CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Diretoria de Pesquisa e Pós-Graduação

Curso de Mestrado em Modelagem Matemática e Computacional

Aplicação de Métodos Heurísticos aos Problemas de Fluxo Multiproduto Mono-objetivo e Multiobjetivo

Dissertação de Mestrado, submetida ao Curso de Mestrado em Modelagem Matemática e Computacional, como parte dos requisitos para a obtenção do título de Mestre em Modelagem Matemática e Computacional.

Aluno: Fábio Pires Mourão

Orientador: Prof. Dr. Sérgio Ricardo de Souza (CEFET/MG)

Co-orientador: Prof. Dr. Marcone Jamilson Freitas Souza (UFOP)

Belo Horizonte, setembro de 2009.

Livros Grátis

http://www.livrosgratis.com.br

Milhares de livros grátis para download.

Às pessoas que são meus maiores exemplos de vida, meu pai e minha mãe,
Newton e Maria José.

Agradecimentos

Ao concluir este trabalho, tenho a certeza de que é importante agradecer a todos que contribuíram para mais essa realização em minha vida.

Em primeiro lugar, agradeço a Deus, pois nos meus momentos de maior desespero e cansaço, encontrei nele, por meio de minhas orações, toda a força e esperança de que tudo iria dar certo. Sempre soube que se Ele me atribuiu este árduo trabalho, é porque, com toda sua sabedoria, sabia desde o início que eu teria capacidade. Muito obrigado Senhor.

Ao meu orientador, Prof. Dr. Sérgio Ricardo de Souza, por ter se mostrado mais que um orientador, uma pessoa confiável e presente em todos os momentos que tive nessa caminhada.

À minha mãe, Maria José, que soube me educar durante toda a minha vida e por ter me apoiado sempre, por tudo que fez por mim, e por sempre ter buscado o melhor para mim e para os meus irmãos. Deixo aqui minha eterna gratidão por tudo.

Ao meu pai, Newton, que infelizmente não está mais presente, mas a confiança que ele sempre teve em meu sucesso, sempre me faz buscar o melhor para a minha vida e para as pessoas que me rodeiam.

Aos meus dois irmãos, Leanderson e Daniel, por terem sido grandes amigos e pessoas nas quais eu sempre poderei confiar.

À minha namorada, Juliane, por ter visto em mim qualidades até então desconhecidas, e por todos os momentos que tivemos e que ainda teremos.

Ao meu co-orientador, Prof. Dr. Marcone Jamilson Freitas Souza, por todo o conhecimento e incentivo. Agradeço a ele desde o primeiro dia, quando o conheci.

Aos meus amigos do Mestrado, por todo apoio e por terem se tornado verdadeiros amigos para a vida toda. Em particular, gostaria de agradecer ao Filipe e à Mariana.

A todos os funcionários do CEFET-MG, que sempre foram cordiais e me deram a estrutura necessária para o desenvolvimento desse trabalho, em particular à Manuela, Sandra, Bruno e ao pessoal do CCC.

À minha tia Maria Cristina e ao seu marido, Walter, pela ajuda nos dias em que fiquei em Belo Horizonte.

Ao Sr. Filipe Alvelos, pela atenção e por toda a contribuição durante a realização deste trabalho.

Ao Prof. Eduardo Gontijo Carrano, por todo o suporte e pelas valiosas contribuições dadas.

Ao meu grande amigo, Luiz André Vieira Leite, por ter acreditado em mim e por ter me dado motivação para a busca desse feito.

A todas as demais pessoas que contribuíram direta e indiretamente para a realização deste trabalho.

Ao CNPq, pelo apoio financeiro.

“O único lugar onde o sucesso vem antes do trabalho é no dicionário.”
Albert Einstein

Resumo

Este trabalho apresenta métodos heurísticos aplicados ao problema de fluxo multiproduto binário mono-objetivo e multiobjetivo. O Problema de Fluxo Multiproduto pertence à classe dos problemas NP-difíceis e possui diversas aplicações práticas, principalmente nas áreas de telecomunicações e logística. O uso de métodos heurísticos se justifica pelo fato do problema tratado ser NP-difícil e, também, porque o número de variáveis envolvidas na resolução do problema é, geralmente, elevado, além de ser um problema combinatório. Problemas de Fluxo Multiproduto surgem quando vários produtos trafegam e competem pelos arcos capacitados de uma rede identificada por um grafo orientado. Os nós são pontos de oferta ou demanda desses produtos e o objetivo é determinar, ao menor custo, o fluxo destes produtos pela rede, de forma a atender as restrições de capacidade, que limitam o fluxo de produtos em cada arco, e as restrições de conservação de fluxo, responsáveis por gerenciar o fluxo de produtos na rede. As variações deste problema consistem no problema linear, inteiro e binário. Neste trabalho, foi considerado o problema binário, em que cada unidade de cada produto é indivisível, além do fluxo de cada produto também o ser. Foi desenvolvida uma heurística construtiva para geração de soluções iniciais e implementado um Algoritmo Genético (AG), além da metaheurística *Iterated Local Search* (ILS). Os testes foram realizados utilizando as duas heurísticas separadamente e, também, utilizando o AG seguido pelo ILS. Os resultados obtidos foram bons, quando comparados aos obtidos por meio de uma decomposição baseada no problema da mochila e aos obtidos por um algoritmo de partição e geração de colunas com cortes, baseado numa decomposição por caminhos. Além do problema mono-objetivo, foi considerado também o problema multiobjetivo, adicionando-se, ao problema mono-objetivo, uma nova função objetivo que mede o congestionamento da rede. A metodologia multiobjetivo se baseia no algoritmo NSGA-II e não foram feitas comparações por não terem sido encontradas abordagens semelhantes na literatura. Os resultados obtidos mostram a possibilidade de uso desta formulação na análise do comportamento de redes multiproduto.

Palavras-Chave: Problema de Fluxo Multiproduto; Problema Multiobjetivo de Fluxo Multiproduto; Problema de Minimização do Congestionamento de Rede; *Iterated Local Search*; Algoritmos Genéticos; Métodos NSGA-II.

Abstract

This dissertation is concerned the application of heuristic methods to two problems: the multicommodity flow problem (MFP) and the multicommodity flow multiobjective problem. Multicommodity Flow Problems arise when different products compete and travel for the capacitated arcs of a network identified by a directed graph. The nodes are points of supply or demand for these products and the objective is to determine the minimum cost flow of products through the network in order to guarantee the capacity flow constraints in each arch, and the conservation flow constraints. The multicommodity flow problem belongs to the NP-hard class problems and has many practical applications, especially in the areas of telecommunications and logistics. The use of heuristic methods is justified by the fact that the problem be NP-hard and, as it is a combinatorial problem, the number of variables involved in solving it is usually high. Variations of this problem consist in the linear, integer and binary problems. In this work, it is considered the binary multicommodity flow problem, where each unit of each product is indivisible as well as the flow of each product. A constructive heuristic is used to generate initial solutions, being implemented a genetic algorithm (GA) and the Iterated Local Search (ILS) metaheuristic. Computational tests are performed using the two implemented metaheuristics and an hybrid one, formed by applying GA and ILS jointly. The obtained results are good, when compared with the ones obtained using exact methods. The multicommodity flow multiobjective problem treated here arise when it is added, to the MFP, a new objective function associated with network congestion. This problem is solved using an adaptation of NSGA-II algorithm. No comparisons are made since, up to author knowledge, there is no similar approach in the literature.
Keywords: Multicommodity Flow Problem; Multicommodity Flow Multiobjective Problem; Network Congestion Minimization Problem; Iterated Local Search; Genetic Algorithm; NSGA-II Method.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução Geral

1.1 Preliminares

Problemas de Fluxo Multiproduto (PFM) foram estudados, inicialmente, no início da década de 60, com contribuições iniciais de Fulkerson e Ford (1962) e Hu (1963). Muitas aplicações reais podem ser formuladas por meio de um PFM, principalmente nas áreas de telecomunicações e transportes, dentre as quais podem ser citadas: roteamento de tráfego na internet (Buriol, 2003); roteamento de mensagens em redes de telecomunicações (Hu, 1969); seqüenciamento de operações em refinarias de petróleo (Milidiu et al., 2001); seqüenciamento de carga (Shan, 2007); otimização em redes de fibra ótica (Ozdaglar e Bertsekas, 2003) e alocação de trens em uma rede ferroviária (Mendes, 1999).

O problema em tela é NP-difícil, de acordo com (Garey e Johson, 1979), além de ser um problema de Otimização Combinatória, que apresenta, em geral, um elevado número de variáveis e restrições. Esse fato justifica a utilização de técnicas heurísticas na tentativa de solucioná-lo, pois tais métodos podem, mesmo sem garantir otimalidade, gerar bons resultados e em menores tempos computacionais quando comparados a métodos exatos de otimização. O problema é modelado por meio de uma rede identificada por um grafo, no qual vários produtos trafegam pelos arcos capacitados da rede a um determinado custo e competem pela capacidade desses arcos. Os nós dessa rede representam pontos de oferta e demanda para os vários produtos e o problema é, então, o de determinar o fluxo desses produtos ao menor custo possível, de forma a atender basicamente dois conjuntos de restrições: restrições de conservação de fluxo, que gerenciam o fluxo dos produtos pela rede, e restrições de capacidade, que limitam o tráfego de produtos nos arcos.

O PFM pode ser dividido em três classes. A primeira consiste no Problema Linear, que considera que cada unidade de cada produto é divisível e as variáveis envolvidas são reais. A segunda, consiste no Problema Inteiro, em que cada unidade de cada produto é indivisível, porém o fluxo de um determinado produto pode ser dividido por diferentes caminhos. A terceira consiste no Problema Binário, que considera que, além de cada unidade de cada produto ser indivisível, os fluxos destes produtos devem fazer uso apenas de uma rota. Neste trabalho, foi considerada a classe dos problemas binários. Assim, surgem duas novas restrições no problema, além das mencionadas, que são a restrição de integralidade, cujo objetivo é garantir que as variáveis envolvidas sejam inteiras, e a restrição de não-negatividade, que

garante que as variáveis sejam maiores ou iguais a zero.

Existem várias estratégias que buscam solucionar o PFM. Algumas delas podem ser encontradas descritas em (Ahuja et al., 1993), como, por exemplo, a aplicação de Relaxação Lagrangeana, do método de Geração de Colunas e da Decomposição de Dantzig-Wolfe. Estas estratégias de resolução citadas foram também descritas e utilizadas por (Alvelos, 2005). Segundo (Castro, 2003), existem quatro classes de métodos para a resolução de um PFM: métodos baseados no método simplex; métodos de decomposição; métodos de aproximação; métodos de pontos interiores. A estratégia utilizada neste trabalho consiste na aplicação de métodos heurísticos populacionais ou baseados em busca local. A escolha se deve ao fato do problema tratado ser, como já dito, um problema NP-difícil, ao lado de ser um problema de Otimização Combinatória, que, por sua própria natureza, apresenta, em geral, um elevado número de variáveis e restrições.

Além da formulação clássica do problema, o presente trabalho também apresenta uma formulação multiobjetivo. Essa abordagem multiobjetivo busca, além de minimizar o custo do fluxo dos produtos pela rede, balanceá-la, a fim de encontrar o congestionamento mínimo da rede. Assim, no Problema Multiobjetivo de Fluxo Multiproduto (PMFM), são consideradas duas funções a serem otimizadas, a saber: função de custo do PFM e função de congestionamento, baseada na função apresentada por (Fortz e Thorup, 2000a), (Fortz et al., 2002) e (Buriol, 2003). A função que mede o congestionamento na rede, segundo esta proposta, é convexa e linear por partes e possui o objetivo de penalizar cada arco de forma independente, de acordo com sua utilização. Com isso, o objetivo é balancear a rede, de forma a uniformizar os fluxos dos arcos e, consequentemente, levando ao balanceando da rede multifluxo.

Este trabalho apresenta uma abordagem mono-objetivo, para a qual são feitas comparações com resultados presentes na literatura, e uma abordagem multiobjetivo. Em ambos os casos são utilizadas técnicas heurísticas aplicadas à solução de cada problema.

1.2 Motivação

O estudo de Problemas de Fluxo Multiproduto é motivado pelo fato de diversos problemas práticos, principalmente nas áreas de transportes e telecomunicações, podem ser modelados por meio de da estrutura do Problema de Fluxo Multiproduto, como em Buriol (2003), (Ozdaglar e Bertsekas, 2003), (Mendes, 1999), (Shan, 2007) e (Frangioni e Gendron, 2007). Em muitos trabalhos encontrados na literatura, não é dada importância ao balanceamento da rede, porém, dada a necessidade de evitar o congestionamento das mesmas, como mostrado em (Buriol, 2003) e (Buriol et al., 2003), é formulada uma função que visa encontrar o balanceamento da rede.

Os Problemas de Fluxo Multiproduto são NP-difíceis e, de acordo com (Silva, 2007), a complexidade do problema inteiro aumenta com a inserção de novos produtos ao problema. Neste trabalho é feita uma abordagem do problema binário, que representa uma variação do problema inteiro. A motivação do uso de técnicas heurísticas se dá pela dificuldade da obtenção de soluções ótimas, devido à complexidade combinatória do problema e pelo elevado número de variáveis e restrições. Técnicas heurísticas têm sido amplamente utilizadas na resolução de problemas combinató-

rios, pois, mesmo sem garantir otimalidade, podem gerar boas soluções.

Foram utilizadas heurísticas como o *Iterated Local Search* (ILS), apresentada por (Lourenço et al., 2002) e o uso desta se dá pelo fato da mesma ser amplamente utilizada e por apresentar sucesso em diversos trabalhos encontrados na literatura, como em (Bauer et al., 2008) e o Algoritmo Genético, proposto inicialmente por (Holland, 1975), além do Método da Descida Randômica. Todos estes foram aplicados ao problema mono-objetivo, no qual é considerada somente a função de custo do PFM. Foi desenvolvida uma heurística construtiva aleatória, para a geração de soluções iniciais para o problema, a fim de perder a aleatoriedade e a diversidade, que são características dos algoritmos evolutivos. Na abordagem multiobjetivo, foi utilizado o método *Nondominated Sorting Genetic Algorithm* (NSGA) II, proposto por (Deb et al., 2002), por apresentar estratégias elitistas e de diversificação, possuir fácil implementação e ser eficiente na busca do conjunto ótimo de Pareto.

Os algoritmos mono-objetivos foram testados com instâncias geradas aleatoriamente pelo GenMCF (*Generator Multicommodity Flow*) e os resultados obtidos foram comparados aos encontrados em (Alvelos, 2005). Estas instâncias foram utilizadas também para os testes dos algoritmos multiobjetivos e não foram feitas comparações com resultados contidos na literatura, por não ter sido encontrada, até o momento, uma abordagem semelhante.

1.3 Caracterização do Problema

A modelagem do PFM se dá via grafo, no qual os vários nós da rede representam pontos de oferta e demanda para os produtos que trafegam pelos arcos da rede e competem pela capacidade dos mesmos. O objetivo é o de determinar o fluxo destes produtos, a um custo mínimo, de forma a atender as restrições de capacidade, que limitam a quantidade de produtos que devem fluir pelos arcos, de conservação de fluxo, que gerenciam o fluxo dos produtos pelos arcos da rede, e de integralidade e não-negatividade, oriundas da própria formulação do problema tratado, o problema binário. A cada aresta do grafo e a cada produto há um custo associado, sendo o somatório desses custos a função a ser otimizada na abordagem mono-objetivo. As variáveis de decisão representam os fluxos dos produtos em cada arco.

A abordagem multiobjetivo é composta pela função e restrições mencionadas, além de outra função, cujo objetivo é determinar o congestionamento mínimo da rede, fazendo o balanceamento da mesma. A idéia desta segunda função é penalizar cada arco de acordo com a quantidade de produtos que trafegam pelo mesmo, sendo mais caro um produto trafegar por um arco cuja utilização está próxima ou é superior à capacidade do mesmo em relação a outro arco que estiver sendo pouco utilizado, ou até mesmo ocioso.

Em suma, o problema mono-objetivo consiste em determinar o fluxo de custo mínimo de produtos por uma rede capacitada, de forma a atender aos conjuntos de restrições, e o problema multiobjetivo consiste em, além de determinar o fluxo de custo mínimo de cada produto, balancear a rede, homogeinizando o fluxo dos produtos envolvidos pelos arcos.

1.4 Metodologia Proposta

A metodologia proposta para a abordagem mono-objetivo se dá por meio de heurísticas, populacionais e de busca local, aplicadas à solução do PFM. Primeiramente, é proposta uma heurística construtiva aleatória, utilizada para a geração de soluções iniciais, e, em seguida, são utilizados o ILS e o AG, tendo sido aplicado o método da Descida Randômica aos dois algoritmos, pois o ILS necessita da execução de um método de busca local e ao AG foi necessário a aplicação de busca local, com o objetivo de melhorar a qualidade das soluções.

A metodologia da abordagem multiobjetivo consiste num método constituído de duas fases, sendo a Fase 1 um AG mono-objetivo que busca encontrar soluções factíveis para serem usadas como soluções iniciais no método NSGA II. Também foram propostos métodos de refinamento do conjunto ótimo de Pareto, a fim de encontrar um maior número de soluções candidatas, ou de encontrar soluções dominantes em relação às soluções encontradas pelo NSGA II.

Os testes computacionais incidem sobre instâncias geradas aleatoriamente pelo GenMCF, desenvolvido por Alvelos (2005). No caso do problema mono-objetivo, foram feitas comparações com resultados obtidos por métodos exatos, encontrados em (Alvelos, 2005). No caso multiobjetivo, não foram feitas comparações com outros resultados da literatura, pelo fato de não ter sido encontrada, até o momento, uma abordagem semelhante.

1.5 Objetivos

1.5.1 Objetivo Geral

O objetivo geral deste trabalho é propor métodos heurísticos aplicados à solução do Problema de Fluxo Multiproduto Binário e analisar os algoritmos por meio de comparações com resultados obtidos por meio de métodos exatos, no caso do problema mono-objetivo.

Além de propor e analisar métodos heurísticos aplicados ao caso mono-objetivo, é objetivo também formular e analisar o caso multiobjetivo do problema, considerando o balanceamento da rede, além de propor algoritmos heurísticos aplicados à geração das soluções candidatas a ótimas de Pareto e, de uma maneira geral, propor um algoritmo evolutivo cuja idéia pode ser aplicada a outros problemas combinatórios de otimização multiobjetivo com restrições, por meio de um algoritmo evolutivo aplicado exclusivamente à geração de soluções iniciais.

1.5.2 Objetivos Específicos

Para que este objetivo geral seja alcançado, é necessário atingir os seguintes objetivos específicos:

(i) Conhecimento da literatura relevante associada aos principais temas envolvidos no trabalho, qual seja, Problemas de Fluxo Multiproduto, Classificação dos Problemas de Fluxo Multiproduto, Algoritmos aplicados à soluções de tais problemas, Otimização Multiobjetivo, Métodos de Resolução de Problemas Multiobjetivos e Problemas de Congestionamento Mínimo;

(ii) Estudo e implementação computacional de métodos heurísticos mono e multi-objetivos;

(iii) Comparação, quando possível, dos resultados obtidos pelos métodos propostos com resultados conhecidos na literatura;

1.6 Estrutura da Dissertação

Este trabalho se estrutura da seguinte forma: o Capítulo 1 apresenta uma introdução geral sobre o trabalho. O Capítulo 2 apresenta uma visão geral de Problemas de Fluxo Multiproduto, com as formulações existentes e, também, com uma revisão bibliográfica sobre o tema. Ainda neste capítulo, é apresentada a formulação do outro problema similar ao tratado, o de congestionamento mínimo. O Capítulo 3 exibe definições de otimização multiobjetivo e também apresenta descrições de alguns métodos tradicionais de resolução de problemas multiobjetivos. O Capítulo 4 apresenta descrições de métodos heurísticos aplicados tradicionalmente a problemas mono e multiobjetivos de otimização combinatória. As metodologias e os resultados para o problema mono-objetivo e multiobjetivo são apresentadas nos Capítulos 5 e 6, respectivamente e, finalmente, o Capítulo 7 apresenta as conclusões e os trabalhos futuros.

Capítulo 2

Problemas de Fluxo Multiproduto

Este capítulo apresenta o Problema de Fluxo Multiproduto; discute algumas de suas aplicações no modelamento de problemas reais; apresenta uma revisão literária dos textos que são relacionados ao que se pretende discutir no presente trabalho; introduz as definições quanto ao problema, pertinentes ao entendimento necessário para o escopo desta dissertação; e apresenta o modelamento matemático que lhe é próprio.

2.1 Introdução

Problemas de Fluxo Multiproduto (PFM) possuem uma larga variedade de aplicações, principalmente nas áreas de transporte e telecomunicações. Dentre essas aplicações, podem ser citadas:

- Roteamento de tráfego na internet, como em Buriol (2003);
- Otimização em redes de fibra ótica Ozdaglar e Bertsekas (2003);
- Alocação de trens em uma rede ferroviária (Mendes, 1999);
- Seqüenciamento de carga (Shan, 2007);
- Problema de projeto de redes, como em Frangioni e Gendron (2007).

Os Problemas de Fluxo Multiproduto (PFM) podem ser classificados em três grandes classes, a saber:

- problema linear: nesta situação, as variáveis de decisão, correspondentes ao fluxo de cada produto em cada arco, podem assumir valores não-inteiros. Portanto, considera-se que cada unidade de cada produto é divisível e as variáveis envolvidas são reais. O modelo apresentado na expressão (2.2) é um exemplo de um problema linear, acrescentando-se a restrição $x_{ij}^k \in \Re_+$;
- problema inteiro: nesta formulação do PFM cada variável assume um valor inteiro, o que significa que cada unidade de cada produto é indivisível, porém, o fluxo de um produto não precisa necessariamente seguir uma única rota, saindo do nó-origem e indo para o nó-destino, podendo ser dividido por diferentes

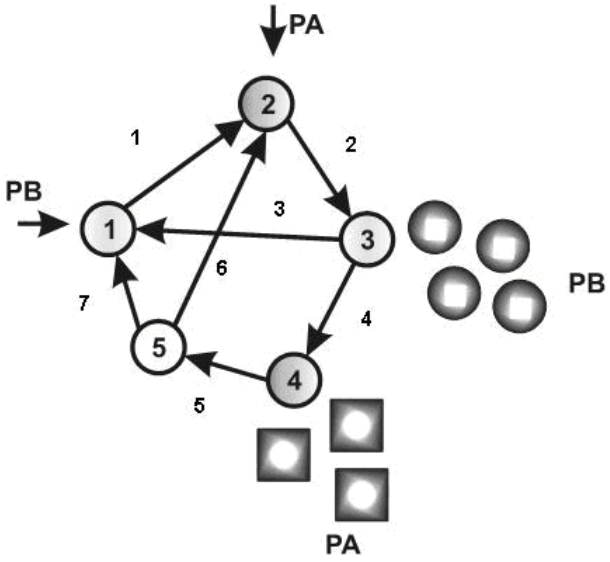

Figura 2.1: Exemplo de uma rede multiproduto.

caminhos, de acordo com (Alvelos, 2005). Assim, acrescentando-se a restrição $x_{ij}^k \in I_+$ à expressão (2.2), o problema passa, então, a ser caracterizado como inteiro.

- problema binário: nesta formulação do problema, todas as unidades de um produto devem fazer uso apenas de uma rota, sendo, assim, um caso particular do problema inteiro. Contudo, diferentemente do anterior, o fluxo dos produtos não pode ser dividido, ou seja, devem fazer uso apenas de uma rota. O problema binário pode ser visto como um conjunto de problemas de caminhos mínimos, um para cada produto, com restrições que relacionam estes produtos. É adicionada a restrição da integralidade dos fluxos, além da integralidade dos valores numéricos das variáveis, presente nos problemas inteiros. Assim, as soluções podem ser representadas por variáveis binárias. Uma formulação do Problema Binário será apresentada na Seção 2.2.

A modelagem deste problema é feita por meio de um grafo direcionado, em que os diversos produtos trafegam, a um determinado custo, de forma a atender a oferta e a demanda, competindo pela capacidade dos arcos. Cada produto trafega em um certo arco a um determinado custo. Os nós correspondem aos pontos de oferta/demanda dos produtos. Em linhas gerais, existem dois conjuntos de restrições que precisam ser consideradas, independente da classe do problema de fluxo tratado:

- Restrições de Conservação de Fluxo: possuem a finalidade de gerenciar o fluxo na rede, considerando os nós de oferta/demanda;
- Restrições de Capacidade dos Arcos: limitam a quantidade de produtos que podem trafegar em cada arco, não permitindo trafegar em um arco uma quantidade de produtos maior que a suportada por ele.

O Problema de Fluxo Multiproduto consiste, então, em determinar o fluxo dos produtos pela rede, ao menor custo possível, de modo a atender às restrições de conservação de fluxo e às restrições de capacidade dos arcos.

A Figura 2.1 apresenta um exemplo de uma rede de fluxo multiproduto com 5 nós, 7 arcos e 2 produtos, de forma que os nós 4 e 2 são de oferta e demanda, respectivamente, para o produto *PA*, e os nós 3 e 1 são de oferta e demanda, para o produto *PB*. Neste exemplo, caso o PFM esteja definido como um problema linear, as variáveis de decisão são reais e mostram os valores de fluxos pelos arcos. Caso se estude a versão inteira, as variáveis de decisão continuam sendo os fluxos pelos arcos, contudo, como variáveis inteiras. Neste caso, acrescenta-se, ao conjunto de restrições anteriores, as restrições de integralidade das variáveis de decisão. Caso os fluxos façam uso de apenas uma rota, as variáveis de decisão do problema passam a definir a passagem ou não de um dado produto por um certo arco, sendo, portanto, variáveis binárias. Então, deve-se acrescentar, ao conjunto de restrições, a restrição quanto ao tipo de variáveis envolvidas.

2.2 Revisão Bibliográfica

Os trabalhos iniciais sobre o PFM datam do início da década de 60, com contribuições iniciais de Hu (1963) e Fulkerson e Ford (1962). Esta Seção discute trabalhos publicados recentemente a respeito do PFM. Wille et al. (2005) utilizam as metaheurísticas Busca Tabu e Algoritmos Genéticos (AG) para gerar soluções para um problema multiproduto topológico de redes IP. O objetivo é minimizar o custo da distribuição de pacotes pela rede, atendendo a um conjunto de restrições, entre elas a qualidade de serviços. Os resultados computacionais obtidos mostram uma melhor eficiência da abordagem feita pelo AG, gerando boas soluções para redes de tamanho médio, em comparação com o método Busca Tabu.

Em Guardia e Mello (2007) são apresentados um estudo e uma implementação numérica do método de pontos interiores primal-dual para o problema não-linear convexo de fluxo multiproduto. A cada iteração, o método proposto resolve um sistema linear, usando o algoritmo do gradiente conjugado com um pré-condicionador. Os testes foram realizados para redes de diversas dimensões, com funções de custo não-lineares e, segundo Guardia e Mello (2007), os resultados computacionais mostram a eficiência do método de pontos interiores para o problema de fluxo multiproduto.

