500	0,000084	0,0311	0,007468	0,062
Média	0,0000746	0,01564	0,004431	0,05049

Na Tabela 4.32 mostra-se, para cada tamanho de instância, o tempo (em segundos) gasto pela metaheurística AGHM.

Tabela 4.32. Tempos computacionais (em segundos) da metaheurística AGHM

$q = 50$	$q = 100$	$q = 150$	$q = 200$	$q = 250$	$q = 300$	$q = 350$	$q = 400$	$q = 450$	$q = 500$
1,15	7,34	17,02	44,25	104,23	120,45	168,43	283,14	466,74	553,23

A Figura 4.41 ilustra o conjunto de pontos gerado pelo algoritmo B&B e pela metaheurística AGHM, para o problema com 300 itens. Fazendo uma ampliação da figura na faixa $11500 \leq f_1 \leq 12000$, observa-se que a metaheurística apresenta bom comportamento na geração de pontos eficientes. Analogamente, na Figura 4.42 ilustra-se o conjunto de pontos gerado pelo algoritmo B&B e pela metaheurística AGHM, para o problema com 500 itens

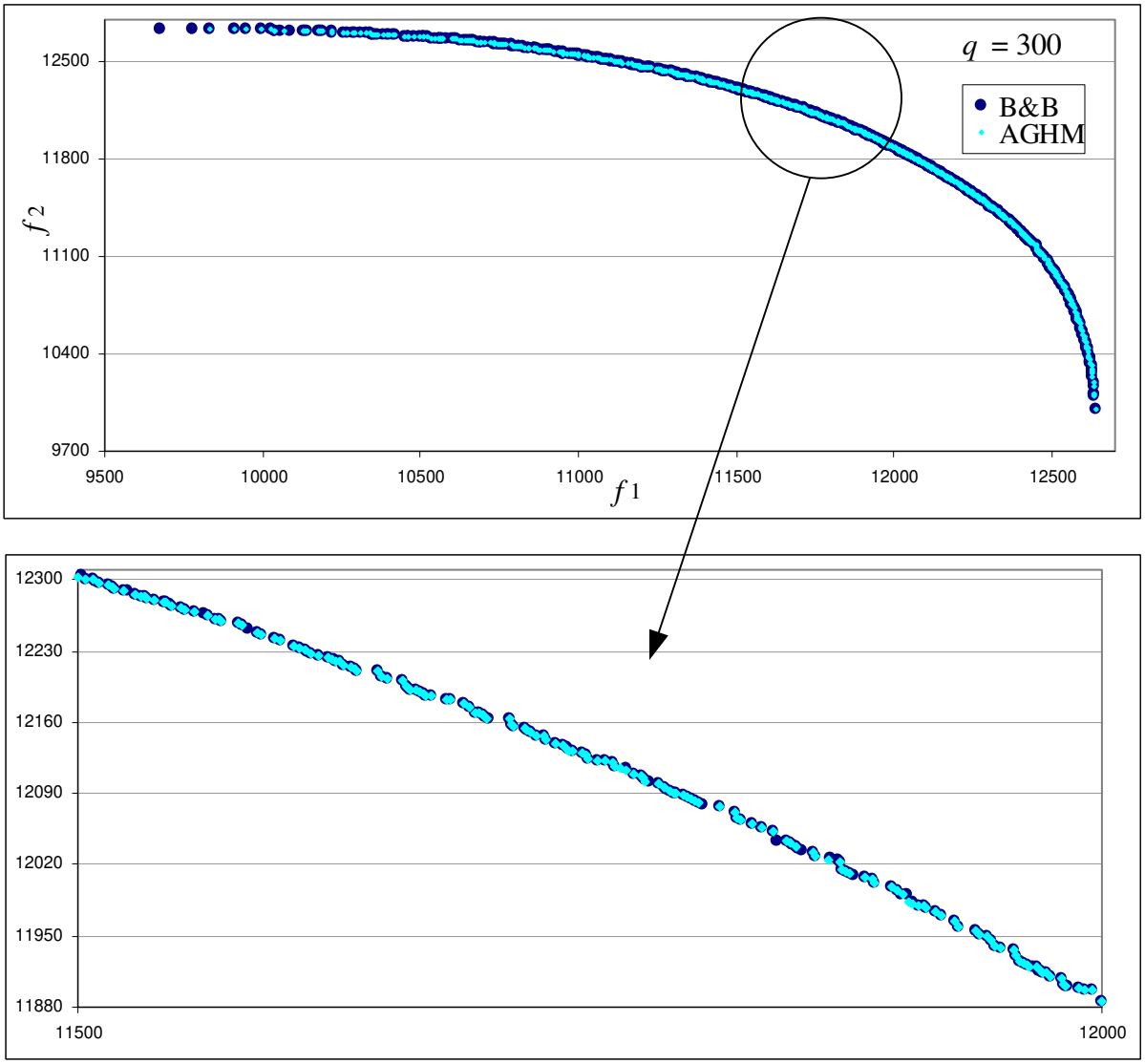


Figura 4.41. Pontos dominantes encontrados pela metaheurística AGHM (q = 300).

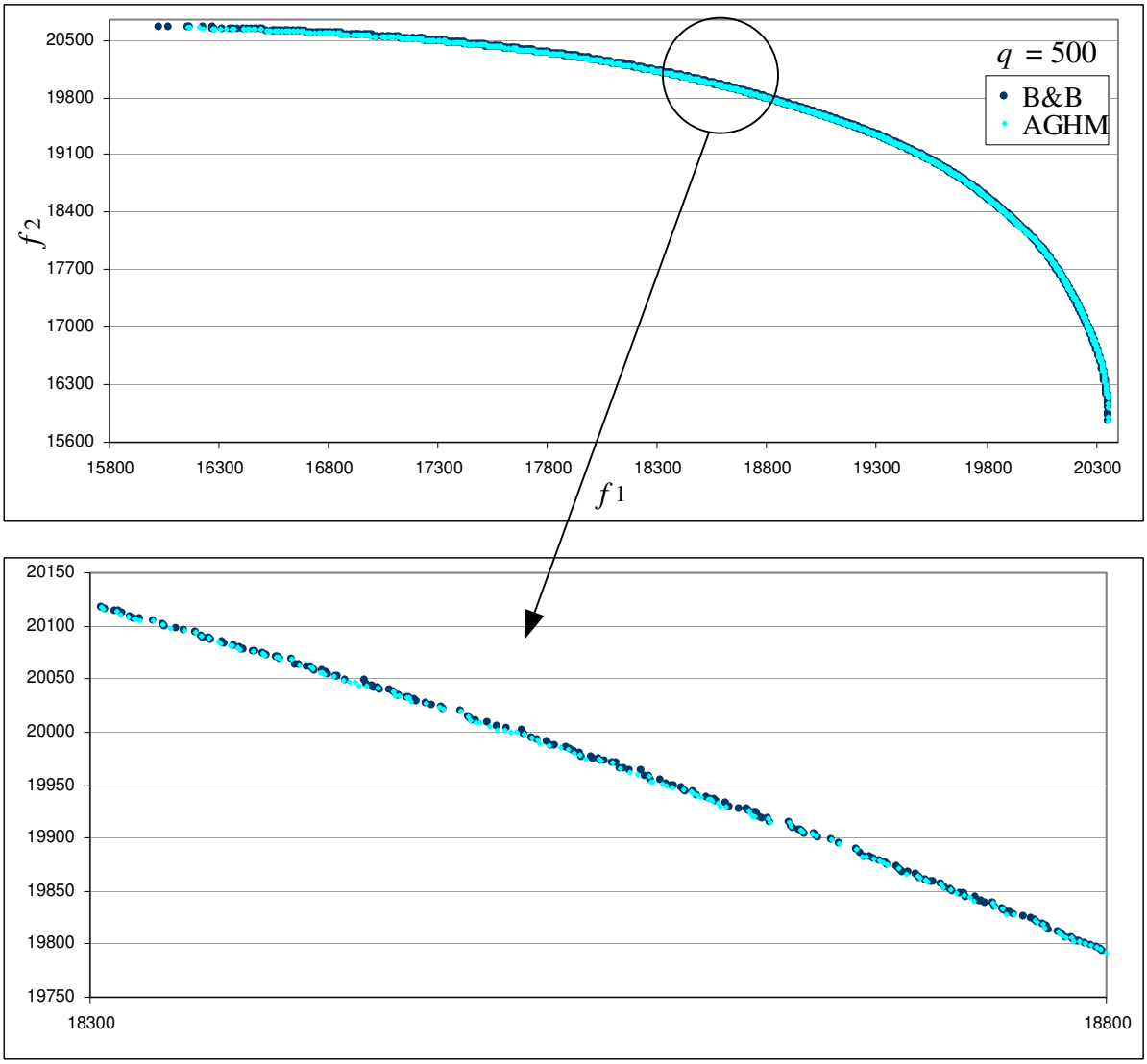


Figura 4.42. Pontos dominantes encontrados pela metaheurística AGHM (q = 500).

Capítulo 5

Busca Tabu Para Otimização Combinatória Multiobjetivo

Busca Tabu (BT) é uma metaheurística baseada em busca local ou busca em vizinhança, proposta por Glover (1989, 1990) para resolver problemas complexos de otimização combinatória. BT é um processo iterativo que procura explorar o espaço de soluções do problema, movendo-se sucessivamente de uma solução $\boldsymbol{x}$ para outra solução $\boldsymbol{x}'$, dentre sua vizinhança $\boldsymbol{N}(\boldsymbol{x})$. Para o caso mono-objetivo, os movimentos são realizados com a finalidade de encontrar boas soluções, segundo a avaliação de alguma função objetivo $f(\boldsymbol{x})$ a ser otimizada. No entanto, em cada iteração o valor da função $f(\boldsymbol{x}')$ não necessariamente é melhor que $f(\boldsymbol{x})$, isto é, a busca tabu pode aceitar soluções que não sejam melhores que a solução atual $\boldsymbol{x}$.

A característica principal da metaheurística BT é que, no processo de busca, ela incorpora uma memória adaptativa e estratégias de busca baseadas em memória. 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, prevenir ciclagem e direcionar a busca para regiões não exploradas. A proibição ocorre através do armazenamento dos atributos dos movimentos já realizados em uma lista denominada *tabu* que é consultada cada vez que se realiza um novo movimento. Caso os atributos deste movimento estejam presentes na lista tabu, este movimento 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.

Em algumas aplicações, o uso de memória de curto prazo é suficiente para produzir soluções de boa qualidade. Porém, existem casos onde a BT se torna mais eficiente ao se incluir memória de longo prazo e as estratégias associadas a esta. 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 BT têm sido aplicado com muito sucesso a uma variedade de problemas mono-objetivo (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 capítulo apresentamos um estudo sobre extensões de busca tabu para otimização multiobjetivo. Na seção 5.1 é apresentada uma revisão dos principais algoritmos de busca tabu multiobjetivo da literatura. Na seção 5.2 apresenta-se a descrição da implementação de uma nova metaheurística busca tabu para resolver problemas de otimização combinatória multiobjetivos. Nas duas últimas seções, apresentamos a aplicação do método proposto e os resultados computacionais obtidos.

5.1. Busca Tabu da Literatura

Na literatura, existem poucos trabalhos de aplicações de busca tabu para resolver problemas de otimização multiobjetivo (Ehrgott e Gandibleux, 2000; Jones et al., 2002). Nesta seção apresentam-se as contribuições mais importantes de alguns métodos baseados em busca tabu encontrados na literatura.

Hansen (1997) apresenta uma busca tabu multiobjetivo que explora um conjunto de soluções (***S***) em paralelo, cada uma com sua própria lista tabu. Para cada solução determina-se um vetor de pesos associado aos objetivos para definir uma direção de busca sobre o espaço objetivo. A escolha de uma solução vizinha é baseada na otimização da soma ponderada dos objetivos e o método tenta atingir diversificação através de aleatoriedade dos pesos dos objetivos. Hansen aplica a busca tabu proposta para resolver o problema da mochila multiobjetivo. A seguir apresentamos os detalhes desta busca tabu para o caso de minimização de *r* objetivos.

Algoritmo 5.1. Busca Tabu de Hansen (BTH)

Notações:

- ***D***: Conjunto de soluções dominantes encontrados até o momento.
- $\boldsymbol{\Lambda} = (\lambda_1, ..., \lambda_r)$: Vetor de pesos associado com cada objetivo.
- $\Delta_k = \max f_k - \min f_k$: Faixa do objetivo *k*, onde $\max f_k$ e $\min f_k$ são respectivamente o valor máximo e mínimo do objetivo f_k no conjunto ***D***.
- $\boldsymbol{\Pi} = (\pi_1, ..., \pi_r)$ onde $\pi_k = \dfrac{1}{\Delta_k}\left[\sum_{i=1}^{r}\dfrac{1}{\Delta_i}\right]^{-1}$ é o fator de normalização usado para normalizar a faixa do objetivo *k*.
- $g(d) = \dfrac{1}{d(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{y}), \boldsymbol{\Pi})}$ uma função de aproximação, para medir a distância entre duas soluções ***x*** e ***y***, onde $d(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{y}), \boldsymbol{\Pi}) = \sum_{k=1}^{r} \pi_k \left| f_k(\boldsymbol{x}) - f_k(\boldsymbol{y}) \right|$.
- ***LTx***: Lista tabu associada à solução ***x***.
- A(***x***, ***y***): Atributo do movimento realizado a partir de ***x*** para obter ***y***.
- $\boldsymbol{\Lambda f}(\boldsymbol{x}) = \sum_{k=1}^{r} \lambda_k f_k(\boldsymbol{x})$.

Entrada: N_{max} (*número de soluções a serem exploradas*)

Saída: ***D*** (*conjunto de soluções dominantes*)

1) Construa um conjunto inicial $\boldsymbol{S}$ de N_{max} soluções. Estas soluções podem ser geradas aleatoriamente ou usando alguma heurística construtiva.

   Faça $\boldsymbol{D} = \varnothing$.

   Faça $\boldsymbol{LTx}_k = \varnothing$, $\forall \boldsymbol{x}_k \in \boldsymbol{S}$, $k = 1, \ldots, N_{max}$.

   Faça $\pi_k = 1/r$, $\forall k = 1,\ldots, r$.

2) Para cada $\boldsymbol{x}_i \in \boldsymbol{S}$ faça

3) $\lambda_k = 0$, $\forall k = 1,\ldots, r$.

4) Para cada $\boldsymbol{x}_j \in \boldsymbol{S}$ tal que $\boldsymbol{x}_j$ não é dominada por $\boldsymbol{x}_i$ e $\boldsymbol{f}(\boldsymbol{x}_j) \neq \boldsymbol{f}(\boldsymbol{x}_i)$ faça

5) $\delta = g(\, d(\boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j), \boldsymbol{\Pi})\,)$.

6) Para cada objetivo k tal que $f_k(\boldsymbol{x}_i) < f_k(\boldsymbol{x}_j)$ faça $\lambda_k = \lambda_k + \pi_k\, \delta$.

7) Se $\lambda_k = 0$, $\forall k = 1,\ldots,r$, então selecione pesos aleatórios tal que: $\lambda_k \geq 0$ $\forall k$, e $\sum_{k=1}^{r} \lambda_k = 1$.

8) Seja V = infinito (um valor suficientemente grande).

9) Para cada solução vizinha $\boldsymbol{y}_i \in \boldsymbol{N}(\boldsymbol{x}_i)$ faça

10) Atualize o conjunto de soluções dominantes $\boldsymbol{D}$ com $\boldsymbol{y}_i$.

11) Se $A(\boldsymbol{x}_i, \boldsymbol{y}_i) \notin \boldsymbol{LTx}_i$ e $\boldsymbol{\Lambda f}(\boldsymbol{y}_i) < V$, então

12) Faça $V = \boldsymbol{\Lambda f}(\boldsymbol{y}_i)$ e $\boldsymbol{y} = \boldsymbol{y}_i$.

13) Senão, se $\boldsymbol{y}_i \in \boldsymbol{D}$, então $V = \boldsymbol{\Lambda f}(\boldsymbol{y}_i)$ e $\boldsymbol{y} = \boldsymbol{y}_i$ (*Critério de Aspiração*).

14) Adicione $A(\boldsymbol{y}, \boldsymbol{x}_i)$ a $\boldsymbol{LTx}_i$ e faça $\boldsymbol{x}_i = \boldsymbol{y}$.

15) Atualize os valores de π_k, $\forall k = 1,\ldots, r$.

16) Se a ativação do critério *drift* é satisfeito, então selecione aleatoriamente duas soluções $\boldsymbol{x}$ e $\boldsymbol{y}$ de $\boldsymbol{S}$ e faça $\boldsymbol{x} = \boldsymbol{y}$.

17) Se a condição de parada não é satisfeita, então volte ao passo 2, senão pare.

No algoritmo BTH, para cada solução $\boldsymbol{x}_i$ do conjunto $\boldsymbol{S}$, determina-se um vetor de pesos (passos 3 a 7) associado aos objetivos para definir uma direção de busca sobre o espaço objetivo. Os pesos são determinados de tal maneira que as soluções no conjunto $\boldsymbol{S}$, geralmente, estejam dispersos sobre a fronteira de busca. O valor dos pesos de uma solução $\boldsymbol{x}_i \in \boldsymbol{S}$ depende da proximidade de $\boldsymbol{x}_i$ com as outras soluções do conjunto $\boldsymbol{S}$. Portanto, compara-se a solução $\boldsymbol{x}_i$

com todas outras soluções do conjunto $\boldsymbol{S}$. Note que, se a solução $\boldsymbol{x}_i$ domina (ou é igual) às outras soluções do conjunto $\boldsymbol{S}$, então os pesos são determinados aleatoriamente, determinando uma nova direção de busca. Nos passos 9 a 14 do algoritmo BTH, são executadas as estratégias básicas da busca tabu. Para cada solução $\boldsymbol{x}_i \in \boldsymbol{S}$, gera-se a vizinhança $\boldsymbol{N}(\boldsymbol{x}_i)$ e seleciona-se a melhor solução vizinha (obtida a partir de um movimento não proibido) para substituir $\boldsymbol{x}_i$. Note que, uma solução obtida a partir de um movimento proibido é aceita para substituir $\boldsymbol{x}_i$ se ela é dominante, ou seja, se ela foi adicionada ao conjunto $\boldsymbol{D}$ (critério de aspiração). No passo 15 atualizam-se os fatores de normalização das faixas dos objetivos, e no passo 16 testa-se a ativação do critério *drift*. Este critério consiste em substituir uma solução de $\boldsymbol{S}$ (selecionada aleatoriamente) por outra solução de $\boldsymbol{S}$ (também selecionada aleatoriamente), ou seja, gera-se uma cópia de uma solução em $\boldsymbol{S}$. Esta estratégia influencia na redefinição de pesos (passos 3-7), em outras palavras, na redefinição de uma direção de busca. Segundo Hansen (1997), isto ajuda a localizar pontos dominantes sem hiperplanos suporte. A intensificação no algoritmo BTH é ativada a cada t de iterações da busca tabu, onde t é um número fixado. Hansen (1997), na aplicação do algoritmo BTH para o problema da mochila multiobjetivo, utiliza $t = 20$.

Recentemente, Viana e Pinho de Sousa (2000) aplicaram a busca tabu de Hansen para resolver um problema de programação de projetos com recursos limitados, minimizando três objetivos.

Baykasoglu et al. (1999) propõem uma busca tabu para encontrar o conjunto Pareto-ótimo em problemas de otimização multiobjetivo. Este método mantém três conjuntos: o conjunto de soluções proibidas (lista tabu), o conjunto das soluções dominantes encontradas até o momento da busca e um conjunto de soluções candidatas. Este último conjunto armazena soluções dominantes da vizinhança não pertencentes ao conjunto dominante que podem ser utilizadas na busca tabu se elas preservam o status de dominantes nas próximas iterações. Neste método, a diversificação é baseada na escolha de uma solução do conjunto de soluções candidatas, reiniciando a busca a partir desta solução e direcionando a busca para uma nova região da fronteira dominante. Os autores aplicam esta busca tabu para otimizar funções contínuas não lineares. A seguir apresentam-se os detalhes desta busca tabu.

Algoritmo 5.2. Busca Tabu de Baykasoglu et al.

Saída: $\boldsymbol{D}$ (*conjunto de soluções dominantes*)

1) *Inicialização*:

   Faça $\boldsymbol{LT} = \varnothing$, $\boldsymbol{D} = \varnothing$ e $\boldsymbol{LC} = \varnothing$.

   Construa uma solução inicial $\boldsymbol{x}$.

   Faça $\boldsymbol{LT} = \boldsymbol{LT} \cup \{\boldsymbol{x}\}$, $\boldsymbol{D} = \boldsymbol{D} \cup \{\boldsymbol{x}\}$.

2) *Geração da Vizinhança*:

   Construa a vizinhança de $\boldsymbol{x}$: $\boldsymbol{N}(\boldsymbol{x})$.

   $\overline{\boldsymbol{N}}(\boldsymbol{x}) = \{\boldsymbol{x}' \in \boldsymbol{N}(\boldsymbol{x})$: $\boldsymbol{x}'$ não é dominado por $\boldsymbol{x}$ e $\boldsymbol{x}' \notin \boldsymbol{LT}\}$.

3) *Identificação das soluções candidatas*:

   $\boldsymbol{C}$ = soluções dominantes de $\overline{\boldsymbol{N}}(\boldsymbol{x})$.

   Elimine de $\boldsymbol{C}$ as soluções dominadas por $\boldsymbol{LC}$ e $\boldsymbol{D}$.

4) *Seleção da próxima solução* $\boldsymbol{x}$:

   Se $\boldsymbol{C} = \varnothing$, então

      Se $\boldsymbol{LC} = \varnothing$, então pare.

      Senão

         Escolha $\boldsymbol{x}$ como sendo a última solução adicionada em $\boldsymbol{LC}$.

   Senão

      Selecione aleatoriamente uma solução, $\boldsymbol{x}$, dentre todas as soluções em $\boldsymbol{C}$.

5) *Atualizar* $\boldsymbol{LC}$, $\boldsymbol{D}$ *e* $\boldsymbol{LT}$:

   Faça $\boldsymbol{LC}$ = soluções dominantes de ($\boldsymbol{LC} \cup \boldsymbol{C}$). Se $\boldsymbol{x} \in \boldsymbol{LC}$, então faça $\boldsymbol{LC} = \boldsymbol{LC} - \{\boldsymbol{x}\}$.

   Elimine de $\boldsymbol{D}$ as soluções dominadas por soluções de $\boldsymbol{C}$.

   Faça $\boldsymbol{D} = \boldsymbol{D} \cup \{\boldsymbol{x}\}$ e $\boldsymbol{LT} = \boldsymbol{LT} \cup \{\boldsymbol{x}\}$.

6) Se a condição de parada não é satisfeita, então volte ao passo 2, senão pare.

Note que a busca tabu de Baykasoglu explora uma solução a cada iteração, gerando uma única trajetória de busca. O conjunto de soluções dominantes $\boldsymbol{D}$ é inicializado com a solução inicial factível $\boldsymbol{x}$. No passo 2, determinam-se o conjunto $\overline{\boldsymbol{N}}(\boldsymbol{x})$ das soluções vizinhas

(de $\boldsymbol{x}$) não proibidas e não dominadas por $\boldsymbol{x}$, e no passo 3 determina-se o conjunto $\boldsymbol{C}$ das soluções dominantes de $\overline{\boldsymbol{N}}(\boldsymbol{x})$. A próxima solução a ser explorada é selecionada aleatoriamente do conjunto $\boldsymbol{C}$ e as soluções não selecionadas são armazenadas na lista de soluções candidatas $\boldsymbol{LC}$. Na Figura 5.1 ilustra-se um exemplo da vizinhança no espaço objetivo, onde $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x})$. Observe que o conjunto $\boldsymbol{C}$ é formado pelos pontos a, b, c, d e suponha que o ponto escolhido aleatoriamente seja a. A busca continua a partir do ponto $\boldsymbol{z} = a$. Se o conjunto de soluções dominantes $\boldsymbol{C}$ é vazio, então a próxima solução a ser explorada pela busca é escolhida da lista $\boldsymbol{LC}$. Note que se a lista $\boldsymbol{LC}$ é vazia, a busca tabu para. Após selecionar a próxima solução $\boldsymbol{x}$, os conjuntos $\boldsymbol{LC}$, $\boldsymbol{D}$ e $\boldsymbol{LT}$ são atualizados. Como as soluções do conjunto $\boldsymbol{D}$ dominadas por soluções de $\boldsymbol{C}$ são eliminadas, então $\boldsymbol{x}$ é adicionada a $\boldsymbol{D}$.

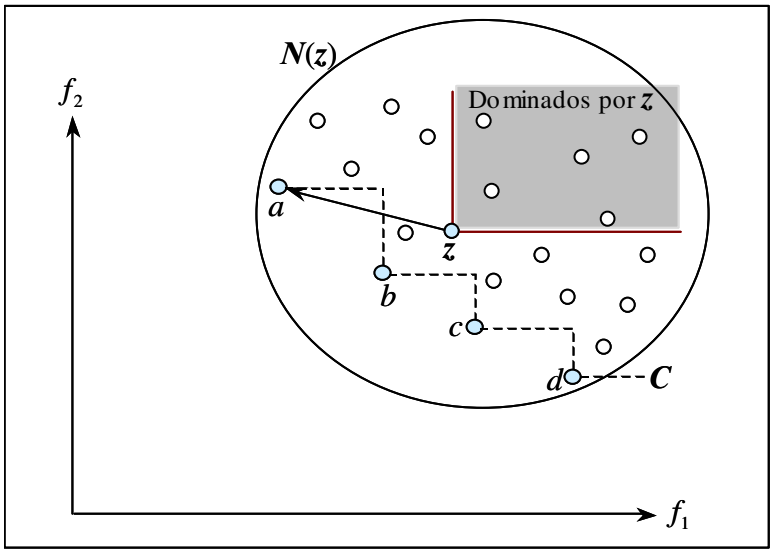


Figura 5.1. Busca local de Baykasoglu.