Em Alvelos (2005), a abordagem realizada é por meio de geração de colunas para programação linear e inteira. Os testes computacionais são feitos em instâncias geradas pelo GenMCF (*Generator Multicommodity Flow*), desenvolvido por Alvelos (2005).

Bocanegra et al. (2000) propõem um algoritmo para resolver o problema não linear de fluxo multiproduto, utilizando planos de corte e centros analíticos. O problema é relaxado utilizando a função lagrangeana parcial, construída a partir de hiperplanos suporte. Foi também utilizado o método de pontos interiores. Segundo Bocanegra et al. (2000), o algoritmo proposto se destaca por gerar uma sequência monótona estritamente crescente de cotas para alcançar a solução e, consequentemente, segue trajetórias centrais associadas ao máximo da função modelo.

Park et al. (2002) utilizam a técnica de geração de colunas para resolver duas classes de problemas de fluxo multiproduto inteiro, sendo que em um dos problemas são dados o conjunto de produtos com suas respectivas demandas e custo unitário

em cada arco. O objetivo desse problema é selecionar um subconjunto de produtos para serem roteados e associar, a cada um deles, uma única rota ligando o nó de origem ao nó destino, tendo, como finalidade, maximizar o benefício da distribuição. Para avaliar o benefício, é utilizado um valor constante para cada produto em cada rota. O segundo problema consiste na seleção de todos os produtos necessários, sendo o objetivo o de determinar o custo mínimo.

Em Lim e Smith (2007) é considerado um problema de interdição de arcos numa rede de fluxo multiproduto, de forma que, para cada interdição de arco, existe um custo associado. O trabalho aborda o problema discreto, de forma que um arco precisa ser interditado totalmente, e também o problema contínuo, em que um determinado arco pode ser parcialmente interditado.

Chekuri et al. (2004) propõem um algoritmo para resolver problemas de fluxo multiproduto dada uma formulação sobre um grafo capacitado e não direcionado. No trabalho, foi também abordado um algoritmo *on-line*, no qual o método precisa decidir imediatamente sobre a chegada, ou não, de um produto a um determinado nó.

Goffin et al. (1995) apresentam um algoritmo que aborda técnicas de decomposição usando relaxação lagrangeana. São aplicados os métodos de plano de corte e centro analítico para resolver o problema de maximização de uma função dual não-diferencial.

Já Buriol (2003) trata de problemas relacionados ao gerenciamento de redes sobre o protocolo OSPF (*Open Shortest Path First*), de forma que pesos são atribuídos aos arcos da rede e o objetivo é o de minimizar o congestionamento.

2.3 Formulação Matemática do PFM

Primeiramente, será apresentada uma formulação básica do PFM.

Seja uma rede $\mathcal{R}$, representada por um grafo direcionado $\mathcal{G}(\mathcal{N}, \mathcal{A})$, com n nós, a arcos e p produtos. Define-se o conjunto dos nós como o conjunto $\mathcal{N}$, o dos arcos como $\mathcal{A}$ e o dos produtos como $\mathcal{P}$, de modo que $|\mathcal{N}| = n$, $|\mathcal{A}| = n$ e $|\mathcal{P}| = p$. O objetivo do problema é o de determinar o fluxo de produtos na rede, de forma a minimizar o custo, atendendo às restrições.

O problema de fluxo de custo mínimo multiproduto é dado, então, por:

$$\min \quad \sum_{k=1}^{p} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k \tag{2.1a}$$

$$\text{suj. a} \quad \sum_{(i,j)\in A} x_{ij}^k - \sum_{(j,i)\in\mathcal{A}} x_{ji}^k = b_i^k, \quad \forall\, i \in \mathcal{N}, 1 \leqslant k \leqslant p \tag{2.1b}$$

$$\sum_{k\in\mathcal{P}} x_{ij}^k \leqslant u_{ij}, \quad \forall (i,j) \in \mathcal{A} \tag{2.1c}$$

$$x_{ij}^k \geq 0, \quad \forall\, k \in \mathcal{P}, \quad \forall\, (i,j) \in \mathcal{A} \tag{2.1d}$$

em que:

- a expressão (2.1a) representa a função a ser minimizada, que indica os valores de custo de passagem dos k produtos pelos arcos (i, j);

- a expressão (2.1b) representa as restrições de conservação de fluxo, que são responsáveis pelo gerenciamento do fluxo na rede;
- a expressão (2.1c) representa as restrições de capacidade, sendo responsáveis por limitar a capacidade de cada arco.

No problema considerado neste trabalho, as variáveis são inteiras, porém, na formulação geral, o problema pode ser linear contínuo (não-inteiro). Além disso:

- c_{ij}^k representa o custo unitário do produto no arco (i, j);
- x_{ij}^k representa a variável de decisão, que é o fluxo do produto k pelo arco (i, j). Caso este valor seja não-nulo e positivo, significa que o produto faz uso do arco em questão;
- u_{ij} representa a capacidade do arco (i, j);
- b_i^k indica se o nó i é oferta ou demanda para o produto k. Considerando d^k como a oferta/demanda para o produto k, i.e., d^k é a quantidade de produto k que precisa sair de um nó origem e chegar a um nó destino e considerando que a cada produto k está associado um par origem-destino, então b_i^k pode assumir os seguintes valores:

$$b_i^k = \begin{cases} d^k, \text{ se o nó } i \text{ é oferta para o produto } k; \\ -d^k, \text{ se o nó } i \text{ é demanda para o produto } k; \\ 0, \text{neste caso, } i \text{ é um nó de transbordo.} \end{cases}$$

Deve-se ressaltar que um nó é definido como de transbordo se não é caracterizado como de origem de ou destino e, contudo, os produtos o utilizam como nó de passagem.

O modelo apresentado em (2.1a) pode também ser posto como na expressão (2.2), a seguir:

$$\begin{aligned} \min \quad & \sum_{k=1}^{p} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k && \text{(2.2a)} \\ \text{suj. a} \quad & Bx^k = b^k, \ \ \forall\, k, \ \ 1 \leq k \leq p && \text{(2.2b)} \\ & \sum_{k\in\mathcal{P}} x_{ij}^k \leqslant u_{ij}, \ \ \forall (i,j) \in \mathcal{A} && \text{(2.2c)} \\ & x_{ij}^k \geq 0, \ \ \forall k \in \mathcal{P}, \ \ \forall (i,j) \in \mathcal{A} && \text{(2.2d)} \end{aligned}$$

A diferença entre as formulações apresentadas nas expressões (2.1) e (2.2) está na forma de representação das restrições de conservação de fluxo, de modo que:

- a matriz B é a matriz de incidência nó-arco;
- b^k é vetor de oferta/demanda do produto k, responsáveis pelo gerenciamento do fluxo na rede;
- x^k é o vetor de fluxo do produto k.

A matriz B é a matriz de incidência e pode ser definida pela relação nó-arco e pela relação arco-rota. Estas relações serão definidas nas próximas seções. Neste trabalho, foi feita uma abordagem pela relação nó-arco. Sendo considerada a relação nó-arco, os elementos b_{ij} pertencentes à matriz B são:

$$b_{ij} = \begin{cases} 1, \text{ se o nó } i \text{ é fonte para o nó } j; \\ -1, \text{ se o nó } j \text{ incide no nó } i; \\ 0, \text{ em caso contrário.} \end{cases}$$

Caso $b_{ij} = -1$, diz-se que nó i é um nó sumidouro.

A matriz B é uma matriz totalmente unimodular, ou seja, o determinante de qualquer submatriz quadrada de B pertence ao conjunto $\{-1, 0, +1\}$. Isto significa que, mesmo adicionando variáveis de folga ou artificiais e sendo o conjunto de oferta/demanda valores inteiros, a matriz continua sendo totalmente unimodular, o que garante a solução inteira em qualquer vértice do politopo convexo gerado pelo conjunto de restrições. Uma demonstração dessa propriedade pode ser encontrada em (Bazaraa et al., 1990).

A seguir, serão discutidas a formulação nó-arco e a formulação arco-rota. Em cada formulação, o fluxo da rede é representado de uma maneira distinta. Neste trabalho, será utilizada a formulação nó-arco.

2.3.1 Formulação nó-arco

A formulação apresentada pela Equação (2.2) é uma formulação nó-arco.

Seja o mesmo exemplo da rede apresentada na Figura 2.1, com 5 nós, 7 arcos e 2 produtos, atribuindo, a cada arco, uma capacidade e um custo, ou seja, cada produto trafega a um custo, que só depende do arco em questão. Obviamente, em alguns casos o custo pode depender também do produto. Neste trabalho foram considerados ambos os casos. A Tabela 2.1 apresenta os custos e capacidades de cada arco, para o exemplo em questão.

Tabela 2.1: Exemplo: Arco, capacidade, custo.

Arco	Capacidade	Custo
1	5	2
2	7	3
3	10	4
4	5	1
5	8	1
6	10	4
7	5	1

A Tabela 2.2 apresenta a relação de oferta/demanda de produtos para esta rede. A coluna "Origem" apresenta o nó de oferta do produto; a coluna "Destino" apresenta o nó de demanda; e a coluna "Demanda" corresponde à quantidade de produtos que precisam fluir do nó-origem ao nó-destino.

A matriz de incidência nó-arco B é responsável por gerenciar o fluxo na rede e está diretamente relacionada às restrições de conservação de fluxo, representadas

Tabela 2.2: Exemplo: Oferta/demanda de produtos

Produto	Origem	Destino	Demanda
PA	4	2	3
PB	3	1	4

pela expressão (2.2b). Na formulação nó-arco, cada linha representa um nó da rede, enquanto cada coluna representa um arco da mesma. No caso deste exemplo, ela é dada, então, por:

$$B = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{bmatrix}$$

Na matriz de incidência nó-arco, o nó 1 (linha 1) é um nó fonte para o arco 1 (coluna 1), porém é um nó sumidouro para o arco 3 (coluna 3), i.e., o arco 3 incide no nó 1, pois $b_{13} = -1$. Obviamente, esta matriz é de dimensão $n \times a$.

O vetor de capacidades é representado por u e possui dimensão $a \times 1$. Os vetores b^1 (oferta/demanda de PA) e b^2 (oferta/demanda de PB) são as colunas da matriz b, a qual representa oferta/demanda dos produtos PA e PB. A matriz de oferta/demanda b é de dimensão $n \times p$.

$$u = \begin{bmatrix} 5 \\ 7 \\ 10 \\ 5 \\ 8 \\ 10 \\ 5 \end{bmatrix}, \quad b = \begin{bmatrix} 0 & -4 \\ -3 & 0 \\ 0 & 4 \\ 3 & 0 \\ 0 & 0 \end{bmatrix}$$

Ainda existem variações da formulação apresentada, de acordo com a definição dos nós origem e destino. (Jones et al., 1993) apresentam uma aplicação do método de Decomposição de Dantzig-Wolf a três formulações nó-arco, de modo que

- Destino-Específico (DSP): um produto p pode partir de qualquer nó, porém seu destino é um nó específico;
- Produto-Específico (CSP): cada produto é especificado por seu índice p, podendo fluir de qualquer nó-origem para qualquer nó-destino;
- Origem-Destino Específico (ODP): todos os produtos são identificados por seu par de nós origem e destino.

Normalmente, a escolha de uma dada abordagem depende da análise dos conjuntos de restrições e do método de resolução proposto. No caso do presente trabalho, a abordagem adotada é a Origem-Destino Específico (ODP), pois os problemas de interesse possuem uma estrutura na forma de um nó-origem e um nó-destino específicos.

Além disso, neste trabalho considera-se que a quantidade de um determinado produto que sai de um nó-origem é igual à quantidade que chega ao nó-destino. Além do mais, é considerado que todas as unidades de um produto são indivisíveis e que devem fazer uso somente de um caminho.

A abordagem utilizada neste trabalho implica no acréscimo de restrições que garantam que o fluxo em cada arco seja inteiro e não seja dividido. Também é garantido que a cada produto está associado um, e somente um, par de nós origem-destino.

2.3.2 Formulação arco-rota

Nesta formulação, a matriz de incidência não relaciona nó-arco, mas, sim, arco-rota, de modo a indicar se um determinado arco pertence a uma das possíveis rotas de um produto.

Seja d^k a quantidade de um produto $k \in \mathcal{P}$, que precisa ser transportado de um nó-origem s a um nó-destino t. Assim, $\mathcal{R}(k)$ é o conjunto de todas as possíveis rotas que ligam o nó s ao nó t para o transporte do produto k. A variável de decisão x_r^k representa, então, a quantidade do produto k que deve fluir do nó s ao nó t, passando pela rota $r \in \mathcal{R}(k)$.

O custo c_r^k corresponde ao custo total da rota, i.e., à soma dos custos de todos os arcos contidos na rota r, por onde trafega o produto k.

A matriz de incidência arco-produto B é definida relacionando-se um determinado arco a uma rota r, de forma que b_{ij}^r é igual a 1 se o arco (i, j) faz uso da rota r ou b_{ij}^r é igual a 0, em caso contrário.

Seja um problema definido com n nós, a arcos e p produtos, de forma que $\mathcal{N}$, $\mathcal{A}$ e $\mathcal{P}$ representam os conjuntos de nós, arcos e produtos, respectivamente. A formulação é, então, representada pela expressão (2.3).

$$\min \quad \sum_{k=1}^{p} \sum_{r \in \mathcal{R}(p)} c_r^k x_r^k \tag{2.3a}$$

$$\text{suj. a} \quad \sum_{k=1}^{p} \sum_{r \in \mathcal{R}(p)} b_{ij}^r x_r^k \leqslant u_{ij}, \quad \forall (i,j) \in \mathcal{A} \tag{2.3b}$$

$$\sum_{r \in \mathcal{R}(p)} x_r^k = d^k, \quad 1 \leq k \leq p \tag{2.3c}$$

$$x_p^k \geq 0, \quad \forall r \in \mathcal{R}(p), \quad 1 \leq k \leq p \tag{2.3d}$$

As restrições de capacidade, representadas pela expressão (2.3) garantem que, em um dado arco (i, j), não trafegue uma quantidade de produtos superior que a suportada por ele. Estas restrições garantem que a quantidade total de produtos que utilizam um arco em todas as rotas seja menor ou igual à capacidade do arco, definida como u_{ij}. As restrições apresentadas pela expressão (2.3) possuem a função de distribuir os produtos pelas possíveis rotas que ligam o nó-origem ao nó-destino, atendendo às demandas.

Assim, as seguintes definições são válidas:

- c_r^k = soma dos custos de cada arco contido na rota r pela qual trafega o produto k;
- x_r^k = variável de decisão que corresponde à quantidade de produto k que flui do nó s ao nó t, sendo s e t correspondentes aos nós de oferta e demanda, respectivamente;
- u_{ij} = capacidade do arco (i, j);
- b_{ij}^r = elemento da matriz de incidência arco-rota, de forma que:

$$b_{ij}^r = \begin{cases} 1, \text{ se o arco } (i,j) \text{ usa a rota } r \text{ pertencente à } \mathcal{R}(k); \\ 0, \text{ em caso contrário.} \end{cases}$$

- d^k = demanda do produto k, que deve trafegar do nó s para o nó t;

A formulação apresentada na expressão (2.3) é baseada na apresentada por (Mccallum, 1977).

2.4 Problema de Fluxo Multiproduto Binário

Como já mencionado, o problema tratado neste trabalho é o Problema de Fluxo Multiproduto Binário. Devido a isso, será apresentada, nesta Seção, uma formulação mais detalhada do mesmo, que também pode ser encontrado na literatura com o nome de Problema de Fluxo Multiproduto Não-Bifurcado.

A representação matemática deste problema é apresentada a seguir:

$$\min \quad \sum_{k=1}^{p} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k d^k x_{ij}^k \tag{2.4a}$$

$$\text{suj. a} \quad \sum_{(i,j)\in A} x_{ij}^k - \sum_{(j,i)\in\mathcal{A}} x_{ji}^k = b_i^k, \quad \forall\, i \in \mathcal{N},\ 1 \leqslant k \leqslant p \tag{2.4b}$$

$$\sum_{k\in\mathcal{P}} d^k x_{ij}^k \leqslant u_{ij}, \quad \forall (i,j) \in \mathcal{A} \tag{2.4c}$$

$$x_{ij}^k \in \{0,1\}, \quad \forall\, k \in \mathcal{P}, \quad \forall\, (i,j) \in \mathcal{A} \tag{2.4d}$$

Na expressão (2.4), tem-se que

- c_{ij}^k representa o custo unitário do produto k no arco (i, j);
- d^k representa a demanda do produto k, que deve fluir do nó origem até o nó destino, associados ao produto k;
- x_{ij}^k representa a variável de decisão, que é a utilização ou não do arco (i, j) pelo produto k. Caso este valor seja nulo, significa que o produto k não faz uso do arco (i, j), e no caso de $k = 1$, significa que o produto k faz uso do arco (i, j).
- a expressão (2.4b) representa as restrições de conservação de fluxo;

- a expressão (2.4c) corresponde ao conjunto de restrições de capacidade;
- u_{ij} representa a capacidade do arco (i, j);
- b_i^k indica se o nó i é oferta ou demanda para o produto k, sendo que b_i^k pode assumir os seguintes valores:

$$b_i^k = \begin{cases} 1, \text{ se o nó } i \text{ é oferta para o produto } k; \\ -1, \text{ se o nó } i \text{ é demanda para o produto } k; \\ 0, \text{neste caso, } i \text{ é um nó de transbordo.} \end{cases}$$

A fim de simplificar a implementação dos métodos propostos, foi feita uma alteração nessa formulação, de modo que as variáveis deixam de ser binárias para assumir os valores 0, no caso de não utilização do arco, ou d^k, no caso de utilização do arco em questão. Essa alteração, feita durante a implementação, não descaracteriza o problema binário abordado neste trabalho.

2.5 Problemas de Fluxo de Custo Mínimo Multiobjetivo

Problemas de Fluxo de Custo Mínimo constituem uma importante classe de problemas de fluxo em redes. O objetivo é encontrar o fluxo de custo mínimo de um ou mais produtos pela rede, atendendo às restrições de capacidade e de conservação de fluxo. Cada produto trafega em cada arco a um determinado custo. O problema multiobjetivo consiste em adicionar mais uma função, com valores diferentes dos custos para os produtos trafegarem pelos arcos. Tais custos podem representar, em problemas reais, custo de combustível, de segurança, de deteriorização de bens, caso seja estudado um problema logístico.

O problema de fluxo de custo mínimo multiobjetivo vem chamando a atenção de vários autores, embora não exista uma extensa literatura a seu respeito, quando comparado à existente relativa ao problema mono-objetivo.

Em Calvete e Mateo (1995) trata-se da obtenção de soluções ótimas para o problema multiobjetivo de fluxo em redes com prioridades preventivas. A abordagem proposta permite a manutenção da estrutura da rede do problema e, consequentemente, desenvolver algoritmos de rede baseados na metodologia que provaram ser mais eficientes que os algoritmos gerais. Além disso, foi proposta uma aplicação para o problema de avaliar a performance de um sistema hidrológico.

Mustafa e Goh (1998) descreve um método para ajustar de forma integral as soluções não-inteiras obtidas por meio do DINAS (*Dynamic Interactive Network Analysis System*), que é um pacote aplicado à resolução de problemas multiobjetivos. O trabalho enfatiza os problemas bi e tri-critérios de fluxo de custo mínimo e também descreve algumas características das soluções eficientes dos dois conjuntos de problemas.

Já em Noda e Martin (2000) é proposto um método que busca todas as soluções extremas do espaço objetivo para o problema multiobjetivo de fluxo de custo mínimo para apenas um produto que trafega pela rede. Em Noda e Martin (2001) é apresentado o problema bi-objetivo utilizando apenas um produto, com soluções

inteiras. É proposto um algoritmo que encontra todas as soluções eficientes dentro de um limite.

Em Hamacher et al. (2007) é feita uma revisão sobre problemas de fluxo de custo mínimo multiobjetivos, sendo também feita uma revisão de algoritmos exatos e aproximados aplicados ao problema. Foram estudados o problema contínuo e o problema inteiro.

Já Nikolova (2001) e Nikolova (2003) apresentam dois algoritmos iterativos para encontrar soluções inteiras do problema de fluxo de custo mínimo multiobjetivo.

2.5.1 Formulação Matemática

A formulação do problema é apresentada na expressão (2.5) e é baseada na formulação posta em (Calvete e Mateo, 1995).

O problema é modelado por um grafo direcionado $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, com n nós e a arcos. A formulação apresentada será *single-commodity* (um único produto) e, portanto, a variável de decisão x_{ij} representa a quantidade de produtos que fluem no arco (i, j). Em uma formulação *multicommodity* (vários produtos), a variável de decisão é dada por x_{ij}^k, que representa o fluxo do produto k que trafega pelo arco (i, j). A apresentação do modelo *single-commodity* é justificada, pois a grande maioria dos textos encontrados na literatura trazem essa abordagem. Seja, então, um problema de fluxo de custo mínimo com m objetivos.

$$\begin{aligned}
\min \quad & Z_1(x) = \sum_{(i,j)\in\mathcal{A}} c_{ij}^1 x_{ij} && \text{(2.5a)}\\
\min \quad & Z_2(x) = \sum_{(i,j)\in\mathcal{A}} c_{ij}^2 x_{ij} && \text{(2.5b)}\\
& \vdots \\
\min \quad & Z_m(x) = \sum_{(i,j)\in\mathcal{A}} c_{ij}^m x_{ij} && \text{(2.5c)}\\
\text{suj. a} \quad & \sum_{j\in\mathcal{N}} x_{ij} - \sum_{j\in\mathcal{N}} x_{ji} = b_i, \quad \forall i \in \mathcal{N}, \quad j \neq i && \text{(2.5d)}\\
& t_{ij} \leq x_{ij} \leq u_{ij}, \quad \forall (i,j) \in \mathcal{A} && \text{(2.5e)}\\
& x_{ij} \geq 0, \quad \forall (i,j) \in \mathcal{A} && \text{(2.5f)}
\end{aligned}$$

em que:

- x_{ij} corresponde ao fluxo no arco (i, j);
- u_{ij} representa a capacidade do arco (i, j);
- t_{ij} representa o limitante inferior do fluxo no arco (i, j). Em problemas de fluxo, esse valor normalmente é zero;
- b_{ij} representa a oferta/demanda do nó (i, j);
- $c_{ij}^1, c_{ij}^2, \ldots, c_{ij}^m$ representam os custos unitários para tráfego em cada função objetivo.

Quando $m = 2$, o problema é chamado de bi-critério e, obviamente, se $m = 3$, trata-se do problema tri-critério.

É possível observar que existe uma alteração dos custos em cada objetivo, incluindo, inclusive, a alteração do significado, ou seja, dependendo do problema, estes custos podem representar distância, tempo de entrega ou custo do transporte, por exemplo.

2.6 Problema de Minimização do Congestionamento em Rede

Esta Seção trata do Problema de Minimização do Congestionamento na rede de fluxo multiproduto. O objetivo deste segundo problema tratado nesta dissertação é distribuir o fluxo dos produtos pelos arcos da rede, de forma homogênea, sem sobrecarregar nenhum arco, penalizando por arco e minimizando a soma dos custos.

Este problema tem importantes aplicações na área de controle de fluxo de rede pela Internet. Como trabalhos relacionados nesse tema, podem ser citados (Buriol et al., 2003), (Buriol, 2003), em que o objetivo é o de minimizar o congestionamento de redes do tipo OSPF (*Open Shortest Path First*); (Ericsson et al., 2002), em que é utilizado algoritmo genético para atribuição de pesos em arcos em uma rede OSPF; além de (Fortz e Thorup, 2000a), (Fortz et al., 2002) e (Fortz e Thorup, 2000b), em que também foi estudado o problema de congestionamento mínimo, sendo que os testes realizados foram feitos utilizando dados reais do *backbone* da AT&T e também em redes sintéticas, criadas especificamente para testes.

2.6.1 Formulação Matemática

Considere, assim, uma rede $\mathcal{R}$, representada por um grafo direcionado $\mathcal{G}(\mathcal{N}, \mathcal{A})$, com n nós, a arcos e p produtos. Define-se, então, o conjunto dos nós como o conjunto $\mathcal{N}$, o conjunto dos arcos como $\mathcal{A}$ e o conjunto dos produtos como $\mathcal{P}$, conforme especificada na Seção 2.3.

Seja a variável l_{ij}. A quantidade de produtos que trafegam pelo arco (i, j), valor denominado como carga no arco (i, j), é dada por:

$$l_{ij} = \sum_{k \in P} x_{ij}^k \tag{2.6}$$

A taxa ou fator de utilização deste arco (i, j), dada por t_{ij}, é definida como:

$$\begin{aligned} t_{ij} &= \frac{l_{ij}}{u_{ij}} \\ &= \frac{\sum_{k \in P} x_{ij}^k}{u_{ij}} \end{aligned} \tag{2.7}$$

sendo o numerador correspondente ao fluxo total de produtos no arco (carga no arco) e o denominador correspondente à capacidade do arco (i, j).

Para medir o congestionamento total da rede, é utilizada a função proposta por (Fortz e Thorup, 2000a). Seja, então, a função de congestionamento em um arco (i, j), representada pela função $\Phi_{ij}(l_{ij})$. Esta função, convexa e linear por partes, é apresentada na Figura 2.2, para um arco com capacidade unitária.

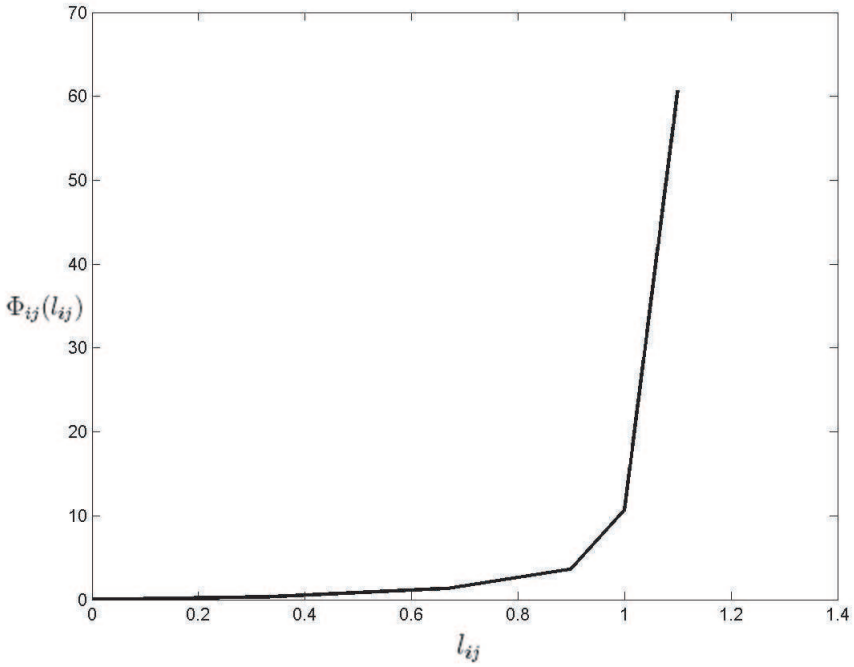

Figura 2.2: Função $\Phi_{ij}(l_{ij})$ de congestionamento de um arco (i, j) com capacidade unitária, conforme (Fortz et al., 2002).

A função de custo do congestionamento na rede é representada pela expressão:

$$\Phi = \sum_{(i,j)\in\mathcal{A}} \Phi_{ij}(l_{ij}) \tag{2.8}$$

Caso o fluxo do arco (i, j) seja nulo, então $l_{ij} = 0$ e, portanto, $\Phi_{ij}(l_{ij}) = 0$.

A função $\Phi_{ij}(l_{ij})$, aqui considerada, é uma adaptação da descrita em (Fortz e Thorup, 2000a), sendo definida, então, como:

$$\Phi_{ij}(l_{ij}) = \begin{cases} l_{ij}, & \text{se } 0 \leq l_{ij} < (1/3)u_{ij}; \\ 3l_{ij} - (2/3)u_{ij}, & \text{se } (1/3)u_{ij} \leq l_{ij} < (2/3)u_{ij}; \\ 10l_{ij} - (16/3)u_{ij}, & \text{se } (2/3)u_{ij} \leq l_{ij} < (9/10)u_{ij}; \\ 70l_{ij} - (178/3)u_{ij}, & \text{se } (9/10)u_{ij} \leq l_{ij} < u_{ij}; \\ 500l_{ij} - (1468/3)u_{ij}, & \text{se } u_{ij} \leq l_{ij} < (11/10)u_{ij}; \\ 5000l_{ij} - (16381/3)u_{ij}, & \text{se } l_{ij} \geq (11/10)u_{ij}. \end{cases} \tag{2.9}$$

Em (Fortz e Thorup, 2000a), estes valores foram obtidos experimentalmente. Estes parâmetros definem de forma direta a penalização que um arco pode sofrer. Como a função proposta é linear em intervalos do seu domínio, estes parâmetros são os coeficientes angulares da equação da reta que contém o segmento definido no intervalo dado.

A diferencial da função Φ_{ij} em relação a l_{ij} é apresentada abaixo:

$$\frac{\partial \Phi_{ij}(l_{ij})}{\partial l_{ij}} = \begin{cases} 1, \text{ se } 0 \leq l_{ij} < (1/3)u_{ij}; \\ 3, \text{ se } (1/3)u_{ij} \leq l_{ij} < (2/3)u_{ij}; \\ 10, \text{ se } (2/3)u_{ij} \leq l_{ij} < (9/10)u_{ij}; \\ 70, \text{ se } (9/10)u_{ij} \leq l_{ij} < 1.u_{ij}; \\ 500, \text{ se } 1.u_{ij} \leq l_{ij} < (11/10)u_{ij}; \\ 5000, \text{ se } l_{ij} \geq (11/10)u_{ij}. \end{cases} \tag{2.10}$$

A idéia por trás da função de custo do congestionamento $\Phi_{ij}(l_{ij})$ é penalizar cada arco de acordo com a quantidade de produtos carregada no mesmo, distribuindo, de forma mais homogênea, o fluxo de produtos pelos arcos da rede. O princípio adotado é que é mais barato fazer um produto trafegar por um arco menos violado que por um arco mais utilizado. Assim, busca-se evitar a sobrecarga de alguns arcos em relação a uma baixa utilização de outros.

O problema de congestionamento pode ser formulado como um problema de fluxo multiproduto, ou seja:

$$\min \quad \Phi = \sum_{(i,j)\in\mathcal{A}} \Phi_{ij}(l_{ij}) \tag{2.11a}$$

$$\text{suj. a} \quad \sum_{(i,j)\in\mathcal{A}} x_{ij}^k - \sum_{(j,i)\in\mathcal{A}} x_{ji}^k = b_i^k, \quad \forall i \in \mathcal{N}, \quad 1 \leqslant k \leqslant p \tag{2.11b}$$

$$l_{ij} = \sum_{k\in\mathcal{P}} x_{ij}^k, \quad \forall (i,j) \in \mathcal{A} \tag{2.11c}$$

$$\Phi_{ij}(l_{ij}) \geq l_{ij}, \quad (i,j) \in \mathcal{A} \tag{2.11d}$$

$$\Phi_{ij}(l_{ij}) \geq 3l_{ij} - (2/3)u_{ij}, \quad (i,j) \in \mathcal{A} \tag{2.11e}$$

$$\Phi_{ij}(l_{ij}) \geq 10l_{ij} - (16/3)u_{ij}, \quad (i,j) \in \mathcal{A} \tag{2.11f}$$

$$\Phi_{ij}(l_{ij}) \geq 70l_{ij} - (178/3)u_{ij}, \quad (i,j) \in \mathcal{A} \tag{2.11g}$$

$$\Phi_{ij}(l_{ij}) \geq 500l_{ij} - (1468/3)u_{ij}, \quad (i,j) \in \mathcal{A} \tag{2.11h}$$

$$\Phi_{ij}(l_{ij}) \geq 5000l_{ij} - (16318/3)u_{ij}, \quad (i,j) \in \mathcal{A} \tag{2.11i}$$

$$x_{ij}^k \geq 0, \quad \forall k \in \mathcal{P}, \quad \forall (i,j) \in \mathcal{A} \tag{2.11j}$$

A função objetivo, dada pela expressão (2.11a), minimiza o custo do congestionamento na rede, somando o custo de congestionamento de cada arco. A expressão (2.11c) define a carga em um arco (i,j) da rede. As restrições de conservação de fluxo, cuja função é gerenciar o fluxo de produtos na rede, são representadas pela expressão (2.11b). As restrições (2.11d) a (2.11i) são utilizadas para definir o custo de acordo com a carga no arco (i,j), caracterizando a forma clássica que transforma um modelo linear por partes em um modelo linear, segundo (Buriol, 2003).

Ainda segundo (Buriol, 2003), este problema é uma relaxação do roteamento OSPF, pois o fluxo que passa por cada nó i não é dividido igualmente entre os caminhos mínimos partindo de i.

2.6.2 Generalização do Modelo Matemático

Nesta Seção será apresentada uma formulação genérica do Problema de Congestionamento em Redes.

Seja então a mesma rede definida na Seção 2.3, com a arcos, n nós e p produtos. Seja também o modelo com a função $\Phi_{ij}(l_{ij})$ linear por partes proposta por (Fortz e Thorup, 2000a) e utilizada também por (Buriol, 2003), correspondente à função de congestionamento em cada arco da rede.

Primeiramente, é preciso definir os intervalos, para, em seguida, definir as regras para a função Φ em cada arco.

Sejam m intervalos, definidos para cada arco da rede e dados de acordo com a capacidade de cada arco:

$$\text{Intervalos} = \begin{cases} [0, w_1 u_{ij}), \text{ primeiro intervalo;} \\ [w_1 u_{ij}, w_2 u_{ij}), \text{ segundo intervalo;} \\ \quad \vdots \qquad\qquad \vdots \\ [w_{m-3} u_{ij}, u_{ij}), \text{ antepenúltimo intervalo, com a taxa de} \\ \quad \text{utilização próxima de 1, sendo } u_{ij} = w_{m-2}; \\ \quad \text{sendo este intervalo o mais próximo da capacidade e factível;} \\ \\ [u_{ij}, w_{m-1} u_{ij}), \text{ penúltimo intervalo, de forma que } w_{m-1} > 1; \\ [w_{m-1} u_{ij}, \infty), \text{ m-ésimo intervalo, mais penalizado;} \end{cases}$$

Os fatores w_r, $1 \leq r \leq m$, consistem em fatores multiplicativos que limitam, inferior ou superiormente, os intervalos. A definição dos intervalos pode também ser feita por meio de uma penalização única no caso de infactibilidade. Os dois últimos intervalos mostrados caracterizam infactibilidade quanto às restrições de capacidade e, neste trabalho, o penúltimo intervalo implicará em uma menor penalização na solução, quando comparado ao último intervalo.

Na Figura 2.2, a diferencial definida para o primeiro intervalo é igual a 1, o que significa que, neste primeiro intervalo, o arco não é penalizado e a função $\Phi_{ij}(l_{ij})$ pode ser definida partindo-se da definição dos intervalos e das demais diferenciais. Considerando a diferencial escolhida e também o intervalo dado, a regra da função neste intervalo é:

$$\Phi_{ij}(l_{ij}) = t_{ij} = l_{ij} \tag{2.12}$$

pois $u_{ij} = 1$.

Ainda considerando a função representada pela Figura 2.2, o segundo intervalo definido foi $[(1/3)u_{ij}, (2/3)u_{ij})$. Considerando que a diferencial definida para este intervalo é igual a 3, é possível obter a função apresentada na equação (2.13), obtendo o termo independente igual a $-2/3$, sendo que o ponto $(1/3, 1/3)$ pertence a este segmento de reta.

$$\Phi_{ij}(l_{ij}) = 3t_{ij} - 16/3 = 3l_{ij} - 16/3, \text{pois } u_{ij} = 1 \tag{2.13}$$

As demais regras da função $\Phi_{ij}(l_{ij})$ são definidas de maneira análoga, dependendo do intervalo e da taxa de crescimento definida no intervalo.

Após definir os intervalos, é preciso definir as diferenciais da função em cada intervalo escolhido. Esta definição é importante pelo fato das derivadas serem diretamente responsáveis pela penalização nos arcos. Assim, pode-se generalizar como:

$$\Phi' = \begin{cases} \delta_1, \text{ se } 0 \leq l_{ij} < w_1 u_{ij}; \\ \delta_2, \text{ se } w_1 u_{ij}; \leq l_{ij} < w_2 u_{ij}; \\ \vdots \quad , \quad \vdots \quad ; \\ \delta_{m-2}, \text{ se } w_{m-3} u_{ij} \leq l_{ij} < u_{ij}; \\ \delta_{m-1}, \text{ se } u_{ij} \leq l_{ij} < w_{m-1} u_{ij}; \\ \delta_m, \text{ se } l_{ij} > w_{m-1} u_{ij}. \end{cases}$$

para $\delta_r, 1 \leq r \leq m$ como as as diferenciais definidas.

Em seguida, a função de congestionamento em um arco (i, j), dada por $\Phi_{ij}(l_{ij})$, é definida como:

$$\Phi_{ij}(l_{ij}) = \begin{cases} \delta_1 l_{ij} + \lambda_1 u_{ij}, \text{ se } 0 \leq l_{ij} < w_1 u_{ij}; \\ \delta_2 l_{ij} + \lambda_2 u_{ij}, \text{ se } w_1 u_{ij} \leq l_{ij} < w_2 u_{ij}; \\ \vdots \quad , \quad \vdots \quad ; \\ \delta_{m-2} l_{ij} + \lambda_{m-2} u_{ij}, \text{ se } w_{m-3} u_{ij} \leq l_{ij} < u_{ij}; \\ \delta_{m-1} l_{ij} + \lambda_{m-1} u_{ij}, \text{ se } u_{ij} \leq l_{ij} < w_{m-1} u_{ij}; \\ \delta_m l_{ij} + \lambda_m u_{ij}, \text{ se } l_{ij} > w_{m-1} u_{ij}. \end{cases}$$

O cálculo, então, é realizado de acordo com a carga l_{ij} no arco (i, j), sendo $\lambda_r, \quad 1 \leq r \leq m$ um parâmetro, utilizado para encontrar os termos independentes na função.

A função Φ, de uma solução qualquer, corresponde à soma dos congestionamentos em cada arco.

As premissas existentes neste modelo são, então, os intervalos e as diferenciais e devem ser passadas como parâmetro para o cálculo do valor funcional de cada solução do problema.

2.7 Conclusão

Neste capítulo foram apresentadas as duas formulações básicas de um PFM, quais sejam, a formulação nó-arco e a formulação arco-rota. Foi feita uma revisão bibliográfica, contendo trabalhos relacionados encontrados na literatura. A formulação tratada neste trabalho consiste na formulação nó-arco, em que os nós da rede representam pontos de oferta e demanda para os diversos produtos que competem pela capacidade de cada arco, sendo que, para cada produto, existe um custo associado para trafegar em cada arco. O objetivo do problema é determinar o fluxo de custo mínimo dos diversos produtos, atendendo às restrições de capacidade e conservação de fluxo. Foi também apresentado um exemplo, considerando a formulação nó-arco, a fim de ilustrar a formulação matemática apresentada. Em seguida, foi apresentada a formulação matemática da classe dos Problemas Binários, considerada neste trabalho.

Por fim, o PFM foi dividido em três classes, a classe dos Problemas Lineares, a classe dos Problemas Inteiros e a classe dos Problemas Binários. Neste trabalho, será tratada a classe dos Problemas Binários, o que adiciona as restrições de integralidade e não-negatividade à formulação básica de um PFM.

Para finalizar o capítulo, foi apresentado o problema de congestionamento mínimo de redes. A função apresentada é baseada na proposta por Fortz e Thorup (2000a) e utilizada também por Buriol (2003). Foi incluída nesta Seção uma introdução ao problema, discutindo o objetivo da função proposta. Em seguida, foi apresentada uma função utilizando os valores de diferenciais e intervalos já mencionados na literatura sobre o assunto e, finalmente, foi apresentada uma formulação genérica do problema de congestionamento mínimo, apresentado uma discussão sobre as premissas do modelo.

Capítulo 3

Revisão de Otimização Multiobjetivo

3.1 Introdução

Uma grande quantidade de problemas de decisão encontrados no mundo real são de natureza multiobjetivo. A simples decisão de compra de um computador está associada à busca pelo máximo desempenho e, obviamente, ao menor custo. Porém, um computador que possui as melhores características de desempenho normalmente possui o maior valor; em contrapartida, geralmente o computador com menor custo financeiro para o comprador apresenta as configurações de menor desempenho. Este é um exemplo de uma situação do cotidiano na qual qualquer pessoa se depara com um problema de decisão tendo claro conflito entre os objetivos que se busca satisfazer. Neste caso, o tomador de decisões (ou decisor), no caso o comprador, precisa ponderar qual é a sua opção entre preço e/ou desempenho. Este problema de tomada de decisão é uma formulação típica de Otimização Multiobjetivo, que também é conhecida como Otimização Vetorial.

Uma solução denominada de ***utópica*** deste problema é um computador com o mais alto desempenho e o mais baixo custo, o que simplesmente não existe no mundo real.

Porém, pode-se dizer que existem soluções que são melhores que outras, soluções cujo desempenho é maior ou igual por um custo menor ou igual quando comparadas a outras. Tais soluções, que superam outras sob aspectos de decisão considerados pelo decisor, são as chamadas soluções ***não-dominadas***; já as soluções que são superadas por outras são as chamadas ***soluções dominadas***.

Deste modo, o principal objetivo da Otimização Multiobjetivo é encontrar o conjunto de soluções não-dominadas, conhecido também como ***Conjunto Ótimo de Pareto*** ou simplesmente ***Soluções Ótimas de Pareto***. Estas soluções podem ser usadas por um tomador de decisões para propiciar uma escolha da solução que seja a mais adequada para o problema tratado.

Neste capítulo, conceitos de Otimização Multiobjetivo são revisados, à luz dos interesses aqui postos para o desenvolvimento de soluções dos problemas em estudo. Não se trata, portanto, de uma apresentação exaustiva deste amplo campo de pesquisa, mas, sim, de introdução de formulações que possam auxiliar no entendimento do que se será tratado nos capítulos seguintes.

3.2 Formulação Multiobjetivo

De acordo com (Castro, 2001), um problema multiobjetivo pode ser descrito com um vetor y, com n objetivos, que dependem de um vetor x de m variáveis independentes. A formulação de um problema multiobjetivo é apresentada na expressão (3.1), de acordo com (Castro, 2001).

$$\begin{aligned} \min / \max \quad & y = (f_1(x),\ f_2(x),\ \ldots,\ f_n(x)) && \text{(3.1a)} \\ \text{suj. a} \quad & x = (x_1,\ x_2,\ \ldots,\ x_m) \in \mathcal{X} && \text{(3.1b)} \\ & y = (y_1,\ y_2,\ \ldots,\ y_n) \in \mathcal{Y} && \text{(3.1c)} \end{aligned}$$

para:

- y: vetor de funções objetivo;
- $f_i(\cdot)$, $1 \leq i \leq n$: i-ésima função objetivo;
- $\mathcal{Y}$: espaço das funções objetivo;
- x: vetor de variáveis de decisão;
- x_i, $1 \leq i \leq m$: i-ésima variável de decisão;
- $\mathcal{X}$: espaço das variáveis de decisão.

3.3 Definições

A otimização multiobjetivo, nos termos postos pelo modelo matemático representado na expressão (3.1), busca encontrar todas as soluções que melhorem o valor de pelo menos uma função objetivo, sem acarretar a piora dos valores das demais, ou seja, busca determinar o Conjunto Ótimo de Pareto. Nesta Seção, são apresentados conceitos importantes da otimização multiobjetivo, como a definição de dominância, a qual leva a identificar tais soluções.

Definição 1 ((Collette e Siarry, 2003)) *Dominância*
Dizemos que um vetor $A = (a_1,\ \ldots,\ a_m)$ domina outro vetor $B = (b_1,\ \ldots,\ b_m)$ se:

- *o vetor A é melhor ou igual a B em todos os objetivos; e*
- *o vetor A é estritamente melhor que o vetor B em pelo menos um objetivo.*

Em termos matemáticos, esta formulação pode ser posta da seguinte forma: um vetor $A = (a_1,\ \ldots,\ a_m)$ domina outro vetor $B = (b_1,\ \ldots,\ b_m)$ se:

$$f_i(A) \leq f_i(B)\ \ \forall i \in [1, n] \quad \text{e} \quad \exists j\ :\ f_j(A) < f_j(B)\ ,\ \ j \in [1, n] \tag{3.2}$$

Definição 2 ((Collette e Siarry, 2003)) *Solução Não-dominada*
São soluções que dominam outras soluções, mas não são dominadas por nenhuma outra.

Matematicamente, uma solução é dita não-dominada se:

$$\nexists B \in \mathcal{X} \ : \ f_i(B) \leq f_i(A) \quad \text{e} \quad \exists j \ : \ f_j(B) < f_j(A) \ ; \quad 1 \leq i, \ \ j \leq n. \tag{3.3}$$

Pela expressão (3.3), conclui-se que uma solução A é não-dominada se não existe nenhuma solução que é melhor que A em pelo menos um objetivo sem que seja pior que em outro. Uma solução não-dominada é também chamada de solução ótima no sentido de Pareto ou, simplemente, solução ótima de Pareto.

O Conjunto Ótimo de Pareto é, portanto, formado pelas soluções não-dominadas de todo o espaço de busca. A Figura 3.1 mostra as soluções Y_A a Y_F. Para verificar

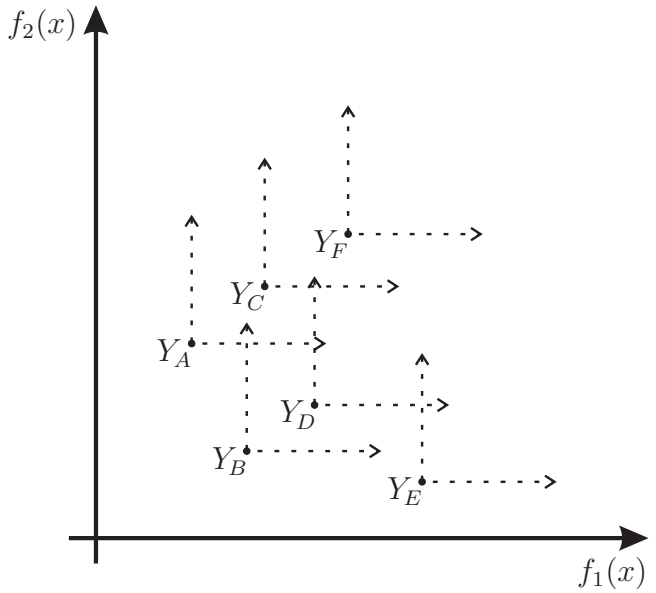

Figura 3.1: Relações de dominância entre soluções.

graficamente o conceito de não-dominância, considere o seguinte procedimento. Em cada um dos pontos, são colocados cones paralelos aos eixos coordenados, tendo estes pontos como vértices. Os pontos interiores a cada cone são dominados pelo ponto localizado no vértice. Assim, a partir das representações mostradas na Figura 3.1, pode-se concluir que:

- Não existe relação de dominância entre as soluções Y_A, Y_B e Y_E;
- Y_A domina Y_C e Y_F;
- Y_B domina Y_C, Y_D e Y_F;
- Não existe relação de dominância entre as soluções Y_C, Y_D e Y_E;
- Y_C e Y_D dominam Y_F;
- Y_E e Y_F não dominam nenhuma outra solução.

Associada à definição de solução não-dominada, a definição abaixo é fundamental:

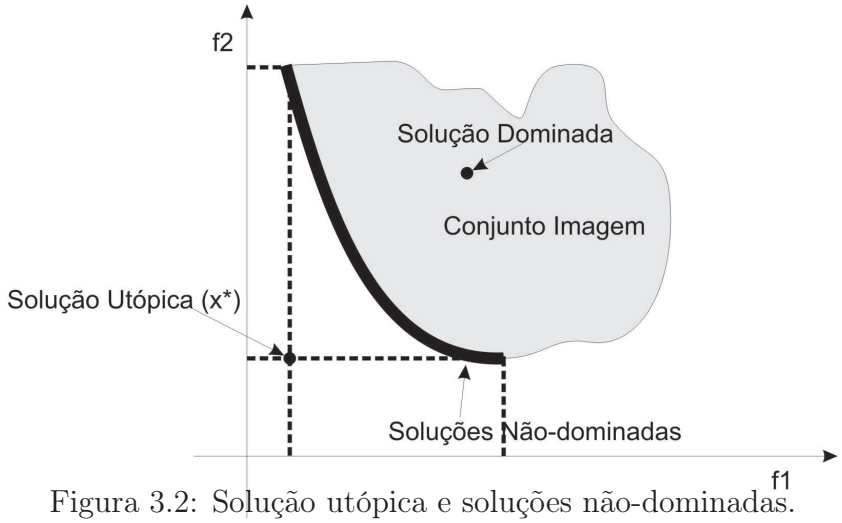

Figura 3.2: Solução utópica e soluções não-dominadas.

Definição 3 ((Miettenen, 1999)) *Solução Utópica*
Diz-se que uma solução x^* é utópica se x^* é a solução ótima de todas as funções objetivo.

É importante ter claro, no entanto, que esta solução, geralmente, não existe. Caso exista, é a própria solução do problema formulado na expressão (3.1). No entanto, a existência de conflitos entre os objetivos impede que esta solução seja alcançada. Este fato motiva o uso de técnicas de solução próprias para problemas de Otimização Multiobjetivo (ou Otimização Vetorial) que se diferenciam das utilizadas para a solução de problemas de Otimização Mono-objetivo (ou Otimização escalar).

A Figura 3.2 apresenta a solução utópica, bem como a fronteira de soluções não-dominadas de Pareto, considerando duas funções objetivo, ou seja, $n = 2$. Nesta figura, é possível observar que a solução utópica, ou seja, a solução que minimiza ambas as funções, não pertence ao conjunto imagem, ou seja, não atende a todas as restrições do problema.

Exemplo 3.1 *Como exemplo simples das questões associadas à Otimização Multi-objetivo, sejam duas funções quadráticas $f_1(x)$ e $f_2(x)$, definidas como na expressão (3.4):*

$$F = (f_1(x), f_2(x)) \tag{3.4a}$$
$$f_1(x) = x^2 \tag{3.4b}$$
$$f_2(x) = (x-3)^2 \tag{3.4c}$$

Este exemplo é baseado no apresentado por (Schaffer, 1984), em que também são apresentadas duas funções quadráticas.

O gráfico apresentado na Figura 3.3 exibe as duas funções. A parte do gráfico em negrito contém as soluções não-dominadas, sendo que as soluções eficientes estão no intervalo $[0, 3]$.

A Figura 3.4 apresenta o gráfico de $f_1(x)$ e $f_2(x)$, tendo a parte em negrito mostrando o Conjunto Ótimo de Pareto, nos pontos onde $0 \leq f_1 \leq f_2$.

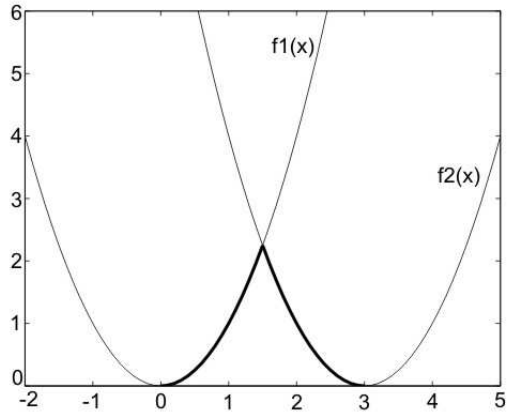

Figura 3.3: Gráfico das funções f_1 e f_2.

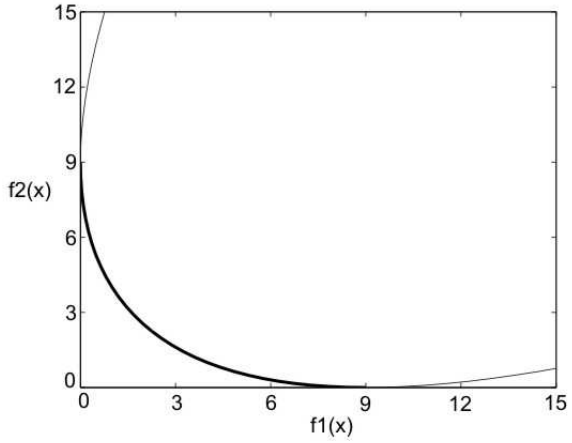

Figura 3.4: Fronteira de Pareto, funções f_1 e f_2.

3.4 Métodos de Resolução

Nesta Seção, serão discutidos métodos de geração de soluções eficientes para um problema de otimização vetorial. Dentre as técnicas de Otimização Multiobjetivo mais utilizadas, estão o Método da Soma Ponderada, o Método ϵ-Restrito, brevemente descritos a seguir.

3.4.1 Método da Soma Ponderada

Este método consiste em transformar as funções em uma única função, transformando o problema multiobjetivo em um problema mono-objetivo. Assim, podem ser usadas técnicas de Otimização Escalar na resolução do problema. É considerado um método clássico de otimização multiobjetivo e é muito utilizado.