A busca local multiobjetivo (BLM) proposta no Capítulo 2 (seção 2.2) possui uma estratégia parecida com a estratégia da busca sugerida por Baykasoglu. Em BLM, ao invés de explorar uma única solução, explora-se um conjunto de N_{max} soluções (distribuídas por toda a fronteira dominante) gerando N_{max} vizinhanças. O próximo conjunto a ser explorado é formado pelas soluções dominantes dentre todas as soluções das N_{max} vizinhanças. Vale lembrar que se existem mais do que N_{max} soluções dominantes, seleciona-se somente N_{max}

soluções e as soluções restantes são armazenadas numa lista para serem exploradas nas próximas iterações, caso estas não sejam dominadas por novas soluções encontradas.

A estratégia adotada em BLM permite gerar varias trajetórias de busca em paralelo tentando obter novas soluções dominantes com direções distintas. Nos testes realizados foi observado que BLM requer poucas iterações para obter soluções dominantes de boa qualidade.

Ben et al. (1999) e Gandibleux e Fréville (2000) apresentam metaheurísticas baseadas em busca tabu para resolver especificamente o problema da mochila multiobjetivo.

5.2. Proposta de um Algoritmo de Busca Tabu Multiobjetivo (BTMO)

Nesta seção apresenta-se a descrição de uma nova busca tabu multiobjetivo (BTMO) para resolver problemas de otimização combinatória envolvendo mais de um objetivo. No método proposto é usado fortemente o conceito de dominância de Pareto. Com o objetivo de encontrar uma variedade de pontos distribuídos sobre toda a fronteira dominante, explora-se um conjunto de soluções em paralelo, cada uma com sua própria lista tabu, de modo similar ao método proposto por Hansen (1997).

A metaheurística BTMO inicia com um conjunto $\boldsymbol{S}$ de N_{max} soluções dominantes. As soluções deste conjunto podem ser geradas aleatoriamente ou através de uma heurística construtiva. Para cada solução $\boldsymbol{x}_k$ do conjunto $\boldsymbol{S}$, gera-se a vizinhança $\boldsymbol{N}(\boldsymbol{x}_k)$ e determina-se o próximo conjunto $\boldsymbol{S}$' formado pelas soluções dominantes de $\bigcup_{k=1}^{N_{\max}} \boldsymbol{N}(\boldsymbol{x}_k)$. Na Figura 5.2 ilustra-se três vizinhanças construídas a partir dos elementos do conjunto $\boldsymbol{S} = \{\boldsymbol{x}_1^0, \boldsymbol{x}_2^0, \boldsymbol{x}_3^0\}$ ($N_{max} = 3$), e o conjunto $\boldsymbol{S}$' constituído pelas soluções dominantes dentre todas as soluções das vizinhanças. Note que, as soluções dominantes no conjunto $\boldsymbol{S}$', geralmente, estão distribuídas sobre a fronteira dominante.

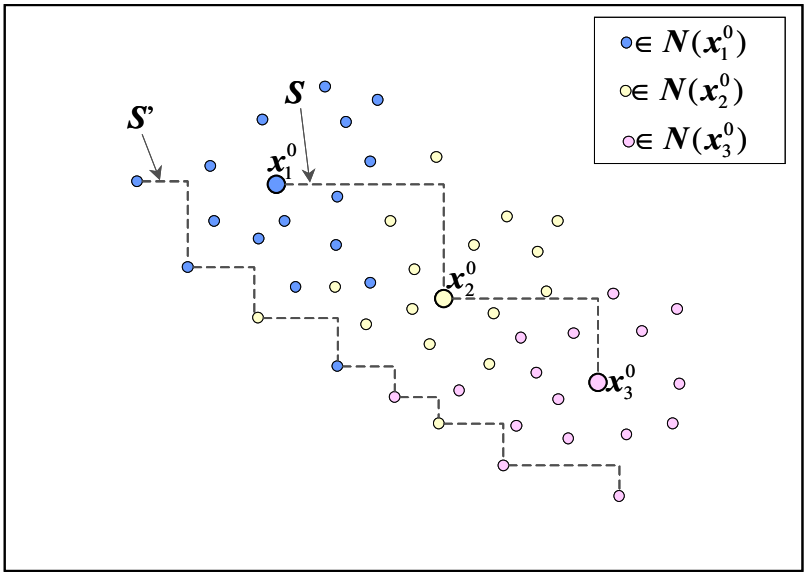


Figura 5.2. Vizinhanças construídas a partir do conjunto inicial $\boldsymbol{S} = \{\boldsymbol{x}_1^0, \boldsymbol{x}_2^0, \boldsymbol{x}_3^0\}$. $\boldsymbol{S}$' é o conjunto das soluções dominantes.

Para continuar com o processo de busca deve-se selecionar do conjunto $\boldsymbol{S}$', as soluções que formarão o próximo conjunto de partida $\boldsymbol{S}$. Se $|\boldsymbol{S}'| \leq N_{max}$, então o novo conjunto $\boldsymbol{S}$ é formado por todas as soluções de $\boldsymbol{S}$' ($\boldsymbol{S} = \boldsymbol{S}'$), caso contrário seleciona-se, através de algum critério, N_{max} soluções do conjunto $\boldsymbol{S}$'. Então, o novo conjunto de partida $\boldsymbol{S}$ será formado pelas soluções selecionadas. Para selecionar soluções de uma fronteira dominante pode ser usado o método de redução aleatória (Algoritmo 2.4) válido somente para o caso bi-objetivo ou o método de redução por *clustering* (Algoritmo 2.5) proposto por Morse (1980). Estes métodos, selecionam N_{max} pontos de tal forma que estejam distribuídos sobre a fronteira dominante. Na Figura 5.3 ilustra-se os $N_{max} = 3$ pontos selecionados do conjunto dominante na primeira iteração, $(\boldsymbol{x}_1^1, \boldsymbol{x}_2^1, \boldsymbol{x}_3^1)$ e na segunda iteração, $(\boldsymbol{x}_1^2, \boldsymbol{x}_2^2, \boldsymbol{x}_3^2)$ da busca tabu. Observe que, a cada iteração sempre são selecionadas $N_{max} = 3$ soluções da fronteira dominante $\boldsymbol{S}$'. Nesta Figura também se observa que, as soluções dominantes *a*, *b*, *c*, *d*, *e* (geradas na iteração $t = 1$), *f*, *g*, *h* (geradas na iteração $t = 2$) não foram selecionadas. As soluções não selecionadas numa dada iteração são guardadas num conjunto que é denominado *lista de soluções candidatas* $\boldsymbol{LC}$. As soluções candidatas são comparadas com as soluções pertencentes à próxima fronteira dominante. Se uma solução candidata domina alguma solução da fronteira dominante gerada,

então esta solução candidata fará parte da fronteira dominante. As soluções dominadas, tanto da lista soluções candidatas como da fronteira dominante, são descartadas. Resumindo, a cada iteração antes de selecionar as novas soluções de partida, atualiza-se a fronteira dominante gerada com a lista de soluções candidatas:

$$\boldsymbol{S}' = \text{soluções dominantes de } (\boldsymbol{S}' \cup \boldsymbol{LC}).$$

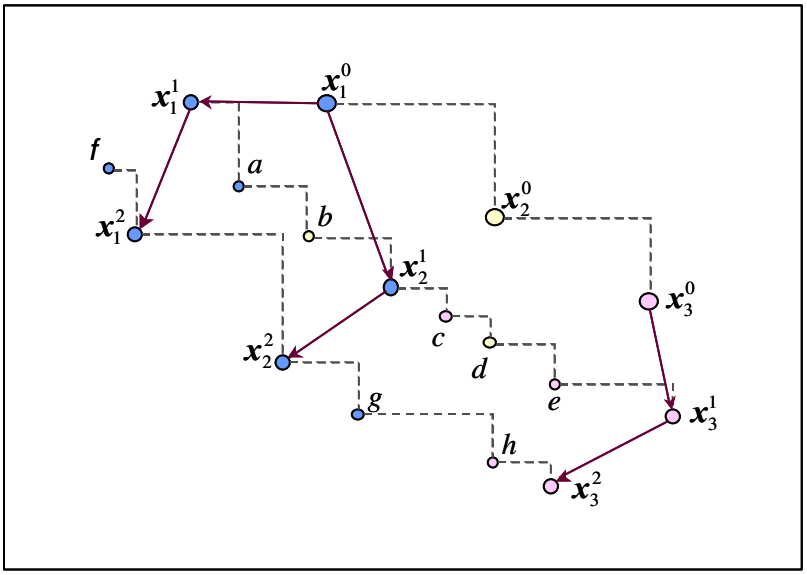


Figura 5.3. Trajetórias geradas pela metaheurística BTMO nas duas primeiras iterações. $\boldsymbol{x}_i^t$ (i = 1,2,3) são os pontos explorados na iteração t.

Observe na Figura 5.3 que, as soluções candidatas da iteração t = 1 (a, b, c, d, e) todas são dominadas pelas soluções da fronteira dominante gerada na iteração t = 2. Na Figura 5.4 ilustra-se a fronteira dominante $\boldsymbol{S}'$ gerada a partir das soluções selecionadas na iteração 2 ($\boldsymbol{x}_1^2, \boldsymbol{x}_2^2, \boldsymbol{x}_3^2$). Observe que algumas soluções desta fronteira são dominadas pelas soluções candidatas g e h. Observe também que, a solução candidata f é dominada por soluções da fronteira $\boldsymbol{S}'$. Portanto, estas soluções dominadas são descartadas. Na Figura 5.5 mostra-se a nova fronteira $\boldsymbol{S}'$ = soluções dominantes de ($\boldsymbol{S}' \cup \boldsymbol{LC}$). Note que esta fronteira inclui as soluções candidatas g e h. A partir desta fronteira $\boldsymbol{S}'$, seleciona-se as soluções de partida da iteração t = 3 ($\boldsymbol{x}_1^3, \boldsymbol{x}_2^3, \boldsymbol{x}_3^3$).

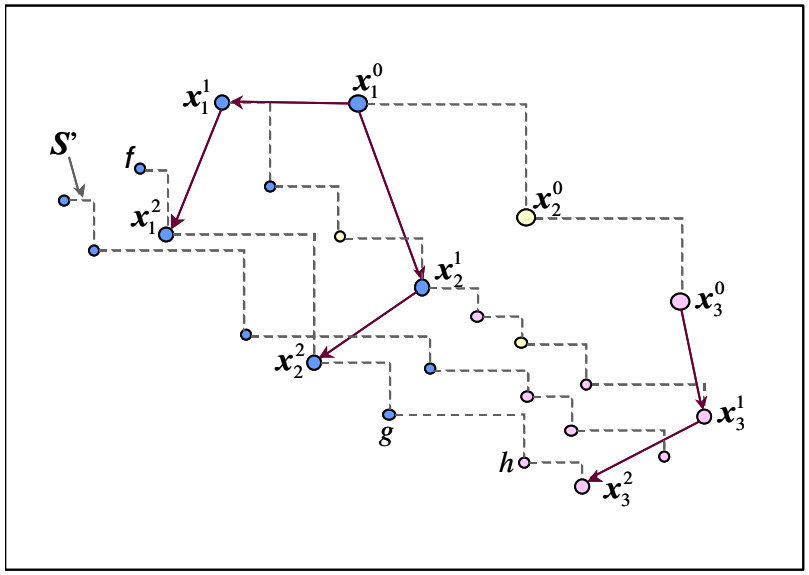


Figura 5.4. Fronteira dominante gerada a partir dos pontos x_1^2, x_2^2, e x_3^2.

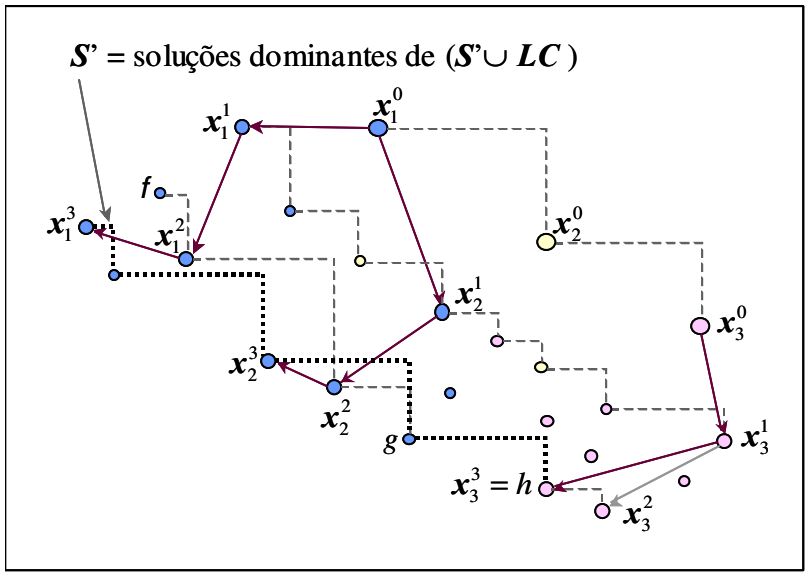


Figura 5.5. Uso das soluções candidatas na metaheurística BTMO

A metaheurística BTMO sempre explora N_{max} soluções numa iteração e cada uma com sua própria lista tabu, gerando N_{max} caminhos ou trajetórias de busca. Quando uma solução candidata é selecionada como próxima solução da trajetória, esta solução herdará a lista tabu

da solução que a gerou. Neste caso, ocorre um desvio ou modificação da trajetória de busca. Por exemplo na Figura 5.5, a solução candidata *h* herda a lista tabu da solução $\boldsymbol{x}_3^1$ (note que, *h* foi gerada a partir de $\boldsymbol{x}_3^1$). Outra característica a ser destacada na metaheurística BTMO é que algumas trajetórias podem terminar durante o transcurso da busca, por gerarem soluções dominadas pelas soluções de outras trajetórias. As trajetórias eliminadas são substituídas por novas trajetórias provenientes de trajetórias existentes. Na Figura 5.6 ilustra-se um exemplo no qual a trajetória iniciada no ponto $\boldsymbol{x}_3^0$ é finalizada, sendo substituída por uma nova trajetória que nasce do ponto $\boldsymbol{x}_2^1$. Isto ocorre, como já mencionado, quando um ponto escolhido e explorado não gera nenhuma solução vizinha na fronteira dominante. Na Figura 5.6, a solução $\boldsymbol{x}_3^1$ não possui nenhum vizinho na fronteira dominante seguinte.

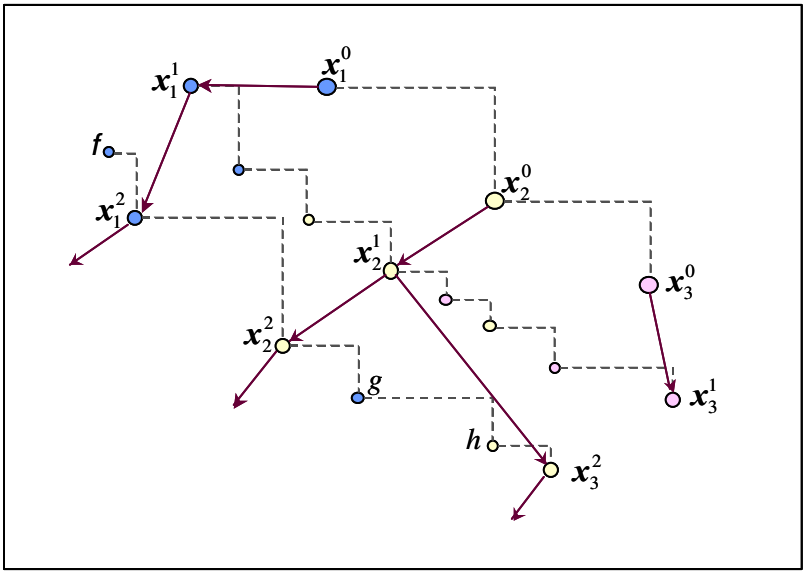


Figura 5.6. Estrutura da metaheurística BTMO

5.2.1. Pseudocódigo da Metaheurística BTMO

A metaheurística BTMO, incorporando somente memória de curto prazo, pode ser resumida no seguinte algoritmo:

Algoritmo 5.3. BTMO

Entrada: N_{max} (*número máximo de soluções a serem exploradas*)

Saída: $\boldsymbol{D}$ (*conjunto de soluções dominantes*)

1) *Inicialização*

   Construir um conjunto de N_{max} soluções dominantes $\boldsymbol{S}$.

   Faça $\boldsymbol{D} = \boldsymbol{S}$, $\boldsymbol{LC} = \varnothing$ e $t = 0$.

   Faça $\boldsymbol{LTx}_k^t = \varnothing$, $\forall\ k = 1,..., N_{max}$ (lista tabu correspondente à solução $\boldsymbol{x}_k$).

   Fase principal da busca tabu

2) Faça $\boldsymbol{S}' = \varnothing$.

3) Para cada solução $\boldsymbol{x}_k^t \in \boldsymbol{S}$ faça

4) Para cada solução vizinha $\boldsymbol{y}_j \in \boldsymbol{N}(\boldsymbol{x}_k^t)$ faça

5) Atualize o conjunto de soluções dominantes $\boldsymbol{D}$ com $\boldsymbol{y}_j$.

6) Se $A(\boldsymbol{x}_k^t, \boldsymbol{y}_j) \notin \boldsymbol{LTx}_k^t$, então faça $\boldsymbol{S}'$ = soluções dominantes de $(\boldsymbol{S}' \cup \{\boldsymbol{y}_j\})$.

7) Senão

8) Se $\boldsymbol{y}_j$ foi adicionado ao conjunto $\boldsymbol{D}$, então faça $\boldsymbol{S}'$ = soluções dominantes de $(\boldsymbol{S}' \cup \{\boldsymbol{y}_j\})$ (*Critério de Aspiração*).

9) Faca $\boldsymbol{S}'$ = soluções dominantes de $(\boldsymbol{S}' \cup \boldsymbol{LC})$ e $\boldsymbol{LC} = \varnothing$.

10) Se $|\boldsymbol{S}'| > N_{max}$ então

11) Aplicando alguma técnica de redução, reduza o conjunto $\boldsymbol{S}'$ a N_{max} soluções.

12) Armazene as soluções descartadas ou não selecionadas na lista $\boldsymbol{LC}$.

13) Para cada $\boldsymbol{y} \in \boldsymbol{S}'$ adicione o atributo $A(\boldsymbol{x}, \boldsymbol{y})$ à lista $\boldsymbol{LTx}$ ($\boldsymbol{y}$ foi gerada a partir de $\boldsymbol{x}$).

14) Faça $\boldsymbol{S} = \boldsymbol{S}'$.

15) Se a condição de parada não é satisfeita, então faça $t = t + 1$ e volte ao passo 2,
    Senão pare.

5.3. Aplicação da metaheurística BTMO para Resolver o Problema de Flowshop Multiobjetivo

Nesta seção descreve-se a implementação dos componentes da metaheurística BTMO aplicada ao problema de flowshop multiobjetivo.

Os componentes da metaheurística BTMO incluem: geração do conjunto de soluções iniciais, estratégia de geração de vizinhança, atributos e regras de proibição, duração tabu, critério de aspiração e critério de parada. Estes componentes determinam o desempenho da metaheurística e são dependentes do problema a ser resolvido.

Para definir os componentes e parâmetros do algoritmo BTMO, foi considerado um conjunto de 108 instâncias do problema de flowshop com número de tarefas n = 20, 50, 80, número de máquinas m = 5, 10, 20 e datas de entrega definidas através de quatro cenários (veja seção 2.1.2.1). Descreve-se a seguir cada um dos componentes da metaheurística BTMO.

5.3.1. Componentes da Metaheurística BTMO

Soluções Iniciais

Para obter o conjunto $\boldsymbol{S}$ de soluções iniciais para o algoritmo BTMO aplicado ao problema de flowshop multiobjetivo, foi usada a heurística multiobjetivo de enumeração parcial (HMEP), descrita no Capítulo 2 (Arroyo e Armentano 2000). Se a heurística HMEP gera um conjunto $\boldsymbol{S}$ com $|\boldsymbol{S}| > N_{max}$, então consideram-se em $\boldsymbol{S}$, N_{max} soluções. As outras soluções são armazenadas na lista de candidatos $\boldsymbol{LC}$. Se a heurística HMEP gera um conjunto $\boldsymbol{S}$ com $|\boldsymbol{S}| \leq N_{max}$, então se consideram todas as soluções em $\boldsymbol{S}$ como soluções iniciais da busca.

O parâmetro N_{max} foi fixado em 10, o que significa que a cada iteração podem ser exploradas no máximo 10 soluções gerando trajetórias distintas. Para reduzir um conjunto dominante selecionando as soluções a serem exploradas, foi utilizado o método de *redução* por *clustering* proposto por Morse (1980) (veja Algoritmo 2.5). Este método tem um bom

desempenho, em termos de tempo computacional, quando o conjunto dominante a reduzir possui poucas soluções.

Estratégia de Geração de Vizinhança

Para o bom desempenho de um algoritmo de busca local, é fundamental fazer uma escolha apropriada de vizinhança. A vizinhança estabelece uma relação entre as soluções no espaço de decisões e, geralmente, é dependente do problema abordado. Para o problema de flowshop multiobjetivo foram testadas duas estratégias de geração de vizinhança:

Seja uma seqüência de n tarefas (solução) $\boldsymbol{s} = (t_1, t_2,...,t_n)$.

- *Inserção de tarefas*: A vizinhança de $\boldsymbol{s}$, $\boldsymbol{N}(\boldsymbol{s})$ é construída, inserindo-se cada uma das tarefas t_i, $i = 1,...,n$, em todas as outras posições da seqüência. Esta estratégia gera uma vizinhança de tamanho $(n\text{-}1)^2$.
- *Troca de tarefas*: A vizinhança de $\boldsymbol{s}$, $\boldsymbol{N}(\boldsymbol{s})$ é gerada através da troca de posições de duas tarefas t_i e t_j $\forall i, j$ com $i \neq j$. Neste caso, o tamanho da vizinhança é $\frac{n}{2}(n-1)$.

Estas estratégias de geração de vizinhanças são muito usadas na literatura para o problema de flowshop de permutação (Armentano e Ronconi, 2000; Gupta et al., 2000a; Taillard, 1990).

Atributos do Movimento e Regras de Proibição

Para cada tipo de vizinhança selecionou-se alguns atributos e regras específicas de proibição. Para o movimento de inserção tem-se:

- *Inser1*: Se a tarefa t_i é inserida em alguma posição da seqüência, esta tarefa é adicionada na lista tabu, e portanto a tarefa t_i não pode ser escolhida para inserção.
- *Inser2*: Se uma tarefa t_i é inserida numa posição k da seqüência ($k \neq i$), os pares $(t_{i\text{-}1}, t_i)$ e (t_i, t_{i+1}) são armazenados na lista tabu. Neste caso, a tarefa t_i não poderá retornar entre a as tarefas $t_{i\text{-}1}$ e t_{i+1} enquanto a duração tabu não finalize. Se $i = 1$, a

tarefa t_i não poderá ser inserida na primeira posição da seqüência. Se $i = n$, a tarefa t_i não poderá ser inserida na ultima posição da seqüência.

Para o movimento de troca, considerou-se a seguinte regra:

- *Troca-ij*: Se forem trocadas as posições das tarefas t_i e t_j, estas tarefas são adicionadas na lista tabu. Portanto, as tarefas t_i e t_j não poderão ser escolhidas para trocar posições com outras tarefas.

Duração Tabu

A duração tabu (ou tamanho da lista tabu) é um parâmetro muito importante da busca tabu. Durações tabus muito pequenas podem causar ciclagem da busca, enquanto altas durações tabus podem restringir a exploração do espaço de busca. 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 $[dt_{min}, dt_{max}]$ com distribuição uniforme. Foram testados e analisados diferentes valores para dt_{min} e dt_{max}, e os valores que geraram os melhores resultados são mostrados na Tabela 5.1.

Tabela 5.1. Intervalos de geração da duração tabu

Movimento	Atributo-Proibição	Intervalo da duração tabu		
		$n < 50$	$n = 50$	$n = 80$
Inserção	*Inser1*	$[n/4, 3n/4]$	$[n/5, 5n/8]$	$[n/6, n/2]$
	Inser2	$[n/2, n]$	$[n/4, 3n/4]$	$[n/6, n/2]$
Troca	*Troca-ij*	$[n/4, n/2]$	$[n/5, n/3]$	$[n/6, n/4]$

Na Figura 5.7, para o movimento de inserção atributo *Inser2*, apresenta-se a variação da duração tabu mínima e máxima de acordo com o número de tarefas.

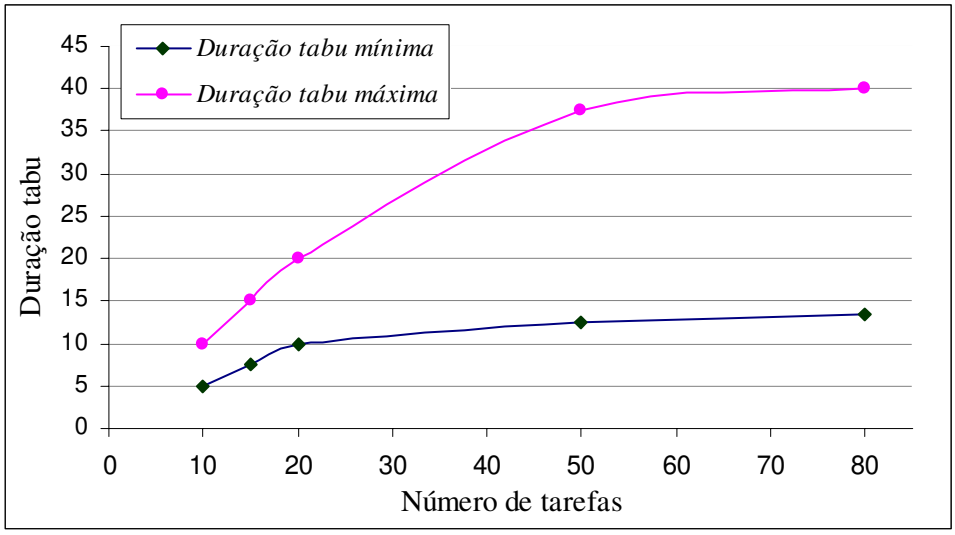


Figura 5.7. Variação da duração tabu dt_{min} e dt_{max} para o movimento de inserção atributo *Inser2*.

Critério de Aspiração

A idéia do critério de aspiração é que um movimento proibido pode ser liberado da condição tabu, caso possua potencial para conduzir a busca para regiões de boas soluções. Na implementação da metaheurística BTMO, se um movimento considerado proibido gera uma nova solução dominante, ou seja o conjunto de soluções dominantes encontradas até o momento pela busca é atualizado, então este movimento é liberado da condição tabu aceitando-se a solução gerada.

Critério de Parada

O critério de parada escolhido para a metaheurística BTMO aplicado ao problema de flowshop é o mesmo utilizado para a metaheurística AGHM apresentada no Capítulo 4. A busca termina após a avaliação de um número máximo de soluções. Com o intuito de realizar comparações com outras metaheurísticas, este número foi fixado em $(800 + 10n)n^2$, onde n é o número de tarefas. Segundo os testes experimentais realizados, observou-se que a avaliação deste número de soluções era suficiente para um bom desempenho da metaheurística.

5.3.2. Avaliação da Metaheurística BTMO Implementada com as Diferentes Estratégias de Geração de Vizinhança

Nesta seção avalia-se o desempenho da metaheurística BTMO, aplicado ao problema de flowshop para a minimização do par de objetivos (C_{max}, T_{max}), com as duas estratégias de geração de vizinhança e os respectivos atributos de movimento e regras de proibição, descritos acima. As implementações da metaheurística BTMO com a vizinhança de inserção e os dois atributos *Inser1* e *Inser2* são denotados por BTMO-*Inser1* e BTMO-*Inser2*, respectivamente. BTMO-*Troca-ij* denota a implementação da metaheurística BTMO considerando a vizinhança de troca e o atributo *Troca-ij*. As três versões de implementação da metaheurística BTMO foram avaliadas sobre um conjunto de problemas cujas dimensões são: número de tarefas n = 20, 50, 80 e número de máquinas m = 5, 10, 20. Para cada combinação de n e m, gerou-se três diferentes matrizes de tempos de processamento, e para cada uma destas foram selecionadas quatro distintos cenários de datas de entrega (veja seção 2.1.2.1), ou seja foram testados 9×3×4 = 108 problemas.

Vale ressaltar que na implementação das três versões da metaheurística BTMO, foram exploradas no máximo N_{max} = 10 soluções em paralelo. O desempenho das três versões da metaheurística BTMO é avaliado usando as três metodologias de avaliação descritas no Capítulo 1: medidas de cardinalidade, medidas de distância proposta por Czyzak e Jaszkiewicz (1998) e Ulungu et al. (1998) e erro de utilidade de Daniels (1992).

Na Tabela 5.2, mostra-se o número de soluções dominantes no conjunto de referência $\boldsymbol{R}$, o número total de soluções e o número de soluções dominantes obtidas por cada versão da metaheurística BTMO. Para as 108 instâncias testadas, foram encontradas 2886 soluções dominantes que são as soluções de referência. As versões BTMO-*Inser1*, BTMO-*Inser2* e BTMO-*Troca-ij* obtiveram, respectivamente, 1981, 2257 e 1150 soluções dominantes.

Tabela 5.2. Número de soluções encontradas pelas três versões da metaheurística BTMO

$\lvert\boldsymbol{R}\rvert$	BTMO-*Inser1*		BTMO-*Inser2*		BTMO-*Troca-ij*	
	NTS	NSD	NTS	NSD	NTS	NSD
2886	2671	1981	2696	2257	2273	1150

$|\boldsymbol{R}|$: número de soluções de referência.
NTS: número total de soluções obtidas.
NSD: número de soluções dominantes obtidas.

Na Figura 5.8 mostra-se a média de número de soluções dominantes, para cada cenário de datas de entrega, obtido pelas três versões da metaheurística BTMO. Observa-se que, a média de número de soluções dominantes é maior para os cenários 2 e 4, e a versão BTMO-*Inser2* identificou maior quantidade de soluções dominantes nos cenários 1, 2 e 4. Para o cenário 3, a média de número de soluções obtida pela versão BTMO-*Inser1* é ligeiramente maior que a média obtida por BTMO-*Inser2*. Note que, a quantidade de soluções dominantes obtidas pela versão BTMO-*Inser1* é bem próxima à obtida pela versão BTMO-*Inser2*.

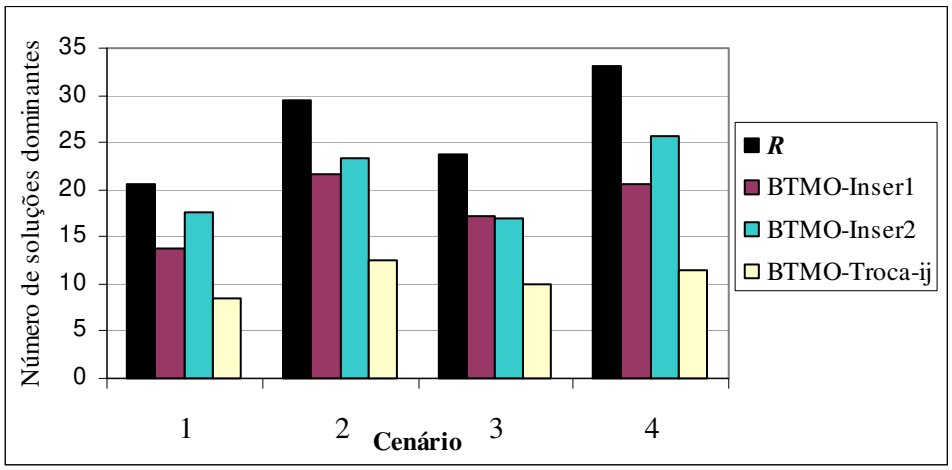
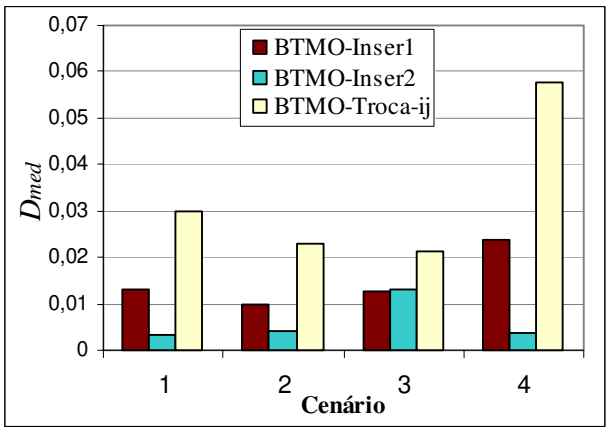
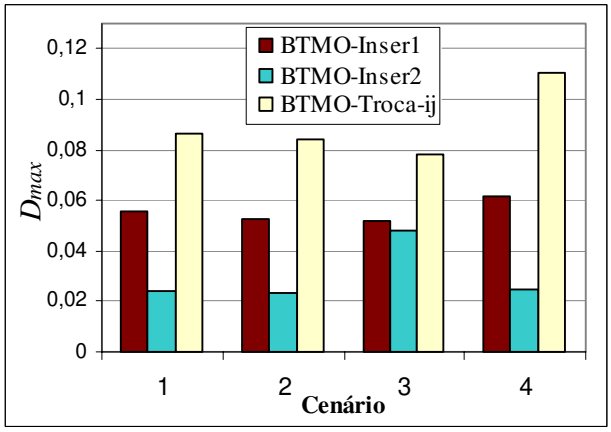


Figura 5.8. Média de número de soluções dominantes por cenário obtidas pelas três versões da metaheurística BTMO.

O desempenho das três versões da metaheurística BTMO é avaliado de acordo com as medidas de distância D_{med} e D_{max} e erros de utilidade de E_{med} e E_{max}. Nas Figuras 5.9 (a) e (b) ilustram-se os valores médios de D_{med} e D_{max}, sobre as 108 instâncias. As Figuras 5.10 (a) e (b) ilustram os valores médios de E_{med} e E_{max}. Observe que BTMO-*Inser2*, geralmente, possui melhor desempenho em todos os cenários e em relação às duas medidas. A versão BTMO-*Troca-ij* apresenta o pior desempenho, para todos os cenários, em relação às duas medidas.

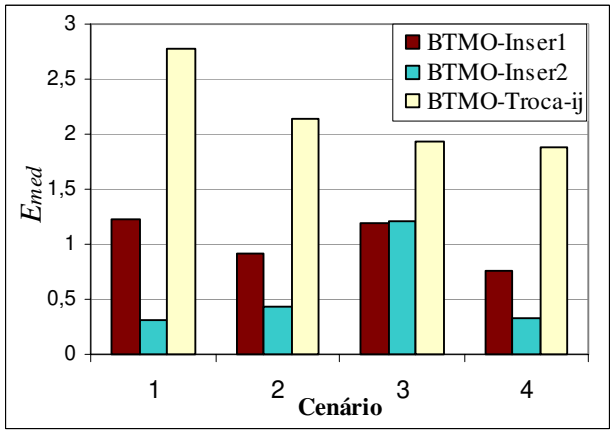


(a)

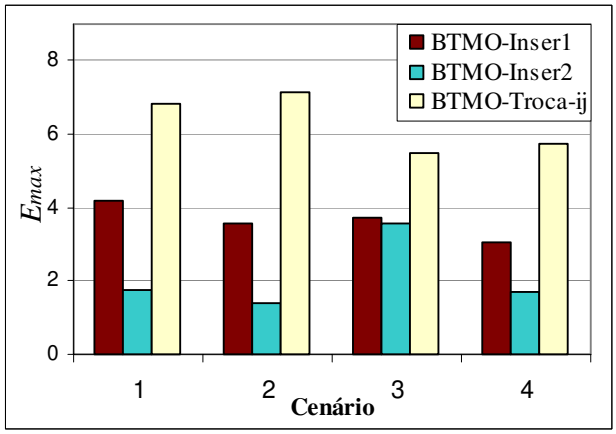


(b)

Figura 5.9. Distância média e máxima por cenário de data de entrega.



(a)



(b)

Figura 5.10. Erro de utilidade médio e máximo por cenário de data de entrega.

Para as três implementações da metaheurística BTMO foram analisados o comportamento dos menores valores do makespan (C_{max}) e atraso máximo (T_{max}) (pontos extremos da fronteira dominante obtida). Das 108 instâncias testadas, as versões BTMO-*Inser1*, BTMO-*Inser2* e BTMO-*Troca-ij* obtiveram o menor valor de C_{max} em 71, 87 e 44 instâncias e o menor valor de T_{max} em 88, 95 e 54 instâncias, respectivamente.

Das avaliações realizadas acima se pode concluir que a versão BTMO-*Inser2* apresenta o melhor desempenho seguido pela versão BTMO-*Inser1*.

Para observar o comportamento gráfico da metaheurística BTMO-*Inser2*, considera-se a exploração de N_{max} = 6 soluções em paralelo. Na Figura 5.11 ilustra-se o comportamento da metaheurística BTMO-*Inser2* nas nove primeiras iterações para uma instância com n = 20 e m = 10. Observe que, a metaheurística melhora rapidamente a fronteira inicial (obtida pela heurística construtiva HMEP) obtendo em cada iteração soluções dominantes bem distribuídos.

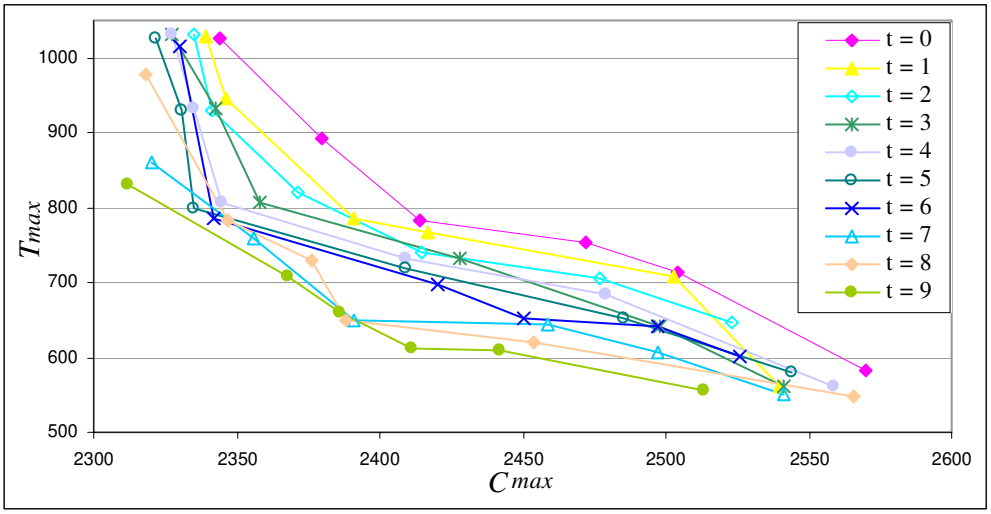


Figura 5.11. Fronteiras dominantes ***S***' obtidas pela metaheurística BTMO-*Inser2*.

As Figuras 5.12 (a) e (b) ilustram as variações dos objetivos C_{max} e T_{max} quando é explorada uma única solução (N_{max} = 1). Note que, quando o valor de um objetivo melhora o outro objetivo tende a piorar.

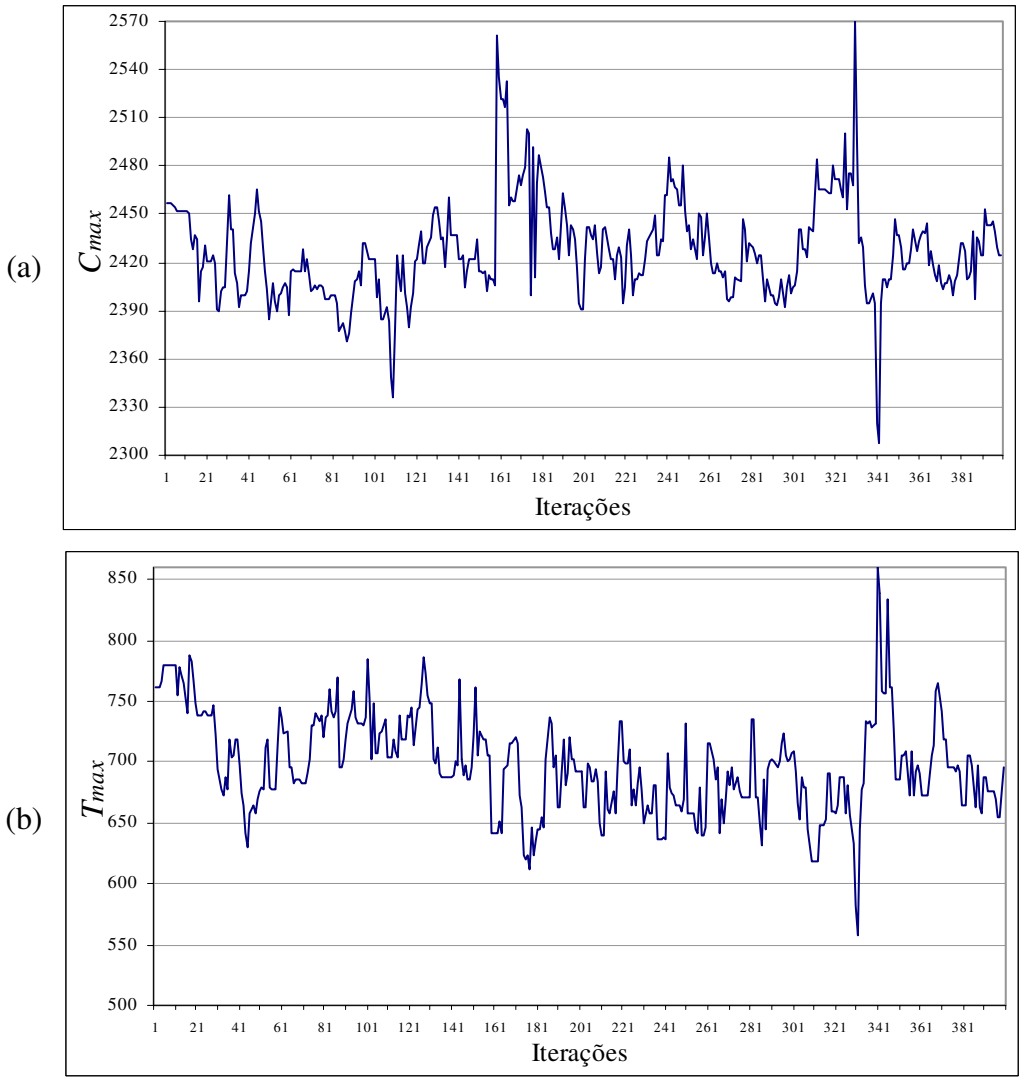


Figura 5.12. Variação dos objetivos na exploração de uma trajetória de busca.

5.3.3. Implementação de uma Estratégia de Diversificação

A estratégia de diversificação consiste em conduzir a busca para regiões ainda não exploradas. Na implementação da metaheurística BTMO-*Inser2* para o problema de flowshop, foi incorporada uma estratégia de diversificação na qual as soluções geradas através de movimentos muito freqüentes são penalizadas, enquanto as soluções geradas através de movimentos pouco freqüentes são incentivadas; o objetivo é explorar soluções com novas características.

A estratégia de diversificação é baseada na freqüência de ocorrência de cada tarefa inserida entre duas tarefas. A freqüência de inserção de uma tarefa t_j entre as tarefas t_i e t_k é armazenada em uma matriz de residência $\boldsymbol{M} = (M_{l,c})$ de ordem $(n+1)\times(n+1)$ ilustrada na Figura 5.13.

Tarefa	1	2	. . .		n	n+1
0	3	0	0	2	0	-
1	-	0	7	4	0	1
2	1	-	0	0	4	10
.	0	0	-	2	0	0
.	0	5	0	-	0	0
n	3	0	0	5	-	0

Tarefa

Figura 5.13. Matriz de residência.

A matriz de residência é zerada no inicio da busca. As posições $M_{i,j}$ e $M_{j,k}$, da matriz são incrementadas em uma unidade toda vez que a tarefa t_j é inserida entre as tarefas t_i e t_k. São consideradas duas tarefas fictícias t_0 e t_{n+1}. Quando uma tarefa t_j é inserida na primeira posição da seqüência e antecedendo a tarefa t_k, então as posições $M_{0,j}$ e $M_{j,k}$ da matriz são incrementadas em uma unidade. Analogamente, quando uma tarefa t_j é inserida na ultima posição da seqüência e sucedendo a tarefa t_i, então as posições $M_{i,j}$ e $M_{j,n+1}$ da matriz são incrementadas em uma unidade.

Se uma solução é obtida, inserindo uma tarefa t_j entre as tarefas t_i e t_k, então para esta solução calcula-se uma medida de penalidade baseada na freqüência de ocorrência da tarefa t_j entre as tarefas t_i e t_k. Esta medida de penalização é calculada da seguinte maneira:

$$Pen = \frac{M_{i,j} + M_{j,k}}{\max M_{l,c}},$$

onde $\max M_{l,c}$ é a maior freqüência da matriz $\boldsymbol{M}$.

Os valores dos r objetivos da solução são alterados para novos valores penalizados, de seguinte maneira:

$$f_k^* = f_k + d_k \times Pen, \ k = 1,...,r.$$

onde f_k é valor original do objetivo k e d_k é um parâmetro ajustável da diversificação associado com o objetivo k. Grandes valores de d_k acarretam um peso maior para a freqüência de ocorrência de uma tarefa dentre duas tarefas. Para os parâmetros d_k, foram testados vários valores que estão relacionados com o valor original do objetivo k. Foram obtidos bons resultados adotando $d_k = 1+ 0{,}10\times f_k$, $k = 1,...,r$.

O processo de diversificação é acionado depois de *Iini* iterações sem atualização do conjunto de soluções dominantes, e este processo é interrompido logo após a execução de *Ifim* iterações consecutivas. Foram testados vários valores para *Iini* e *Ifim*. Os valores que apresentaram melhores resultados foram *Iini* = 12 e *Ifim* = 2. Nas análises realizadas observou-se que, a qualidade das soluções dominantes é deteriorada quando o processo de diversificação é executado por mais de duas iterações. Na Figura 5.14 ilustra-se o comportamento da diversificação ativada após a iteração t = 54 da busca tabu. Nesta Figura mostra-se a execução de três iterações da diversificação. Observa-se que, a busca é guiada para regiões pouco exploradas deteriorando a qualidade das soluções dominantes. Esta estratégia de diversificação pode ser vista como um reinício da busca com uma nova fronteira dominante. Quando a execução da diversificação é interrompida, a busca é iniciada a partir da fronteira construída na última iteração da diversificação. A busca tentará novamente melhorar a qualidade das soluções dominantes pertencentes às próximas fronteiras. Na Figura 5.15 ilustram-se os pontos dominantes gerados na ultima iteração pela busca tabu BTMO-*Inser2* sem diversificação e com diversificação. Neste exemplo, os pontos A, B, C, D gerados pela busca tabu sem diversificação são dominados respectivamente pelos pontos a, b, c, c, gerados pela busca tabu com diversificação. Note também que o ponto d gerado pela busca tabu com diversificação é dominado pelo ponto E gerado pela busca tabu sem diversificação. As Figuras 5.16 (a) e (b) ilustram as variações dos objetivos C_{max} e T_{max} quando a busca tabu com diversificação explora uma única solução. Observe que em relação às Figuras 5.12 (a) e (b), nos instantes em que a diversificação é acionada, ocorre uma degradação dos objetivos.

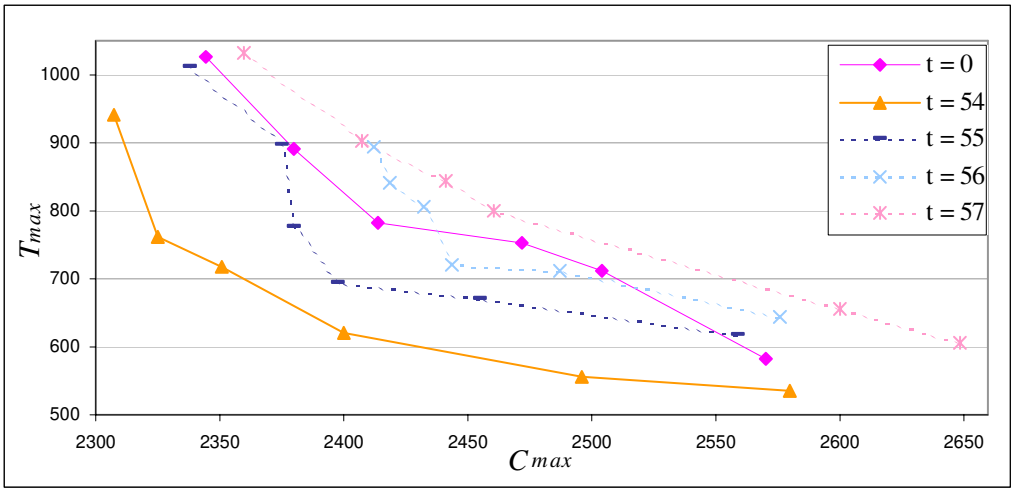


Figura 5.14. Comportamento da estratégia de diversificação.

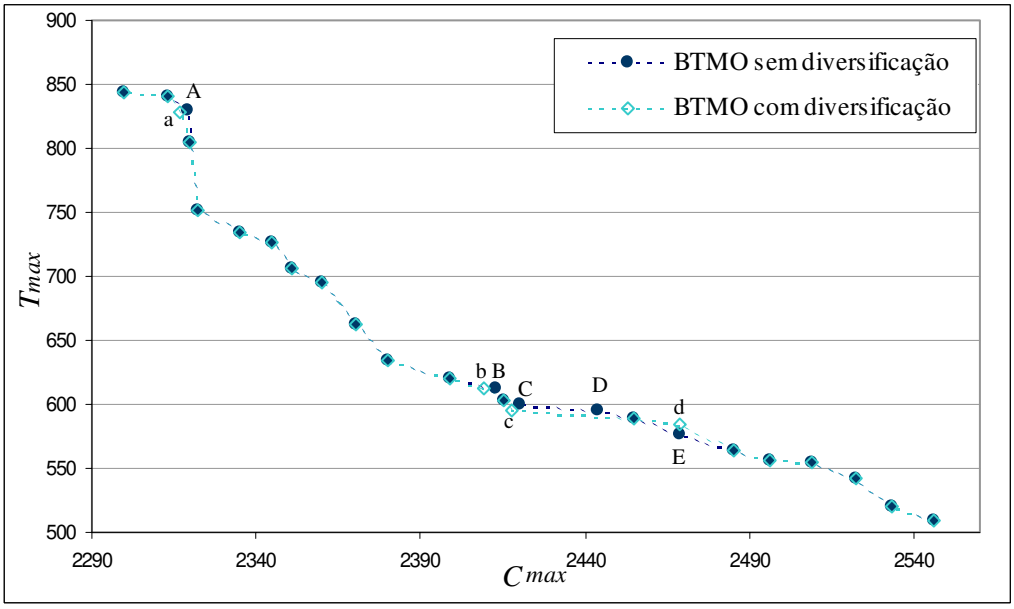
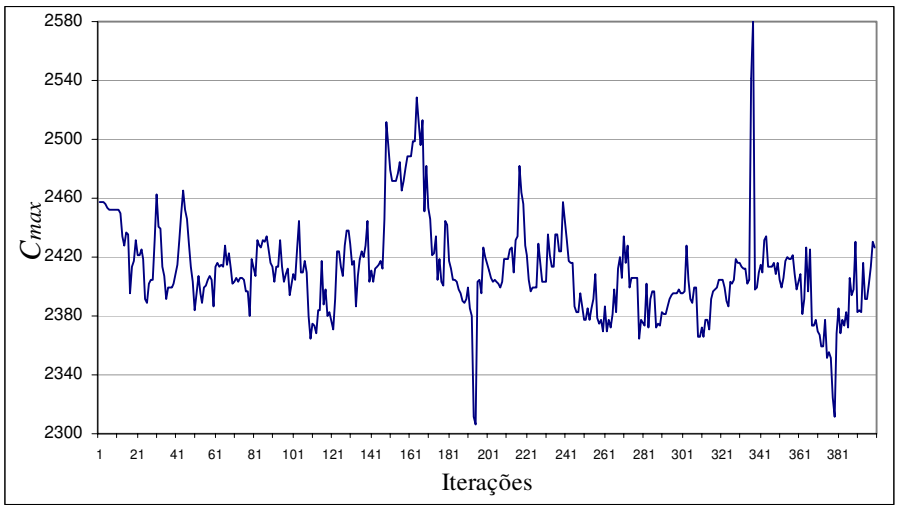


Figura 5.15. Pontos dominantes gerados pela busca tabu BTMO-*Inser2* sem diversificação e com diversificação.