O problema de otimização multiobjetivo pode ser escrito na seguinte forma:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \lambda_i f_i(x) \\ \text{suj. a} \quad & x \in \mathcal{X} \end{aligned} \tag{3.5}$$

Assim, se x^* é solução do problema representado pela expressão (3.5), então x^*

é uma solução eficiente para o Problema de Otimização Multiobjetivo se:

- x^* é a solução única do Problema ou,
- $\lambda_i > 0,\ i = 1,\ \ldots,\ m.$

Obviamente, pode ser considerada a situação em que a ponderação λ seja tal que $\lambda \geq 0$ e $\sum_{i=1}^{n} \lambda_i = 1$, considerando n funções objetivo.

Este método é computacionalmente simples, mantém o número de restrições do problema original e mantém características das funções originais, tais como convexidade e diferenciabilidade. Porém, este método não se aplica a problemas não-convexos. No caso de problemas convexos, é possível gerar todo o conjunto de soluções eficientes, de acordo com (Gonçalves, 2007).

3.4.2 Método ϵ-Restrito

Este método baseia-se no Lema 3.1 a seguir.

Lema 3.1 *Se $x^* \in \mathcal{X}$ é eficiente, então existem $i \in \mathcal{Z}_+ \ : \ i = 1,\ \ldots,\ n$ e números reais $\epsilon_j,\ j = 1,\ \ldots,\ n$, tais que x^* resolve:*

$$\begin{array}{rcll} x^* & = & \arg \min & f_i(x) \\ & & \text{suj. a} & f_i(x) \leq \epsilon_j, \quad j = 1,\ \ldots,\ n \quad (i \neq j) \end{array} \tag{3.6}$$

A idéia por trás deste método é otimizar uma função e transformar as demais funções em restrições, escolhendo-se adequadamente cada ϵ_j como limitante de cada restrição tratada. Esta escolha dos limitantes geralmente não é uma tarefa trivial.

Com essa abordagem, é possível gerar completamente o conjunto de soluções eficientes, mesmo que o problema não seja convexo. No entanto, esta tarefa não é computacionalmente simples para problemas não-lineares.

Como desvantagens deste método, pode-se destacar, além de não ser simples computacionalmente para problemas não-lineares, o aumento no número de restrições do problema. De acordo com o número de objetivos tratados, a quantidade de restrições pode aumentar muito. Por fim, normalmente é difícil encontrar os parâmetros ϵ_j, pois, dependendo das escolhas, o algoritmo poderá retornar soluções infactíveis.

Existem outras abordagens a problemas multiobjetivos, porém não serão tratadas neste trabalho. Por exemplo, podem ser citadas a Abordagem Híbrida, que utiliza os dois métodos descritos anteriormente, e a Programação-alvo, que consiste na minimização de uma norma vetorial p em relação a um vetor referência (alvo).

3.5 Conclusão

Neste capítulo foi apresentada uma formulação de problemas de otimização multiobjetivo, bem como definições importantes para a leitura dos capítulos seguintes deste trabalho, como os conceitos de dominância e de soluções ótimas de Pareto. Foi apresentado um exemplo, com o objetivo de otimizar, simultaneamente, duas funções quadráticas e foi construído o conjunto de soluções ótimas de Pareto. Finalmente, foram apresentados alguns métodos de resolução de problemas multiobjetivos.

Capítulo 4

Métodos Heurísticos

4.1 Introdução

Um método heurístico pode ser definido como um procedimento desenvolvido a partir de regras observadas pelos desenvolvedores.

Segundo (Ziviane, 2004), *"uma heurística é um algoritmo que pode produzir um bom resultado, ou até mesmo obter a solução ótima, mas pode também não produzir solução alguma ou uma solução que está distante da solução ótima"*.

Dessa forma, métodos heurísticos não garantem soluções ótimas, porém, podem chegar a soluções viáveis, e até mesmo a boas soluções, em tempo computacional menor quando comparado a métodos exatos.

As heurísticas podem ser classificadas em duas classes, segundo a forma de construção de uma solução:

- **Construtivas**: constroem passo a passo uma solução, que normalmente é usada como solução inicial para outro procedimento heurístico. As heurísticas construtivas podem ser aleatórias, quando os passos para a construção da solução são totalmente aleatórios, ou gulosas, quando a construção da solução segue alguma regra. Como exemplo de uma construção aleatória, pode ser citado uma construção de uma solução para o Problema do Caixeiro Viajante (PCV), em que a escolha de um novo cliente é feita de forma totalmente aleatória; como exemplo de uma construção gulosa, basta considerar que a escolha de um novo cliente no PCV é feita segundo o cliente que possuir a menor distância da cidade atual. Muitas vezes as soluções geradas por Heurísticas Construtivas são infactíveis e necessitam de um método de refinamento;

- **Refinamento**: são utilizadas para melhorar uma solução inicial, que pode ser obtida por uma Heurística Construtiva (HC). A melhoria se dá por meio de modificações na solução, gerando vizinhos da solução atual.

4.2 Heurísticas de Busca Local

O principal objetivo de métodos de busca local é encontrar soluções melhores que a solução inicial construída previamente por uma Heurística Construtiva. A idéia é explorar as soluções denominadas vizinhas, em busca de uma solução melhor.

Um método de busca local bastante utilizado é o Método da Descida Randômica, que é uma variação do método clássico da Descida. Este método se destaca tendo em vista que o tempo computacional da Descida Randômica é menor que o método clássico de descida.

4.2.1 Método da Descida

Antes de descrever o Método da Descida, são necessárias algumas considerações iniciais. Seja $f(\cdot)$ uma função a ser otimizada e $\mathcal{X}$ o espaço de busca associado, ou seja, o domínio da função. Seja x uma solução, tal que $x \in \mathcal{X}$. Denomina-se $\mathcal{N}(\cdot)$ como uma função que caracterize uma vizinhança de x, de acordo com um movimento m sobre a solução x. Obviamente, todas os vizinhos gerados a partir do movimento m devem ser tais que pertençam a $\mathcal{X}$. Denota-se a vizinhança como $\mathcal{N}(\mathcal{X})$.

Cada solução $x' \in \mathcal{N}(x)$ é chamada de vizinho de x. Um movimento é uma ação que modifica x e o transforma em x'. Por exemplo, se a solução é um vetor binário, um movimento m pode ser definido como trocar um *bit* em uma posição do vetor; a vizinhança são as possíveis soluções geradas por este movimento em x.

O método da Descida consiste basicamente de três passos:

(i) encontrar uma solução inicial x por meio de uma heurística construtiva, aleatória ou gulosa;

(ii) encontrar $x' \in \mathcal{X}$, de forma que $f(x') < f(x)$; i.e., encontrar o melhor vizinho de x e substituir a solução atual pela melhor solução encontrada;

(iii) repetir (i) e (ii) até que nenhuma solução vizinha melhore a solução atual.

A Figura 4.1 apresenta o pseudo-código do método clássico da Descida.

```
1 Procedimento Descida(f(.), N(.), x); V = x' ∈ N(x)|f(x') < f(x);
2 enquanto (|V| > 0) faça
3     Selecione x' ∈ V, onde x' = arg min.{f(x')|x' ∈ V}; x ← x';
      V = x' ∈ N(x)|f(x') < f(x);
4 fim
5 Retorne x;
```

Figura 4.1: Pseudo-código do método clássico da descida (Souza, 2007).

O Método Clássico da Descida, descrito nesta Seção, explora toda a vizinhança de uma solução, de modo tal que é realizada uma busca exaustiva. Pode-se observar ainda que este método pode levar à estagnação em um ótimo local da função objeto de otimização. A Figura 4.2 mostra uma interpretação geométrica deste problema, para o caso de função contínua.

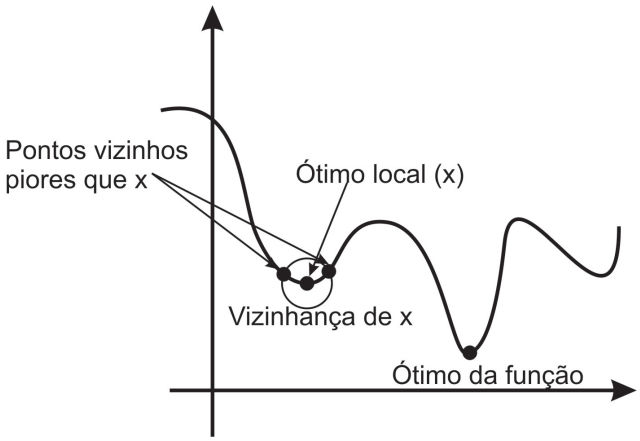

Figura 4.2: Descida - ótimo local.

```
1  Procedimento RandomicoDescida(f(.), N(.), itermax, x);
2  iter = 0; {contador de iterações sem melhora}
3  enquanto (iter < itermax) faça
4      iter = iter + 1;
5      Selecione aleatoriamente x' ∈ N(x);
6      se (f(x') < f(x)) então
7          iter = 0;
8          x ← x';
9      fim
10 fim
11 Retorne x;
```

Figura 4.3: Pseudo-código do método da descida randômica, (Souza, 2007).

4.2.2 Método da Descida Randômica

Em busca de um menor tempo computacional, é interessante avaliar o comportamento de uma variação do método clássico da Descida, descrito anteriormente.

O método escolhido foi o da Descida Randômica, que evita a busca exaustiva pelo melhor vizinho. A idéia é aceitar um vizinho somente se o mesmo for de melhora. Se o vizinho não melhorar a solução atual, então é gerado um novo vizinho. O processo é interrompido quando se atinge um número máximo de iterações sem melhora, definido pelo parâmetro (*itermax*). De acordo com (Souza, 2007), a solução final não é necessariamente um ótimo local, fato que pode ser percebido intuitivamente, já que existe uma grande chance de não se verificar toda a vizinhança. O pseudo-código é apresentado na Figura 4.3, de acordo com (Souza, 2007).

4.3 Metaheurísticas

Um dos principais problemas de heurísticas simples, como o Método da Descida, é que o Algoritmo pode retornar a uma solução ótima local em relação a uma vizinhança.

As metaheurísticas são procedimentos heurísticos que possuem estratégias de fuga de ótimos locais. Assim, as chances de obtenção de melhores soluções são maiores, quando comparadas a heurísticas simples. Tais métodos exploram melhor o espaço de busca, buscando soluções em outras regiões.

Becceneri (2007) definiu uma metaheurística como "*uma ferramenta algorítmica geral, que pode ser aplicada a diferentes problemas de otimização, com modificações relativamente pequenas, para torná-las adaptáveis a um problema específico*".

Ambas as definições apresentadas se encaixam perfeitamente na proposta deste trabalho.

A metaheurística utilizada neste trabalho é a metaheurística *Iterated Local Search* (ILS) e será descrita na Seção 4.3.1. A justificativa pela escolha desta metaheurística está em sua simplicidade computacional e sua aplicabilidade a diversos problemas de Otimização, em particular ao Problema de Fluxo Multiproduto Inteiro Mino-objetivo e Multiobjetivo.

4.3.1 Método *Iterated Local Search* (ILS)

Proposto inicialmente por Lourenço et al. (2002), o método *Iterated Local Search* (ILS) é classificado como uma metaheurística, pois apresenta uma estratégia de fuga de ótimos locais, através da idéia de melhorar um processo de busca local. No método ILS, aplica-se perturbações na solução ótima local e, em seguida, aplica-se um método de busca local a esta solução. Tal perturbação não pode ser muito fraca, pois neste caso a solução poderá não sair do ótimo local encontrado, tampouco muito forte, para evitar um reinício aleatório.

Segundo (Lourenço et al., 2002), o método possui quatro componentes:

1. Geração de uma solução inicial;
2. Busca Local, utilizada para retornar uma solução melhor;
3. Perturbação, utilizada para alterar a solução encontrada pela busca local (solução corrente);
4. Critério de aceitação, em que se decide em qual solução a próxima perturbação será aplicada.

A Figura 4.4 apresenta o pseudo-código do método ILS.

Esta dissertação apresenta, na Seção 5.4, uma implementação do método ILS para a solução do Problema de Fluxo Multiproduto. É utilizado o Método da Descida Randômica como método de busca local associado ao ILS.

4.4 Computação Evolutiva

A idéia básica da Computação Evolutiva, surgida nos anos 50, é aplicar o processo de evolução natural como um paradigma de solução de problemas. Segundo Von Zuben (2000), três algoritmos para computação evolutiva foram desenvolvidos independentemente:

- Algoritmos Genéticos;

```
1  Procedimento ILS;
2  x_0 ← SolucaoIncial;
3  x ← BuscaLocal(x_0);
4  iter ← 0;
5  enquanto (iter < itermax) faça
6      iter = iter + 1;
7      x' ← perturbacao(x, historico);
8      x'' ← BuscaLocal(x');
9      x ← CriterioAceitacao(x, x', x'');
10 fim
11 Retorne x.
```

Figura 4.4: Pseudo-código do método ILS.

- Programação Evolutiva;
- Estratégias Evolutivas.

Ainda segundo Von Zuben (2000), *"a vantagem mais significativa da computação evolutiva está na possibilidade de resolver problemas pela simples descrição matemática do que se quer ver presente na solução, não havendo necessidade de se indicar explicitamente os passos até o resultado, que certamente seriam específicos para cada caso."*

A idéia básica de um procedimento evolutivo é iniciar por uma população de soluções iniciais e, a partir desta, buscar melhores soluções através de um processo semelhante ao da evolução. Dessa forma, a teoria da computação evolutiva possui uma família de algoritmos inspirados na Teoria da Evolução de Darwin, em que os mais adaptados sobrevivem de uma geração para outra.

A estrutura básica de um algoritmo evolutivo é apresentada na Figura 4.5, de acordo com Mccallum (1977).

```
1  Procedimento AlgoritmoEvolutivo;
2  t ← 0;
3  Inicialize P(t);{População Inicial} Avalie P(t);
4  enquanto CritriodeParada faça
5      t ← (t + 1);
6      Selecione P(t)(de)P(t − 1);
7      Altere P(t);
8      Avalie P(t);
9  fim
10 Fim AlgoritmoEvolutivo;
```

Figura 4.5: Estrutura básica de um algoritmo evolutivo (Mccallum, 1977).

Assim, segundo o algoritmo apresentado na Figura 4.5, um algoritmo evolutivo precisa receber uma população de soluções e cada solução precisa ser avaliada por

```
1 Procedimento ACO;
2 enquanto (CriterioDeParada) faça
3     S = s_1, ..., s_n; Constrói soluções utilizando o rastro de feromônio
      S' ← BuscaLocal(S);
4     Atualiza feromônio;
5 fim
6 Fim ACO;
```

Figura 4.6: Pseudo-código do método colônia de formigas (Dorigo e Stützle, 2002).

uma função que mede sua capacidade de adaptação para a próxima geração. Esta função é denominada função *fitness*, função de aptidão ou, simplesmente, função de avaliação.

Como exemplos de algoritmos evolutivos, podem ser citados o Algoritmo Genético (AG), o Otimização por Colônia de Formigas (*Ant Colony Optimization* - ACO), o Evolução Diferencial (*Differential Evolution* - DE), o Otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO), dentre outros. Estes métodos serão brevemente apresentados discutidos nas próximas seções. Será apresentado, mais detalhadamente, o AG, método utilizado neste trabalho tanto para a solução do Problema de Fluxo Multiproduto como para o Problema Multiobjetivo de Fluxo Multiproduto.

4.4.1 Otimização por Colônia de Formigas - *Ant Colony Optimization* (ACO)

Proposto por (Dorigo et al., 1996), este método é baseado no comportamento real das formigas. A idéia é que as formigas, enquanto caminham em busca de alimentos, depositam no solo uma substância denominada feromônio, criando uma trilha desta substância, que pode ser seguida por outras formigas. Dessa forma, uma formiga qualquer terá maior probabilidade de seguir um caminho com maior concentração de feromônio.

Como a escolha do caminho é probabilística, com alta probabilidade de se escolher o caminho com maior quantidade de feromônio, o algoritmo possui chances de fugir de ótimos locais.

De acordo com (Dorigo et al., 1996), o ACO é:

- versátil, pois pode ser aplicado em variações de um mesmo tipo de problema, como em variações do Problema do Caixeiro Viajante;
- robusto, pois pode ser aplicado a outros problemas de otimização, sofrendo apenas pequenas mudanças;
- populacional, o que faz o método explorar melhor o espaço de busca.

O pseudo-código desse método é apresentado na Figura 4.6 e pode ser encontrado em (Dorigo e Stützle, 2002).

Uma descrição mais detalhada do método e de suas diversas variantes pode ser encontrada em (Dorigo, 1992), (Dorigo et al., 1996), (Dorigo e Stützle, 2002) e (Dorigo e Gambardella, 1997). Em (Dorigo e Stützle, 2002), é encontrado um detalhamento dos procedimentos apresentados pela Figura 4.6.

4.4.2 Evolução Diferencial - *Differential Evolution* (DE)

Proposto por Storn e Price (1995), este método surgiu com a tentativa de solucionar o ajuste polinomial de Chebychev. Um exemplo de aplicação deste método pode ser encontrado em (Arantes et al., 2006), em que é apresentado o método e o mesmo é aplicado a dois problemas de engenharia, sendo que o primeiro visa minimizar o custo da produção de cimento *Portland*, considerando o co-processamento e a adição de mineralizadores, e o segundo consiste no despacho econômico de energia elétrica. Em (Lobato et al., 2007) o método é aplicado para determinar instantes em que ocorrem ativação e desativação de restrições de desigualdade e dos coeficientes da aproximação polinomial proposta para o controle, já que é tratado o Problema de Controle Ótimo.

De acordo com (Arantes et al., 2006), a inicialização do algoritmo se dá por meio da geração de uma população inicial aleatória. Em seguida, é selecionado aleatoriamente um indivíduo para ser substituído. São selecionados três pais e um destes é escolhido como genitor principal, enquanto os demais são os genitores secundários. Em seguida, modifica-se, com dada probabilidade, cada variável do genitor principal. Ao valor atual da variável, adiciona-se a diferença entre as outras duas variáveis (genitores secundários), ponderada por uma taxa de perturbação $\boldsymbol{F}$. Este último procedimento caracteriza o operador de mutação presente no método *Differential Evolution*.

Como este método não será aplicado ao problema proposto neste trabalho, não será dada uma descrição mais detalhada sobre o mesmo, que podem ser encontrados em (Storn e Price, 1995), (Lampinen e Zelinka, 2000), (Arantes et al., 2006), (Lobato et al., 2007) e (Das et al., 2009). Em (Vesterstrøm e Thomson, 2004) é apresentada uma comparação entre este método e o método de Otimização por Enxame de Partículas, mostrado na Seção 4.4.3.

4.4.3 Otimização por Enxame de Partículas - *Particle Swarm Optimization* (PSO)

O método *Particle Swarm Optimization* (PSO) foi proposto por (Kennedy e Eberhart, 1995) e é baseado no comportamento social de determinadas espécies de pássaros, ou seja, na forma como essas partículas (pássaros) se comportam na busca de alimentos ou de um lugar para construir seus ninhos, conforme apresentado na Figura 4.7.

Segundo (Medeiros, 2005), no método PSO, cada partícula corresponde a um ponto no espaço de busca. Os valores associados a cada partícula correspondem à adequação da mesma como solução do problema tratado. A cada partícula também está associada uma velocidade, e essa velocidade é alterada levando-se em conta a melhor posição da partícula e a melhor posição do grupo e, ao longo do tempo, o grupo encontra o alimento.

Figura 4.7: Representação do método de enxame de partículas.

4.5 Algoritmo Genético (AG)

Neste trabalho, a abordagem é feita pelo Algoritmo Genético (AG). A motivação do uso do AG está na possibilidade de, junto a métodos de busca local, encontrar boas soluções para o Problema de Fluxo Multiproduto, já que os mesmos possuem um elevado dimensionamento quanto ao número de variáveis envolvidas. É óbvio que não existe nenhuma garantia das soluções encontradas serem ótimas. No entanto, podem ser geradas boas soluções e em tempos computacionais menores quando comparados a métodos exatos, para problemas muito complexos, como muitos problemas de otimização combinatória.

No AG, um indivíduo representa um *cromossomo*, que contém a *codificação* (genótipo) de uma possível *solução* (fenótipo). Na Figura 4.8, são mostradas representações binárias, em que o cromossomo corresponde a um vetor binário. Cada posição desse vetor representa um gene e os possíveis valores de cada gene, no caso 0 ou 1, identificam os alelos. Obviamente, esta é uma definição particular, pois, em outros problemas, cada indivíduo pode ser representado de forma diferente e conseqüentemente, cada termo descrito poderá ser visto de outra maneira, associado ao próprio problema.

De acordo com (Von Zuben, 2000), o processo evolutivo executado por um AG corresponde a um procedimento de busca em um espaço de soluções potenciais para o problema.

É necessário discutir alguns parâmetros e operadores do AG, já que os mesmos influenciam diretamente no sucesso do algoritmo.

Operador de *crossover*

É também chamado de operador de recombinação. Possui a função de criar novos indivíduos (filhos) a partir da seleção de dois ou mais indivíduos (pais). O operador mais comum e mais usado é o *crossover* de um ponto, em que dois pais são selecionados e dois novos filhos são gerados, a partir de um ponto de corte escolhido aleatoriamente. Na Figura 4.8, é apresentado um exemplo deste tipo de operador de *crossover* utilizando soluções com representação binária. Nesta figura, é mostrado que o filho 1 (2) "herda" os valores da posição 1 até o ponto de corte do pai 1 (2) e os demais valores do pai 2 (1). No algoritmo genético clássico, cada indivíduo tem

uma probabilidade fixa de ser selecionado para este operador; no entanto, de acordo com a necessidade, podem ser selecionados pais de acordo com a função de aptidão. Por exemplo, pode ser escolhido um pai entre os 30% melhores indivíduos e outro pai pode ser escolhido de forma aleatória dentre todos os indivíduos da população. Neste trabalho, todos os indivíduos podem ser selecionados para a ação de *crossover*, com probabilidade uniforme.

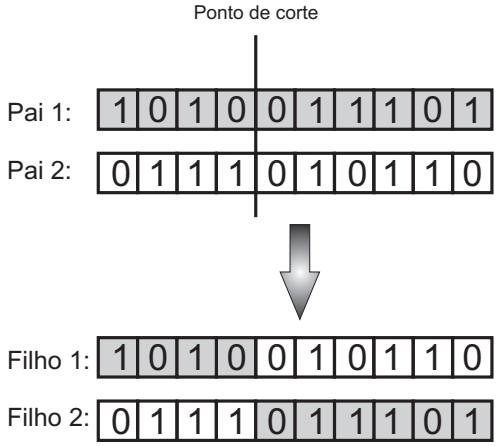

Figura 4.8: *Crossover* 1 ponto.

Outro operador de *crossover* é o de 2 pontos, em que são escolhidos dois pontos de corte, em vez de apenas 1. Existem diversos outros operadores de *crossover*, como o *crossover* Uniforme, utilizado neste trabalho, e proposto por (Syswerda, 1989). No operador de *crossover* Uniforme, para cada bit no primeiro filho, é decidido, com alguma probabilidade, qual será o pai que irá repassar o valor daquela posição ao filho. Este operador é capaz de diversificar mais as soluções geradas, pois um filho não herda necessariamente uma faixa de bits única de cada pai.

A Figura 4.9 ilustra o processo de geração de dois filhos, após a seleção de dois pais, a partir do *crossover* Uniforme. Primeiramente, é gerado um vetor binário, e o filho 1 (2), herda do pai 1 (2) os genes correspondentes às posições do vetor binário que contém o valor 1 (0); as demais posições do filho 1 (2) são compostas pelos valores do pai 2 (1).

Outros exemplos de operador de *crossover* são o operador PMX (*Partially Matched Crossover*), o CX (*Cycle Crossover*) e o OX (*Order Crossover*), que possuem características específicas e são detalhados em (Silva e Oliveira, 2006). Neste artigo, são feitas comparações entre implementações de Algoritmo Genético aplicadas ao Problema do Caixeiro Viajante utilizando tais operadores.

Operador de mutação

É utilizado para modificar um ou mais genes do indivíduo gerado pelo operador de *crossover*. Cada indivíduo normalmente possui uma determinada probabilidade de sofrer mutação, chamada de *taxa de mutação*. O principal objetivo é diversificar o espaço de busca, evitando uma possível convergência prematura e fugindo de ótimos locais. A taxa de mutação não pode ser alta, pois assim modificaria muito os novos indivíduos e descaracterizaria o crossover. Este operador de mutação tem o

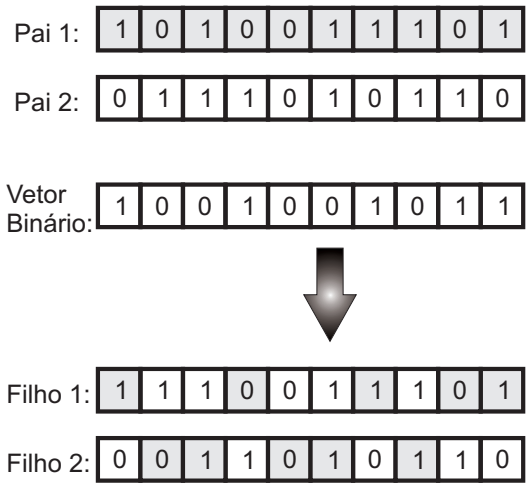

Figura 4.9: Crossover Uniforme.

importante papel de alterar os genes de um indíviduo, com uma baixa probabilidade disso acontecer, caracterizando ainda mais o processo evolutivo.

Considerando os exemplos binários exibidos nesta Seção, uma mutação em um gene de um filho gerado consiste em alterar o valor do bit para 0, se o filho contiver 1, ou vice-versa.

Seleção de indivíduos para a próxima geração

Existem diversas formas de seleção de indivíduos, porém um método mais usado de seleção de indivíduos para a próxima geração é o método da roleta (Mccallum, 1977). Neste método, quanto melhor o valor da função de aptidão do indivíduo, mais chances ele tem de ir para a próxima geração. Outro tipo de operador de seleção é a seleção baseada em *rank*, que é realizada a partir dos valores da função de aptidão de cada indivíduo. Um método bastante utilizado e que de uma forma geral é uma variação do operador baseado em *rank*, é simplesmente conhecido como *elitismo*, que garante que um percentual dos melhores indivíduos não seja perdido.

No método da roleta, cada indivíduo recebe uma probabilidade de ir para a próxima geração proporcional, ao valor da sua função de aptidão. Assim, os indivíduos com maior probabilidade são os que possuem maiores (melhores) valores de aptidão.

A estratégia de *rank* ordena os indivíduos de acordo com os valores funcionais de aptidão e passa os melhores para a próxima geração. Neste caso, é sempre garantido que a melhor solução passará para a próxima geração, fato que não ocorre com o método da roleta. Uma alternativa ao uso do método da roleta e garantir que a melhor solução não será perdida é sempre mantê-la na geração seguinte.

A Figura 4.10 apresenta o pseudo-código de um AG.