(a)



(b)

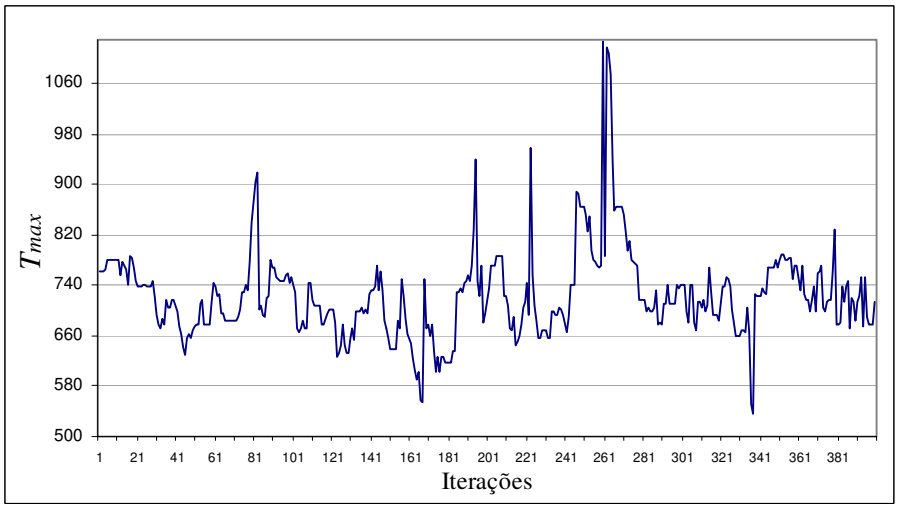


Figura 5.16. Comportamento da busca com diversificação - variação dos objetivos na exploração de uma trajetória de busca.

Apresenta-se a seguir os resultados computacionais da metaheurística BTMO-*Inser2* com diversificação. A avaliação da metaheurística foi realizada considerando o mesmo conjunto de 108 problemas usados na seção 5.3.2. São comparados os desempenhos da metaheurística BTMO-*Inser2* sem diversificação e com diversificação.

Na Tabela 5.3, mostra-se o número de soluções dominantes no conjunto de referência ***R***, o número total de soluções e o número de soluções dominantes obtidas por cada versão da metaheurística BTMO-*Inser2*. Para as 108 instâncias testadas, foram encontradas 2915

soluções dominantes que são as soluções de referência. As versões BTMO-*Inser2* e BTMO-*Inser2* com diversificação obtiveram, respectivamente, 2268 e 2374 soluções dominantes.

Tabela 5.3. Número de soluções encontradas pela metaheurística BTMO-Inser2 sem diversificação e com diversificação.

$\|R\|$	BTMO-*Inser2*		BTMO-*Inser2* com Diversificação	
	NTS	NSD	NTS	NSD
2915	2696	2268	2708	2374

$|R|$: número de soluções de referência.
NTS: número total de soluções obtidas.
NSD: número de soluções dominantes obtidas.

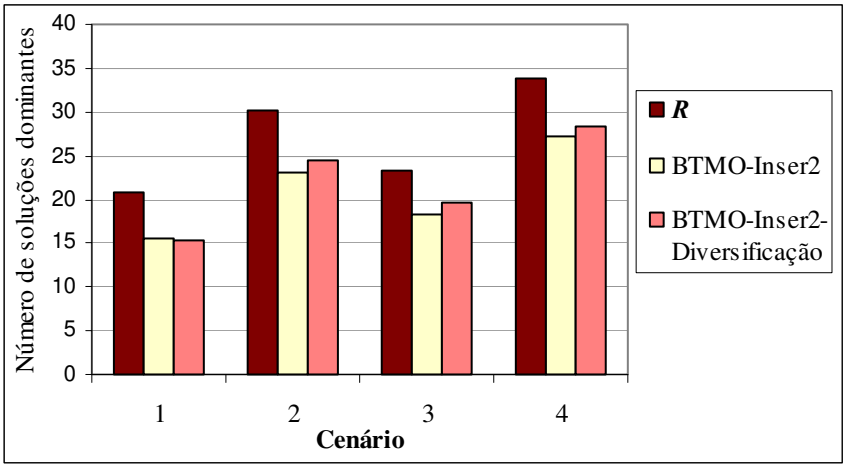


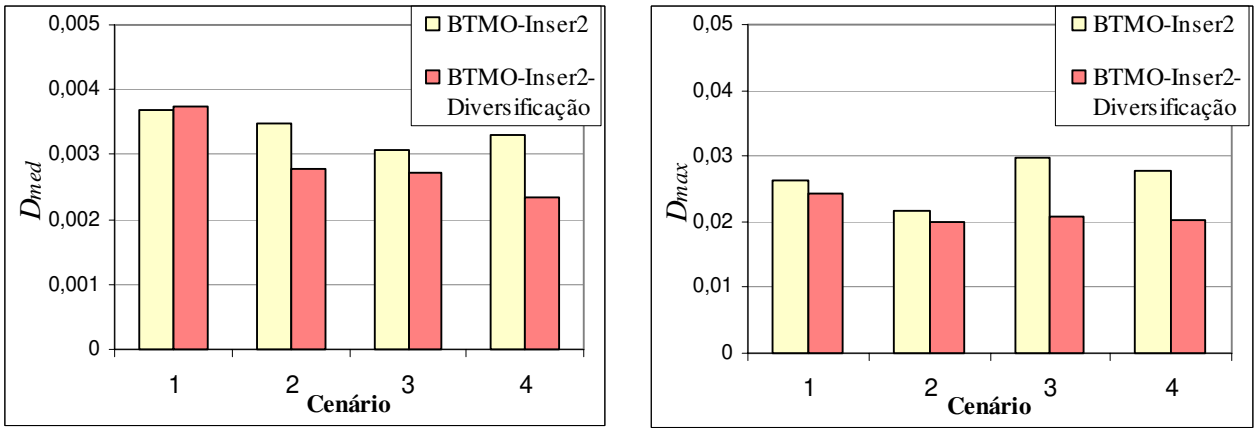
Figura 5.17. Média de número de soluções dominantes por cenário.

Na Figura 5.17 mostra-se a média de número de soluções dominantes, para cada cenário de datas de entrega, obtida pelas versões BTMO-*Inser2* e BTMO-*Inser2* com diversificação. Observa-se que, a versão BTMO-*Inser2* com diversificação identificou maior quantidade de soluções dominantes nos cenários 2, 3 e 4. Para o cenário 1, a média de número de soluções obtida pela versão BTMO-*Inser2* sem diversificação é ligeiramente maior. Note que, a média do número de soluções dominantes obtidas pelas duas versões são muito próximas.

O desempenho da metaheurística BTMO-*Inser2* sem diversificação e com diversificação é avaliado de acordo com as medidas de distância D_{med} e D_{max} e erros de utilidade de E_{med} e E_{max}. Nas Figuras 5.18 (a) e (b) ilustram-se os valores médios de D_{med} e

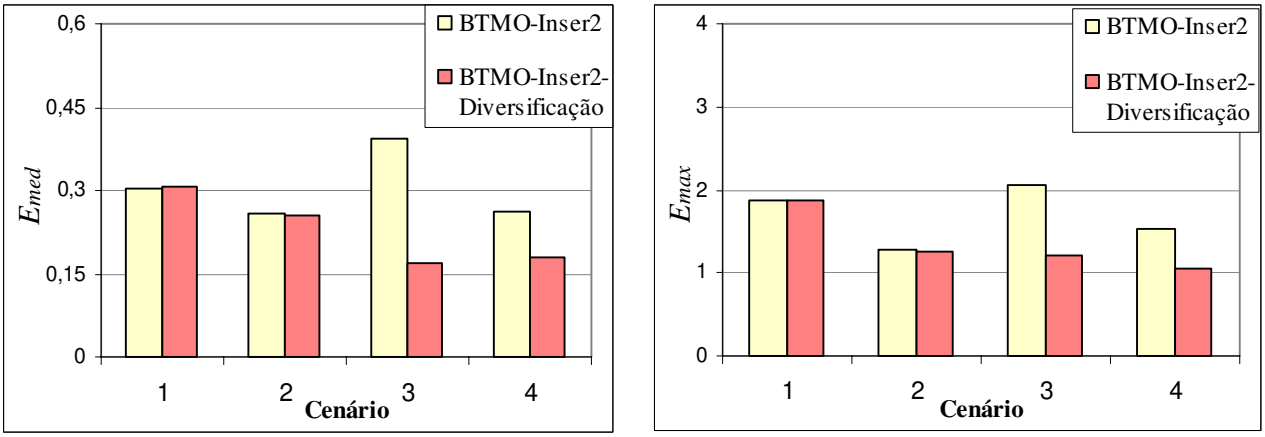
D_{max}, sobre as 108 instâncias. As Figuras 5.19 (a) e (b) ilustram os valores médios de E_{med} e E_{max}. Observe que, BTMO-*Inser2* com diversificação possui melhor desempenho nos cenários 2, 3 e 4.

Também foi analisado o comportamento das duas versões de BTMO-*Inser2* em obter os menores valores do makespan (C_{max}) e atraso máximo (T_{max}). Das 108 instâncias testadas, a metaheurística BTMO-*Inser2* sem diversificação obteve os menores valores de C_{max} e T_{max} em 84 e 95 instâncias, respectivamente. BTMO-*Inser2* com diversificação obteve os menores valores de C_{max} e T_{max} em 87 e 103 instâncias, respectivamente.



(a) (b)

Figura 5.18. Distância média e máxima por cenário de data de entrega.



(a) (b)

Figura 5.19. Distância média e máxima por cenário de data de entrega.

Através dos testes realizados acima, pode-se constatar que a incorporação da estratégia de diversificação na metaheurística BTMO-*Inser2* influencia no aumento do número de soluções dominantes e na qualidade das soluções dominantes. Mesmo que as melhorias médias sejam pequenas, a estratégia de diversificação implementada mostrou um bom desempenho para o problema de flowshop.

Vale lembrar que a estratégia de diversificação não causa efeito considerável no tempo de execução da busca tabu. A operação básica da diversificação consiste em, a cada iteração, atualizar algumas posições da matriz de residência identificando sempre a maior freqüência da matriz. Esta operação, computacionalmente, é executada em tempo constante, ou seja, produz um acréscimo insignificante no tempo total da metaheurística.

Na próxima seção apresentam-se extensos testes computacionais no qual a metaheurística BTMO-*Inser2* com diversificação é comparada com métodos exatos, enumeração completa e Branch-and-Bound, e com a busca tabu multiobjetivo proposta por Hansen (1997).

5.4. Resultados Computacionais para o Problema de Flowshop Multiobjetivo

Nesta seção apresentam-se os resultados computacionais obtidos pela metaheurística BTMO aplicada ao problema de flowshop. A metaheurística é implementada considerando a vizinhança de inserção com o atributo *Inser2* e com a estratégia de diversificação descrita na seção anterior. O desempenho da metaheurística é testado na minimização dos seguintes pares de objetivos: i) (C_{max}, T_{max}) e ii) (C_{max}, T). São considerados três conjuntos de instâncias:

- ***Conjunto 1***: n = 10, 11; m = 5, 10; 20 matrizes de tempo de processamento; 4 cenários de datas de entrega ; totalizando 4×20×4 = 320 instâncias.
- ***Conjunto 2***: n = 10, 15, 20, 25; m = 2; 10 matrizes de tempo de processamento; 4 cenários de datas de entrega; totalizando 4×10×4 =160 instâncias.
- ***Conjunto 3***: n = 20, 50, 80; m = 5, 10, 20; 10 matrizes de tempo de processamento; 4 cenários de datas de entrega; totalizando 9×10×4 = 360 instâncias.

Para as instâncias do primeiro conjunto, o desempenho da metaheurística BTMO é comparado com enumeração completa. Para as instâncias do segundo conjunto (instâncias com duas máquinas) a busca tabu BTMO é comparada com os algoritmos Branch-and-Bound propostos, respectivamente, por Daniels e Chambers (1990) e Liao et al. (1997) (veja Apêndice B). Para instâncias de grande porte (pertencentes ao conjunto 3) a metaheurística proposta é comparada com a busca tabu multiobjetivo proposta por Hansen (1997), denotada por BTH.

Na implementação da metaheurística BTH adaptada para o problema de flowshop multiobjetivo, considerou-se os seguintes elementos:

- *Número de soluções a serem exploradas*: $N_{max} = 10$.
- *Conjunto de soluções iniciais*: Obtido pela heurística construtiva de enumeração parcial HMEP. Se a heurística HMEP gera um conjunto com menos de N_{max} soluções, o conjunto inicial é completado com soluções geradas aleatoriamente.
- *Vizinhança e atributo de movimento*: Inserção de tarefas – *Inser2*.
- *Duração tabu*: A mesma usada em BTMO (veja Tabela 5.1).
- *Ativação da diversificação*: Após 30 iterações sem atualização do conjunto de soluções dominantes.
- *Critério de parada*: O mesmo usado em BTMO, avaliação de no máximo $(800 + 10n)n^2$ soluções.

As metaheurísticas e os métodos exatos foram implementados na linguajem de programação C++. Os testes computacionais foram executados em uma estação de trabalho SUN Ultra 60.

5.4.1. Resultados Computacionais para $f = (C_{max}, T_{max})$

Nesta seção apresentam-se os resultados computacionais obtidos na solução do problema de flowshop considerando o par de objetivos (C_{max}, T_{max}). Os desempenhos das metaheurísticas implementadas foram testados sobre os três conjuntos de instancias, geradas como descrito no Capítulo 2, seção 2.1.2.1.

5.4.1.1. Resultados Para o Conjunto 1 de Instâncias

Os resultados das metaheurísticas BTMO e BTH são comparados com os resultados ótimos obtidos pela enumeração completa. Na Tabela 5.4 apresenta-se, para cada tamanho de problema, o número de soluções eficientes e o número de soluções obtidas pelas metaheurísticas BTMO e BTH associadas com 80 instâncias. Nas 320 instâncias, a enumeração completa obteve 3004 soluções eficientes. As metaheurísticas BTMO e BTH identificaram 2966 (97,80%) e 2873 (95,63%) soluções eficientes, respectivamente. Além disso, as metaheurísticas encontraram o conjunto Pareto-ótimo para 271 e 247 instâncias, respectivamente. Observe que, em cada grupo de problemas, a metaheurística BTMO obteve maior quantidade de soluções eficientes.

Tabela 5.4. Número de soluções encontradas pela Enumeração Completa e pelas metaheurísticas comparadas

Problema	EC	BTMO		BTH	
$n \times m$	NSE	NTS	NSEM	NTS	NSEM
10×5	582	577	572	564	559
10×10	786	778	778	776	767
11×5	687	679	668	673	647
11×10	949	932	920	927	900
Total	3004	2966	2938	2940	2873

NSE: número de soluções eficientes encontradas pelo método exato.
NTS: número total de soluções encontradas pela metaheurística.
NSEM: número de soluções eficientes encontradas pela metaheurística.

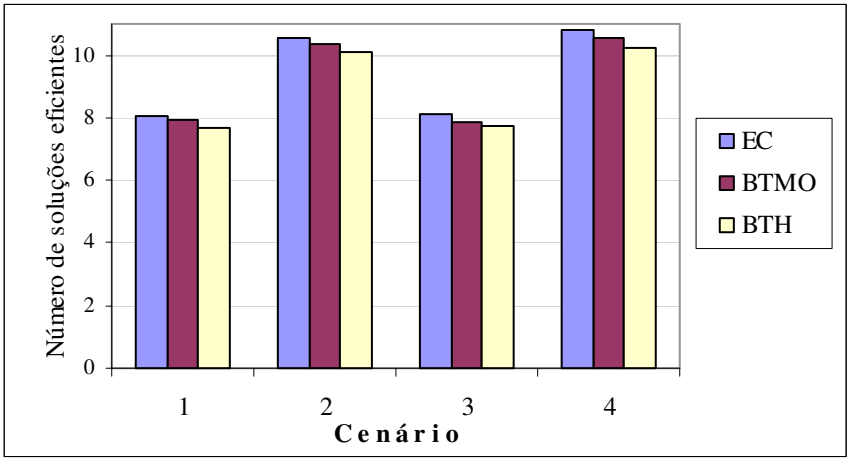


Figura 5.20. Média de número de soluções eficientes por cenário.

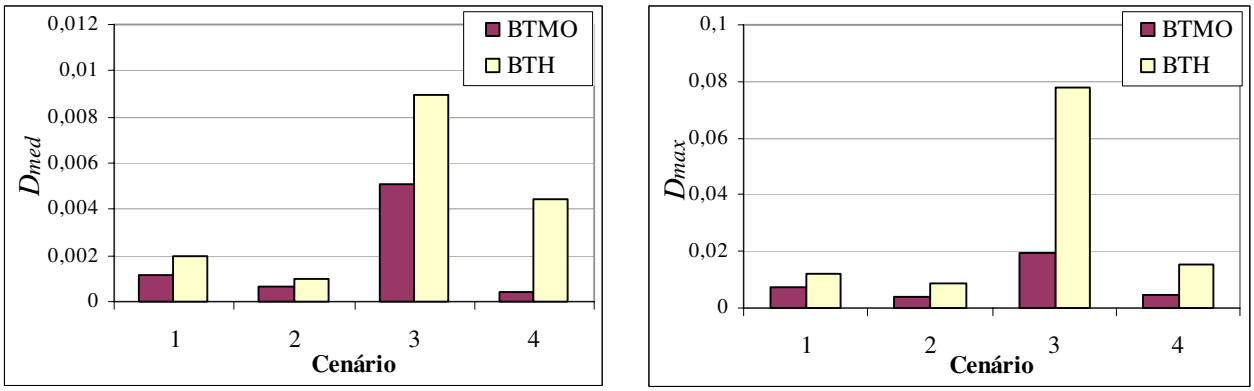
Para cada cenário de datas de entregas, a Figura 5.20 mostra a média do número de soluções eficientes obtida pelo método exato EC e pelas metaheurísticas. Pode-se observar que, a metaheurística BTMO identificou uma média maior de número de soluções eficientes em todos os cenários.

Os desempenhos das metaheurísticas BTMO e BTH em relação às medidas de distância e erro de utilidade de Daniels (1992), são mostrados na Tabela 5.5. Note que, a metaheurística BTMO apresenta melhor desempenho considerando as duas medidas.

Tabela 5.5. Desempenho das metaheurísticas BTMO e MTH

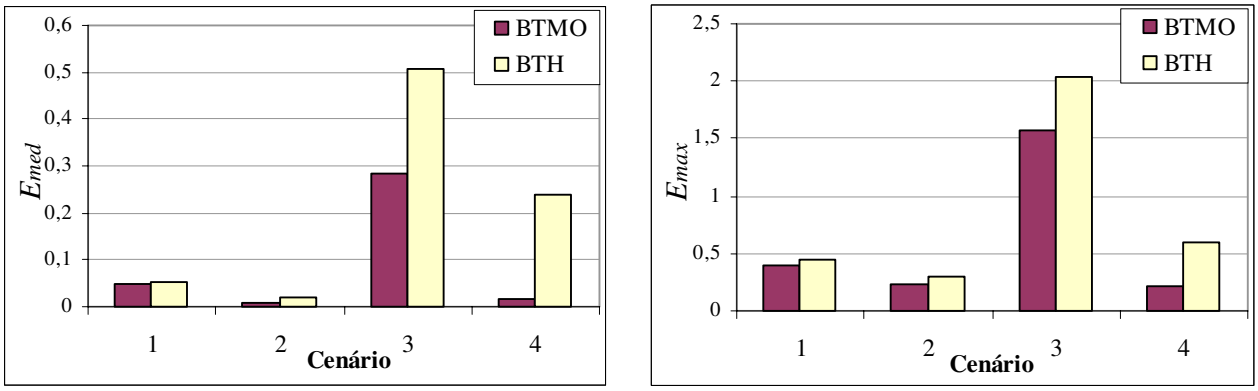
$n \times m$	Medida de distância				Erro de utilidade (%)			
	BTMO		BTH		BTMO		BTH	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×5	0,0029	0,0110	0,0060	0,0178	0,137	1,016	0,381	1,470
10×10	0,0002	0,0021	0,0008	0,0071	0,006	0,102	0,013	0,171
11×5	0,0011	0,0061	0,0063	0,0194	0,129	0,528	0,393	1,327
11×10	0,0030	0,0150	0,0033	0,0171	0,099	0,767	0,116	0,749
Média	0,0018	0,0086	0,0041	0,0154	0,093	0,603	0,226	0,929

Para cada cenário de datas de entrega, as Figuras 5.21 (a) e (b) ilustram os valores médios de D_{med} e D_{max}, enquanto as Figuras 5.22 (a) e (b) ilustram os valores médios de E_{med} e E_{max}. Segundo a análise por cenários, nota-se que a metaheurística BTMO possui melhor desempenho em todos os cenários. Observe que, as metaheurística BTMO e BTH apresentam valores mais altos de distâncias e erros de utilidade no cenário 3 que está associado com pequenas faixas de datas de entrega.



(a) (b)

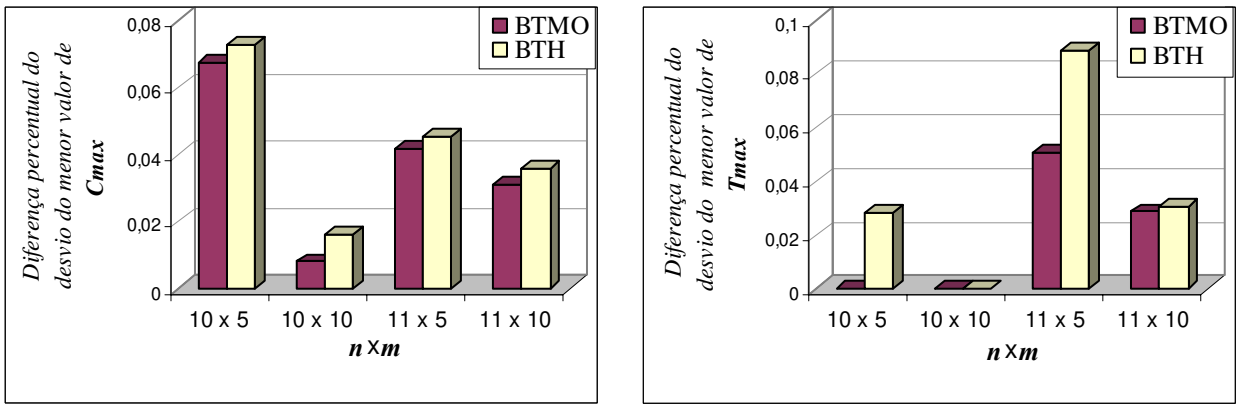
Figura 5.21. Distância média e máxima por cenário de data de entrega.



(a) (b)

Figura 5.22. Erro de utilidade médio e máximo por cenário de data de entrega.

Para cada instância, o menor valor do makespan (C_{max}) e atraso máximo (T_{max}) obtidos pelas metaheurísticas são comparados com os respectivos valores ótimos. Do total de 320 instâncias, as metaheurísticas BTMO e BTH obtiveram o valor ótimo de C_{max} em 301 e 299 instâncias e o valor ótimo de T_{max} em 318 e 314 instâncias, respectivamente. Para cada tamanho de problema, as Figuras 5.23 (a) e (b) mostram a diferença percentual média do desvio dos menores valores de C_{max} e T_{max}, obtidas pelas metaheurísticas, em relação aos valores ótimos obtidos pela enumeração completa. Observe que, para cada de tamanho de problema, a metaheurística BTMO obteve, em média, os menores valores de C_{max} e T_{max}. BTMO identificou os valores ótimos de T_{max} para todas as instâncias com 10 tarefas.



(a) (b)

Figura 5.23. Diferença percentual média do desvio dos menores valores de C_{max} e T_{max} obtidas pelas metaheurísticas em relação aos valores ótimos.

Na Tabela 5.6 são mostrados os tempos computacionais gastos pelas metaheurísticas BTMO e BTH.

Tabela 5.6. Média de tempos computacionais (em segundos)

n×*m*:	10×5	10×10	11×5	11×10
BTMO	0,577	0,934	0,852	1,322
BTH	0,624	0,857	0,813	1,214

5.4.1.2. Resultados Para o Conjunto 2 de Instâncias

Os resultados obtidos pelas metaheurísticas BTMO e BTH também são comparados com os resultados ótimos gerados pelo algoritmo Branch and Bound (B&B) desenvolvido por Daniels e Chamber (1990). A Tabela 5.7 mostra, para cada tamanho de problema, o número de soluções obtidas pelo algoritmo B&B e pelas metaheurísticas. De um total de 160 instâncias testadas foram obtidas 507 soluções eficientes pelo algoritmo B&B. As metaheurísticas BTMO e BTH encontraram 499 (98,42%) e 477 (94,08%) soluções eficientes e identificaram exatamente o conjunto Pareto-ótimo para 155 e 145 instâncias, respectivamente. Observe que as metaheurísticas BTMO e BTH encontraram o conjunto Pareto-ótimo em todos os problemas com 10 tarefas. BTMO também obteve o conjunto Pareto-ótimo em todos os problemas com 15 tarefas.