4.6 Algoritmos Genéticos Multiobjetivos

Nesta Seção, serão apresentados métodos evolutivos aplicados a problemas multiobjetivos.

```
1  Procedimento AG;
2  t ← 0;
3  P(t) ← GeraPopulacaoIncial;
4  Avalia(P(t));
5  enquanto CriterioParada faça
6      t ← (t + 1);
7      P(t) = GeraPopulacao(P(t − 1));
8      Altera(P(t));
9      Avalia(P(t));
10     PopulacaoSobrevivente ← P(t);
11 fim
12 Fim AG;
```

Figura 4.10: Pseudo-código de um AG.

Segundo (Castro, 2001), os métodos evolucionários apresentam certas características mais apropriadas para a resolução de problemas multiobjetivos, principalmente na obtenção do Conjunto Ótimo de Pareto, já que tais métodos utilizam um conjunto de soluções e, por isso, exploram melhor o espaço de busca. Ainda segundo (Castro, 2001), a primeira implementação prática do método para tratar problemas multiobjetivos data de 1984, com Schaffer, e, após isso, um modelo de ordenamento revolucionário de soluções não-dominadas foi proposto por Goldberg, em 1986. Alguns destes algoritmos evolucionários aplicados a problemas multiobjetivo são citados a seguir.

VEGA (*Vector Evaluated Genetic Algorithm*) : desenvolvido por (Shaffer, 1984), de acordo com (Garcia, 2005). Este algoritmo tem, como principal diferença em relação ao AG convencional, o uso de um operador especial de seleção. Neste algoritmo, é criada uma subpopulação para cada objetivo tratado. Um problema observado em (Garcia, 2005) foi a criação de novas espécies durante o processo de evolução.

Considerando cada subpopulação como sendo uma espécie, é razoável avaliar que este método simula melhor um processo de evolução mais amplo quando comparado ao AG convencional, que tecnicamente considera somente uma espécie.

É também intuitivo inferir que cada espécie possui características distintas, ou seja, podem ser mais adaptáveis para uma função do que para outra função. Fazendo uma analogia ao mundo real, é como se alguns animais tivessem maior aptidão para sobreviverem em ambientes mais secos, enquanto outros são mais aptos a sobreviverem em ambientes mais úmidos.

O problema desta abordagem é que não é interessante ter indivíduos excelentes em relação a um objetivo e péssimos em relação a outros. Portanto, a idéia de espécies não é um bom método pelo fato de otimizar funções de forma independente.

MOGA (*Multiobjective Optimization Genetic Algorithm*) : proposto no início da década de 90 por Fonseca e Fleming (1993). A idéia geral deste algoritmo

é estabelecer uma nova ordenação aos indivíduos, sendo que as soluções dominadas são penalizadas. Segundo (Avila, 2006), a dificuldade está em interpolar estes dois grupos de soluções, dominadas e não-dominadas, de forma a conseguir uma boa aproximação da fronteira Pareto Ótima.

NSGA (*Nondominated Sorting Genetic Algorithm*) : proposto por Srinivas e Deb (1994), o princípio deste algoritmo é ordenar as soluções através da camada em que as mesmas se encontram no conjunto de Pareto. Dessa forma, soluções não-dominadas são mantidas de uma geração para outra e também soluções dominadas podem permanecer. No entanto, com probabilidade menor de sobrevivência em relação à primeira camada (soluções não-dominadas). Outra versão do NSGA é o NSGA II, proposto por Deb et al. (2002), e que foi aplicado ao caso multiobjetivo abordado neste trabalho. Por este motivo, o NSGA II é descrito de forma mais detalhada na Seção 4.7.

NSGA II (*Nondominated Sorting Genetic Algorithm II*) : proposto por Deb et al. (2002), este método traz algumas diferenças de seu antecessor. De acordo com Deb et al. (2002), o NSGA possui três principais pontos críticos, que foram melhorados na nova versão. Estes pontos são: *i)* complexidade computacional no ordenamento de soluções não-dominadas; *ii)* abordagem não-elitista; e *iii)* necessidade de um parâmetro de compartilhamento.

Para corrigir o primeiro ponto crítico do algoritmo, foi proposto um outro método de ordenamento de soluções com complexidade $\mathcal{O}(MN^2)$, melhorando o anterior, que possui complexidade $\mathcal{O}(MN^3)$.

No caso do segundo ponto crítico do NSGA, o NSGA II possui uma abordagem elitista, de forma que as soluções são distribuídas em *fronts*, com o primeiro *front* sendo das soluções não-dominadas, e o segundo *front* contém as soluções dominadas por pelo menos uma solução contida no primeiro *front*, e os demais *fronts* são definidos de maneira análoga. As soluções contidas nos menores *fronts* não sobrevivem, portanto cada solução dominada possui uma probabilidade de sobreviver de acordo com a quantidade de soluções que a domina e também de acordo com o *front* dessas soluções que a domina.

Para corrigir o terceiro ponto crítico presente no NSGA, o NSGA II utiliza o valor da *crowding distance* (distância de agrupamento), que será definida na Seção 4.7.

SPEA (*Strength Pareto Evolutionary Algorithm*) : proposto por Zitzler e Thiele (1998), constitui-se de um método evolutivo multiobjetivo elitista. Segundo Castro (2001), o algoritmo funciona com a manutenção de uma população externa que armazena, a cada geração, todas as soluções não-dominadas, desde a população inicial. A aptidão de cada indivíduo é feita primeiramente combinando a população atual com a população externa e, em seguida, atribuindo a cada solução não-dominada um valor de aptidão baseado no número de soluções que elas dominam.

SPEA 2 (*Strength Pareto Evolutionary Algorithm 2*) : proposto por Zitzler et al. (2001), este método foi desenvolvido para melhorar a eficiência e tentar eliminar as restrições do seu antecessor. Segundo Machado (2006), as três principais diferenças entre o SPEA e o SPEA 2 são:

- uma estratégia de atribuição da aptidão que considera, para cada indivíduo, tanto o número de soluções que a dominam quanto o número de soluções dominadas por ela;
- uma estimativa da densidade da vizinhança incorporada à aptidão dos indivíduos;
- a substituição de um algoritmo de agrupamento por um método alternativo de truncamento, que preserva soluções externas.

Estes são apenas exemplos de métodos evolutivos encontrados na literatura com estratégias de Otimização Multiobjetivo. Obviamente, existem outros métodos, mas percebe-se que basicamente a preocupação de todos os métodos não é encontrar apenas uma solução satisfatória, como em métodos heurísticos desenvolvidos para otimização mono-objetivo, mas, sim, encontrar um conjunto de soluções que mais se aproxime da fronteira Pareto Ótima.

4.7 *Nondominated Sorting Genetic Algorithm* II

Proposto por Deb et al. (2002), este método difere da primeira versão do NSGA em três pontos principais:

- possui complexidade computacional de ordenamento por não-dominância igual a $\mathcal{O}(MN^2)$, com M sendo o número de objetivos e N o tamanho da população. Seu antecessor, o NSGA, é $\mathcal{O}(MN^3)$ e, de acordo com (Deb et al., 2002), o método NSGA é caro computacionalmente para grandes tamanhos de populações;
- abordagem elitista, devido a um ordenamento de soluções com base nas soluções que a dominam. Zitzler et al. (2000) mostraram que o elitismo melhora significamente o desempenho de um AG, além de evitar a perda de boas soluções.
- não necessidade de um parâmetro de compartilhamento, utilizado para manter a diversidade na população, necessário em seu antecessor.

Tais alterações possuem o intuito de reduzir o tempo computacional, tornar o método elitista e também melhorar o método evolutivo quanto a diversidade. Tais alterações implicam em ganhos quanto à velocidade e convergência.

```
1  Procedimento FastNonDominatedSorting(S);
2  para cada (s ∈ S) faça
3      S_s = ∅;
4      n_s = 0;
5      para cada (q ∈ S) faça
6          se (s ≺ q) então
7              S_s = S_s ⋃{q}; {q é adicionada ao conjunto de soluções dominadas
               por s.}
8          fim
9          se (q ≺ s) então
10             n_s = n_s + 1;
11         fim
12     fim
13     se (n_s = 0) então
14         s_rank = 1;{insere s no front 1}
15         F_1 = F_1 ⋃{s};
16     fim
17 fim
18 enquanto (F_i ≠ ∅) faça
19     Q = ∅;
20     para cada (s ∈ F_i) faça
21         para cada (q ∈ S_s) faça
22             n_q = n_q − 1;
23             se (n_q = 0) então
24                 q_rank = i + 1; {q pertence ao próximo front}
25                 Q = Q ⋃{q}; {insere em outro nível de não-dominância}
26                 i = i + 1;
27                 F_i = Q;
28             fim
29         fim
30     fim
31 fim
32 Fim FastNonDominatedSorting;
```

Figura 4.11: Pseudo-código - *Fast NonDominated Sorting* (Deb et al., 2002).

4.7.1 *Fast NonDominated Sorting*

O NSGA-II inicia-se com a identificação da primeira fronteira (conjunto de soluções não dominadas) em uma população de tamanho N. Para encontrá-la, é necessário que todas as soluções sejam classificadas em dominadas ou não-dominadas. Isso é feito comparando-se cada solução com todas as outras soluções. Ao final desse processo, é encontrado o conjunto de soluções não-dominadas, que irá compor a primeira fronteira. Primeiramente, para cada solução s, encontra-se o número de soluções que a dominam, denotado por n_s (contador de dominância) e o conjunto

$\mathcal{S}_s$, formado pelas soluções dominadas por s. De acordo com (Deb et al., 2002), isso requer $\mathcal{O}(MN^2)$ comparações, de forma que M é a quantidade de objetivos e N consiste no tamanho da população.

Então, todas as soluções não-dominadas possuem $n_s = 0$ e formarão o primeiro *front*. Em seguida, para cada solução s do *front 1*, são visitadas todas as soluções q pertencentes ao conjunto $\mathcal{S}_s$ e o contador de dominância n_q é decrementado em uma unidade. Os elementos q com contador de dominância iguais a zero são, então, separados para uma lista Q e estas soluções pertencem ao segundo nível de não-dominância, ou seja, pertencem ao *front 2*. O processo é repetido para encontrar os demais níveis de não-dominância.

Este método de ordenação por não-dominância é conhecido como *Fast NonDominated Sorting* e seu pseudo-código é apresentado na Figura 4.11, de acordo com (Deb et al., 2002). Na Figura 4.11, S representa a população e a representação $s \prec q$ significa que a solução s domina a solução q e F_i corresponde ao i-ésimo *front*.

Ao fim do procedimento *Fast NonDominated Sorting*, as soluções são alocadas em diferentes *fronts*, conforme mostra a Figura 4.12.

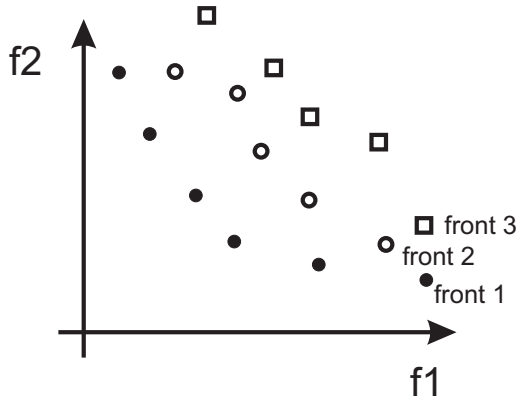

Figura 4.12: Atribuição de *fronts* para cada solução pelo *Fast NonDominated Sort*.

4.7.2 *Crowding Distance*

Quanto à diversidade, o método NSGA II também difere do NSGA. Como mencionado anteriormente, o NSGA necessita de um parâmetro de compartilhamento denotado por σ_{share}, necessário no método da função de compartilhamento e utilizado para manter a diversidade da população. Este parâmetro está relacionado com a distância métrica escolhida para calcular a medida da proximidade entre dois indivíduos da população.

De acordo com (Deb et al., 2002), o desempenho do método está relacionado com a escolha desse parâmetro σ_{share}. Este método exige a comparação de cada solução com todas as outras soluções da população, o que gera um custo computacional de $\mathcal{O}(N^2)$, sendo N o tamanho da população.

No NSGA II, o método de compartilhamento foi substituído por uma abordagem que utiliza uma *crowded-comparasion*, a qual não possui necessidade de especificar nenhum parâmetro de compartilhamento e também possui uma melhor complexidade computacional quando comparado ao NSGA.

A *Crowding distance* é definida como sendo o perímetro do cubóide criado a partir de um ponto e tendo como vértices os vizinhos deste ponto. O cálculo é feito com soluções de um mesmo *front* e as soluções extremas recebem um valor infinito e sempre terão preferência no método elitista.

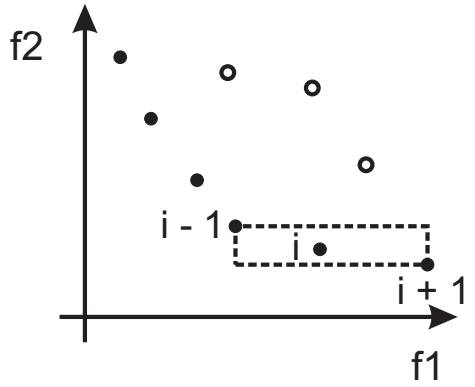

Figura 4.13: Cubóide gerado a partir de uma solução.

Na Figura 4.13, é exibida uma representação gráfica do cubóide gerado a partir de uma solução, tendo como vértices os pontos correspondentes às soluções vizinhas. O cálculo da *crowding distance* necessita do ordenamento de cada objetivo em or-

```
1  Procedimento Crowding Distance;
2  l = |I|; {l corresponde ao número de soluções em I}
3  para cada (i ∈ I) faça
4      I[i]_cd = 0;
5  fim
6  para cada (cada objetivo m) faça
7      I = ordena(I, m);{ordena I em relação ao objetivo m em ordem
       crescente}
8      I[1]_cd = I[l]_cd = ∞; {atribui infinito às soluções extremas de I}
9      para cada (i = 2 até (i − 1)) faça
10         I[i]_cd = I[i]_cd + ((I[i + 1].m − I[i − 1].m)/(f_m^max − f_m^min))
11     fim
12 fim
```

Figura 4.14: Pseudo-código de Procedimento *Crowding Distance*.

dem crescente; em seguida, cada solução extrema do *front* recebe um valor infinito. Todas as soluções intermediárias receberão um valor igual à diferença absoluta normalizada em função dos valores das soluções extremas. O cálculo é repetido para cada objetivo. O valor global de cada solução corresponde à soma dos valores em relação a cada objetivo.

A Figura 4.14 apresenta o pseudo-código do algoritmo para o cálculo da crowding distance, sendo $\mathcal{I}$ um conjunto de soluções de um mesmo nível de não-dominância e $\mathcal{I}[i]_{cd}$ representa o valor da crowding distance da solução $i \in \mathcal{I}$, $\mathcal{I}[i].m$ representa o valor da função objetivo m da solução i e, f_m^{max} e f_m^{min} corresponde, aos valores mínimos e máximos, respectivamente, dos valores do objetivo m.

4.7.3 Algoritmo NSGA II

Primeiramente é gerada, de forma aleatória, uma população inicial P_0 e a essa população é aplicado o procedimento *Fast non Dominated Sort*, descrito na Seção 4.7.1. Cada solução recebe um *rank* de acordo com seu nível de não-dominância, o que define o índice da fronteira (*front*) à qual a solução pertence. Em seguida, por meio dos operadores comuns dos algoritmos genéticos, é gerada uma nova população Q_0. Generalizando, na geração t, temos P_t e Q_t, sendo essa última gerada por meio de operadores de seleção *crossover* e mutação.

O próximo passo é gerar uma população $R_t = P_t \bigcup Q_t$. Em seguida, o algoritmo *Fast non Dominated Sort* recebe a população R_t e são geradas as fronteiras. As soluções pertencentes a $\mathcal{F}_1$ são as não-dominadas e, conseqüentemente, formam o conjunto de soluções candidatas a ótimas de Pareto. Neste trabalho, após determinar os valores dos *fronts*, os valores da *crowding distance* são calculados para o último *front* da seguinte forma. Seja a solução n_{pop}, sendo n_{pop} o número de indivíduos do NSGA II, pertencente a um determinado front n_{rank}. Se a solução $n_{pop}+1$ pertencer ao front $n_{rank}+1$, então o cálculo da *crowding distance* não se faz necessário, pois todas as soluções até n_{pop} serão aceitas para a próxima geração. Mas se a solução $n_{pop}+1$ pertencer ao mesmo front n_{rank}, então é efetuado o cálculo da *crowding distance* em cada solução do front n_{rank} e as soluções que passarão para a próxima geração serão as soluções até o front n_{rank}, sendo que neste *front* serão escolhidas as soluções que tiverem os maiores valores de *crowding distance*.

Quanto ao elitismo, primeiramente são preferidas as soluções que pertencem a fronts menores e, em seguida, é efetuado o cálculo da *crowding distance* no *front* em que isso se faz necessário, e são preferidas as soluções com maiores valores de *crowding distance*. A Figura 4.15 ilustra este procedimento.

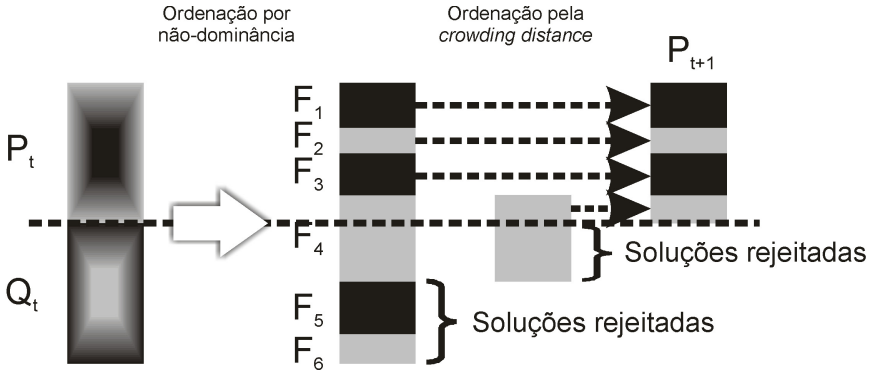

Figura 4.15: Procedimento NSGA II.

A Figura 4.16, semelhante à encontrada em (Deb et al., 2002), mostra o pseudo-código do método NSGA II, sendo N_{pop} o tamanho da população.

4.8 AG e NSGA II aplicados a Problemas Mono-objetivos e Multiobjetivos de Fluxo em Rede

Tanto o AG quanto o NSGA II têm sido amplamente aplicados à otimização de problemas de fluxo em redes. As primeiras pesquisas e simulações computacionais

1 **Procedimento** *NSGA II*;
2 $R_t = P_t \bigcup Q_t$;{combina população com os filhos gerados}
3 $\mathcal{F} = FastNonDominatedSort(R_t)$;{$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, ...)$, todos os fronts de R_t}
4 $P_{t+1} = \emptyset ei = 1$;{inicialização da próxima população e da variável i}
5 **enquanto** $((|P_{t+1}| + |\mathcal{F}_i|) \leq N_{pop})$ **faça**
6 $CrowdingDistance(\mathcal{F}_i)$;{cálculo da Crowding Distance em $\mathcal{F}_i$}
7 $P_{t+1} = P_{t+1} \bigcup F_i$;{inclui $\mathcal{F}_i$ na população}
8 $i = i + 1$;{verifica a próxima fronteira para inclusão}
9 **fim**
10 $Ordena(\mathcal{F}_i, \prec)$;{ordena em ordem decrescente}
11 $P_{t+1} = P_{t+1} \bigcup \mathcal{F}_i[1 : (N_{pop} - |P_{t+1}|)]$;{escolha dos primeiros elementos de $\mathcal{F}_i$}
12 $Q_{t+1} = CriaNovaPopulacao(P_{t+1})$;{cria nova população por meio dos operadores genéticos}
13 $t = t + 1$;

Figura 4.16: Pseudo-código NSGA II.

utilizando AG's foram efetuadas inicialmente por (Holland, 1975), enquanto os estudos com o método NSGA no início da década de 90, originalmente com Srinivas e Deb (1994), quando o método NSGA foi proposto e aprimorado em (Deb et al., 2002), quando o algoritmo NSGA II foi proposto. Além do NSGA, existem outros métodos evolucionários multiobjetivos, como o *Multiobjective Optimization Genetic Algorthm* (MOGA) (Fonseca e Fleming, 1993), o *Vector Evaluated Genetic Algorithm* (VEGA) (Shaffer, 1984), *Strength Pareto Evolutionary Algorithm* (SPEA), (Zitzler e Thiele, 1998) e o SPEA 2, (Zitzler et al., 2001).

A seguir são descritos alguns trabalhos encontrados na literatura, que utilizam ou o AG mono-objetivo ou algum método evolucionário multiobjetivo aplicados a problemas de otimização de fluxos em redes.

Em Erickson et al. (2002), é descrito um problema de otimização de roteamento do tráfego na internet, com o objetivo de minimizar o congestionamento na rede. É apresentado um algoritmo genético para resolver o problema, sendo os resultados comparados àqueles mais conhecidos obtidos por outras heurísticas desenvolvidas especificamente para este tipo de problema. De acordo com (Erickson et al., 2002), o algoritmo proposto foi capaz de produzir soluções de boa qualidade para a maioria das instâncias.

Sheng et al. (2006) propõem um algoritmo genético para resolver um problema de transporte com o ponto de corte do cruzamento e o pivot randômico da mutação desenvolvidos para uma evolução eficiente. Segundo os autores, o algoritmo proposto produz soluções de melhor qualidade quando comparadas aos algoritmos genéticos baseados na matriz de códigos (Michalewicz et al., 1991).

Em Costa et al. (2003), são apresentadas técnicas de otimização combinatória e de computação evolucionária para determinar opções de traçado para atingir novos pontos e realizar o dimensionamento dos tubos de uma rede de distribuição de gás em expansão, considerando o atendimento das demandas e buscando minimizar os custos envolvidos na implantação da rede.

Carrano et al. (2006) apresenta um método evolucionário multiobjetivo para o problema de projeto de redes de distribuição elétrica, em que os objetivos são definidos como um índice de custo monetário (incluindo custo de instalação e custo de perda de energia) e um índice de falha no sistema.

4.9 Conclusão

Neste capítulo foram definidas heurísticas construtivas e de refinamento, além de terem sido apresentadas duas heurísticas de busca local, sendo que o Método da Descida Randômica foi utilizado neste trabalho. Quanto às metaheurísticas, que apresentam estratégias de fuga de ótimos locais, foi apresentada a metaheurística ILS, também utilizada neste trabalho.

Em seguida foram descritos alguns métodos evolutivos presentes na literatura, tais como ACO, DE, PSO e AG e também foram descritos alguns métodos evolutivos multiobjetivos. Neste trabalho os métodos evolutivos aplicados foram o AG e o NSGA II.

Finalmente foram apresentados trabalhos co-relatados, que utilizam métodos evolutivos com o objetivos de solucionar problemas de fluxo em redes.

Capítulo 5

Metodologia Mono-objetivo

Este capítulo se dedica a apresentar uma metodologia de solução do Problema de Fluxo Multiproduto em sua versão mono-objetiva, problema este descrito na Seção 2.4 e representado pela expressão (2.3), bem como os resultados encontrados a partir da adoção desta metodologia. São apresentados os algoritmos desenvolvidos para solucionar este problema e os resultados encontrados são comparados com os obtidos pela aplicação de métodos exatos para as mesmas instâncias aqui tratadas.

5.1 Introdução

A metodologia proposta para a solução do Problema de Fluxo Multiproduto envolve o desenvolvimento de heurísticas construtivas, para gerar soluções iniciais, a serem usadas pelas heurísticas de refinamento e pelas metaheurísticas Algoritmo Genético (AG) e *Iterated Local Search* (ILS). Foram implementados os Métodos da Descida Randômica, *Iterated Local Search* e AG para solucionar o PFM. A escolha da Descida Randômica se deve ao fato de tal método ser mais barato computacionalmente, quando comparado ao método clássico da descida. O ILS foi escolhido pela simplicidade de implementação e pelo sucesso que obteve em muitos trabalhos encontrados na literatura e o AG se deve ao fato de ser um método evolutivo e, por este motivo, poder explorar melhor o espaço de busca.

Antes de descrever cada método, é necessário discutir algumas particularidades computacionais. São elas:

- **Representação de uma solução:** Uma solução é representada por uma matriz x de dimensão $a \times p$, em que são representados os fluxos dos produtos em cada arco, sendo a a quantidade de arcos e p o número de produtos. Cada elemento dessa matriz é uma variável, de modo que o elemento x_{ij} representa o fluxo do produto j no arco i. Cada coluna j da matriz solução representa o fluxo do produto j. Cada unidade de cada produto possui uma única rota, mesmo essa matriz não possuindo uma representação binária.

- **Vizinhança de uma solução:** Para explorar o espaço de solução do problema, é aplicado um movimento que consiste em trocar o fluxo de um produto escolhido aleatoriamente. Primeiramente, é escolhido, de forma aleatória, um produto, e, em seguida, é traçada uma nova rota para ele. Esta nova rota

atende à restrição de conservação de fluxo associada ao par origem-destino. Essa definição de vizinhança será utilizada pelo método de busca local utlizado (Método da Descida Randômica).

Na próxima Seção, será apresentada a Heurística Construtiva Aleatória baseada na Matriz de incidência nó-arco e utilizada para geração de soluções iniciais para os métodos propostos neste trabalho.

5.2 Heurística Construtiva

Para gerar uma solução inicial, foi desenvolvida uma Heurística Construtiva baseada nas matrizes N (incidência nó-arco) e b (oferta/demanda) do modelo posto pela expressão (2.3). Para descrever este algoritmo, considere a rede apresentada pela Figura 2.1 e novamente apresentada pela Figura 5.1.

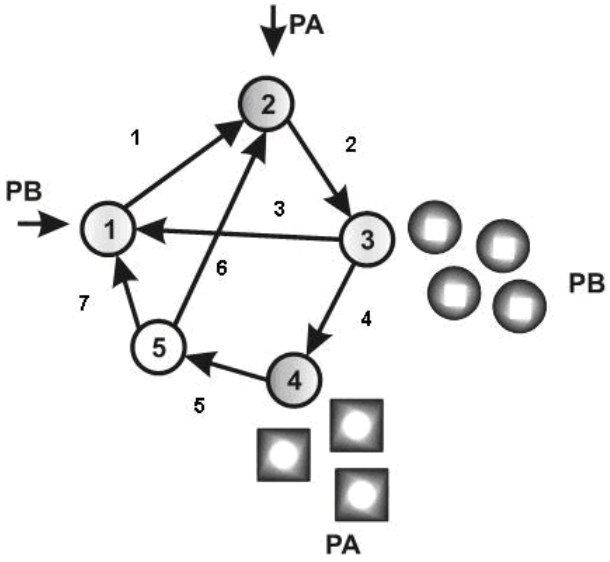

Figura 5.1: Exemplo de uma rede multiproduto.