Tabela 5.7. Número de soluções encontradas pela Enumeração Completa e pelas metaheurísticas comparadas

Problema	B&B	BTMO		BTH	
n×*m*	NSE	NTS	NSEM	NTS	NSEM
10×2	122	122	122	122	122
15×2	140	140	140	138	131
20×2	144	142	138	140	129
25×2	101	101	99	100	95
Total	507	505	499	500	477

Os desempenhos das metaheurísticas de acordo com a medida de distância e erro de utilidade são mostrados nas Tabelas 5.8. A metaheurística BTMO apresenta melhor desempenho segundo as duas medidas.

Tabela 5.8. Desempenho das metaheurísticas

$n \times m$	Medida de distância				Erro de utilidade (%)			
	BTMO		BTH		BTMO		BTH	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×2	0	0	0	0	0	0	0	0
15×2	0	0	0,0077	0,225	0	0	0,685	1,964
20×2	0,0038	0,0148	0,0213	0,048	0,266	0,994	1,219	3,669
25×2	0,0027	0,009	0,0044	0,016	0,281	0,855	0,394	1,281
Média	0,0016	0,0060	0,0084	0,0723	0,13675	0,46225	0,575	1,729

Para cada instancia deste conjunto, o menor valor do makespan (C_{max}) e atraso máximo (T_{max}) obtidos pelas metaheurísticas são comparados com os respectivos valores ótimos. Do total de 160 instâncias testadas, as metaheurísticas BTMO e BTH obtiveram o valor ótimo de C_{max} em todas as instâncias e o valor ótimo de T_{max} em 158 e 155 instâncias, respectivamente.

Na Tabela 5.9 são mostrados os tempos computacionais gastos pelas metaheurísticas BTMO e BTH.

Tabela 5.9. Média de tempos computacionais (em segundos)

Problema $n \times m$	BTMO	BTH
10×2	0,383	0,333
15×2	1,201	1,023
20×2	2,426	2,344
25×2	4,384	4,412

5.4.1.3. Resultados Para o Conjunto 3 de Instâncias

Os desempenhos das metaheurísticas BTMO e BTH foram avaliados em instâncias de grande porte pertencentes a este conjunto. Para obter o conjunto de soluções de referência $\boldsymbol{R}$, determinam-se as soluções dominantes dentre todas as soluções encontradas por ambas metaheurísticas.

Na Tabela 5.10, para cada tamanho de problema mostra-se o número de soluções de referência, e para cada metaheurística mostra-se o número total de soluções e o número de soluções dominantes obtidas. Para as 360 instâncias testadas, foram encontradas 10119 soluções de referência. As metaheurísticas BTMO e BTH obtiveram, respectivamente, 9033 e 2127 soluções dominantes. Note que, a metaheurística BTMO gerou a maior quantidade de soluções dominantes para todos os tamanhos de problemas. A tabela 5.10 mostra que o número de soluções dominantes geradas por BTMO é muito maior, especialmente quando a razão n/m decresce, para n fixo.

Tabela 5.10. Número de soluções encontradas pelas metaheurísticas comparadas

Problema	$\lvert \boldsymbol{R} \rvert$	BTMO		BTH	
$n \times m$		NTS	NSD	NTS	NSD
20×5	506	484	420	456	280
20×10	906	876	746	784	343
20×20	1133	1097	893	1006	470
50×5	487	482	463	490	110
50×10	1297	1258	1191	1137	107
50×20	1815	1794	1674	1533	141
80×5	377	371	362	366	120
80×10	1130	1106	1032	1005	338
80×20	2468	2441	2252	2052	218
Total	10119	9909	9033	8829	2127

$\lvert \boldsymbol{R} \rvert$: número de soluções de referência.
NTS: número total de soluções obtidas por uma metaheurística..
NSD: número de soluções dominantes obtidas por uma metaheurística.

As médias de número de soluções dominantes obtidas pelas metaheurísticas, para cada cenário de datas de entrega, são mostradas na Figura 5.24. Observa-se que, a metaheurística BTMO obteve as maiores médias em todos os cenários.

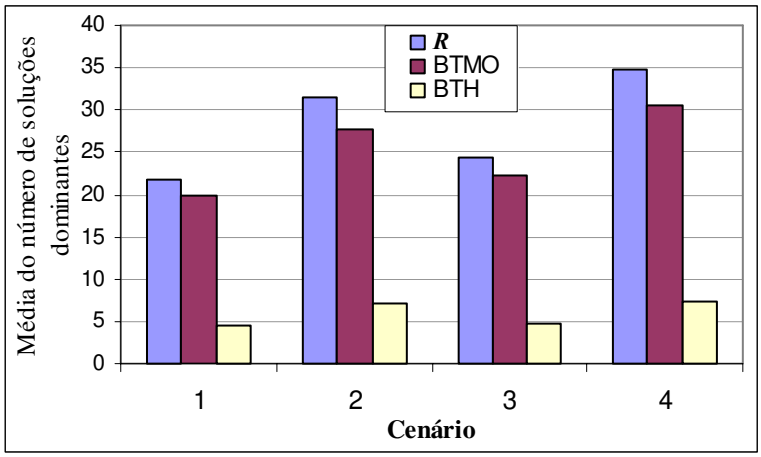


Figura 5.24. Média de número de soluções dominantes por cenário

Para cada tamanho de problema e cada cenário, na Figura 5.25 ilustra-se a média do número de soluções dominantes obtidas pela metaheurística BTMO. Observe que, para instancias com cenários 2 e 4 de datas de entrega sempre são obtidos maiores quantidades de soluções dominantes e o número de soluções dominantes cresce à medida que a razão *n*/*m* decresce.

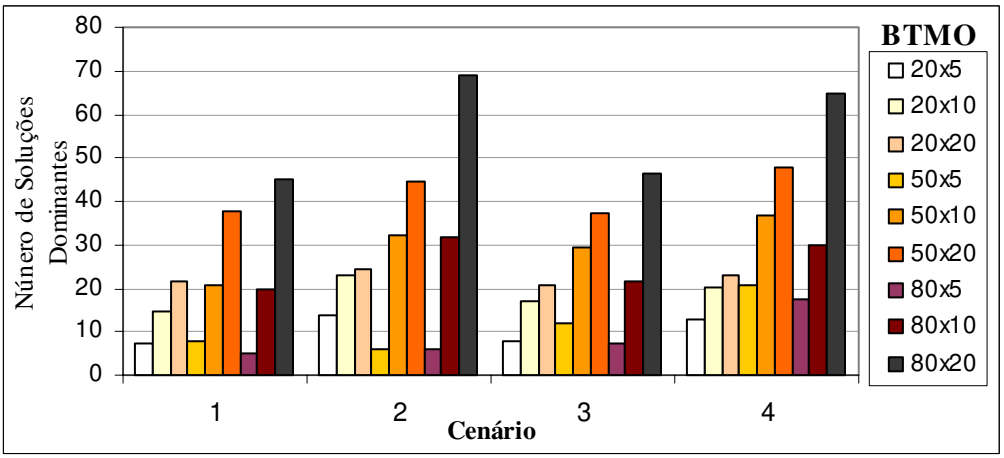


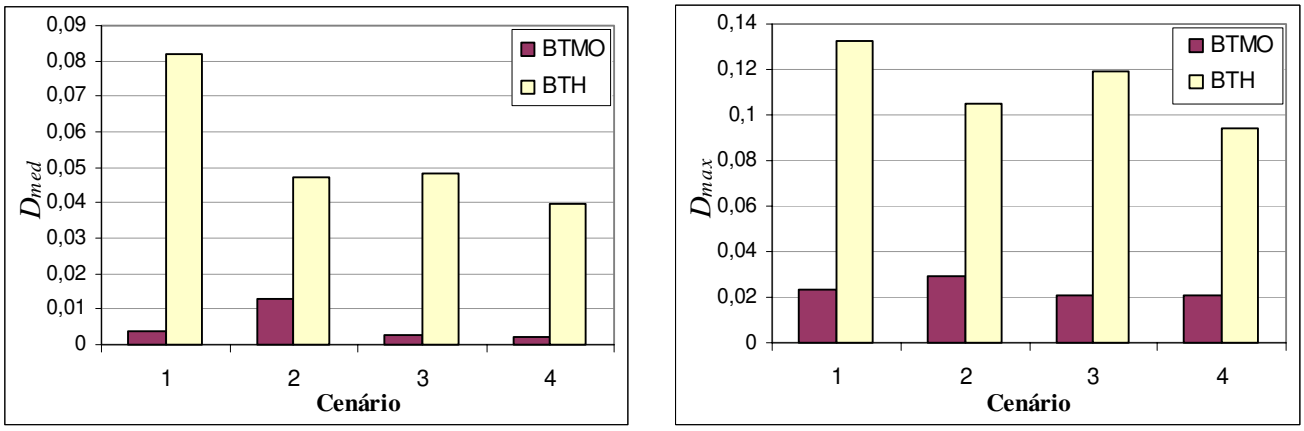
Figura 5.25. Média de número de soluções dominantes por tamanho de instância e por cenário correspondente à metaheurística BTMO.

Na Tabela 5. 11 são mostrados os desempenhos das metaheurísticas de acordo com a medida de distância e erro de utilidade percentual. Observe que, para todos os tamanhos de problemas a metaheurística BTMO possui melhor desempenho em relação às duas medidas.

Tabela 5.11. Desempenho das metaheurísticas

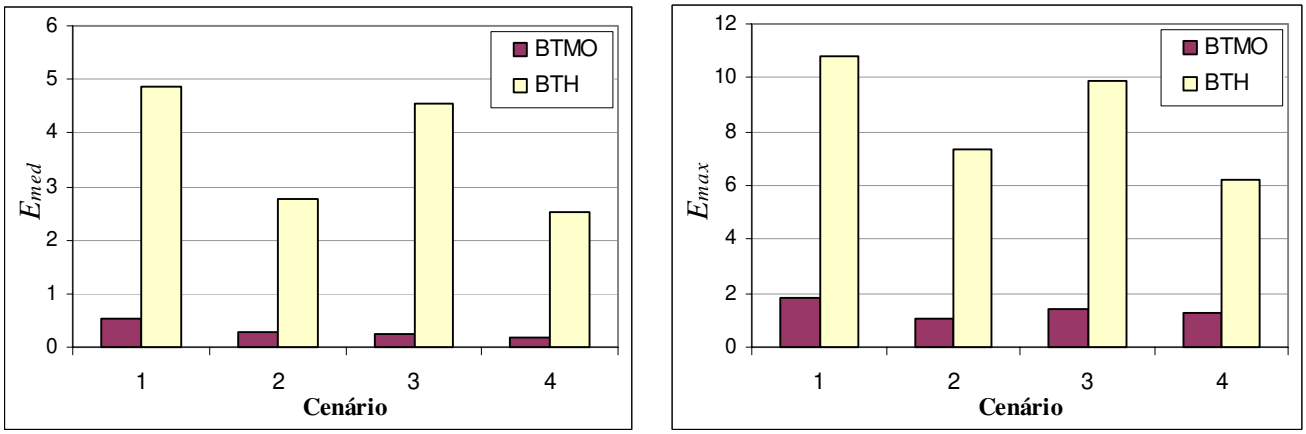
$n \times m$	Medida de distância				Erro de utilidade (%)			
	BTMO		BTH		BTMO		BTH	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
20×5	0,007	0,037	0,031	0,092	0,632	2,599	2,947	7,641
20×10	0,004	0,038	0,020	0,073	0,521	2,478	1,609	5,252
20×20	0,003	0,034	0,016	0,068	0,448	2,202	1,232	4,674
50×5	0,026	0,034	0,070	0,131	0,211	0,570	3,489	8,557
50×10	0,001	0,015	0,061	0,131	0,149	1,150	5,479	11,081
50×20	0,001	0,018	0,054	0,121	0,143	1,151	5,315	10,560
80×5	0,0002	0,002	0,070	0,129	0,007	0,084	2,956	7,350
80×10	0,002	0,017	0,060	0,140	0,326	1,508	5,265	11,917
80×20	0,001	0,013	0,049	0,127	0,116	0,677	4,786	9,987
Média	0,004	0,019	0,044	0,102	0,284	1,380	3,675	8,558

Para cada cenário, as Figuras 5.26 (a) e (b) mostram os valores médios (sobre 90 instâncias) das distâncias D_{med} e D_{max}, e as Figuras 5.27 (a) e (b) mostram os valores médios de E_{med} e E_{max}. A metaheurística BTMO apresenta menores valores da distância e erro de utilidade para todos os cenários.



(a) (b)

Figura 5.26. Distância média e máxima por cenário de data de entrega



(a) (b)

Figura 5.27. Erro de utilidade percentual médio e máximo por cenário de data de entrega.

Para este conjunto de instâncias, também são comparados os menores valores do makespan (C_{max}) e atraso máximo (T_{max}) obtidos pelas metaheurísticas. Nas 360 instâncias testadas, as metaheurísticas BTMO e BTH obtiveram o menor valor de C_{max} em 299 e 167 instâncias e o menor valor de T_{max} em 340 e 225 instâncias, respectivamente.

Na Tabela 5.12 estão as medidas dos tempos computacionais (em segundos) gastos pelas metaheurísticas BTMO e BTH.

Tabela 5.12. Média de tempos computacionais (em segundos)

n:	20			50			80		
m:	5	10	20	5	10	20	5	10	20
BTMO	4,46	7,716	12,46	73,42	135,42	252,82	338,07	617,25	1092,4
BTH	4,56	7,422	12,24	76,12	128,35	247,41	331,64	628,36	1074,3

5.4.2. Resultados Computacionais para $f = (C_{max}, T)$

Nesta seção apresentam-se os resultados computacionais obtidos na solução do problema de flowshop considerando o par de objetivos (C_{max}, T). Os desempenhos das metaheurísticas implementadas foram testados para os mesmos três conjuntos de instâncias usados na seção anterior.

5.4.2.1. Resultados Para o Conjunto 1 de Instâncias

Com o propósito de analisar o comportamento das metaheurísticas BTMO e BTH sobre instâncias deste conjunto, geram-se as soluções eficientes de cada instância através do método de enumeração completa.

Na Tabela 5.13 apresenta-se, para cada tamanho de problema, o número de soluções eficientes e o número de soluções obtidas pelas metaheurísticas BTMO e BTH associadas com 80 instâncias. De um total de 320 instâncias, a enumeração completa obteve 3442 soluções eficientes. As metaheurísticas BTMO e BTH encontraram 3380 (98,20%) e 3367 (97,82%) soluções eficientes, respectivamente. Além disso, as metaheurísticas encontraram o conjunto Pareto-ótimo para 277 e 273 instâncias, respectivamente. Observa-se que, os números de soluções eficientes obtidas pelas metaheurísticas são muito próximos, no entanto, a metaheurística BTMO aparenta ser ligeiramente superior.

Tabela 5.13. Número de soluções encontradas pela Enumeração Completa e pelas metaheurísticas comparadas

Problema	EC	BTMO		BTH	
$n \times m$	NSE	NTS	NSEM	NTS	NSEM
10×5	755	740	737	743	736
10×10	820	820	818	820	819
11×5	825	811	800	816	794
11×10	1042	1035	1025	1026	1018
Total	3442	3406	3380	3405	3367

NSE: número de soluções eficientes encontradas pelo método exato.
NTS: número total de soluções encontradas pela metaheurística.
NSEM: número de soluções eficientes encontradas pela metaheurística.

Na Figura 5.28 ilustra-se, para cada cenário de datas de entregas, a média de número de soluções eficientes obtidas pela enumeração completa (EC) e pelas duas metaheurísticas comparadas. Nota-se que, em todos os cenários a média de número de soluções eficientes obtida pelas duas metaheurísticas é bastante próxima à média do método exato. No entanto, comparando as metaheurísticas BTMO e BTH, a primeira apresenta médias ligeiramente maiores para os cenários 1, 2 e 4; e a segunda metaheurística apresenta média maior para o

cenário 3. Na minimização dos objetivos (C_{max}, T) também se pode notar que, instâncias com cenários 2 e 4 de datas de entrega possuem, em média, maior número de soluções eficientes.

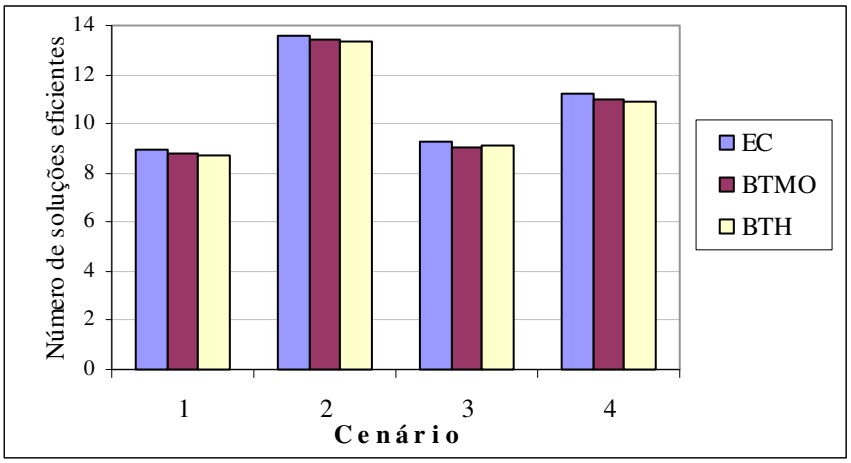


Figura 5.28. Média de número de soluções eficientes por cenário.

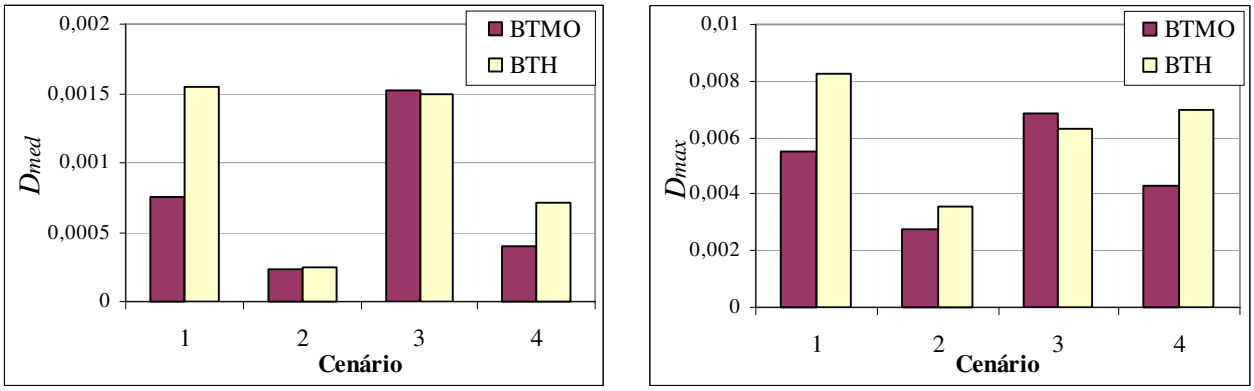
Os desempenhos das metaheurísticas BTMO e BTH em relação às medidas de distância e erro de utilidade, são mostrados na Tabela 5.14. Note que, a metaheurística BTMO em média apresenta melhor desempenho para todos os grupos de problemas, exceto para os problemas com 10 tarefas e 10 máquinas.

Tabela 5.14. Desempenho das metaheurísticas BTMO e MTH

$n \times m$	Medida de distância				Erro de utilidade (%)			
	BTMO		BTH		BTMO		BTH	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×5	0,0011	0,0073	0,0012	0,0073	0,044	0,466	0,055	0,474
10×10	0,0002	0,0016	0,00001	0,0003	0,007	0,053	0,000	0,000
11×5	0,0013	0,0070	0,0023	0,0111	0,064	0,504	0,164	0,606
11×10	0,0004	0,0047	0,0005	0,0062	0,023	0,3002	0,045	0,415
Média	0,0008	0,0052	0,0010	0,0062	0,035	0,331	0,066	0,374

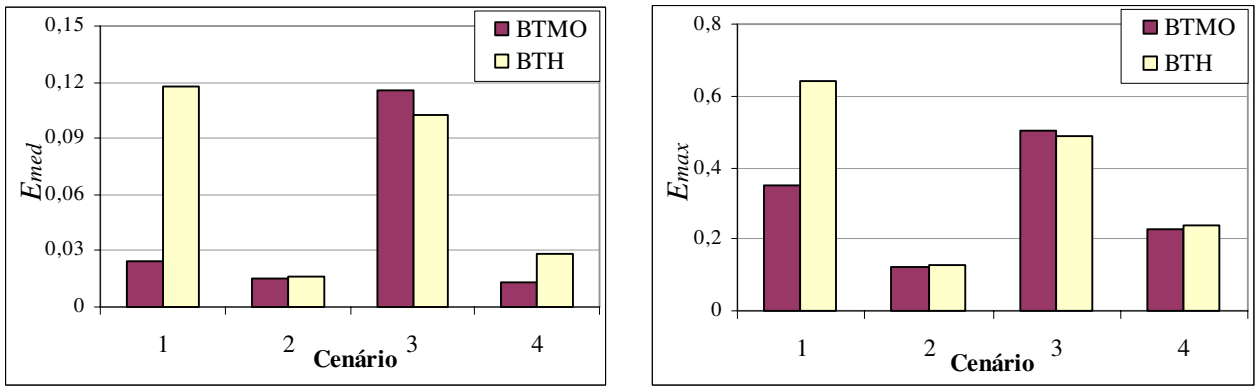
O desempenho das metaheurísticas também é analisado para cada cenário de datas de entrega. As Figuras 5.29 (a) e (b) ilustram os valores médios de D_{med} e D_{max}, enquanto as

Figuras 5.30 (a) e (b) ilustram os valores médios de E_{med} e E_{max}. Nota-se que, a metaheurística BTMO possui melhor desempenho para todos cenários, exceto para o cenário 3.



(a) (b)

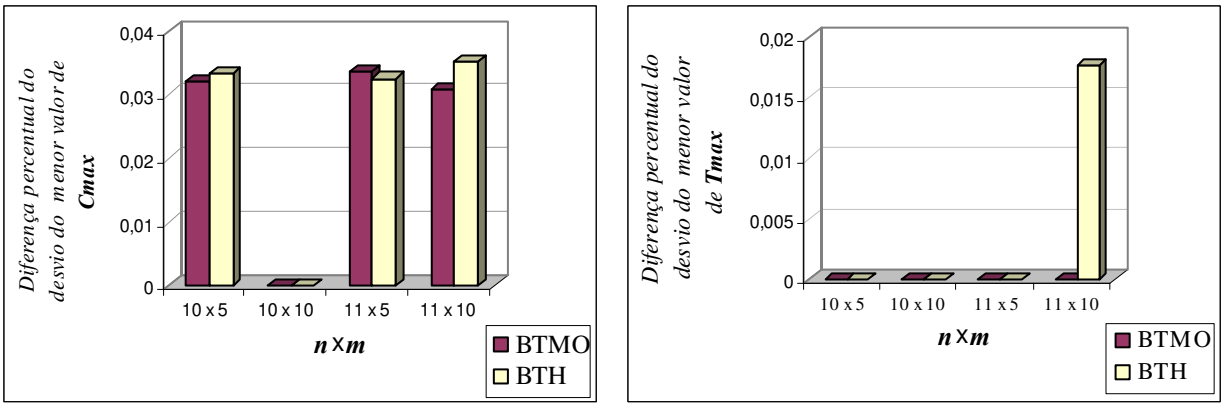
Figura 5.29. Distância média e máxima por cenário de data de entrega.



(a) (b)

Figura 5.30. Erro de utilidade médio e máximo por cenário de data de entrega.

Para cada instância, o menor valor de makespan (C_{max}) e soma dos atrasos (T) obtidos pelas metaheurísticas são comparados com os respectivos valores ótimos. Do total de 320 instâncias, as metaheurísticas BTMO e BTH obtiveram o valor ótimo de C_{max} em 302 e 304 instâncias, respectivamente. A metaheurística BTMO obteve o valor ótimo de T em todas as instâncias e a metaheurística BTH obteve o valor ótimo de T em 318 instâncias. Para cada tamanho de problema, as Figuras 5.31 (a) e (b) mostram a diferença percentual média do desvio dos menores valores de C_{max} e T, obtidas pelas metaheurísticas, em relação aos valores ótimos obtidos pela enumeração completa.



(a) (b)

Figura 5.31. Diferença percentual média do desvio dos menores valores de C_{max} e T_{max} obtidas pelas metaheurísticas em relação aos valores ótimos.

Na Tabela 5.15 são mostrados os tempos computacionais gastos pelas metaheurísticas BTMO e BTH.

Tabela 5.15. Média de tempos computacionais (em segundos)

	$n \times m$	10×5	10×10	11×5	11×10
BTMO		0,584	0,882	0,826	1,257
BTH		0,592	0,854	0,812	1,303

5.4.2.2. Resultados Para o Conjunto 2 de Instâncias

Para o par de objetivos (C_{max}, T), os resultados obtidos pelas metaheurísticas BTMO e BTH são comparados com os resultados ótimos gerados pelo algoritmo Branch and Bound (B&B) proposto por Liao et al. (1997). A Tabela 5.16 mostra, para cada tamanho de problema, o número de soluções obtidas pelo algoritmo B&B e pelas metaheurísticas. De um total de 120 instâncias testadas foram obtidas 474 soluções eficientes. As metaheurísticas BTMO e BTH encontraram 464 (97,89%) e 425 (89,66%) soluções eficientes e identificaram exatamente o conjunto Pareto-ótimo para 112 e 94 instâncias, respectivamente. Observe que, a metaheurística BTMO encontrou o conjunto Pareto-ótimo em todos os problemas com 10 tarefas.