Considere cada arco com capacidade igual a 5. Assim, as matrizes N (incidência nó-arco), b (oferta/demanda), c (matriz de custo) e o vetor de capacidade u ficam postos da seguinte maneira:

$$N = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 0 & -4 \\ -3 & 0 \\ 0 & 4 \\ 3 & 0 \\ 0 & 0 \end{bmatrix}, \quad c = \begin{bmatrix} 2 & 3 & 4 & 1 & 1 & 4 & 1 \\ 2 & 3 & 4 & 1 & 1 & 4 & 1 \end{bmatrix}, \quad u = \begin{bmatrix} 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \end{bmatrix}$$

A partir da matriz b, foi construída uma matriz reduzida M de oferta/demanda, importante na construção da solução.

$$M = \begin{bmatrix} 4 & 1 & 3 \\ 2 & 1 & -3 \\ 3 & 2 & 4 \\ 1 & 2 & -4 \end{bmatrix} \tag{5.1}$$

Na matriz reduzida M, cada elemento na primeira coluna representa um nó de oferta ou demanda; a segunda coluna apresenta os produtos; e a terceira coluna representa a oferta/demanda do produto correspondente à linha. A dimensão da matriz reduzida é $2p \times 3$, sendo p o número de produtos.

O pseudo-código da Heurística Construtiva proposta é apresentado na Figura 5.2 e descrito a seguir.

```
1  Procedimento Heurística Construtiva(N, M);
2  x = matriz nula a × p
3  para cada (i ∈ M) faça
4      j = 0 // contador de infactibilidade
5      aux1 = M(i, 1)
6      enquanto aux1 ≠ M(i, 2) faça
7          Armazenar, num vetor, todos os indices das colunas que possuem valor
           1 na matriz N, na linha aux1, pois são números de arcos para os quais
           aux1 é origem
8          Selecionar, aleatoriamente, um número armazenado neste vetor e fazer
           aux2 = ndicedacolunaselecionada
9          se arco selecionado estiver for violado somando-se M(i, 3) e
           j ≤ q_max então
10             aux1 = M(i, 1)
11             j = j + 1
12         fim
13         Para o arco selecionado, existe um único nó destino, linha onde está o
           valor −1
14         x(aux2, i) = x(aux2, i) + M(i, 3)
           aux1 = ndicedalinhaondeestovalor − 1 //quando aux1 = M(i, 1), o
           fluxo para o produdo i é concluído
15     fim
16 fim
```

Figura 5.2: Pseudo-código da Heurística Construtiva.

Seja M a matriz reduzida. Esta heurística construtiva aleatória primeiramente atribui o valor contido em $M(i, 1)$ a uma variável auxiliar $aux1$ e, em seguida, verifica quais são os possíveis arcos, procurando na linha $aux1$ da matriz N. Para isso, o algoritmo armazena em um vetor $vaux$ todos os índices das colunas cujo valor presente na linha $aux1$ da matriz N seja 1, pois, neste caso, o nó armazenado em $aux1$ será um nó de oferta para o arco. Em seguida, seleciona-se aleatoriamente um

elemento do vetor dos índices $vaux$ e atribui-se esse valor a uma variável auxiliar $aux2$. Assim, na matriz solução x, o elemento $x(p_j, aux1)$ receberá o valor contido em $M(aux1, 3)$, sendo p_j correspondente ao produto j para o qual está sendo traçado o fluxo. Antes de fazer essa atribuição na matriz solução, é verificado se o arco $aux2$ não será violado ao receber tal valor. Caso ocorra violação da capacidade do arco $aux2$, então um novo elemento do vetor $vaux$ será selecionado aleatoriamente, gerando um novo valor para $aux2$. Esta nova seleção aleatória em caso de violação do arco $aux2$ é feita até um determinado número de vezes, chamado neste trabalho de q_max. Caso não houver possibilidade de fazer uma atribuição a um arco tal que ele não ultrapasse sua capacidade, então a violação será aceita. Em seguida, o índice da linha em que há um valor correspondente a -1 na coluna $aux2$ da matriz N é atribuído a $aux1$ e o processo é feito novamente para decidir outro arco $aux2$ pelo qual o produto poderá passar. Este processo é reiniciado desde a origem no caso do nó atribuído a $aux1$ ser um nó tal que não existe um valor igual a 1 na matriz N na linha $aux1$, e ele é diferente do valor armazenado em $M(i+1, 1)$, correspondente ao nó de demanda para o produto em questão. No caso de $aux1$ receber um valor igual ao armazenado em $M(i+1, 1)$, então o fluxo do produto em questão foi determinado aleatoriamente e o processo continua para o próximo produto. Obviamente, a próxima atribuição de $aux1$ será o valor armazenado em $M(i+2, 1)$.

O procedimento proposto na Figura 5.2 garante factibilidade quanto às restrições de conservação de fluxo, mas não garante factibilidade quanto às restrições de capacidade. O exemplo a seguir, baseado na Figura 5.1, mostra sua utilização.

A geração de uma solução x qualquer para o exemplo é feita então da forma descrita a seguir. Seja, inicialmente, uma solução x em que todos os elementos são nulos.

Passo 1: Atribuir a $aux1$ o valor contido em $M(1, 1)$ e atribuir à variável i o índice da primeira linha da matriz. Portanto $aux1 = 4$ e $i = 1$, conforme Figura 5.3.

$$\begin{pmatrix} 4 & 1 & 3 \\ 2 & 1 & -3 \\ 3 & 2 & 4 \\ 1 & 2 & -4 \end{pmatrix}$$

Figura 5.3: Passo 1.

Passo 2: Selecionar aleatoriamente uma arco na linha $aux1$ na matriz N, de forma que a entrada do arco selecionado na linha $aux1$ e na coluna selecionada seja 1. No exemplo, só existe uma possibilidade e o arco selecionado foi o arco 5. Este valor é atribuído a $aux2$. Se não houver nenhum valor 1 na linha correspondente e ainda se $aux1 \neq M(i+1, 1)$, voltar ao **Passo 1**. Veja Figura 5.4.

Passo 3: O arco selecionado foi o arco 5. Este passo consiste em verificar se este arco não está violado quanto à restrição de capacidade, ou seja, verificar se a

$$N = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{pmatrix}$$

Figura 5.4: Passo 2.

soma dos produtos que estão passando por ele não superam a sua capacidade. Se violar, voltar ao **Passo 2** e se necessário voltar ao **Passo 2** até q_max e aceitar o último arco selecionado. Se não violar, ir ao **Passo 4**.

Passo 4: Como o produto para o qual o fluxo está sendo gerado é o produto 1, cujo fluxo estará na primeira coluna da matriz solução, atribuir o valor armazenado em $M(i, 3)$ à posição $x(aux2, 1)$ na matriz solução.

Passo 5: Atribuir a $aux1$ o índice da linha que contém o valor -1 na mesma coluna. Neste caso, $aux1 = 5$, conforme Figura 5.5.

$$N = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{pmatrix}$$

Figura 5.5: Passo 5.

Passo 6: Verificar se $aux1 = M(i + 1, 1)$. Em caso afirmativo, o fluxo para o produto 1 termina e os passos recomeçam para o produto 2, com a atribuição $i = i + 2$; em caso negativo, voltar ao **Passo 2**. Refazendo o **Passo 2**, existem duas possibilidades de seleção de um arco: seja, então, selecionado o arco 6, Figura 5.6.

$$N = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{pmatrix}$$

Figura 5.6: Passo 6.

Os Passos **3**, **4** e **5** são feitos normalmente e $aux1 = 2$, pois o nó 2 é o nó para onde o fluxo poderá ir quando estiver no arco 6 neste exemplo, conforme Figura 5.7. Em outras palavras, o arco incide no nó 2.

$$N = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{pmatrix}$$

Figura 5.7: Nó onde o arco incide.

Na verificação contida no **Passo 6**, $aux1 = M(i+1, 1)$, o que significa que terminou a geração do fluxo para o produto 1 e recomeçará a criação de uma rota para o produto 2. Além disso, esse algoritmo retornou a rota para o produto 1 passando pelos arcos 5 e 6, respectivamente. Esta rota é observada na Figura 5.1.

5.3 Método da Descida Randômica Aplicado ao PFM

O Método da Descida Randômica foi aplicado aos indivíduos da população inicial e também aos filhos gerados durante a execução do AG. Além disso, foi o método de busca local aplicado ao método ILS.

Seja x uma solução do problema e x' pertencente a uma vizinhança de x, definida como os elementos do conjunto $N(x)$, de forma que x' é gerado a partir de um movimento m realizado em x.

No Método da Descida Randômica aplicado à população inicial e aos filhos gerados nas gerações do AG, um movimento m em x é definido como gerar uma nova rota para um produto escolhido de forma totalmente aleatória, sendo esta nova rota gerada de forma a manter as restrições de conservação de fluxo.

A função de avaliação deste método consiste em:

$$f(x) = Tr(cx) + \alpha\nu \tag{5.2}$$

sendo α um parâmetro utilizado para penalizar soluções infactíveis quanto às restrições de capacidade e ν uma variável que consiste no somatório das violações das restrições de capacidade em todos os arcos. Obviamente, se uma solução for factível, então a função de aptidão é a própria função objetivo do problema em questão, pois, neste caso, $\nu = 0$.

Em caso de melhora, então a solução ótima corrente é atualizada; caso contrário, gera-se um novo vizinho.

O critério de parada do método consiste no número máximo de iterações sem melhora, ou seja, no número máximo de vizinhos da solução ótima corrente que são testados. Neste trabalho, este parâmetro foi denominado $itermax$.

5.4 *Iterated Local Search* Aplicado ao PFM

Nesta Seção são discutidas características específicas do método *Iterated Local Search* (ILS) aplicado ao Problema de Fluxo Multiproduto. O método ILS está

descrito na Seção 4.3.1, tendo o pseudo-código apresentado na Figura 4.4. Foi escolhido o Método da Descida Randômica como sendo a técnica de busca local para as soluções geradas a partir de perturbações pelo método ILS.

Nesta implementação, uma perturbação no método ILS consiste em trocar o fluxo de mais um produto, ou seja, na perturbação de nível 1, são trocados os fluxos de dois produtos; na perturbação de nível 2, são trocados os fluxos de três produtos, e assim por diante.

O valor da perturbação é implementado após um número máximo de tentativas dentro de um mesmo nível. Este número máximo de tentativas em um mesmo nível foi chamado neste trabalho de *vezesnvel* e a quantidade máxima de perturbações de $ilsmax$.

Após alterar os fluxos na solução corrente, o Método da Descida Randômica é executado. Caso a solução retornada pelo método de busca seja melhor do que a solução ótima corrente, essa solução é trocada. Para cada nível de perturbação, é considerado o critério de parada por nível como sendo um número máximo de iterações sem melhora, chamado neste trabalho de $vezes\_nvel$.

O critério de parada do ILS é o número máximo de perturbações feitas sem melhora durante as perturbações, denominado no trabalho de $ilsmax$.

5.5 Algoritmo Genético Aplicado ao PFM

Nesta Seção são discutidas as características específicas do Algoritmo Genético (AG) aplicado ao Problema de Fluxo Multiproduto (PFM). A estrutura geral deste Algoritmo Genético segue o que foi exposto na Seção 4.5, tendo o pseudo-código apresentado na Figura 4.10.

Representação da solução: a representação utilizada é análoga à apresentada na Seção 5.1. Em particular, cada cromossomo representa uma solução x na formulação dada na Seção 2.3, sendo a implementação desta em forma de uma matriz. Os atributos de uma solução tratada neste trabalho são os fluxos correspondentes a cada produto. Portanto, cada coluna da matriz solução, em que estão os fluxos de cada produto, é denominada *gene*. Os possíveis fluxos que podem ser determinados para cada produto são denominados *alelos*.

Representação da população: feita por meio de uma estrutura, em que cada registro contém três campos: um para alocar a matriz solução, outro para o valor da função objetivo associada ao indivíduo e outro para armazenar o valor da função de aptidão.

Geração da população inicial: para gerar as soluções iniciais foi utilizada a Heurística Construtiva descrita na Seção 5.2, que garante factibilidade quanto às restrições de conservação de fluxo e ainda mantém a aleatoriedade, característica marcante em algoritmos genéticos. Em cada indivíduo da população inicial foi aplicado o Método de Descida Randômica, por apresentar um baixo custo computacional

quando comparado a outras heurísticas de busca local, a fim de melhorar a solução, reduzindo ou até mesmo, em alguns casos, eliminando as violações quanto às restrições de capacidade.

Seleção de indivíduos: A seleção dos indivíduos para a recombinação é feita de forma aleatória dentre todos os indivíduos da população, com probabilidade uniforme. São escolhidos dois pais aleatoriamente.

***Crossover*:** foi utilizado o *Crossover* Uniforme. Um vetor binário com número de posições igual à quantidade de produtos é criado e então este vetor é preenchido com certa probabilidade. Cada filho herda as colunas de cada pai, de acordo com o valor contido nas posições deste vetor. Foi definido um número específico de posições que contém o valor 1 neste vetor, chamado neste trabalho de *genes_trocados*, enquanto as demais posições deverão conter o valor 0. Mesmo sendo essa escolha aleatória, isso foi feito para garantir maior diversidade, e dessa forma é possível definir o quanto da codificação genética cada filho herdará de cada pai.

Mutação: a operação de mutação pode ser aleatória ou determinística. Após a geração de um filho, o mesmo pode sofrer mutação aleatória, dada uma certa probabilidade, chamada de *taxa_mutacao*. A operação de mutação neste algoritmo consiste em alterar um determinado número de colunas, que são utilizadas para representar a rota de um fluxo. No caso, a quantidade de produtos que terão seus fluxos alterados é chamada de *genes_mutados*. É importante que *genes_mutados* não seja um valor muito alto, para não descaracterizar a operação de *Crossover*. Vale lembrar que o novo fluxo é gerado utilizando o mesmo princípio utilizado na Heurística Construtiva descrita na Figura 5.2 e, por isso, o novo fluxo é factível quanto às restrições de conservação de fluxo. Caso a solução gerada não sofra nenhuma mutação, ela é então passada como parâmetro no Método da Descida Randômica. O Método da Descida Randômica é muito importante para melhorar as soluções criadas pelo *Crossover*, enquanto a operação de mutação mantém a diversidade. A ausência da mutação tradicional pode levar o algoritmo à convergência prematura, enquanto que a ausência de um método de busca local pode levar o algoritmo a encontrar soluções ruins, quando comparadas com soluções encontradas por métodos exatos.

Eliminação: sendo $nind$ o número total de indivíduos, são trocados os $nind/2$ piores indivíduos. Esses são substituídos pelos filhos gerados. Os $nind/2$ melhores indivíduos da população anterior são mantidos e a magnitude da população também continua a mesma.

Função objetivo: A função objetivo é definida como:

$$f(x) = Tr(cx) \tag{5.3}$$

sendo c uma matriz de dimensão $p \times a$ em que cada linha corresponde aos custos dos produtos em cada arco. A matriz x é a matriz solução, de dimensão $a \times p$, em que cada linha contém os fluxos dos produtos em cada arco.

Função de aptidão: A função de aptidão como:

$$f(x) = Tr(cx) + \alpha\nu \tag{5.4}$$

Como dito anteriormente, α é um parâmetro utilizado para penalizar soluções infactíveis quanto às restrições de capacidade e ν consiste no somatório das violações das restrições de capacidade em todos os arcos. Obviamente, se uma solução for factível, então a função objetivo é a mesma função de aptidão, pois, neste caso, $\nu = 0$.

Em se tratando de um problema de Otimização Restrita, o método da Penalidade foi o escolhido para tratar as restrições de conservação de Fluxo. Não foi definido nenhum método para tratamento das restrições de conservação de fluxo porque tais restrições não são violadas em momento algum na execução do algoritmo, pelo fato da forma como tais soluções são construídas, baseando-se na matriz de incidência nó-arco e, também, na matriz de oferta-demanda. Além disso, os movimentos definidos nas heurísticas garantem que as soluções sejam factíveis quanto às restrições de conservação de fluxo.

5.6 Testes Computacionais para a Solução do PFM

Os testes computacionais incidem sobre instâncias geradas aleatoriamente pelo ***GenMCF***, desenvolvido por (Alvelos, 2005). Primeiramente foi aplicado somente o AG, em seguida, foi usada a Metaheurística ILS e, depois, foi aplicado o AG seguido pelo ILS. As seções seguintes apresentam os resultados obtidos em cada caso. Foram realizados *13* testes para cada instância e foram utilizados computadores *Intel Core 2 Duo*, com 3 GHz e 2 GB de memória *RAM*, sob o Sistema Operacional *Windows XP SP3*. Os algoritmos foram implementados em linguagem *C*, por meio do compilador *Borland C++ Builder 5.0*.

5.6.1 Características das Instâncias Testadas

De acordo com Alvelos (2005), as instâncias podem ser classificadas com base em cinco características: a relação capacidade média dos arcos / demanda média dos produtos, o número de produtos, a densidade da rede (número de arcos/ número de produtos), tipo de custo (se o custo depende somente do arco ou se depende do arco e do produto que trafega por ele) e a variação dos custos.

Segundo Alvelos (2005), as 48 instâncias podem ser classificadas em dois principais grupos de 24 instâncias cada: as que possuem *l* e as que possuem *s* no nome. As Tabelas A.1 - A.2, mostradas no Anexo A, apresentam as principais características das instâncias do Pacote Carbin proposto por Alvelos (2005). Nessas tabelas, as colunas #**N**, #**A** e #**P** representam, respectivamente, número de nós, número de arcos e número de produtos. A coluna **C_Max** representa o custo máximo para um produto trafegar por um arco na rede; a coluna **T_Custo** se refere ao tipo de custo, sendo que o tipo *1* significa que o custo depende do produto e do arco, já o tipo *2* indica que o custo depende somente do arco. A coluna **CM** apresenta a média das capacidades dos arcos de cada instância, enquanto a coluna **DM** apresenta a média das demandas em cada instância. As demais colunas indicam somente relações, que podem contribuir diretamente para o desempenho dos algoritmos utilizados na resolução de cada problema teste.

Tabela 5.1: Parâmetros - AG

Grupo	Instâncias	Maxger	Nind	Itermax	Taxa_mut.	Genes_mut.	Alfa
1	bx01 - bx04	200	300	300	30%	10	100000
2	bx05 - bx08	200	100	100	30%	10	100000
3	bx09 - bx12	100	160	100	30%	40	50000
4	bx13 - bx16	80	120	60	30%	40	50000
5	bx17 - bx20	80	100	100	30%	64	50000
6	bx21 - bx24	70	80	40	30%	64	40000

Tabela 5.2: Resultados - AG - blxx

Inst.	fo_AG	T_fo	Menor Viol.	Menor Tempo
bl01	1615947	1308.47	0	1308.47
bl02	1816947	1541.33	0	1541.33
bl03	17340	1491.22	0	1491.22
bl04	618808	1799.13	6	1799.13
bl05	520135	283.53	0	283.53
bl06	493528	270.67	0	270.67
bl07	5999	305.92	0	305.92
bl08	6035	271.80	0	271.80
bl09	8072242	1232.42	29	1232.42
bl10	11726861	1281.67	88	1281.67
bl11	472078	1611.72	8	1611.72
bl12	2074632	1176.44	40	1176.44
bl13	11980387	967.67	148	967.67
bl14	10081602	1054.89	113	1054.89
bl15	8800061	935.55	175	935.55
bl16	6093829	940.38	121	940.38
bl17	15144427	1473.86	20	1473.86
bl18	12515170	1359.28	9	1359.28
bl19	8567771	922.77	169	922.77
bl20	2376923	1302.48	45	1302.48
bl21	34872623	675.92	661	675.92
bl22	30255867	653.47	553	653.47
bl23	16722133	665.81	416	665.81
bl24	22801023	682.11	568	682.11

5.6.2 Resultados - AG

A Tabela 5.1 apresenta os parâmetros utilizados durante a execução do AG. O Algoritmo Genético implementado está descrito na Seção 5.5, segundo o pseudo-código apresentado na Figura 4.10.

As instâncias foram divididas em grupos, de acordo com suas características (número de nós, arcos e produtos). Na Tabela 5.1, a coluna **grupo** é referente ao grupo de instâncias citado anteriormente; **instâncias** corresponde às instâncias testadas, de forma que cada grupo contém 8 instâncias, sendo que, ao Grupo 1, pertencem as instâncias de *bl01* a *bl04* e também as instâncias *bs01* a *bs04*.

Portanto, o "x" existente na chamada das tabelas pode assumir valor "l" ou "s".

A coluna **maxger** corresponde ao número máximo de gerações do AG; a coluna **nind**, ao número de indivíduos; a coluna **itermax** corresponde ao número máximo de iterações sem melhora utilizado na heurística de busca local (Método de Descida Randômica), aplicado aos indivíduos iniciais e aos filhos gerados pelo operador de crossover. A coluna **taxa_mut.** corresponde à probabilidade do indivíduo sofrer mutação aleatória, sendo que, caso o mesmo não sofra esse tipo de mutação, ele é submetido à heurística de busca local. A coluna **genes_mut.** corresponde ao número de colunas alteradas durante o operador de mutação aleatória. A coluna **alfa** apresenta o valor utilizado para penalizar uma solução infactível no AG e no método de busca local. Em cada geração, a quantidade de filhos gerados pelo operador *Crossover*, em particular o *Crossover* Uniforme, é igual à metade do tamanho da população (**nind**).

Tabela 5.3: Resultados - AG - blxx - Médias

Inst.	SM1	SM2	Violação Média	T_Médio	% de Fact.
bl01	1619952.35	1619952.353	0.00	1396.11	100%
bl02	1816947.00	1816947.00	0.00	1619.19	100%
bl03	48188.46	17410.86	0.15	1563.85	92%
bl04	618874.23	-	6.00	1939.16	0%
bl05	549284.00	549284.00	0.00	286.99	100%
bl06	513261.15	513261.15	0.00	272.39	100%
bl07	6258.77	6258.77	0.00	321.18	100%
bl08	6195.46	6195.46	0.00	259.03	100%
bl09	8644450.54	-	40.77	1250.62	0%
bl10	12663345.69	-	108.00	1307.55	0%
bl11	703268.31	-	12.62	1636.53	0%
bl12	3975453.46	-	78.00	1179.22	0%
bl13	13862083.69	-	185.38	988.30	0%
bl14	12930081.08	-	166.00	1042.87	0%
bl15	10433908.15	-	207.69	952.98	0%
bl16	9034503.23	-	179.77	956.28	0%
bl17	15490690.62	-	28.85	1495.28	0%
bl18	13033481.62	-	18.31	1405.22	0%
bl19	11265651.62	-	222.92	948.11	0%
bl20	3430263.31	-	66.08	1307.79	0%
bl21	39022782.15	-	760.31	657.62	0%
bl22	35308732.92	-	670.15	678.55	0%
bl23	22559298.23	-	561.85	668.37	0%
bl24	28651286.08	-	714.15	669.78	0%

Para cada uma das 48 instâncias testadas, foram efetuados 13 testes. Os resultados obtidos pelo AG aplicado ao Problema Mono-objetivo são apresentados nas Tabelas 5.2 a 5.5. Nestas tabelas, a coluna **inst.** é referente à instância testada; já a coluna **fo_AG** apresenta a melhor solução encontrada pelo AG, enquanto a coluna **T_fo** apresenta o tempo computacional, em segundos, para encontrar tal solução. A coluna **menor viol.** mostra a menor violação encontrada dentre as 13 execuções,

Tabela 5.4: Resultados - AG - bsxx

Inst.	fo_AG	T_fo	Menor Viol.	Menor Tempo
bs01	1643776	1420.39	1	1420.39
bs02	1819311	1590.86	2	1510.38
bs03	16828	1553.74	0	1553.74
bs04	20333	2005.41	0	2005.41
bs05	545871	448.56	0	387.88
bs06	616322	611.67	0	503.20
bs07	7741	378.91	0	378.91
bs08	7922	755.56	0	481.55
bs09	10943511	1424.73	85	981.36
bs10	8853690	2001.47	22	1528.47
bs11	5118896	1133.77	101	1127.55
bs12	5026498	1862.74	99	1186.64
bs13	15532397	968.34	220	944.20
bs14	18649618	1476.61	278	968.16
bs15	10048469	1335.28	200	873.14
bs16	12147365	1550.74	242	924.14
bs17	16088264	1447.09	106	973.11
bs18	13814467	2578.94	40	1438.86
bs19	385556	1489.05	9	1471.59
bs20	5345443	2225.69	174	942.03
bs21	32464205	646.11	807	643.14
bs22	26742989	953.38	634	616.20
bs23	20517277	1275.61	681	646.28
bs24	18053353	1571.03	599	660.75

sendo que esta coluna apresenta a soma das violações em relação às restrições de capacidade em cada arco da rede. A coluna **menor tempo** exibe o menor tempo de execução dentre as 13 execuções de cada problema. A coluna **SM1** apresenta a média das soluções obtidas, incluindo as soluções infactíveis, que são penalizadas. A coluna **SM2** apresenta o valor médio somente das soluções factíveis encontradas para cada problema testado, ou seja, esse valor médio não inclui nenhuma solução penalizada. A coluna **violação média** mostra a média das violações das soluções obtidas para cada instância. A coluna **T_médio** exibe o tempo médio de execução e, finalmente, a coluna **% de fact.**, apresenta o percentual de factibilidade obtido pelas execuções do algoritmo para cada instância.

Com base nos resultados apresentados pelo AG, conclui-se que o método não obteve sucesso na obtenção de soluções factíveis para a grande maioria dos problemas testados. Em particular, o AG obteve sucesso para instâncias classificadas como menores por (Alvelos, 2005). Mesmo com o baixo nível de factibilidade, para as instâncias menores (*bl01* a *bl08* e *bs01* a *bs 08*), o método se mostrou eficiente $para 81,25\%$ dos casos, porém, considerando as demais instâncias (instâncias maiores), essa eficiência é reduzida para $27,08\%$, pois não foi obtida nenhuma solução factível para as instâncias classificadas como maiores por Alvelos (2005).

Tabela 5.5: Resultados - AG - bsxx - Médias

Inst.	SM 1	SM2	Violação Média	T_Médio	% de Fact.
bs01	1644089.54	-	1.00	1509.75	0%
bs02	2019675.92	-	6.62	1575.44	0%
bs03	16828.00	16828.00	0.00	1590.19	100%
bs04	43311.23	2234.56	0.46	2293.87	54%
bs05	564292.92	564292.92	0.00	425.95	100%
bs06	639481.77	639481.77	0.00	572.02	100%
bs07	8055.23	8055.23	0.00	585.09	100%
bs08	8173.69	8173.69	0.00	665.94	100%
bs09	11857074.46	-	103.77	1345.54	0%
bs10	9090333.00	-	27.00	1893.36	0%
bs11	6654282.62	-	131.69	1716.06	0%
bs12	6373908.00	-	125.92	1886.35	0%
bs13	18873803.23	-	283.00	1434.46	0%
bs14	21106952.38	-	327.31	1560.82	0%
bs15	12683059.69	-	252.69	1330.12	0%
bs16	15973677.08	-	318.54	1459.29	0%
bs17	17786453.38	-	160.92	1456.80	0%
bs18	13837696.77	-	48.77	2213.48	0%
bs19	515942.08	-	13.31	2347.57	0%
bs20	6671581.00	-	218.15	1486.70	0%
bs21	33957912.54	-	863.85	887.14	0%
bs22	31855539.85	-	803.08	1019.78	0%
bs23	24181296.69	-	803.23	1055.76	0%
bs24	22206472.62	-	737.31	986.84	0%

5.6.3 Resultados - ILS

Esta Seção apresenta os resultados obtidos pela aplicação da Metaheurística *Iterated Local Search* (ILS) ao mesmo conjunto de instâncias testadas anteriormente. A estrutura da implementação realizada está posta na Seção 5.4, nos termos do pseudo-código apresentado na Figura 4.4. As instâncias também foram divididas em grupos e, os parâmetros, obtidos empiricamente para cada grupo, estão apresentados na Tabela 5.6. Como citado anteriormente, o método de busca local utilizado no ILS foi o Método de Descida Randômica.

Tabela 5.6: Parâmetros - ILS

Grupo	vezesnivel	ilsmax	itermax	alfa
1	200	20	300	50000
2	100	15	200	50000
3	80	15	150	30000
4	60	15	100	30000
5	70	15	100	30000
6	50	15	80	30000

Na Tabela 5.6, a coluna **grupo** apresenta os grupos de instâncias, definidos

da mesma forma apresentada na Seção anterior. A coluna **vezesnivel** apresenta o número máximo de iterações sem melhora dentro de um mesmo nível de perturbação; a coluna **ilsmax** apresenta o número máximo de perturbações; e a coluna **alfa** exibe o parâmetro utilizado para penalizar soluções infactíveis quanto às restrições de capacidade.

Os resultados obtidos por meio do ILS são apresentados pelas Tabelas 5.7 - 5.10. As descrições das colunas dessas tabelas são as mesmas apresentadas na Seção 5.6.2.

Tabela 5.7: Resultados - ILS - blxx

Inst.	fo_ILS	T_fo	Menor Viol.	Menor Tempo
bl01	1621156	587.70	0	265.06
bl02	1816947	393.03	0	393.03
bl03	17588	538.05	0	233.59
bl04	70162	788.95	1	440.08
bl05	485914	554.28	0	294.13
bl06	436716	886.41	0	462.77
bl07	5819	804.83	0	324.94
bl08	5724	1713.09	0	304.16
bl09	6660275	821.83	6	188.39
bl10	7052268	485.34	4	324.03
bl11	69942	947.75	0	363.27
bl12	101387	1878.33	1	290.58
bl13	3497101	3578.91	0	787.41
bl14	2979970	3292.03	0	1379.53
bl15	36979	3192.80	0	864.64
bl16	31487	2139.94	0	783.45
bl17	13932878	302.83	2	252.75
bl18	11197717	560.55	0	323.02
bl19	267810	358.63	5	125.52
bl20	208950	641.34	3	207.94
bl21	6447287	3529.73	0	2820.81
bl22	5110330	4804.78	0	3072.67
bl23	61157	5014.73	0	2314.05
bl24	56852	5763.63	0	3148.84

O método ILS se mostrou melhor que o AG, pois obteve maior sucesso na busca de soluções factíveis para o problema, considerando todas as soluções. Para as instâncias menores (ou seja, *bl01* a *bl08* e *bs01* a *bs08*), o ILS foi um pouco menos eficiente quando comparado ao AG, obtendo 75% de factibilidade. Porém, considerando as demais instâncias, esse percentual de factibilidade se altera para $62,5\%$, o que é bem mais significativo quando comparado ao AG.

Tais resultados motivaram a combinação dos dois métodos na busca de uma metodologia mais eficiente na busca de boas soluções, conforme mostrado na Seção 5.6.5.

Tabela 5.8: Resultados - ILS - blxx - Médias

Inst.	SM 1	SM2	Violação Média	T_Médio	% de Fact.
bl01	1635942.46	1635942.46	0.00	535.08	100.00%
bl02	1862934.77	1816947.00	1.00	689.62	69.23%
bl03	171015.77	17791.50	3.08	510.16	15.38%
bl04	434616.85	-	8.31	930.15	0.00%
bl05	498061.85	498061.85	0.00	703.12	100.00%
bl06	446676.23	446676.23	0.00	940.77	100.00%
bl07	5918.77	5918.77	0.00	943.20	100.00%
bl08	5812.85	5812.85	0.00	651.48	100.00%
bl09	7002373.46	-	13.85	539.38	0.00%
bl10	7928074.08	-	25.92	782.61	0.00%
bl11	181944.15	70482.00	3.69	727.96	30.77%
bl12	335937.31	-	8.69	603.16	0.00%
bl13	3696510.77	3552478.75	4.15	1687.64	30.77%
bl14	3063462.23	3063043.30	0.38	2422.97	76.92%
bl15	112689.62	38548.44	2.46	1801.03	69.23%
bl16	51617.46	32790.25	0.62	1845.67	61.54%
bl17	14060567.85	-	8.08	504.27	0.00%
bl18	11195599.62	11239480.00	2.85	524.56	15.38%
bl19	947341.08	-	27.62	616.75	0.00%
bl20	473748.23	-	11.77	489.04	0.00%
bl21	6572153.54	6544378.00	0.77	4234.15	69.23%
bl22	5345333.15	5306255.71	0.85	5374.55	53.85%
bl23	107122.85	63318.78	1.46	3737.09	69.23%
bl24	84425.77	58451.00	0.85	5315.78	61.54%

5.6.4 Comparação entre os resultados usando AG e os resultados via ILS

As seções 5.6.2 e 5.6.3 mostram um melhor desempenho da metaheurística ILS quando comparada ao algoritmo populacional AG. Enquanto o AG encontrou soluções factíveis para $27,8\%$ das instâncias testadas, o ILS foi capaz de encontrar soluções factíveis para $62,5\%$ dos problemas teste. Considerando apenas os problemas teste nos quais ambos encontraram soluções factíveis, o AG superou o ILS em 25%, enquanto o ILS foi melhor em $58,3\%$ dos casos e em $16,7\%$ dos casos houve empate. A única instância para a qual o AG encontrou solução factível, enquanto o ILS não obteve factibilidade, foi a instância $bs04$.

Com base nos resultados obtidos, conclui-se que um AG, com as características implementadas neste trabalho, possui um desempenho inferior, quando comparado com uma metaheurística local, em particular com o ILS.

Na próxima Seção serão apresentados os resultados dos dois algoritmos trabalhando em conjunto, sendo que o método ILS foi aplicado à solução retornada pelo AG, ou seja, primeiro o AG é aplicado, com os mesmos parâmetros descritos na Seção 5.6.2, e, em seguida, o ILS é aplicado.

Tabela 5.9: Resultados - ILS - bsxx

Inst.	fo_ILS	T_fo	Menor Viol.	Menor Tempo
bs01	1643776	366.24	1	328.20
bs02	1845273	892.14	2	283.83
bs03	16828	357.80	0	357.80
bs04	69975	449.24	1	425.05
bs05	520055	830.55	0	325.42
bs06	543343	1408.83	0	398.05
bs07	7668	565.20	0	359.00
bs08	6936	541.83	0	498.30
bs09	6709372	518.77	4	242.06
bs10	7484448	511.92	2	351.77
bs11	398091	618.06	11	148.38
bs12	648326	834.00	19	176.89
bs13	4296742	1909.77	4	828.09
bs14	3742849	2615.08	0	975.11
bs15	41916	1269.39	0	1256.00
bs16	96864	2965.48	2	965.00
bs17	11993703	666.63	0	406.00
bs18	11468676	631.72	5	522.11
bs19	109999	521.97	0	227.05
bs20	327149	966.08	7	358.13
bs21	6379918	3821.84	0	2629.11
bs22	5676742	3184.94	0	2285.08
bs23	63948	4015.34	0	1709.08
bs24	60015	3796.84	0	2238.89

5.6.5 Resultados - AG + ILS

Esta Seção apresenta os resultados obtidos por meio da aplicação conjunta dos dois algoritmos, qual seja, AG e ILS. Os parâmetros utilizados no AG foram os mesmos mostrados na Tabela 5.1, porém os parâmetros utilizados no ILS foram alterados, fato este justificado pelo fato do ILS utilizar uma solução inicial retornada pelo AG. Assim, esta solução inicial é, normalmente, de qualidade superior quando comparada a outra que poderia ser gerada de forma totalmente aleatória, como foi feito para obter os resultados apresentados na Seção 5.6.3.

A Tabela 5.11 apresenta os parâmetros utilizados no ILS, após a aplicação do AG. As Tabelas 5.12 - 5.15 apresentam os resultados obtidos por meio de AG + ILS. Novamente, foram realizados 13 testes para cada uma das 48 instâncias.

O método AG + ILS obteve sucesso na determinação de soluções factíveis em $93,75\%$ dos problemas testados. Por este motivo, conclui-se que o mesmo é superior quando comparado aos métodos AG e ILS, na situação em que estes foram aplicados separadamente. Além disso, em $70,8\%$ dos casos, foram obtidas soluções factíveis para o problema em 100% das execuções; em 75% dos problemas testados, foram obtidas soluções factíveis em mais de 90% das execuções; e, em $77,1\%$ das instâncias testadas, foram obtidas soluções factíveis em mais de 80% das execuções.

Tabela 5.10: Resultados - ILS - bsxx - Médias

Inst.	SM 1	SM2	Violação Média	T_Médio	% de Fact.
bs01	2012409.38	-	8.31	553.72	0.00%
bs02	2208193.15	-	10.77	564.00	0.00%
bs03	151583.15	16828.00	2.69	606.75	30.77%
bs04	300799.15	-	5.62	1076.67	0.00%
bs05	541679.77	541679.77	0.00	815.13	100.00%
bs06	575700.46	575700.46	0.00	1012.81	100.00%
bs07	7809.15	7809.15	0.00	701.20	100.00%
bs08	7478.08	7478.08	0.00	795.87	100.00%
bs09	7360744.85	-	21.08	399.70	0.00%
bs10	7894159.31	-	9.77	648.62	0.00%
bs11	1252347.08	-	39.38	397.97	0.00%
bs12	1167687.77	-	36.31	535.55	0.00%
bs13	4396973.85	-	8.77	1904.12	0.00%
bs14	4147826.77	3742849.00	9.62	2369.72	7.69%
bs15	204573.00	42941.00	5.38	1857.90	15.38%
bs16	316415.54	-	9.23	2228.09	0.00%
bs17	12354991.92	11993703.00	5.00	732.71	7.69%
bs18	11720005.69	-	11.08	797.02	0.00%
bs19	203145.23	110249.50	3.08	676.74	15.38%
bs20	588159.00	-	15.54	921.17	0.00%
bs21	6664364.31	6404416.00	5.15	4831.94	15.38%
bs22	5890862.69	5805032.40	2.38	4706.64	38.46%
bs23	150597.69	65755.67	2.77	3919.15	23.08%
bs24	71792.92	62460.89	0.31	4496.11	69.23%

Tabela 5.11: Parâmetros - ILS - AG + ILS

Grupo	Alfa	vezesnivel	ilsmax	itermax2
Grupo 1	100000	200	15	200
Grupo 2	100000	150	15	100
Grupo 3	50000	100	15	200
Grupo 4	50000	60	15	100
Grupo 5	50000	50	15	150
Grupo 6	40000	50	15	50

5.6.6 Comparações com Métodos Exatos

Para finalizar esta Seção, a seguir é apresentada um comparação entre a implementação AG+ILS e a solução via métodos exatos apresentada em Alvelos (2005), em que são msotrados resultados para a mesma classe de instâncias aqui tratada.

Os melhores resultados obtidos foram comparados com os resultados obtidos por Alvelos (2005). Alvelos (2005) desenvolveu um algoritmo de partição e geração de colunas baseado em uma decomposição de mochila modificado e um algoritmo de partição e geração de colunas com cortes, baseado numa decomposição por caminhos. Segundo (Alvelos, 2005), os resultados dos testes computacionais foram

Tabela 5.12: Resultados - AG + ILS - blxx

Inst.	fo_AG + ILS	T_fo	Menor Viol.	Menor Tempo
bl01	1615947	1403.47	0	1403.47
bl02	1816947	1655.56	0	1655.56
bl03	17340	1571.95	0	1571.95
bl04	220022	2265.64	2	1924.28
bl05	490314	721.30	0	496.97
bl06	434423	937.94	0	498.14
bl07	5816	698.69	0	476.42
bl08	5724	624.67	0	423.17
bl09	6440345	1737.16	0	1662.83
bl10	6585522	3261.77	0	1789.13
bl11	69276	2376.39	0	2176.13
bl12	67370	2429.39	0	1604.11
bl13	3294965	2755.03	0	1783.48
bl14	2721069	3952.11	0	2415.89
bl15	35583	2707.38	0	1742.67
bl16	29940	2765.39	0	1594.30
bl17	13341288	2421.66	0	1953.61
bl18	10739354	2085.66	0	1796.16
bl19	112114	2348.14	0	1656.75
bl20	112742	1948.86	0	1948.86
bl21	6123669	2893.22	0	1589.77
bl22	4704714	4104.31	0	2363.11
bl23	60134	2429.99	0	1661.56
bl24	53448	3809.09	0	2135.77

surpreendentes, pois o software CPLEX 8.1 obteve, em geral, resultados muito melhores do que as duas decomposições. Neste trabalho, os resultados obtidos por meio do método AG + ILS, descrito anteriormente, são comparados aos resultados obtidos pelas duas decomposições propostas por Alvelos (2005) e também com os resultados obtidos por meio do software CPLEX 8.1. Os resultados apresentados por Alvelos (2005) foram divididos em *Instâncias de Menores Dimensões* e *Instâncias de Maiores Dimensões*.

Instâncias de Menores Dimensões

Na Tabela 5.16, a coluna *fo* apresenta os resultados obtidos por meio do AG + ILS; a coluna *f* apresenta os resultados obtidos por (Alvelos, 2005). As colunas P, K e O apresentam os tempos computacionais dos algoritmos baseado em decomposição por caminhos, baseado em uma decomposição de mochila e de uma abordagem via CPLEX 8.1, respectivamente, conforme (Alvelos, 2005). A coluna *fo* apresenta o valor obtido por meio do algoritmo proposto neste trabalho. O desvio de cada solução em relação aos resultados de (Alvelos, 2005) é encontrado pela fórmula $(fo - f^*)/f^*$ na coluna GAP, ou seja, apresenta, em percentual, a distância relativa da solução obtida pelo algoritmo proposto em relação à solução ótima. . Assim, a Tabela 5.16 apresenta o símbolo "**" em lugar de apresentar o tempo computacional.

Tabela 5.13: Resultados - AG + ILS - blxx - Médias

Inst.	SM 1	SM2	Violação Média	T_Médio	% de Fact.
bl01	1619930.35	1619930.353	0.00	1485.62	100.00%
bl02	1816947.00	1816947.00	0.00	1734.81	100.00%
bl03	17361.46	17361.46	0.00	1664.99	100.00%
bl04	588143.69	-	5.69	2077.67	0.00%
bl05	496975.23	496975.23	0.00	706.17	100.00%
bl06	451059.92	451059.92	0.00	716.47	100.00%
bl07	5900.31	5900.31	0.00	589.89	100.00%
bl08	5830.69	5830.69	0.00	536.21	100.00%
bl09	6490474.85	6440345.00	2.15	1928.24	7.69%
bl10	6693486.85	6685345.42	0.08	2504.06	92.31%
bl11	73601.38	69655.33	0.08	2402.49	92.31%
bl12	68401.08	68401.08	0.00	1989.51	100.00%
bl13	3334710.15	3334710.15	0.00	2177.38	100.00%
bl14	2776221.31	2776221.31	0.00	3042.55	100.00%
bl15	36260.31	36260.31	0.00	2234.38	100.00%
bl16	30565.31	30565.31	0.00	2301.88	100.00%
bl17	13453933.62	13433208.56	0.31	2206.26	69.23%
bl18	10810917.08	10810917.08	0.00	1951.81	100.00%
bl19	144697.54	113163.60	0.62	1974.72	76.92%
bl20	157402.31	114622.45	0.85	2310.73	84.62%
bl21	6254214.92	6254214.92	0.00	2285.97	100.00%
bl22	4889032.77	4889032.77	0.00	3390.70	100.00%
bl23	60907.38	60907.38	0.00	2175.27	100.00%
bl24	54950.23	54950.23	0.00	2983.50	100.00%

O símbolo "-", utilizado em algumas linhas, indica que o método proposto por este trabalho não encontrou solução factível para aquela instância específica. Essa mesma simbologia será utilizada na Tabela 5.17.

Comparando-se os resultados apresentados, verifica-se que o desvio médio da solução ótima para o conjunto de instâncias de menores dimensões obtido pela aplicação do método AG+ILS é $2,11\%$. Em três instâncias (*bl04*, *bs01* e *bs02*) não foram encontradas, pelo método proposto, soluções factíveis. O algoritmo proposto por (Alvelos, 2005), em sua versão baseada em decomposição por caminhos, não obteve, também, solução ótima para a instância *bl04*. Quanto a esta instância, cabe observar que o tempo computacional para a obtenção de solução ótima, tanto na versão via decomposição de mochila quanto na abordagem via CPLEX é fortemente superior às demais instâncias do mesmo grupo, conforme pode ser observado na Tabela 5.16.

Instâncias de Maiores Dimensões

A Tabela 5.17 apresenta comparações para as instâncias ditas de maiores dimensões, segundo definido por Alvelos (2005). O símbolo "*" indica que (Alvelos, 2005) não encontrou a solução ótima do problema em um tempo máximo de uma hora para a instância em questão. Já o símbolo "**" indica que (Alvelos, 2005) não

Tabela 5.14: Resultados - AG + ILS - bsxx

Inst.	fo_AG + ILS	T_fo	Menor Viol.	Menor Tempo
bs01	1643776	1498.95	1	1498.95
bs02	1819311	1694.31	2	1614.52
bs03	16828	1672.44	0	1672.44
bs04	20333	2155.16	0	2155.16
bs05	509086	827.53	0	590.78
bs06	529544	1251.63	0	932.86
bs07	7439	1108.45	0	581.77
bs08	6867	1115.75	0	740.13
bs09	6355229	2251.64	0	1676.98
bs10	7280515	3341.94	0	2304.36
bs11	66345	4780.74	0	2129.48
bs12	73394	2922.64	0	2357.78
bs13	3762485	2943.64	0	1650.99
bs14	3253351	3480.19	0	2357.13
bs15	39996	3575.52	0	1681.38
bs16	34581	3296.88	0	2292.94
bs17	11642335	2733.59	0	1417.44
bs18	10866784	2669.45	0	2499.00
bs19	107690	3733.77	0	1925.19
bs20	111435	7095.59	0	1889.81
bs21	5919991	4581.55	0	1973.49
bs22	5087721	8563.53	0	2833.20
bs23	61257	3244.92	0	2113.25
bs24	56705	3915.33	0	2471.30

encontrou solução factível em um tempo máximo de uma hora. O símbolo "***" indica que o problema não foi resolvido em um tempo máximo de uma hora. Caso alguma instância apresente algum tempo computacional em alguma das colunas P, K ou O, isso significa que o algoritmo correspondente, proposto por Alvelos (2005), encontrou a solução ótima para a instância associada.

Comparando-se os resultados apresentados, verifica-se que o desvio médio da solução ótima para o conjunto de instâncias de maiores dimensões obtido pela aplicação do método AG+ILS é $5,26\%$, resultado superior, portanto, ao encontrado para instâncias de menores dimensões. Em todas as instâncias foram encontradas, pelo método proposto, soluções factíveis. O algoritmo proposto por (Alvelos, 2005), no entanto, não obteve, solução factível para a instância *bl10* em qualquer uma de suas três formulações, conforme Tabela 5.17. Desse modo, não é apresentado, para essa instância, o valor de GAP, ou seja, de desvio da solução ótima.

Além disso, em 14 das 32 instâncias desse grupo, a decomposição por caminhos não encontrou solução ótima em um tempo computacional máximo de uma hora; a decomposição via mochila também não obteve solução ótima em 3 das 32 instâncias, enquanto a solução via CPLEX 8.1 não obteve solução ótima também em 8 das 32 instâncias.

A decomposição por caminhos não obteve solução factível em um tempo máximo

de uma hora em 12 instâncias; a decomposição por mochila encontrou não encontrou solução factível em 3 das instâncias, e a solução via CPLEX 8.1 não obteve solução factível em uma única instância. A decomposição via mochila não solucionou o problema em 23 das instâncias propostas.

Em 8 instâncias, os resultados apresentados na coluna f não correspondem a valores ótimos, mas, sim, aos melhores valores encontrados em uma hora de execução computacional, posto que nenhuma das implementações via método exato apresentadas em Alvelos (2005) logrou encontrar a solução ótima para estas instâncias.

Tabela 5.15: Resultados - AG + ILS - bsxx - Médias

Inst.	SM 1	SM2	Violação Média	T_Médio	% de Fact.
bs01	1644089.54	-	1.00	1588.94	0.00%
bs02	1997077.08	-	6.15	1685.33	0.00%
bs03	16828.00	16828.00	0.00	1709.61	100.00%
bs04	43276.31	20391.71	0.46	2455.45	53.85%
bs05	525874.08	525874.08	0.00	750.66	100.00%
bs06	559076.77	559076.77	0.00	1161.54	100.00%
bs07	7606.69	7606.69	0.00	942.76	100.00%
bs08	7290.85	7290.85	0.00	1231.25	100.00%
bs09	6434622.85	6434622.85	0.00	2402.86	100.00%
bs10	7389849.08	7280515	1.85	3293.16	7.69%
bs11	67259.69	67259.69	0.00	3625.94	100.00%
bs12	206079.23	74891.75	2.62	3987.66	30.77%
bs13	3850375.54	3850375.54	0.00	3477.28	100.00%
bs14	3354986.23	3354986.23	0.00	4833.31	100.00%
bs15	40788.38	40788.38	0.00	2919.30	100.00%
bs16	35093.00	35093.00	0.00	4574.48	100.00%
bs17	11776802.38	11776802.38	0.00	2750.58	100.00%
bs18	11063534.69	10956143.71	1.08	3959.41	53.85%
bs19	108704.85	108704.85	0.00	3423.03	100.00%
bs20	138010.31	112404.90	0.85	4272.38	76.92%
bs21	6116618.69	6116618.69	0.00	3835.80	100.00%
bs22	5224818.08	5224818.08	0.00	5137.40	100.00%
bs23	62282.54	62282.54	0.00	3846.09	100.00%
bs24	58116.54	58116.54	0.00	4168.33	100.00%

Tabela 5.16: AG + ILS x Exatos - Instâncias Menores

Inst.	f*	P	K	O	fo	GAP
bl01	1615947	0.7	8.3	0.6	1615947	0.00%
bl02	1816947	1.7	13.1	1.5	1816947	0.00%
bl03	17340	8	4.8	4.6	17340	0.00%
bl04	21370	**	1286.8	54.6	-	-
bl05	474782	0.1	7.6	0.3	490314	3.27%
bl06	411480	0	1.7	0.3	434423	5.58%
bl07	5751	0	3.1	0.3	5816	1.13%
bl08	5688	0	1.3	0.3	5724	0.63%
bs01	1639862	0.3	1.4	0.1	-	-
bs02	1702368	1.6	6	1.9	-	-
bs03	16828	0.1	2.8	0.1	16828	0.00%
bs04	20213	53.8	49.7	2	20333	0.59%
bs05	500870	1.5	2.7	2.4	509086	1.64%
bs06	502151	1.6	4.9	3.7	529544	5.46%
bs07	7223	4.6	9.1	8.1	7439	2.99%
bs08	6471	0.8	1.8	0.8	6867	6.12%

5.7 Conclusões

Dos resultados apresentados neste capítulo, conclui-se que o melhor método utilizado consiste na aplicação da metaheurística *Iterated Local Search* (ILS) após a aplicação do Algoritmo Genético (AG). Os piores resultados obtidos pelas metaheurísticas utilizadas neste trabalho foram encontrados por meio do algoritmo populacional AG. Com isso, concluiu-se que um algoritmo populacional, com as características apresentadas neste trabalho, não é eficiente quando utilizado sem nenhum outro método de otimização. Mesmo com a utilização do Método de Descida Randômica método de busca local no AG, este não se mostrou eficiente, sendo necessário a adição de um método de pós-otimização, em particular o ILS, para a melhoria dos resultados obtidos.

Com base nas Tabelas 5.16 e 5.17, conclui-se que, para instâncias menores, os algoritmos propostos por (Alvelos, 2005) foram mais eficientes quando comparados à metodologia AG + ILS. Porém, para instâncias maiores, é possível perceber que o método proposto neste trabalho superou as duas decomposições propostos por (Alvelos, 2005), encontrando soluções próximas das soluções ótimas, ou das melhores soluções soluções conhecidas. A metodologia proposta neste trabalho foi pior apenas quando comparada à resolução via CPLEX, com a formulação original do problema, mas é muito importante ressaltar que, para a instância *bl10*, foi encontrada uma solução factível. Não pode-se afirmar que a solução para tal instância seja a ótima, pois neste trabalho a abordagem é feita via métodos heurísticos.

Tabela 5.17: AG + ILS x Exatos - Instâncias Maiores

Inst.	f*	P	K	O	fo	GAP
bl09	6261671	**	***	1087.2	6440345	2.85%
bl10	**	**	***	**	6585522	
bl11	69018	470.8	***	17.5	69276	0.37%
bl12	65902	*	***	182.1	67370	2.23%
bl13	3132695	*	1766.6	176.5	3294965	5.18%
bl14	2433011	303.6	*	117.9	2721069	11.84%
bl15	34274	318.5	*	96.3	35583	3.82%
bl16	28074	39.1	1359.4	57.1	29940	6.65%
bl17	13190922	*	***	2881.7	13341288	1.14%
bl18	10496120	**	***	1448.4	10739354	2.32%
bl19	*109556	**	***	*	112114	2.33%
bl20	*111604	**	***	*	112742	1.02%
bl21	*5800149	*	***	*	6123669	5.58%
bl22	4209266	*	***	389.1	4704714	11.77%
bl23	56856	*	***	2250.2	60134	5.77%
bl24	47964	*	***	250.8	53448	11.43%
bs09	6287195	*	***	665.3	6355229	1.08%
bs10	7072735	**	***	643.2	7280515	2.94%
bs11	*65168	**	***	*	66345	1.81%
bs12	71483	**	***	2235.8	73394	2.67%
bs13	3605397	*	*	532.2	3762485	4.36%
bs14	*2872664	*	**	*	3253351	13.25%
bs15	38533	120.7	290.2	99	39996	3.80%
bs16	31124	337.8	630	95.3	34581	11.11%
bs17	*11447995	**	***	*	11642335	1.70%
bs18	10486796	**	***	746.2	10866784	3.62%
bs19	106142	*	***	913	107690	1.46%
bs20	107712	**	***	896.1	111435	3.46%
bs21	*5562469	**	***	*	5919991	6.43%
bs22	*4487045	*	***	*	5087721	13.39%
bs23	57548	*	**	3301.3	61257	6.45%
bs24	50980	*	***	2919.6	56705	11.23%

A principal contribuição apresentada neste Capítulo foi a proposta de uma nova metodologia, baseada em métodos heurísticos, e que supera, para as instâncias maiores, dois métodos de decomposição. Além disso, uma importante contribuição foi a obtenção de uma solução factível para a instância *bl10*.