Tabela 5.16. Número de soluções encontradas pela Enumeração Completa e pelas metaheurísticas comparadas

Problema	B&B	BTMO		BTH	
$n \times m$	NSE	NTS	NSEM	NTS	NSEM
10×2	158	158	158	156	155
15×2	172	171	167	169	157
20×2	144	144	139	145	113
Total	474	473	464	470	425

Os desempenhos das metaheurísticas de acordo com a medida de distância e erro de utilidade são mostrados nas Tabelas 5.8. A metaheurística BTMO apresenta melhor desempenho segundo as duas medidas.

Tabela 5.17. Desempenho das metaheurísticas

$n \times m$	Medida de distância				Erro de utilidade (%)			
	BTMO		BTH		BTMO		BTH	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
10×2	0	0	0,00011	0,0007	0	0	0,00013	0,0023
15×2	0,0007	0,0032	0,0018	0,0062	0,056	0,195	0,058	0,303
20×2	0,0081	0,0174	0,0326	0,054	0,697	1,672	2,166	4,522
Média	0,0029	0,0069	0,0115	0,0203	0,25100	0,62233	0,74138	1,60910

Para cada instancia deste conjunto, o menor valor do makespan (C_{max}) e soma dos atrasos (T) obtidos pelas metaheurísticas, são comparados com os respectivos valores ótimos. Do total de 120 instâncias testadas, as metaheurísticas BTMO e BTH obtiveram o valor ótimo de C_{max} em todas as instâncias e o valor ótimo de T em 117 e 114 instâncias, respectivamente.

Na Tabela 5.18 são mostrados os tempos computacionais gastos pelas metaheurísticas BTMO e BTH.

Tabela 5.18. Média de tempos computacionais (em segundos)

Problema $n \times m$	BTMO	BTH
10×2	0,374	0,355
15×2	1,021	0,943
20×2	2,356	2,224

5.4.2.3. Resultados Para o Conjunto 3 de Instâncias

As metaheurísticas BTMO e BTH foram aplicadas para resolver as instâncias deste conjunto, minimizando o par de objetivos (C_{max}, T). O conjunto dominante gerado por uma metaheurística é comparado com o respectivo conjunto de referência $\boldsymbol{R}$ formado pelas soluções dominantes dentre todas as soluções encontradas por ambas metaheurísticas.

Na Tabela 5.19, para cada tamanho de problema mostra-se o número de soluções de referência, e para cada metaheurística mostra-se o número total de soluções e o número de soluções dominantes obtidas. Para as 360 instâncias testadas, foram encontradas 14994 soluções de referência. As metaheurísticas BTMO e BTH obtiveram, respectivamente, 12864 (85,79%) e 4554 (30,37%) soluções dominantes. Note que, a metaheurística BTMO gerou a maior quantidade de soluções dominantes para todos os tamanhos de problemas e que novamente a maior quantidade se acentua com o decréscimo da razão n/m.

Tabela 5.19. Número de soluções encontradas pelas metaheurísticas comparadas

Problema $n \times m$	$\lvert\boldsymbol{R}\rvert$	BTMO		BTH	
		NTS	NSD	NTS	NSD
20×5	808	763	662	800	446
20×10	1147	1039	855	1083	614
20×20	1216	1161	906	1122	699
50×5	810	800	768	729	351
50×10	2135	1981	1853	1959	388
50×20	2577	2404	2061	2441	516
80×5	613	612	610	571	175
80×10	2165	2117	2070	1940	542
80×20	3523	3311	3079	3447	823
Total	14994	14188	12864	14092	4554

Na Figura 5.32 ilustram-se, para cada cenário de datas de entrega, as médias do número de soluções dominantes obtidas pelas metaheurísticas. Observa-se que, a metaheurística BTMO obteve as maiores médias em todos os cenários.

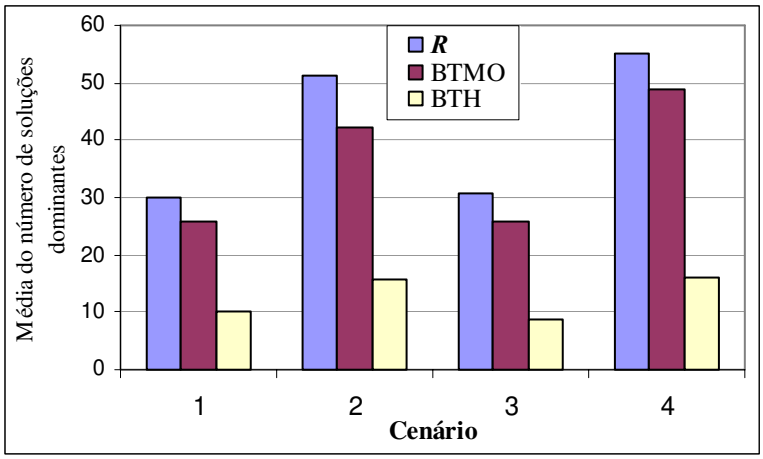


Figura 5.32. Média de número de soluções dominantes por cenário

Para cada tamanho de problema e cada cenário, na Figura 5.33 ilustra-se a média do número de soluções dominantes obtidas pela metaheurística BTMO. Geralmente, para instancias com cenários 2 e 4 de datas de entrega são obtidos maiores quantidades de soluções dominantes e o número de soluções dominantes cresce à medida a razão n/m decresce.

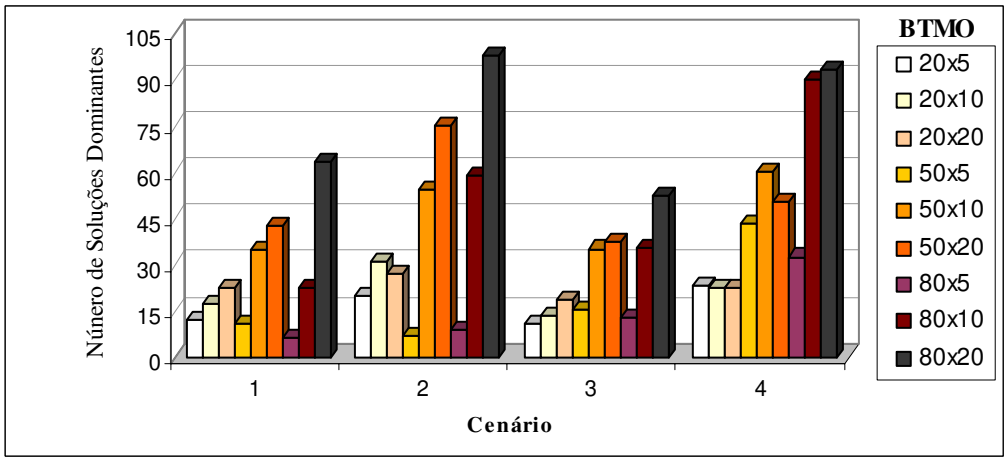


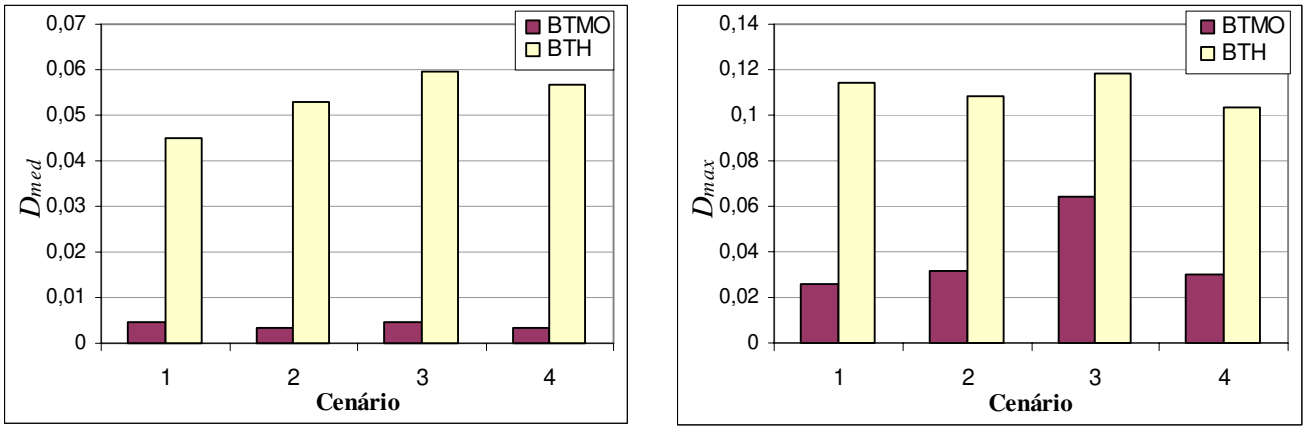
Figura 5.33. Média de número de soluções dominantes por tamanho de instância e por cenário correspondente à metaheurística BTMO.

Na Tabela 5. 20 são mostrados os desempenhos das metaheurísticas de acordo com a medida de distância e erro de utilidade percentual. Observe que, para todos os tamanhos de problemas a metaheurística BTMO possui um desempenho superior em relação às duas medidas.

Tabela 5.20. Desempenho das metaheurísticas

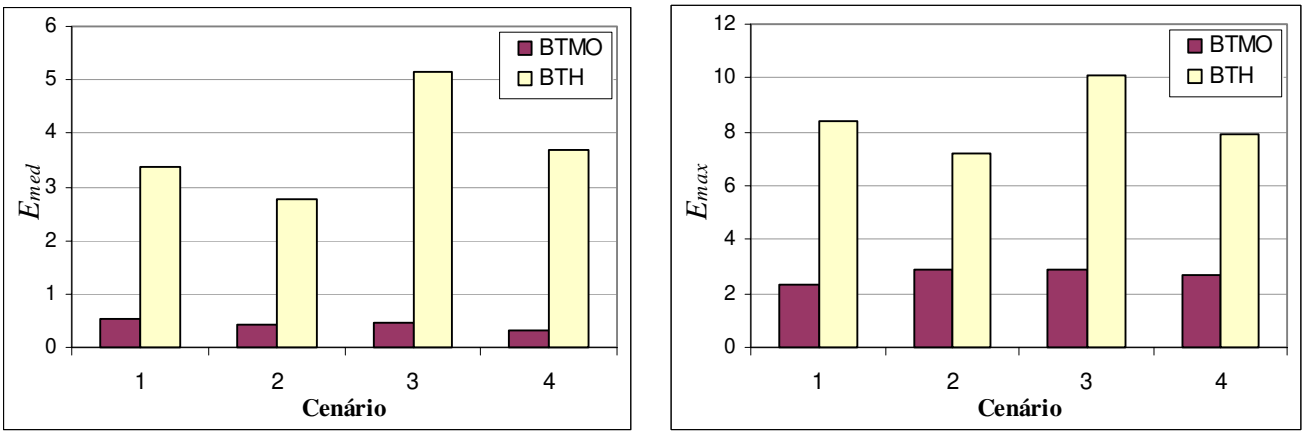
$n \times m$	Medida de distância				Erro de utilidade (%)			
	BTMO		BTH		BTMO		BTH	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
20×5	0,00824	0,0423	0,0184	0,0801	0,768	3,578	1,681	5,559
20×10	0,0073	0,046	0,015	0,066	0,676	3,654	1,503	5,599
20×20	0,0067	0,044	0,012	0,052	0,615	3,608	1,033	3,703
50×5	0,0008	0,007	0,062	0,120	0,019	0,294	4,394	10,129
50×10	0,0034	0,038	0,047	0,107	0,449	3,996	4,079	8,623
50×20	0,0052	0,046	0,041	0,093	0,837	5,079	3,879	8,391
80×5	0,00002	0,00015	0,176	0,270	0,000	0,000	8,302	16,690
80×10	0,0006	0,009	0,066	0,123	0,085	0,894	5,545	10,563
80×20	0,0031	0,030	0,043	0,088	0,491	3,301	4,313	8,167
Média	0,003	0,024	0,051	0,102	0,438	2,711	3,859	8,603

Para cada cenário, as Figuras 5.34 (a) e (b) mostram os valores médios (sobre 90 instâncias) das distâncias D_{med} e D_{max}, e as Figuras 5.35 (a) e (b) mostram os valores médios de E_{med} e E_{max}. A metaheurística BTMO sempre apresenta menores valores da distância e erro de utilidade para todos os cenários.



(a) (b)

Figura 5.34. Distância média e máxima por cenário de data de entrega



(a) (b)

Figura 5.35. Erro de utilidade percentual médio e máximo por cenário de data de entrega.

Para este conjunto de instâncias, também são comparados os menores valores do makespan (C_{max}) e atraso máximo (T_{max}) obtidos pelas metaheurísticas. Nas 360 instâncias testadas, as metaheurísticas BTMO e BTH obtiveram o menor valor de C_{max} em 299 e 167 instâncias e o menor valor de T_{max} em 340 e 225 instâncias, respectivamente.

Na Tabela 5.21 estão as medidas dos tempos computacionais (em segundos) gastos pelas metaheurísticas BTMO e BTH.

Tabela 5.21. Média de tempos computacionais (em segundos)

n	20			50			80		
m	5	10	20	5	10	20	5	10	20
BTMO	5,18	8,13	11,84	77,24	146,43	248,36	344,14	628,15	1112,3
BTH	4,82	8,23	12,16	75,37	137,55	252,72	339,42	620,46	1094,4

5.5. Aplicação da Metaheurística BTMO ao Problema da Mochila Multiobjetivo

Nesta seção apresenta-se a implementação da metaheurística BTMO para resolver o problema da mochila multiobjetivo 0-1 (PMM-1). Este problema foi descrito e formulado no Capítulo 4, seção 4.5. Apresenta-se a seguir a descrição dos componentes da metaheurística BTMO aplicado ao problema da mochila com dois objetivos.

5.5.1. Componentes da Busca Tabu BTMO

As combinações dos componentes que geraram os melhores resultados são apresentadas a seguir.

Soluções Iniciais

Para determinar as soluções de partida da busca tabu BTMO, gera-se um conjunto $\boldsymbol{S}$ de soluções dominantes através da heurística gulosa apresentado no Algoritmo 4.6 (veja seção 4.5.1). Note que, esta heurística é aplicada para resolver o problema ponderado (Pλ) descrito na seção 4.5.1, portanto é necessário definir um conjunto de N vetores pesos $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}^d, \boldsymbol{\lambda}^d = (\lambda_1^d,...,\lambda_r^d)$, $d = 0,..., N-1\}$ para determinar N direções diferentes de busca (veja seção 4.5.1). O valor do parâmetro N é fixado de acordo com o número de itens q do problema (veja Tabela 5.22).

Tabela 5.22. Número de direções consideradas para a heurística gulosa

q = 50, 100	q = 150, 200, 250	q = 300, 350, 400, 450, 500
N = 100	N = 150	N = 200

Como a metaheurística BTMO explora um número máximo N_{max} de soluções em paralelo, as soluções iniciais da busca são selecionadas a partir do conjunto $\boldsymbol{S}$ gerado pela

heurística gulosa. Se $|\boldsymbol{S}| > N_{max}$, então o conjunto $\boldsymbol{S}$ é reduzido a N_{max} soluções que serão os pontos de partida da busca. As soluções não consideradas são armazenadas na lista de candidatos $\boldsymbol{LC}$. Estas soluções podem ser aproveitadas na iteração seguinte caso elas dominem às novas soluções vizinhas geradas. Vale ressaltar que, a redução do conjunto $\boldsymbol{S}$ (escolha de soluções iniciais) é feita utilizando o método de redução aleatória (veja Algoritmo 2.4).

Se a heurística gulosa gera um conjunto $\boldsymbol{S}$ com $|\boldsymbol{S}| \leq N_{max}$, então se consideram todas as soluções em $\boldsymbol{S}$ como soluções iniciais da busca.

O número máximo de soluções a serem explorados pela busca tabu, N_{max}, foi fixado em 12, o que implica na geração de no máximo 12 trajetórias distintas de busca.

Estratégia de Geração de Vizinhança e Regra de Proibição

Para o problema da mochila bi-objetivo, foi considerado uma vizinhança baseada na remoção e inserção de itens (*drop-add*) (veja Algoritmo 4.8). A partir de uma solução $\boldsymbol{x} = (x_1,...,x_v,...,x_u,...,x_q)$, $x_v = 1$, $x_u = 0$, gera-se uma solução vizinha $\boldsymbol{y} = (y_1, ..., y_q)$ removendo um item v presente na mochila e inserindo um item u ausente na mochila: $y_i = x_i$, $i = 1,...,q$, $i \neq v$, $i \neq u$, $y_v = 0$, $y_u = 1$.

A regra de proibição é definida da seguinte maneira: Se o item v é removido da mochila e o item u é inserido, estes itens são adicionados na lista tabu, e portanto os itens v e u não podem ser escolhidos para inserção e remoção, respectivamente.

Duração Tabu

A duração tabu de um movimento é feita de forma dinâmica. A duração tabu dt é gerada aleatoriamente dentro de um intervalo $[dt_{min}, dt_{max}]$ com distribuição uniforme. Foram testados diferentes valores para dt_{min} e dt_{max}, e estes valores são definidos em termos do número de itens q. Os intervalos da duração tabu que geraram resultados razoáveis são mostrados na Tabela 5.23.

Tabela 5.23. Intervalos de geração da duração tabu

$q \leq 100$	$100 < q \leq 250$	$250 < q \leq 500$
$[q/6, q/3]$	$[q/8, q/4]$	$[n/10, q/6]$

Critério de Aspiração

Na implementação da metaheurística BTMO adaptada para o problema da mochila multiobjetivo, o critério de aspiração consiste na liberação da condição tabu de um movimento que gera uma solução dominante. Se um movimento considerado proibido gera uma nova solução dominante, ou seja, o conjunto de soluções dominantes encontradas até o momento pela busca é atualizado, então este movimento é liberado da condição tabu aceitando-se a solução gerada.

Estratégia de Diversificação

Na implementação da metaheurística BTMO para o problema da mochila, foi incorporada uma estratégia de diversificação baseada na freqüência de ocorrência dos itens dentro e fora da mochila.

A memória baseada em freqüência é representada através de dois vetores (V^0 e V^1) cada um de tamanho q. Se o item v é removido da mochila, então a posição v do vetor V^0 é incrementada em 1. Similarmente, a posição u do vetor V^1 é incrementada 1 toda vez que o item u é adicionado na mochila.

Se uma solução $\boldsymbol{x}$' é obtida a partir da solução $\boldsymbol{x}$ removendo o item v e adicionando o item u, então para a solução $\boldsymbol{x}$' calcula-se uma medida de penalidade baseada na freqüência de ocorrência dos itens v e u. Esta medida de penalização é calculada da seguinte maneira:

$$Pen = \frac{V^0(v)}{\max V^0} + \frac{V^1(u)}{\max V^1},$$

onde $\max V^0$ e $\max V^1$ são, respectivamente, as maiores freqüências dos vetores V^0 e V^1.

Os valores dos r objetivos da solução $\boldsymbol{x}$' são alterados para novos valores penalizados, de seguinte maneira:

$$f_k^* = f_k - d_k \times Pen, \ k = 1,...,r.$$

onde f_k é valor original do objetivo k e d_k é um parâmetro ajustável da diversificação associado com o objetivo k. Para os parâmetros d_k, foram testados vários valores que estão relacionados com o valor original do objetivo k. Foram obtidos bons resultados considerando $d_k = 0{,}05 \times f_k$, $k = 1,...,r$.

O processo de diversificação é acionado depois de 10 iterações sem atualização do conjunto de soluções dominantes, e este processo é interrompido após a execução de 2 iterações consecutivas.

Critério de Parada

O critério de parada escolhido para a metaheurística BTMO aplicado ao problema da mochila multiobjetivo é o mesmo utilizado para a metaheurística AGHM apresentada no Capítulo 4. A busca finaliza após de executar 150 iterações da busca em paralelo. Note que, a metaheurística BTMO gera no máximo $150 \times N_{max}$ vizinhanças (N_{max}: número máximo de soluções exploradas paralelamente).

5.5.2. Resultados Computacionais

A qualidade das soluções obtidas pela metaheurística BTMO aplicado ao problema da mochila bi-objetivo é avaliada através da comparação com as soluções Pareto-ótimas obtidas pelo algoritmo Branch-and-Bound (B&B) proposto por Ulungu e Teghem (1995).

Na Tabela 5.24, para cada tamanho de problema, apresenta-se o número de soluções eficientes obtidas pelo algoritmo B&B, o número total de soluções e o número de soluções eficientes obtidas pela busca tabu BTMO. Para as 10 instâncias testadas, o algoritmo B&B encontrou 7003 soluções eficientes sendo que a metaheurística BTMO identificou 3770 (53,83%) soluções eficientes.

Tabela 5.24. Número de soluções obtidas pelo algoritmo B&B e pela metaheurística

Número de itens q	B&B	BTMO		
	NSE	NTS	NSEM	%NSE
50	34	34	34	100
100	172	173	166	96,51
150	244	244	242	99,18
200	439	424	381	86,79
250	629	602	556	88,39
300	713	671	518	72,65
350	871	775	406	46,61
400	1000	875	482	48,20
450	1450	1106	559	38,55
500	1451	1172	426	29,36
Total	7003	6076	3770	53,83

NSE: número de soluções eficientes obtidas pelo algoritmo B&B.
NTS: número total de soluções obtidas pela metaheurística.
NSEM: número de soluções eficientes obtidas pela metaheurística.
%NSE: percentagem de soluções eficientes obtidas pela metaheurística.

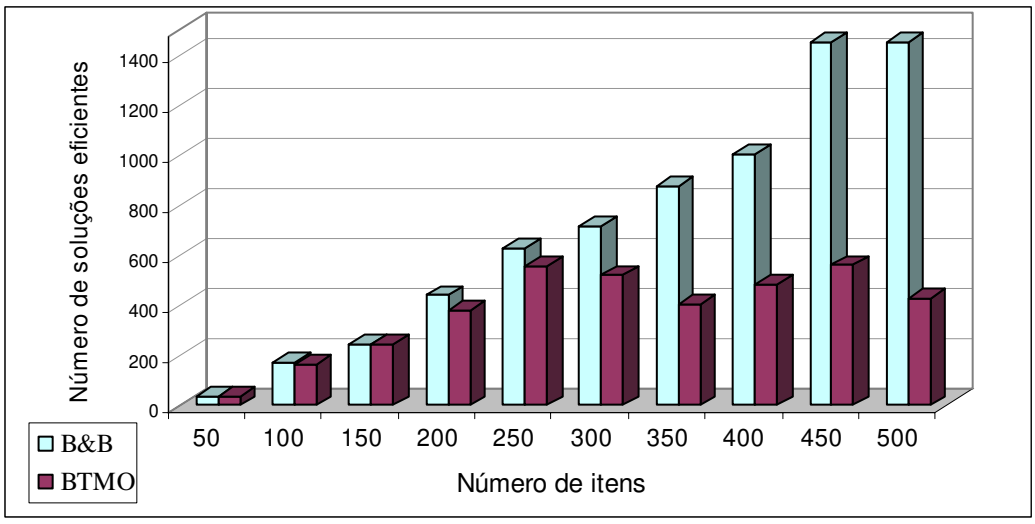


Figura 5.36. Número de soluções eficientes obtidas pelo algoritmo B&B e pela metaheurística BTMO.

Na Figura 5.36 ilustra-se, para cada instância, o número de soluções eficientes obtidos pelo algoritmo B&B e pela metaheurística BTMO. Observe que, o número de soluções eficientes cresce consideravelmente quando o número de itens q é incrementado.

O desempenho da metaheurística BTMO é avaliado através da medida de distância e erro de utilidade, ambos, adaptados para problemas de maximização. Na Tabela 5.25, para cada tamanho de instância, mostra-se o desempenho da metaheurística em relação às duas medidas. Observe que, os valores das medidas de distância e erro de utilidade são muitos pequenos, isto significa que, as soluções encontradas pela metaheurística estão bem próximas às soluções eficientes e bem distribuídas sobre a fronteira dominante.

Na Tabela 5.26 mostra-se, para cada tamanho de instância, o tempo (em segundos) gasto pela metaheurística BTMO.

Tabela 5.25. Desempenho da metaheurística BTMO

Número de itens	Medida de Distância		Erro de Utilidade (%)	
	D_{med}	D_{max}	E_{med}	E_{max}
50	0	0	0	0
100	0,000029	0,005	0	0
150	0,0000026	0,000637	0	0
200	0,000058	0,00291	0,00097	0,0556
250	0,0002253	0,0610	0,00104	0,07575
300	0,000048	0,00202	0,001806	0,0696
350	0,000115	0,0362	0,0128	0,086
400	0,0000075	0,00132	0,00536	0,05911
450	0,000051	0,00089	0,009114	0,05987
500	0,000101	0,0311	0,00761	0,06263
Média	0,0000637	0,014107	0,00387	0,04685

Tabela 5.26. Tempos computacionais (em segundos) da metaheurística BTMO

$q = 50$	$q = 100$	$q = 150$	$q = 200$	$q = 250$	$q = 300$	$q = 350$	$q = 400$	$q = 450$	$q = 500$
1,67	9,37	20,23	54,16	114,63	141,82	176,23	302,61	498,18	616,84

Capítulo 6

Comparação das Metaheurísticas AGHM e BTMO

Este capítulo apresenta a comparação dos resultados obtidos pelas metaheurísticas algoritmo genético multiobjetivo AGHM e busca tabu multiobjetivo BTMO apresentadas nos Capítulos 4 e 5, respectivamente. Avalia-se o desempenho e a eficácia de cada método proposto, em encontrar soluções dominantes para o problema de flowshop multiobjetivo e o problema da mochila multiobjetivo.

6.1. Comparação dos Resultados para o Problema de Flowshop

Nesta seção apresenta-se a comparação dos resultados obtidos pelas metaheurísticas AGHM e BTMO aplicadas ao problema de flowshop. Os desempenhos das metaheurísticas são avaliados para os seguintes pares de objetivos: i) (C_{max}, T_{max}) e ii) (C_{max}, T). Para realizar a avaliação dos resultados são consideradas as instâncias de problemas de grande porte cujas variações do número de tarefas e do número de máquinas são: n = 20, 50, 80 e m = 5, 10, 20. A forma de geração destas instâncias é descrita no Capítulo 2, seção 2.1.2.1.

6.1.1. Resultados Computacionais para $f = (C_{max}, T_{max})$

Nesta seção apresentam-se os resultados computacionais das metaheurísticas AGHM e BTMO obtidos para o problema de flowshop, minimizando o makespan (C_{max}) e o atraso máximo (T_{max}).

A partir dos conjuntos de soluções obtidas pelas metaheurísticas AGHM e BTMO, determina-se o conjunto de referência $\boldsymbol{R}$ que é formado pelas soluções dominantes dentre todas as soluções encontradas por ambas metaheurísticas. Compara-se então o conjunto gerado por uma metaheurística com o respectivo conjunto de referência. Na Tabela 6.1, para cada tamanho de problema mostra-se o número de soluções de referência, e para cada metaheurística mostra-se, o número total de soluções e o número de soluções dominantes obtidas. Para as 360 instâncias testadas, foram encontradas 10066 soluções de referência. As metaheurísticas AGHM e BTMO obtiveram, respectivamente, 6030 (59,9%) e 6011 (59,7%) soluções dominantes. Pela Tabela, pode-se notar que as metaheurísticas AGHM e BTMO possuem comportamentos bastante próximos. Observe que, para cada tamanho (grupo) de problema a diferença do número de soluções dominantes é pequeno. Dos 9 grupos de problemas, a metaheurística AGHM gerou um número maior de soluções dominantes em 5 grupos e a metaheurística BTMO em 4 grupos.

Tabela 6.1. Número de soluções encontradas pelas metaheurísticas comparadas

Problema	$\lvert\boldsymbol{R}\rvert$	AGHM		BTMO	
$n\times m$		NTS	NSD	NTS	NSD
20×5	516	491	450	484	376
20×10	940	867	576	876	674
20×20	1122	1017	742	1097	851
50×5	488	488	369	482	305
50×10	1276	1137	828	1258	850
50×20	1872	1626	939	1794	941
80×5	388	368	307	371	271
80×10	1051	922	567	1106	527
80×20	2413	2096	1252	2441	1216
Total	10066	9912	6030	9909	6011

$\lvert\boldsymbol{R}\rvert$: número de soluções de referência.

NTS: número total de soluções obtidas por uma metaheurística..

NSD: número de soluções dominantes obtidas por uma metaheurística.

Para cada cenário de datas de entrega, a Figura 6.1, ilustra as médias do número de soluções dominantes obtidas pelas metaheurísticas AGHM e BTMO. Observa-se que, para cada cenário, a diferença das médias do número de soluções dominantes é pequena. Para os

cenários 1 e 4, a metaheurística AGHM obteve maiores médias, e para os cenários 2 e 3 maiores medias foram obtidas pela metaheurística BTMO. É importante ressaltar que cada metaheurística obteve, em média, 60% de soluções dominantes que estão no conjunto de referência.

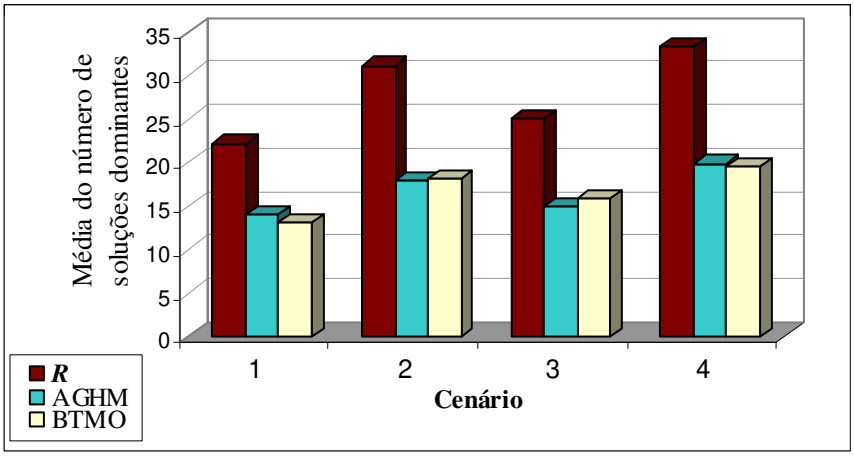


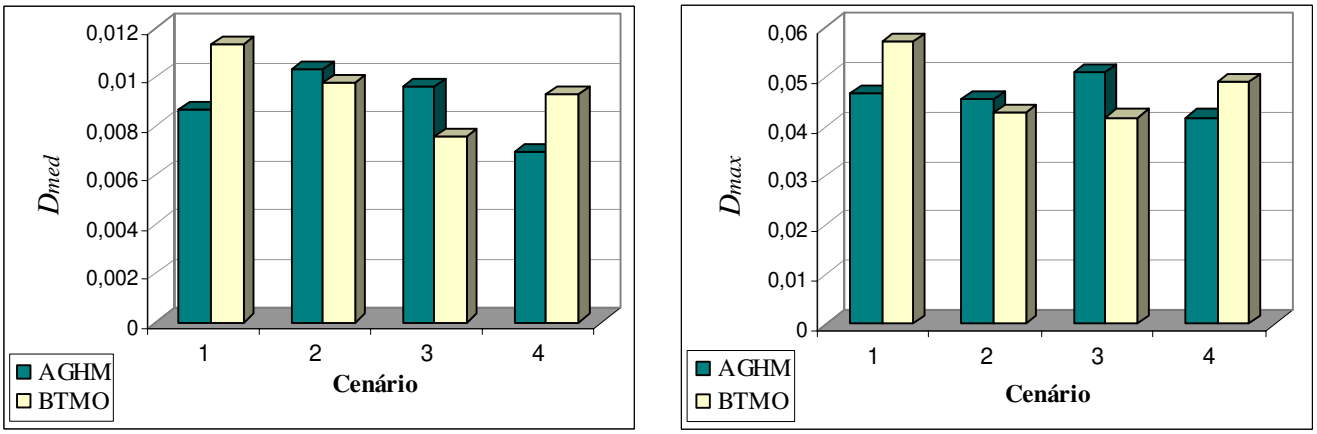
Figura 6.1. Média de número de soluções dominantes por cenário.

Na Tabela 6.2 são mostrados os desempenhos das metaheurísticas AGHM e BTMO em relação à medida de distância e o erro de utilidade percentual. Observe que, os desempenhos das metaheurísticas são bastante próximos. Em alguns grupos de problemas uma metaheurística possui melhor desempenho e em outros grupos o desempenho é um pouco pior.

Tabela 6.2. Desempenho das metaheurísticas

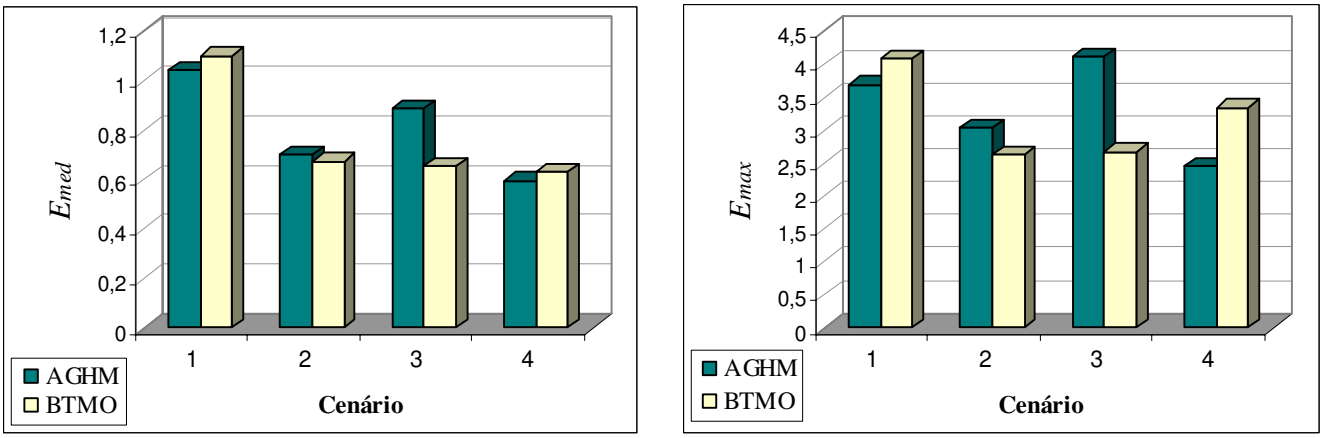
$n \times m$	Medida de distância				Erro de utilidade (%)			
	AGHM		BTMO		AGHM		BTMO	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
20×5	0,0047	0,027	0,011	0,047	0,497	2,097	0,651	2,462
20×10	0,011	0,055	0,006	0,042	1,016	4,037	0,615	2,743
20×20	0,009	0,059	0,004	0,027	1,080	3,897	0,255	1,411
50×5	0,007	0,022	0,008	0,356	0,634	1,901	0,661	2,129
50×10	0,0075	0,046	0,0071	0,041	0,819	3,423	0,612	2,659
50×20	0,017	0,086	0,011	0,056	2,028	6,796	1,255	4,511
80×5	0,002	0,012	0,0320	0,054	0,141	0,865	0,194	1,150
80×10	0,011	0,058	0,022	0,071	1,495	5,014	1,712	5,114
80×20	0,012	0,062	0,011	0,055	1,115	5,088	1,068	4,131
Média	0,008	0,044	0,011	0,078	0,980	3,680	0,780	2,923

Para cada cenário de datas de entrega, as Figuras 6.2 (a) e (b) mostram os valores médios das distâncias D_{med} e D_{max}, e as Figuras 6.3 (a) e (b) mostram os valores médios de E_{med} e E_{max}. Observe que, a metaheurística AGHM apresenta menores valores da distância e erro de utilidade para os cenários 1 e 4, e para a metaheurística BTMO estes valores são menores para os cenários 2 e 3.



(a) (b)

Figura 6.2. Distância média e máxima por cenário de data de entrega.



(a) (b)

Figura 6.3. Erro de utilidade percentual médio e máximo por cenário de data de entrega.

Para todas as instâncias testadas, comparam-se também os menores valores do makespan (C_{max}) e atraso máximo (T_{max}) obtidas por cada metaheurística. Nas 360 instâncias testadas, as metaheurísticas AGHM e BTMO obtiveram o menor valor de C_{max} em 230 e 296 instâncias e o menor valor de T_{max} em 289 e 304 instâncias, respectivamente. Note que, a

metaheurística BTMO apresenta bom comportamento na obtenção de soluções localizadas nos extremos da fronteira dominante (soluções com menores valores de C_{max} ou T_{max}). Na Figura 6.4 ilustram-se os conjuntos de soluções obtidas pelas metaheurísticas AGHM e BTMO para uma instância com n = 20 e m = 10. Observe que, a metaheurística BTMO obteve os menores valores de C_{max} e T_{max}.

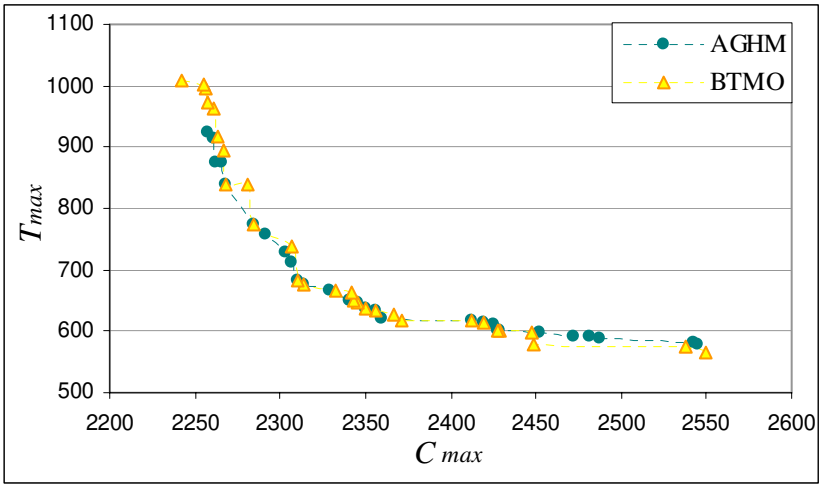


Figura 6.4. Conjunto de soluções obtidas pelas metaheurísticas.

6.1.2. Resultados Computacionais para $f = (C_{max}, T)$

Nesta seção apresentam-se os resultados computacionais das metaheurísticas AGHM e BTMO obtidos na solução do problema de flowshop, considerando o par de objetivos (C_{max}, T).

O conjunto dominante gerado por cada metaheurística é comparado com o respectivo conjunto de referência $\boldsymbol{R}$ formado pelas soluções dominantes dentre todas as soluções encontradas por ambas metaheurísticas. Na Tabela 6.3, para cada tamanho de problema, mostra-se o número de soluções de referência, o número total de soluções obtidas por uma metaheurística e o número de soluções dominantes obtidas. Para as 360 instâncias testadas, foram encontradas 14658 soluções de referência. As metaheurísticas AGHM e BTMO obtiveram, respectivamente, 9965 (67,98%) e 9800 (66,85%) soluções dominantes. Note que,

a metaheurística AGHM gerou uma quantidade maior de soluções dominantes. Dos 9 grupos de instâncias testadas, a metaheurística AGHM gerou um número maior de soluções dominantes em 6 grupos e a metaheurística BTMO em 3 grupos.

Tabela 6.3. Número de soluções encontradas pelas metaheurísticas comparadas

Problema $n \times m$	$\|R\|$	AGHM		BTMO	
		NTS	NSD	NTS	NSD
20×5	791	782	604	763	596
20×10	1075	1047	871	1039	909
20×20	1166	1088	956	1161	1063
50×5	817	771	597	800	586
50×10	2107	1966	1275	1981	1157
50×20	2524	2260	1476	2404	1315
80×5	641	633	501	612	451
80×10	2151	1971	1612	2117	1568
80×20	3386	3250	2073	3311	2155
Total	14658	13768	9965	14188	9800

Na Figura 6.5 ilustram-se, para cada cenário de datas de entrega, as médias do número de soluções dominantes obtidas pelas metaheurísticas. Observa-se que, a metaheurística AGHM apresenta médias ligeiramente maiores para os cenários 1 e 2, e a metaheurística BTMO apresenta maiores médias para os cenários 3 e 4.

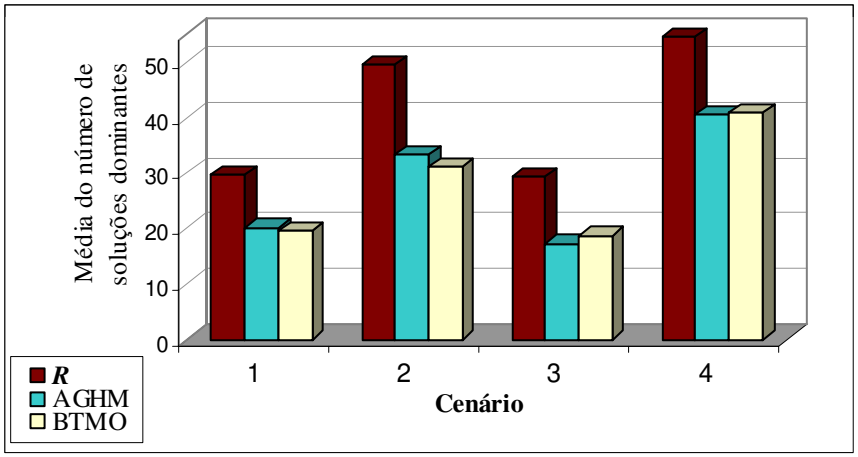


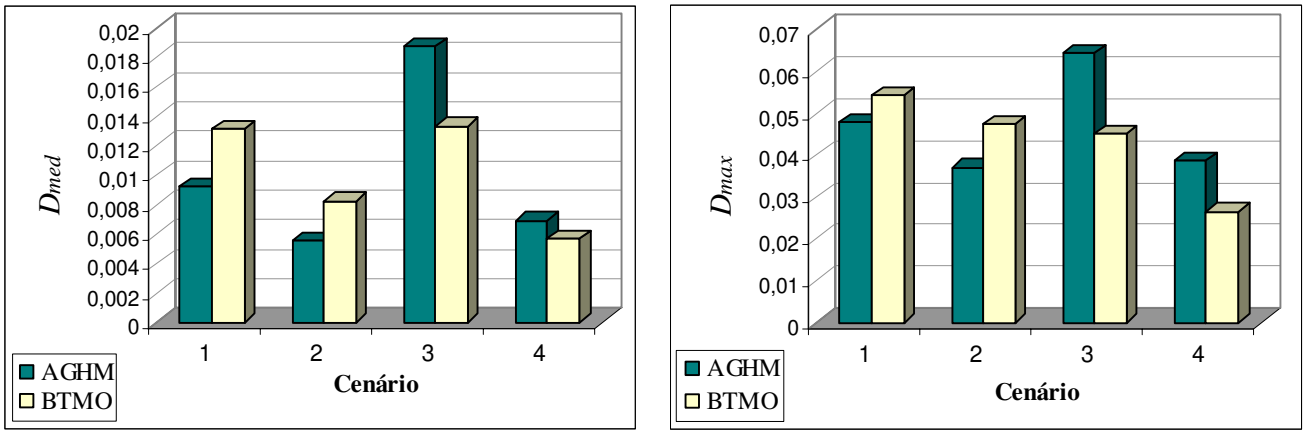
Figura 6.5. Média do número de soluções dominantes por cenário.

Na Tabela 6.4 são mostrados os desempenhos das metaheurísticas de acordo com a medida de distância e erro de utilidade percentual. Observe que, os desempenhos das metaheurísticas são bastante próximos.

Tabela 6.4. Desempenho das metaheurísticas

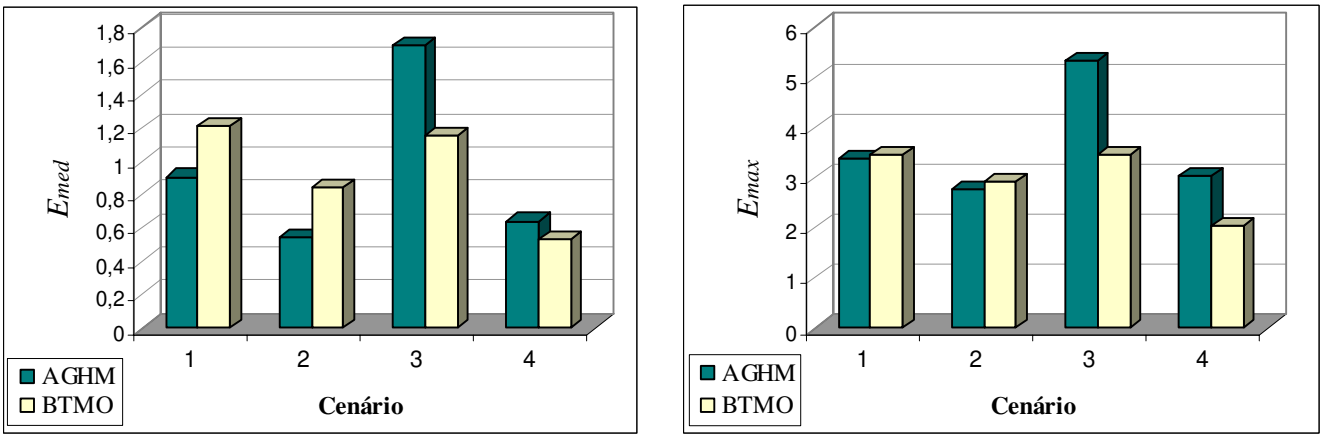
$n \times m$	Medida de distância				Erro de utilidade (%)			
	AGHM		BTMO		AGHM		BTMO	
	D_{med}	D_{max}	D_{med}	D_{max}	E_{med}	E_{max}	E_{med}	E_{max}
20×5	0,00871	0,0467	0,00621	0,0686	0,775	3,374	0,566	2,395
20×10	0,0056	0,036	0,0047	0,027	0,643	2,831	0,420	1,786
20×20	0,0062	0,039	0,0019	0,017	0,703	1,268	0,129	0,926
50×5	0,0090	0,037	0,035	0,065	0,702	2,274	0,820	2,657
50×10	0,0161	0,065	0,0140	0,056	1,682	5,717	1,442	4,307
50×20	0,0171	0,076	0,016	0,056	1,831	6,185	1,615	4,902
80×5	0,0056	0,0221	0,008	0,025	0,341	1,474	0,532	1,671
80×10	0,0075	0,040	0,016	0,053	0,642	3,808	1,051	4,018
80×20	0,0151	0,060	0,011	0,045	1,838	5,638	1,408	4,108
Média	0,010	0,047	0,013	0,046	1,018	3,619	0,887	2,974

Para cada cenário, as Figuras 6.6 (a) e (b) mostram os valores médios das distâncias D_{med} e D_{max}, e as Figuras 6.7 (a) e (b) mostram os valores médios de E_{med} e E_{max}. Observe que, a metaheurística AGHM apresenta menores valores da distância e erro de utilidade para os cenários 1 e 2, e a metaheurística BTMO para os cenários 3 e 4..



(a) (b)

Figura 6.6. Distância média e máxima por cenário de data de entrega



(a) (b)

Figura 6.7. Erro de utilidade percentual médio e máximo por cenário de data de entrega.

Para este conjunto de instâncias, também são comparados os menores valores do makespan (C_{max}) e do atraso total (T), obtidos pelas metaheurísticas. Nas 360 instâncias testadas, as metaheurísticas AGHM e BTMO obtiveram o menor valor de C_{max} em 245 e 319 instâncias e o menor valor de T em 272 e 303 instâncias, respectivamente. Observa-se que, para o par de objetivos (C_{max},T), a metaheurística BTMO também possui bom comportamento em obter os menores valores de C_{max} e T.

Na Tabela 6.5, para cada tamanho de problema, são mostrados os tempos computacionais (em segundos) gastos pelas metaheurísticas AGHM e BTMO considerando os dois pares de objetivos: i) (C_{max}, T_{max}) e ii) (C_{max}, T). Note que, os tempos das metaheurísticas são bastante próximos, sendo que os tempos do algoritmo genético são ligeiramente mais altos.

Tabela 6.5. Média de tempos computacionais (em segundos)

$n \rightarrow$		20			50			80		
$m \rightarrow$		5	10	20	5	10	20	5	10	20
AGHM	i)	5,75	8,16	12,25	80,19	152,42	247,26	351,03	625,82	1106.2
	ii)	6,42	8,82	12,78	83,92	164,35	262,24	353,24	642,81	1152,6
BTMO	i)	4,46	7,716	12,46	73,42	135,42	252,82	338,07	617,25	1092,4
	ii)	5,18	8,13	11,84	77,24	146,43	248,36	344,14	628,15	1112,3

6.2. Comparação dos Resultados para o Problema da Mochila Multiobjetivo

Nesta seção apresenta-se a comparação dos resultados obtidos pelas metaheurísticas AGHM e BTMO aplicadas ao problema da mochila multiobjetivo.

O desempenho das metaheurísticas é avaliado sobre um conjunto de 10 instâncias com dois objetivos na qual o número de itens q varia de 50 a 500. Para cada instância, o conjunto de soluções gerado pelas metaheurísticas é comparado com o respectivo conjunto Pareto-ótimo obtido pelo algoritmo Branch-and-Bound proposto por Ulungu e Teghem (1995). Para as 10 instâncias testadas, o algoritmo B&B encontrou 7003 soluções eficientes. As metaheurísticas AGHM e BTMO identificaram respectivamente 3590 (51,26%) e 3770 (53,83%) soluções eficientes. Na Figura 6.8 ilustra-se, para cada tamanho de instância, o número de soluções eficientes obtidos pelas metaheurísticas. Observe que, o número de soluções geradas pela metaheurística AGHM é um pouco maior para as instâncias 100, 200, 300, 350, 500 itens, e metaheurística BTMO gera maior quantidade de soluções para as instâncias com 150, 250, 400, 450 itens.

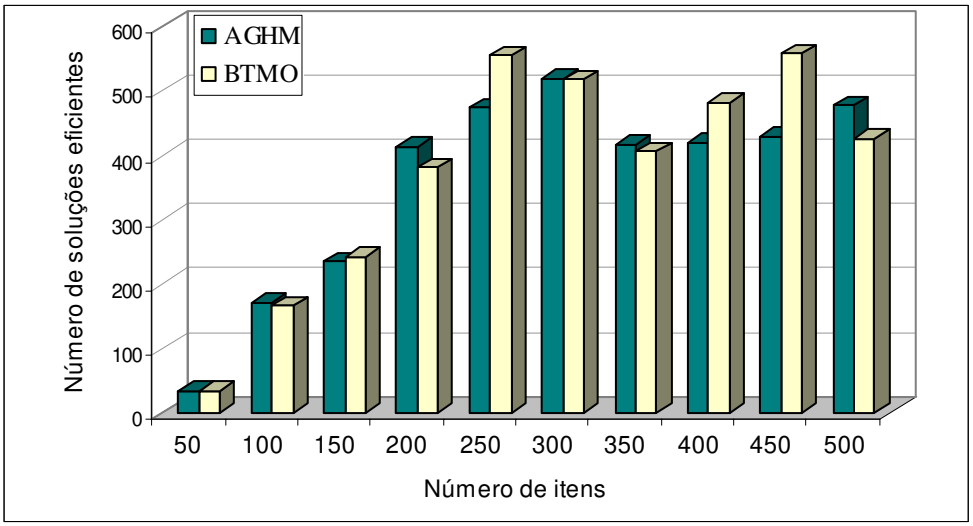


Figura 6.8. Número de soluções eficientes obtidas pelas metaheurísticas para cada tamanho de instância.

Para cada tamanho de instância, as Figuras 6.9 e 6.10 mostram os desempenhos das metaheurísticas de acordo com as medidas de distância D_{med} e D_{max}, respectivamente. Nas Figuras 6.11 e 6.12 ilustram-se os desempenhos das metaheurísticas de acordo com os erros de utilidade E_{med} e E_{max}, respectivamente.

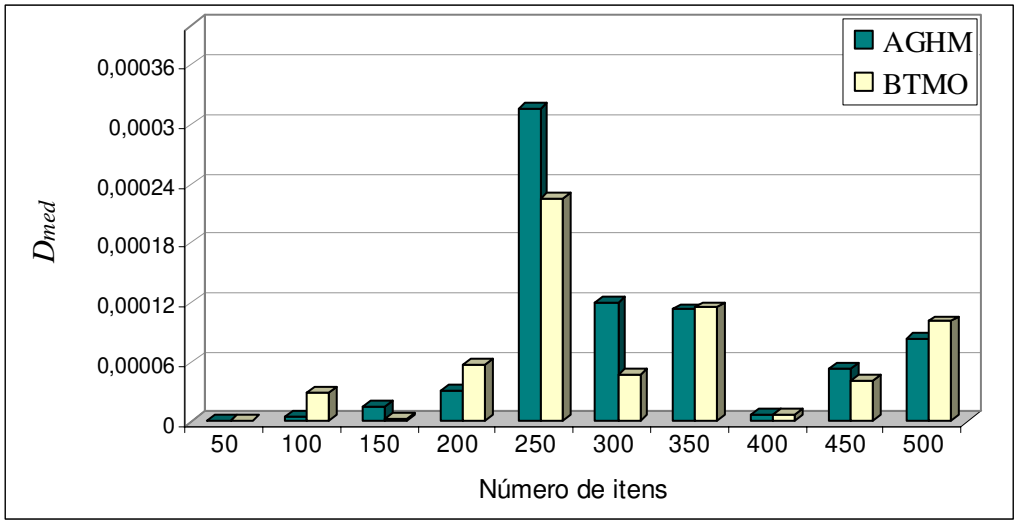


Figura 6.9. Distância média por tamanho de instância.

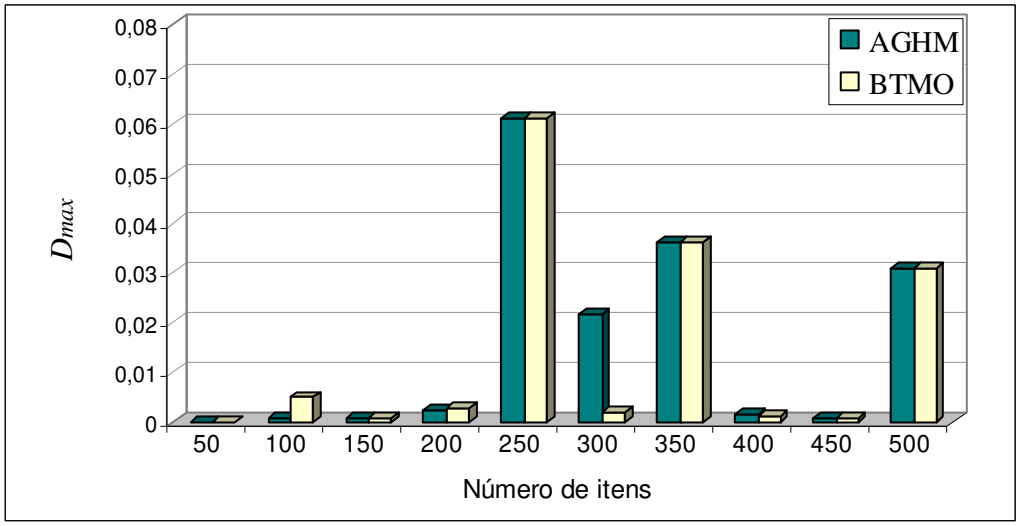


Figura 6.10. Distância máxima por tamanho de instância.

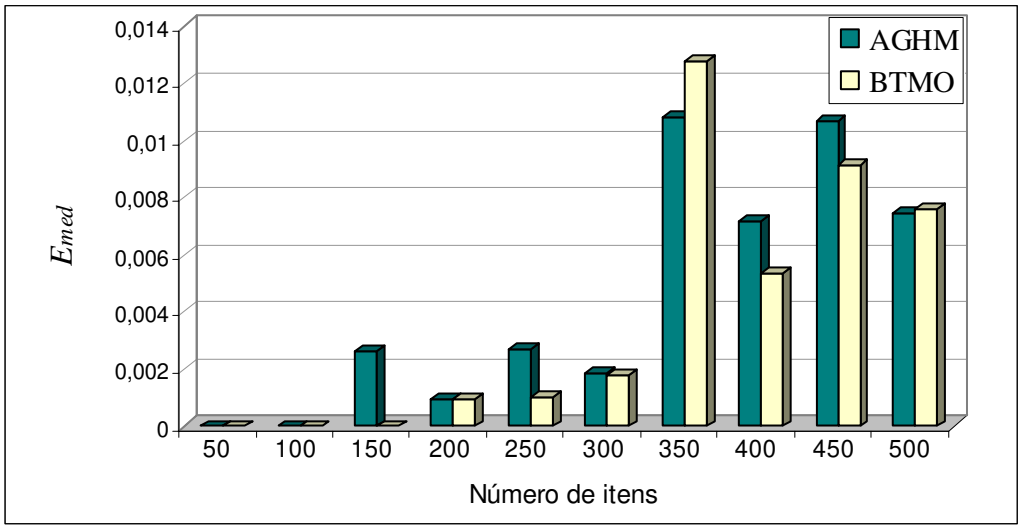


Figura 6.11. Erro de utilidade médio por tamanho de instância.

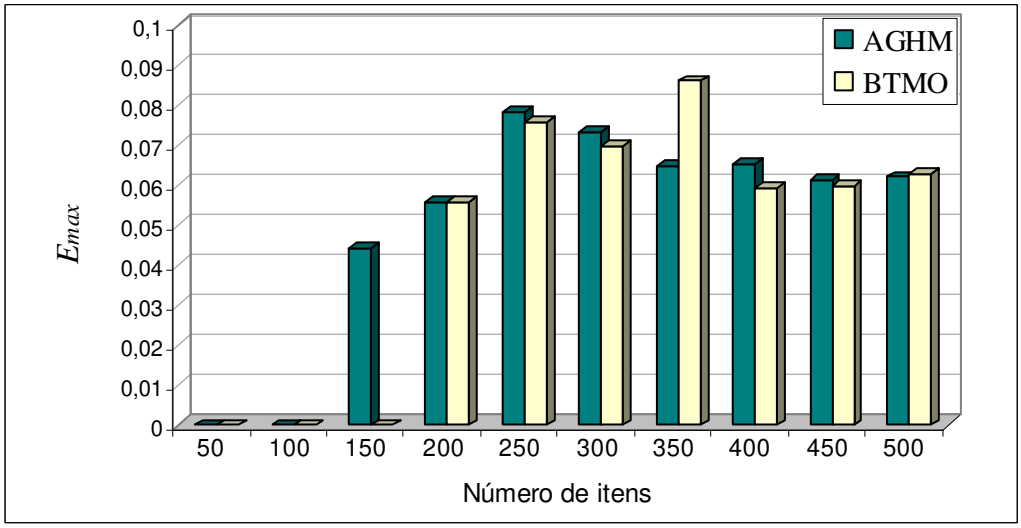


Figura 6.16. Erro de utilidade máximo por tamanho de instância.

Na Tabela 6.6, para cada tamanho de problema, são mostrados os tempos computacionais (em segundos) gastos pelas metaheurísticas AGHM e BTMO.

Tabela 6.6. Tempos computacionais (em segundos) das metaheurísticas

	$q = 50$	$q = 100$	$q = 150$	$q = 200$	$q = 250$	$q = 300$	$q = 350$	$q = 400$	$q = 450$	$q = 500$
BTMO	1,67	9,37	20,23	54,16	114,63	141,82	176,23	302,61	498,18	616,84
AGHM	1,15	7,34	17,02	44,25	104,23	120,45	168,43	283,14	466,74	553,23

Capítulo 7

Conclusões

O objetivo deste trabalho foi desenvolver métodos heurísticos para a resolução de problemas de otimização combinatória multiobjetivo.

Primeiramente, foi desenvolvida uma heurística construtiva de enumeração parcial para obter uma aproximação do conjunto de soluções Pareto-ótimas do problema de flowshop multiobjetivo. Foi considerada a minimização de dois pares de objetivos, (C_{max}, T_{max}) e (C_{max}, T). A heurística construtiva é inspirada na heurística de NEH proposta por Nawaz et al. (1983) e é baseada na inserção de tarefas e no conceito de dominância de Pareto quando comparam-se seqüências parciais e completas. Os resultados computacionais mostram que, a heurística é rápida e gera uma aproximação razoável do conjunto Pareto-ótimo. Portanto, as soluções geradas pela heurística proposta podem ser usadas como pontos de partidas de alguma metaheurística.

Além disso, foi proposto um método heurístico de busca local para resolver problemas de otimização combinatória multiobjetivo. Este método foi testado através de sua aplicação ao problema de flowshop multiobjetivo, apresentando bom desempenho quando comparado com o método enumeração completa e Branch-and-Bound.

Com o objetivo de gerar soluções dominantes próximas das soluções eficientes de um problema de otimização combinatória multiobjetivo, foram desenvolvidas dois métodos metaheurísticos. Inicialmente, foi desenvolvido um algoritmo genético híbrido multiobjetivo AGHM baseado no conceito de dominância de Pareto, elitismo, preservação de diversidade na população e estratégias de busca local. Em seguida, foi proposto um algoritmo de busca tabu

multiobjetivo BTMO que realiza uma busca eficaz através da exploração de várias soluções em paralelo.

As metaheurísticas propostas AGHM e BTMO baseiam-se fortemente no conceito de dominância de Pareto. A eficácia das metaheurísticas propostas foi testada na resolução do problema de flowshop e do problema da mochila, ambos com dois objetivos.

Para o problema de flowshop, a qualidade das soluções aproximadas foi avaliada pela comparação com as soluções eficientes obtidas pelo método de enumeração completa e por dois algoritmos Branch-and-Bound. Soluções de problemas de grande porte geradas pelas metaheurísticas AGHM e BTMO foram comparadas com as soluções geradas por metaheurísticas multiobjetivos da literatura. Ambas metaheurísticas propostas apresentaram um bom comportamento na obtenção de soluções dominantes. Na minimização de C_{max} e T_{max} (em 360 problemas de grande porte) o algoritmo genético híbrido AGHM obteve 4945 soluções dominantes a mais que o algoritmo genético híbrido AGIM proposto por Ishibuchi e Murata (1998). Também, a busca tabu BTMO gerou 6906 soluções dominantes a mais que a busca tabu BTH proposta por Hansen (1997). Na minimização de C_{max} e T (em 360 problemas de grande porte) a metaheurística AGHM gerou 7882 soluções dominantes a mais que a metaheurística AGIM, e a busca tabu BTMO encontrou 8310 soluções dominantes a mais que a busca tabu BTH.

O desempenho dos métodos heurísticos e metaheurísticos foi avaliado através de duas medidas: medida de distância e erro de utilidade. Segundo estas medidas, as metaheurísticas AGHMO e BTMO apresentaram um desempenho superior que as metaheurísticas AGIM e BTH, respectivamente.

Também foi comparada a qualidade das soluções obtidas pelas metaheurísticas AGHMO e BTMO. Nas avaliações realizadas, sobre o problema de flowshop e o problema da mochila, observaram-se que ambas metaheurísticas apresentam comportamentos e desempenhos bastante similares. Isto se deve ao fato de que em ambas heurísticas aplica-se uma mesma estratégia de busca local: exploração de um conjunto de soluções em paralelo. Cabe destacar que, o desempenho da metaheurística AGHMO é bastante superior que o desempenho da versão não híbrida (sem busca local).

De modo geral, este trabalho contribuiu com algumas idéias e estratégias a serem usadas em aplicações de heurísticas e metaheurísticas para resolver problemas de otimização multiobjetivo. Dentre as estratégias propostas, podemos destacar a busca local multiobjetivo. Esta estratégia explora varias soluções em paralelo gerando, rapidamente, várias soluções dominantes distribuídas e bem próximas às soluções Pareto-ótimas.

Como continuação e extensões deste trabalho, podem-se citar:

- Otimização de três ou de mais objetivos no problema de flowshop e no problema da mochila;
- Aplicação das metaheurísticas propostas para resolver outros problemas combinatórios multiobjetivos;
- Comparação das metaheurísticas propostas com outros métodos tais como *simulated annealing* e *scatter search*.

Referências Bibliográficas

Armentano V.A. & Ronconi D.P. (1999) "Tabu search for total tardiness minimization in flowshop scheduling problems", *Computers and Operations Research*, vol. 26, pp.219-235.

Arroyo J.E.C. & Armentano, V.A. (2000) "Uma metaheurística de Aproximação do Conjunto ótimo de Pareto em um Problema de Flowshop com dois objetivos", *Anais do XXXII Simpósio Brasileiro de Pesquisa Operacional*, pp. 789-802.

Baker K.R. & Kanet J.J. (1983) "Job shop scheduling with modified due dates", *Journal of Operations Management*, vol. 4, pp. 11-22.

Baykasoglu A., Owen S. & Gindy N. (1999) "A taboo search based approach to find the Pareto optimal set in multiple objective optimization", *Engineering Optimization*, vol. 31, pp. 731-748.

Ben A.F., Chaouachi J. & Krichen S. (1999) "A hybrid heuristic for multiobjective knapsack problems", In:Voss S,Martello S,Osman I,Roucairol C(eds) Meta-heuristics. Advances and trends in local search paradigms for optimization, pp. 205-212. Kluwer, Dordrecht.

Chankong V. & Haimes Y. Y. (1983) "Multiobjective Decision Making -Theory and Methodology", *New York: North-Holland*.

Coello C.A.C. (2000) "An Updated Survey of GA-Based Multiobjective Optimization Techniques", *ACM Computing Surveys*, vol. 32, nº 2, pp. 109-143.

Cohon J.L. (1978) Multiobjective Programing and Plannig. New York: Academic Press.

Czyzak P. & Jaszkiewicz A. (1998) "Pareto Simulated Annealing – a metaheuristic technique for multiple objective combinatorial optimization", *Journal of Multi-Criteria Decision Analysis*, vol. 7, pp. 34-47.

Daniels R.L. (1992) "Analytical evaluation of multi-criteria heuristics", *Management Science*, vol. 38, pp. 501-513.

Daniels R.L. & Chambers R.J. (1990) "Multiobjective flow-shop scheduling", *Naval Research Logistics*, vol. 37, pp. 981-995.

Deb K. Agrawal S. Pratab A. & Meyarivan T. (2000) "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II". KanGAL report.

Deb K. & Goldberg (1991) "An Investigation of niche and species formation in genetic function optimization", In J.D. Shaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, pp. 42-50. Morgan Kaufmann.

Dileepan P. & Sen T. (1988) "Bicriterion static scheduling research for a single machine", *OMEGA*, vol. 16, pp. 53-59.

Ehrgott M. (2000) "Approximation algorithms for combinatorial multicriteria optimization problems", *International Transactions in Operational Research*, vol. 7, pp. 5-31.

Ehrgott M. & Gandibleux X. (2000) "A survey and annotated bibliography of multicriteria combinatorial optimization", *OR Spektrum*, forthcoming.

Fonseca C.M & Fleming P.J. (1995) "Multiobjective Genetic Algorithms made easy: Selection, Sharing and Matinig restrictions" In *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA 95)*, London, UK, pp. 45-52. The Institution of Electrical Engineers.

Fonseca C.M. & Fleming P.J. (1993) "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization", In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, California, pp. 416-423.

Fonseca C.M. & Fleming P.J. (1998) "Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms-part1: A unified formulation", *IEEE Transactionos on Systems, Man and Cybernetics* vol. 26(1), pp. 26-37.

Fry T.D., Armstrong R.D. & Lewis H. (1989) "A framework for single machine multiple objective sequencing research", *OMEGA*, vol.17, pp. 595-607.

Gandibleux X. & Fréville A. (2000) "Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: The two objectives case", *Journal of Heuristics*, 6, pp. 361-383.

Gandibleux, X., Mezdaoui, N. & Fréville A. (1997) "A tabu search procedure to solve multiobjective combinatorial problems", in: *Advances in Multiple Objective and Goal Programming*, R. Caballero, F. Ruiz and R. Steuer (Editors), volume 455 of Lecture Notes in Economics and Mathematical Systems, pp. 291-300, Springer.

Glass C.A. Potts C.N. & Shade P. (1992) "Genetic algorithms and neighborhood search for scheduling unrelated parallel machines", Preprint series No.OR47, University of Southampton, UK.

Glover F.W. (1989) "Tabu search, Part I", *ORSA Journal on Computing*, 1 pp. 190-206.

Glover F.W. (1990) "Tabu search, Part II", *ORSA Journal on Computing*, 2 pp. 4-32.

Glover F.W. (1998) "A Template for Scatter Search and Path Relinking", in *Artificial Evolution. Lecture Notes in Computer Science* 1363. J.-K. Hao. E. Lutton. E. Ronald. M. Schoenauer and D. Snyers (Eds.). Springer-Verlag. pp. 13-54.

Glover F.W. & Laguna M. (1997) Tabu search, *Kluwer Academic Publishers*.

Goldberg D.E. (1989) "Genetic Algorithms in Search Optimization and Machine Learning", Reading, Massachusetts: *Addison-Wesley*.

Goldberg D.E. & Richardson (1987) "Genetics Algorithm with sharing for multimodal function optimization", In J.J. Grefenstette (Ed.), *Genetics Algorithms and Their Applications: Proceedings of the Second International Conference on Genetics Algorithms* (pp. 41-49). San Mateo, CA:Morgan Kaufmann.

Gupta J.N.D., Hennig K. & Werner F. (2000a) "Local Search heuristics for two-stage flow shop problems with secondary criterion", *Computers and Operations Research,* forthcoming.

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*, forthcoming.

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*, vol. 10, pp. 251-265.

Hajela P. & Lin C.Y. (1992) "Genetic search strategies in multicriterion optimal design" *Structural Optimization*, 4 pp. 99-107.

Hansen M.P. (1997) "Tabu search for multiobjective optimization: MOTS". Technical Report, Technical University of Denmark. Paper presented at *The 13th International Conference on Multiple Criteria Decision Making*, January 6-10, Cape Town, South Africa.

Hansen M.P. & Jaszkiewicz A. (1998) "Evaluating the quality of approximations to the nondominated set", Technical Report, Institute of Mathematical Modelling (1998-7), Technical University of Denmark.

Hapke M., Jaszkiewicz A. & Slowinski R. (1998) "Interactive Analysis of Multi-Criteria Project Scheduling Problems", *European Journal of Operational Research*, vol. 107, pp. 315-324.

Hollan J.H. (1975) "Adaptation in Natural and Artifitial System", *University of Michigan Press, Ann Arbor*.

Horn J. & Nafpliotis N. (1993) "Multiobjective optimization using the niche Pareto genetic algorithm", IlliGAL Report 93005, Illinois Genetic Algorithm Laboratory, University of Illinois, Urbana, Champaign.

Ishibuchi H. & Murata T. (1998) "A multi-objective genetic local search algorithm and its application to flowshop scheduling", *IEEE Transactions on Systems, Man and Cybernetics-part C: Applications and Reviews*", vol. 28, pp. 392-403.

Ishibuchi H., Yamamoto N., Murata T. & Tanaka H. (1994) "Genetic Algorithms and Neighborhood Search Algorithms for Fuzzy Flowshop Scheduling Problems", *Fuzzy Sets and System*s, vol.67, pp.81-100.

Jaskiewicz A. (2002) "Genetic local search for multi-objective combinatorial optimization", *European Journal of Operational Research* vol.137, pp. 50-71.

Johnson S. M. (1954) "Optimal two- and three-stage production schedules with setup times included", *Naval Research Logistics Quarterly* , vol. 1, pp. 61-68.

Jones D.F., Mirrazavi S.K. & Tamiz M. (2002) "Multi-objective meta-heuristics: An overview of the current state-of-art", *European Journal of Operational Research* vol.137, pp. 1-19.

Liao C.J., Yu W.C. & Joe C.B. (1997) "Bicriterion scheduling in the two-machine flowshop", *Journal of Operational Research Society*, vol. 48, pp. 929-935.

Michalewicz Z. (1996) "Genetic Algorithms + Data Structures = Evolution Programs", *Thir Edition*, *Springer-Verlag Berlin Heidelberg New York*.

Morse J.N. (1980) "Reducing the size of the nondominated set: Pruning by clustering", *Computers and Operations Research*, 7(1-2). 55-66.

Murata T., Ishibuchi H. & Tanaka H. (1996) "Multi-objective genetic algorithm and its applications to flowshop scheduling", *Computers and Industrial Engineering*, vol. 30, pp. 957-968.

Nagar A., Haddock J. & Heragu S. (1995a) "Multiple and bicriteria scheduling: A literatura survey", *European Journal of Operational Research*, vol. 81, pp. 88-104.

Nagar A., Heragu S. & Haddock J. (1995b) "A branch and bound approach for a 2-machine flowshop scheduling problem", *Journal of the Operational Research Society*, vol. 46, pp. 721-734.

Nawaz M., Enscore E.E., & Ham I. (1983) "A heuristic algorithm for the m-machine, n-job flowshop sequencing problem", *OMEGA*, vol. 11, pp. 91-98.

Osman I. & Laporte G. (1996) "Metaheurístics: A bibliography", *Annal of Operations Research*, 65, pp 513-623.

Panwalkar S.S., Dudek, R.A. & Smith M.L. (1973) "Sequencing research and the industrial scheduling problem", in: *Symposium on the Theory of Scheduling and its Applications*, S. E. Elmaghraby (Ed.), Springer, Berlin.

Panwalkar S.S. & Iskander W. (1977), "A survey of scheduling rules", *Operations Research*, vol.25, pp. 45-61.

Pareto V. (1896) Cours D'Economie Politique, vol. 1. Lausanne: F. Rouge.

Potts C. N. & Van Wassenhove L. N. (1982), "A decomposition algorithm for the single machine total tardiness problem", *Operations Research Letters*, vol. 1, pp.177-181.

Rajendran C. (1992) "Two-stage flow shop scheduling problem with bicriteria", *Journal of the Operational Research Society*, vol. 43, pp. 871-884.

Rajendran C. (1995) "Heuristics for scheduling in flowshop with multiple objectives", *European Journal of Operational Research*, vol. 82, pp. 540-555.

Sayin S. & Karabati S. (1999) "A bicriteria approach to the two-machine flow shop scheduling problem", *European Journal of Operational Research*, vol. 113, pp. 435-449.

Schaffer J.D. (1985) "Multiple objective optimization with vector evaluated genetic algorithms", *International Conference on Genetic Algorithms*, Morgan Kaufmann, New York, pp. 93-100.

Şerifoğlu F.S. & Ulusoy G. (1998) "A bicriteria two-machine permutation flowshop problem", *European Journal of Operational Research*, vol. 107, pp. 414-430.

Srinivas N. & Deb K. (1995) "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms" *Evolutionary Computation*, vol. 2(3) pp. 221-248.

Steuer R.E. (1986) Multiple Criteria Optimization: Theory, Computation, and Application. New York: Wiley.

Taillard E. (1990) "Some efficient heuristic methods for the flow shop scheduling pproblem", *European Journal of Operational Research*, vol. 47, pp. 65-74.

Taillard E. (1993) "Benchmarks for basic scheduling problems", *European Journal of Operational Research*, vol. 64, pp. 278-285.

Teghem J., Tuyttens D. & Ulungu E.L. (2000) "An iteractive heuristic method for multi-objective combinatorial optimization", *Computer and Operations Research*, vol. 27, pp.621-634.

Tood D.S. & Sen P. (1997) "A Multiple Criteria Genetic Algorithm for Containership Loading", In T. Bäck (Ed), *Proceedings of the Seventh International Conference on Genetics Algorithms*, San Francisco, California, pp. 674-681. Morgan Kaufmann.

Tuyttens D., Teghem J., Fortemps Ph. and Van Nieuwenhuyze K. (2000) "Performance of the MOSA Method for the Bicriteria Assignment Problem", *Journal of Heuristics*, 6(3), pp. 295-310.

Ulungu E.L. & Teghem J. (1995) "The Two Phases Method: An Efficient Procedure to Solve Bi-Objective Combinatorial Optimization Problems" *Foundations of Computing and Decision Sciences* 20(2), 149-165.

Ulungu E.L., Teghem J. & Ost Ch. (1998) "Efficiency of interactive multi-objective simulated annealing through a case study", *Journal of the Operational Research Society*, vol.49, pp. 1044-1050.

Van Veldhuizen D.A. & Lamont G.B. (2000a) "Multiobjective evolutionary algorithms: Analysing the state-of art", *Evolutionary Computation*, vol. 8(2), pp. 125-147.

Van Veldhuizen D.A. & Lamont G.B. (2000b) "On Measuring Multiobjective Evolutionary Algorithm Performance", *2000 Congress on Evolutionary Computation*, volume 1, pp. 204-211, Piscataway, New Jersey.

Viana A. & Pinho de Sousa J. (2000) "Using metaheuristics in multiobjective resource constrained project scheduling", *European Journal of Operational Research*, vol. 120, pp. 359-374.

Visée M., Teghem J., Pirlot M. & Ulungu E.L. (1998) "Two-phases Method and Branch and Bound Procedures to Solve the Bi-objective Knapsack Problem", *Journal of Global Optimization* 12: 139-155.

Volmann T.E., Berry W.L. & Whybark D.C. (1988) "Manufacturing Planning and Control Systems", Dow Jones-Irwin, 2ª edição.

Wierzbick A. P. (1986) "On the completeness and constructivenss of parametric characterization to vector optimization problems", *OR Spektrum*, 8, pp 73-87.

Zhou G. & Gen M. (1999) "Genetic Algorithm Approach on Multi-criteria Minimum Spaning Tree Problem". *European Journal of Operational Research* vol.114, pp. 141-152.

Zitzler E. & Thiele L. (1999) "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach", *IEEE Transactions on Evolutionary Computation*, vol.3, nº 4, pp. 257-271.