Capítulo 6

Metodologia Multiobjetivo

Este capítulo apresenta a solução do Problema Multiobjetivo de Fluxo Multiproduto em sua versão descrita na Seção 2.5 e representado pela expressão (2.5), bem como os resultados encontrados a partir da adoção desta metodologia. É apresentado o algoritmo utilizado para solucionar o problema, inicialmente discutido na Seção 4.7 e os resultados resultantes da aplicação deste algoritmo de solução.

6.1 Introdução

Para a solução do Problema Multiobjetivo de Fluxo Multiproduto (PMFM), é utilizado o método *Nondominated Sorting Genetic Algorithm* II, proposto por Deb et al. (2002). Este método é uma nova formulação do método *Nondominated Sorting Genetic Algorithm*, proposto por Srinivas e Deb (1994), conforme discutido na Seção 4.6.

O método de solução do problema multiobjetivo proposto é constituído por duas fases, sendo, a primeira, um Algoritmo Genético (AG) mono-objetivo, cujo objetivo é o de encontrar o maior número possível de soluções iniciais factíveis para a segunda fase, constituída pelo NSGA II.

O uso de um AG na Fase 1 se dá pela dificuldade encontrada em determinar soluções iniciais factíveis e também pelo AG atingir uma diversidade satisfatória em relação às soluções encontradas.

Além desses dois métodos, foram implementados dois métodos de melhoramento do conjunto de soluções candidatas a ótimas de Pareto, sendo estes métodos baseados em estruturas de vizinhança e diversificações.

As seções seguintes deste capítulo apresentam as aplicações destes algoritmos ao PMFM.

6.2 Algoritmo Genético Aplicado ao Problema Multiobjetivo de Fluxo Multiproduto - Fase 1

Nesta fase, foi proposto um algoritmo genético mono-objetivo, cujo intuito é o de minimizar o somatório das violações das restrições de capacidade, tendo a função de aptidão igual à soma das violações. O algoritmo é semelhante ao apresentado na Seção 5.5.

Geração da população inicial: análoga à apresentada na Seção 5.2. Aos indivíduos da população inicial é aplicado um processo de busca local, em particular o Método da Descida Randômica, buscando reduzir, ou, até mesmo, anular, o somatório das violações (f_{apt}). A diferença deste método de busca local em relação ao método aplicado ao PFM é a definição da função de avaliação, que agora consiste na função f_{apt}.

Função de aptidão: a função de avaliação utilizada é:

$$f_{apt} = \sum_{(i,j)\in\mathcal{A}} \psi_{ij} \tag{6.1}$$

Se o arco (i, j) não estiver violado, então $\psi_{ij} = 0$; e, em caso contrário, tem-se que

$$\psi_{ij} = (\sum_{k=1}^{p} x_{ij}^{k}) - u_{ij} \tag{6.2}$$

Representação da solução: análoga à apresentada na Seção 5.5.

Representação da População: a população é representada por meio de uma estrutura que contém os campos que armazenam a solução e o somatório das violações.

Operador de *Crossover*: foi utilizado o *Crossover* Uniforme, conforme descrito na Seção 5.5.

Seleção de indivíduos: o processo de escolha de pais para o operador de *Crossover* é aleatório, com probabilidade uniforme. A eliminação de indivíduos de uma geração para outra é feita de maneira elitista, sendo escolhidos os indivíduos com melhor valor da função f_{apt}. No caso, foram mantidas as $nind/2$ melhores soluções para as gerações seguintes, para $nind$ igual ao número de indivíduos da população.

Mutação: após a geração do filho pelo processo de *Crossover*, o mesmo pode sofrer dois tipos de mutação, com dada probabilidade, chamada de *taxa de mutação*. O primeiro processo é a mutação aleatória, que altera os valores de determinado número de colunas ($genes\_mutados$) na solução, ou seja, os fluxos de produtos escolhidos aleatoriamente são trocados. Basicamente, a mutação aleatória escolhe produtos aleatoriamente e troca as rotas desses produtos. Caso o filho não sofra mutação aleatória, ele sofrerá um outro processo de mutação, de busca local, que consiste na aplicação do Método da Descida Randômica tendo como função de avaliação a função f_{apt}.

Além da estrutura que armazena os indivíduos da população, também foi criada outra estrutura, denominada $pop\_inicial$, cujo objetivo é armazenar as soluções que serão inseridas como soluções iniciais na Fase 2. Cada solução gerada, seja pela heurística construtiva ou após a mutação aleatória ou não-aleatória, após o operador de *Crossover*, poderá ser inserida nessa lista. Se a solução gerada for factível, então a

mesma é copiada para a lista de soluções iniciais do NSGA II; se a mesma for infactível, ela ainda poderá ser inserida nessa lista, dada certa probabilidade, chamada neste trabalho de $prob\_infact$. Isso foi feito pelo fato do NSGA II também ser evolutivo e elitista e, em alguns casos, o AG poderá não encontrar soluções factíveis e mesmo assim ainda existem chances do NSGA II obter melhores resultados.

O critério de parada do AG é o número máximo de gerações. Porém, se terminarem as gerações sem ainda alcançar o número de soluções necessárias para a Fase 2, o processo continua até a obtenção de um número de indivíduos igual ao tamanho da população utilizado pela Fase 2.

6.3 NSGA II Aplicado ao PMFM - Fase 2

Em primeiro lugar, a estrutura $pop\_inicial$ armazena as soluções iniciais do NSGA II. Os registros dessa estrutura contém um campo para armazenar a matriz solução (x), outro para armazenar o $front$, outro para armazenar o valor da *crowding distance* da solução (cd), um campo para armazenar a função de custo (f) e outro para armazenar o valor da função de congestionamento (Φ). Aos indivíduos dessa estrutura, são aplicados os procedimentos $FastNonDominatedSort()$ e $CrowdingDistance()$, com o objetivo de eliminar algumas soluções, de forma que a população do NSGA II contenha apenas $nind2$ elementos, sendo $nind2$ o número de indivíduos do NSGA II. A estrutura que armazena a população do NSGA II é chamada de pop_nsga e possui os mesmos campos utilizados na estrutura $pop\_inicial$. Como a estrutura $pop\_inicial$ pode conter soluções infactíveis, ao ser aplicado o procedimento $FastNonDominatedSort()$, estas soluções são penalizadas, multiplicando-se a soma das violações em relação às restrições de capacidade por um parâmetro, denominado $alfa\_front$

O primeiro objetivo do NSGA II é minimizar a função de custo do PFM (f), sendo esta função dada como:

$$f(x) = \mathbf{Tr}(cx) \tag{6.3}$$

O segundo objetivo consiste na função de congestionamento da rede. Foram utilizados, para definição dessa função, os mesmos valores de diferenciais e os mesmos intervalos propostos por (Fortz e Thorup, 2000a).

$$\Phi = \sum_{(i,j)\in\mathcal{A}} \Phi_{ij}(l_{ij}) \tag{6.4}$$

Se o fluxo do arco (i, j) for nulo, então $l_{ij} = 0$ e $\Phi(l_{ij}) = 0$, para:

$$\Phi_{ij}(l_{ij}) = \begin{cases} l_{ij}, \text{ se } 0 \leq l_{ij} < (1/3)u_{ij}; \\ 3l_{ij} - (2/3)u_{ij}, \text{ se } (1/3)u_{ij} \leq l_{ij} < (2/3)u_{ij}; \\ 10l_{ij} - (16/3)u_{ij}, \text{se } (2/3)u_{ij} \leq l_{ij} < (9/10)u_{ij}; \\ 70l_{ij} - (178/3)u_{ij}, \text{ se } (9/10)u_{ij} \leq l_{ij} < u_{ij}; \\ 500l_{ij} - (1468/3)u_{ij}, \text{ se } u_{ij} \leq l_{ij} < (11/10)u_{ij}; \\ 5000l_{ij} - (16381/3)u_{ij}, \text{ se } l_{ij} \geq (11/10)u_{ij}. \end{cases} \tag{6.5}$$

A seguir, são apresentados detalhes da implementação do NSGA II para a solução do Problema Multiobjetivo de Fluxo Multiproduto.

Crossover: foi utilizado o operador *crossover* uniforme, conforme Seção 5.5.

Mutação: a mutação, durante a Fase 2, é feita somente de forma aleatória, sem aplicação de busca local. Este processo altera os valores de determinado número de colunas ($genes\_mutados$) na solução, ou seja, os fluxos de produtos escolhidos aleatoriamente são trocados.

Seleção de indivíduos para recombinação: O processo de escolha de pais para o operador de *crossover* é aleatório, com probabilidade uniforme.

Elitismo: característico do próprio NSGA II, por meio da *crowding distance*.

6.4 Melhoramento do Pareto Baseado em Estrutura de Vizinhança

É um método baseado na geração de vizinhos de cada solução, com o intuito de obter mais soluções candidatas, ou de encontrar soluções que dominem as soluções candidatas encontradas pelo NSGA II.

Este procedimento, para cada solução s pertencente ao conjunto ótimo de Pareto, gera vizinhos a partir de um movimento que consiste na troca de uma coluna na matriz solução.

O procedimento é baseado no Método da Descida Randômica e é descrito a seguir. Dada uma solução s, candidata a ótima de Pareto, é gerada uma nova solução s', definida como vizinha de s e gerada a partir da alteração de uma coluna em s, que corresponde à troca do fluxo de um produto, escolhido aleatoriamente. Caso s' seja factível, então ela é armazenada em uma outra lista de soluções; caso ela seja infactível, é incrementado um contador, chamado neste trabalho de $iter2$. No caso de s' ser factível, então o contador $iter2$ é zerado. A cada vizinho gerado, é feita uma verificação para eliminar a geração de soluções repetidas, e o contador $iter2$ não é zerado no caso de uma solução repetida, somente no caso de infactibilidade. O processo aplicado a cada solução é interrompido até $itermax2$, parâmetro que representa o número máximo de iterações sem ser encontrada uma solução factível vizinha de s. Em nenhum momento é efetuada a troca de s por s', pois o objetivo é encontrar o maior número de soluções próxima de s. Somente é efetuada a troca de conjuntos de soluções candidatas após serem aplicados os procedimentos do NSGA II. Para evitar um esforço computacional muito elevado, o número de vizinhos de s é limitado por um parâmetro denominado max_neigh.

Todas as soluções geradas por este procedimento são armazenadas em uma lista. Nesta lista também são copiadas todas as soluções candidatas, a partir das quais as novas soluções foram geradas. Em seguida, o procedimento $FastNondominatedSort()$ e o cálculo da *crowding distance* são aplicados às soluções dessa lista. Então, uma nova fronteira de soluções candidatas é gerada. Para essa nova fronteira e para as seguintes, o processo de melhoramento do Pareto baseado em estrutura de vizinhança é repetido até um número máximo de vezes, denominado neste trabalho de $trial$.

6.5 Melhoramento do Pareto Baseado em Estrutura de Vizinhança e Diversificações

Foi implementado outro método para melhoramento do conjunto de soluções candidatas a ótimas de Pareto pelo NSGA II. Este outro método é baseado na geração de vizinhos e em diversificações, a fim de explorar melhor o espaço de busca.

Este procedimento, para cada solução s pertencente ao conjunto ótimo de Pareto, gera vizinhos a partir de um movimento na solução que consiste na troca da rota do fluxo de um produto escolhido aleatoriamente, ou seja, altera-se os valores de uma coluna escolhida aleatoriamente na matriz solução.

Os movimentos são apenas realizados em s e essa solução não é trocada durante a execução do método, pois o intuito é, inicialmente, encontrar o máximo número de vizinhos dessa solução. A fim de evitar um esforço excessivo, esse número máximo é limitado por um parâmetro denominado max_neigh. Após a geração de um vizinho s', gerado a partir do movimento mencionado, é verificada a factibilidade dessa solução e, no caso da mesma ser factível, ela é copiada para uma outra lista, para onde também são copiadas as soluções Pareto-Ótimas já encontradas. Além de ser feito o teste da factibilidade da solução (em relação às restrições de capacidade), ela também é comparada às demais soluções já encontradas, e, no caso de ser uma solução duplicada, ela não é copiada para essa lista. No caso de s' ser infactível, é então incrementado um valor denominado $iter3$ e o processo é interrompido quando se chega ao número máximo de iterações sem ser encontrada solução factível, denominado $vezes_nivel2$, ou até que se obtenha o número máximo de vizinhos factíveis, denominado max_neigh.

Após essa execução, é incrementado um parâmetro denominado $nivel$, responsável pela diversificação. O *nível* 1 consiste em trocar duas colunas na matriz solução, enquanto o *nível* 2 consiste na troca de 3 colunas, e assim por diante. Esse parâmetro é responsável pela diversificação no algoritmo e a quantidade máxima de diversificações é chamada de $nivel\_max$.

Todos os vizinhos gerados por este procedimento, assim como as soluções ótimas de Pareto já obtidas, são copiadas para uma lista e, a essas soluções, é aplicado o procedimento $FastNondominatedSort()$, presente no NSGA II, e o cálculo da *crowding distance* também é realizado. Em seguida, um novo conjunto de soluções candidatas é gerado e a este conjunto será novamente aplicado o procedimento descrito nesta Seção. O número de repetições desse processo é limitado por uma parâmetro denominado $trial2$.

6.6 Testes Computacionais para a solução do PMFM

Foram testadas as mesmas instâncias utilizadas no problema mono-objetivo, porém foram consideradas as duas funções citadas anteriormente, quais sejam, a primeira função objetivo é dada pela expressão (6.3); a segunda função objetivo é dada pela expressão (6.4). O algoritmo foi executado uma vez para cada instância. A implementação também foi feita por meio da linguagem C, sendo utilizado o com-

pilador *Borland C++ Builder 5.0*. Os testes foram realizados em computadores *Intel Core 2 Duo*, com 3 GHz e 2 GB de memória *RAM*, sob o Sistema Operacional *Windows XP SP3*.

Tabela 6.1: Parâmetros - AG (soluções iniciais) - Fase 1

Grupo	nind	max_ger	filhos	t_mutação	genes_mut	itermax	p_inf	nind2
Grupo 1	300	150	nind/2	30%	10	300	5%	350
Grupo 2	140	100	nind/2	30%	10	100	5%	150
Grupo 3	300	200	nind/2	30%	40	180	5%	250
Grupo 4	300	150	nind/2	30%	40	120	5%	250
Grupo 5	240	150	nind/2	30%	64	100	5%	250
Grupo 6	200	150	nind/3	30%	64	100	5%	250

Tabela 6.2: Parâmetros NSGA II - Fase 2

Grupo	nind	max_ger	filhos	t_mutação	genes_mut	alfa_front
Grupo 1	300	150	nind/2	30%	10	150
Grupo 2	100	150	nind/2	30%	10	150
Grupo 3	200	100	nind/2	30%	40	150
Grupo 4	200	100	nind/2	30%	40	150
Grupo 5	200	100	nind/2	30%	64	150
Grupo 6	200	100	nind/2	30%	64	150

Tabela 6.3: Parâmetros - Melhoramento sem diversificações

Grupo	itermax2	trial	max_neigh
Grupo 1	500	20	70
Grupo 2	100	20	70
Grupo 3	300	15	70
Grupo 4	300	15	70
Grupo 5	150	15	70
Grupo 6	150	15	70

Tabela 6.4: Parâmetros - Melhoramento com diversificações

Grupo	itermax3	vezes_nível	trial	max_neigh
Grupo 1	15	50	20	70
Grupo 2	15	40	20	70
Grupo 3	15	50	15	70
Grupo 4	15	50	15	70
Grupo 5	15	40	15	70
Grupo 6	15	40	15	70

A Tabela 6.1 apresenta os parâmetros utilizados para grupo de instâncias testado durante a execução da primeira fase do método (AG), enquanto a Tabela 6.2 apresenta os parâmetros utilizados pela segunda fase (NSGA II). As Tabelas 6.3 e

6.4 apresentam os parâmetros utilizados pelos métodos de melhoramento descritos anteriormente.

Na Tabela 6.1, a coluna grupo é referente ao grupo de instâncias testado. Já a coluna **max_ger** se refere ao número máximo de gerações do AG mono-objetivo para geração de soluções iniciais. Vale ressaltar que, nesta fase, é possível acontecer do número de gerações ser maior, pois, caso não se atinja o número mínimo de possíveis soluções iniciais para o NSGA II, o AG continua até atingir tal valor ($nind2$). A coluna filhos apresenta a quantidade de filhos gerados pelo processo de *crossover*. O parâmetro T_mutao é a probabilidade de um indivíduo, logo após ser gerado pelo *crossover* uniforme, sofrer mutação aleatória e $genes\_mut$ apresenta a quantidade de colunas (*genes*) alteradas na solução pela mutação aleatória. O parâmetro $Itermax$ corresponde ao número máximo de iterações em melhora na busca local que um indivíduo pode sofrer. A coluna p_infact apresenta a probabilidade de uma solução infactível ser aceita como solução inicial para a segunda fase e $nind2$ corresponde ao número mínimo de soluções iniciais que podem ser usadas pela segunda fase.

A Tabela 6.2 apresenta os parâmetros utilizados na segunda fase, o NSGA II. A coluna **nind**, nesse caso, é referente ao número de indivíduos do NSGA II. Vale observar que os valores das mesmas são maiores que os valores correspondentes na coluna **nind2** da Tabela 6.1. A coluna **alfa_front** indica a penalização de uma solução infactível. O somatório das violações das restrições de capacidade de uma solução infactível é multiplicado pelo parâmetro **alfa_front** e, assim, tal solução é inserida em um *front* que possui um alto índice e, quanto maior for a soma das violações, mais distante essa solução ficará das soluções do nível 1. As demais colunas possuem as mesmas descrições apresentadas para a Tabela 6.1, porém se aplicam ao NSGA II.

A Tabela 6.3 apresenta os parâmetros utilizados no melhoramento do conjunto Pareto-Ótimo, baseado em estruturas de vizinhança, de forma que **itermax2** representa o número máximo de iterações sem obter um vizinho diferente e factível, em relação a cada solução de Pareto; **trial** corresponde ao número máximo de tentativas de melhora de conjunto; e **max_neigh** representa o número máximo de vizinhos gerados para cada solução.

Na Tabela 6.4, **itermax3** representa o número máximo de diversificações, ou perturbações, utilizadas na geração das novas soluções a partir de uma solução do conjunto Pareto-Ótimo; **vezes_nível** corresponde à quantidade máxima de soluções geradas, dentro de um mesmo nível de perturbações, sem que seja gerada uma solução diferente e factível; as colunas **trial** e **max_neigh** possuem as mesmas descrições apresentadas anteriormente.

As Figuras 6.1 - 6.6 apresentam os resultados obtidos, de forma que a aproximação do conjunto Pareto-Ótimo gerado pelo NSGA II é representado por $\bullet$; as soluções representadas por $*$ foram obtidas por meio do melhoramento com estratégia baseada em estrutura de vizinhança; as representadas por $\triangle$ obtidas por meio do melhoramento baseado em vizinhança e diversificações. Nos gráficos, o eixo x representa a função de custo, enquanto que o eixo y, a função de congestionamento da rede.

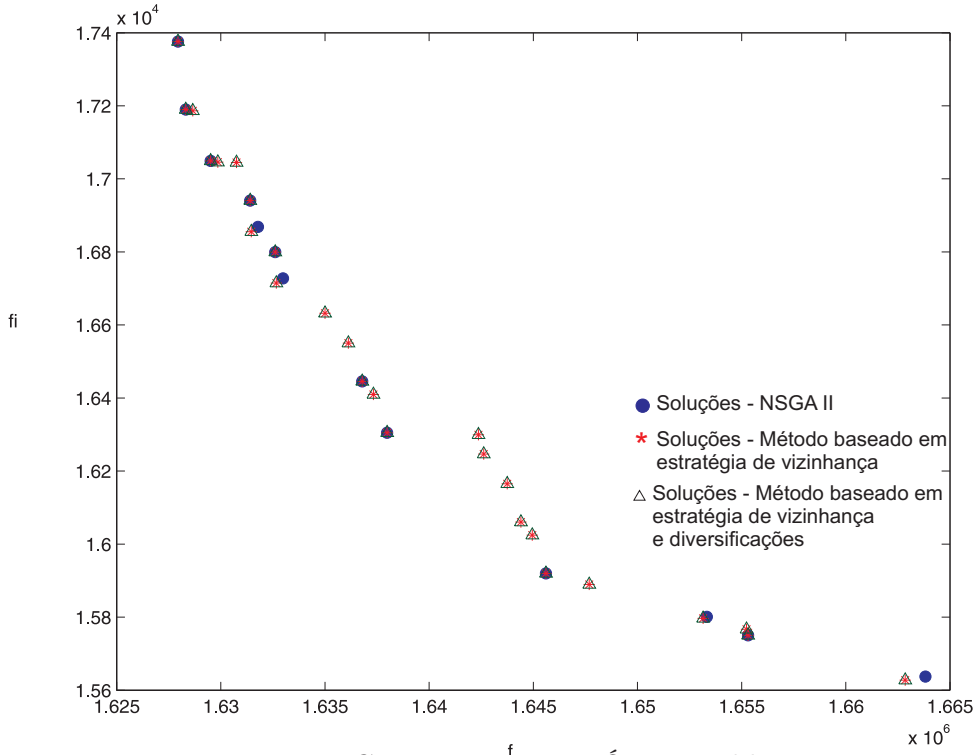

Figura 6.1: Conjuntos Pareto-Ótimos - bl01.

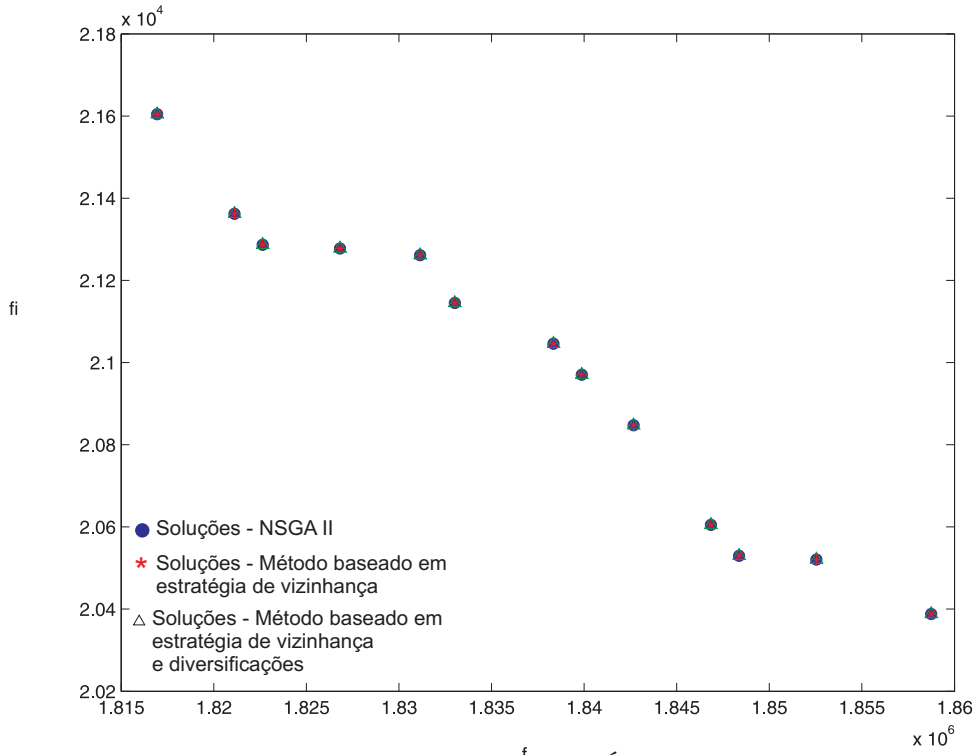

Figura 6.2: Conjuntos Pareto-Ótimos - bl02.

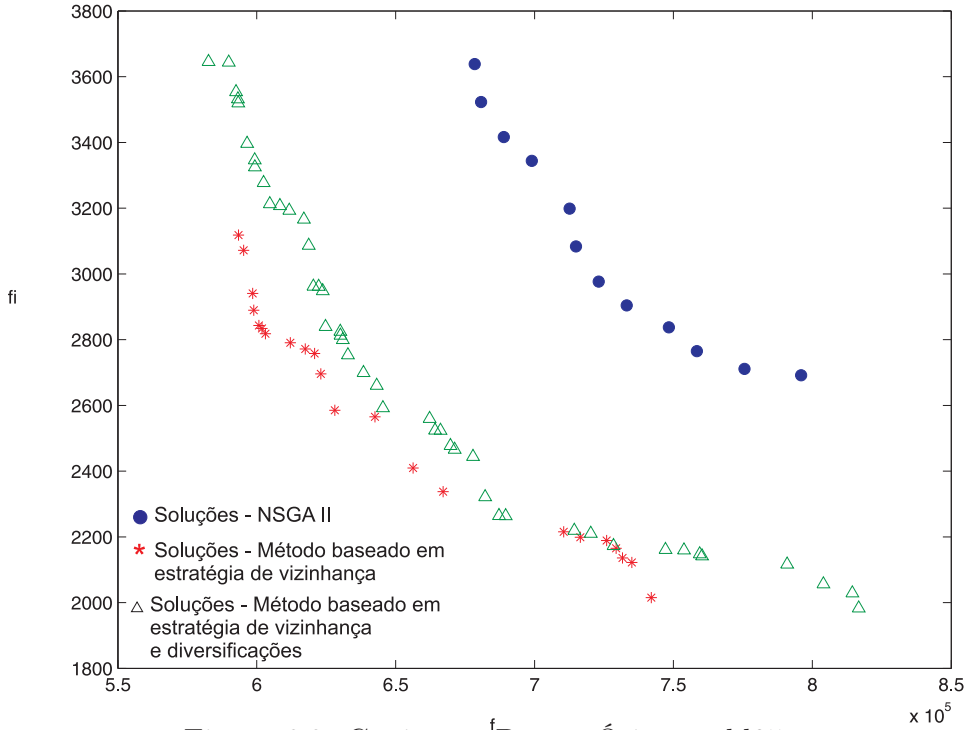

Figura 6.3: Conjuntos Pareto-Ótimos - bl05.

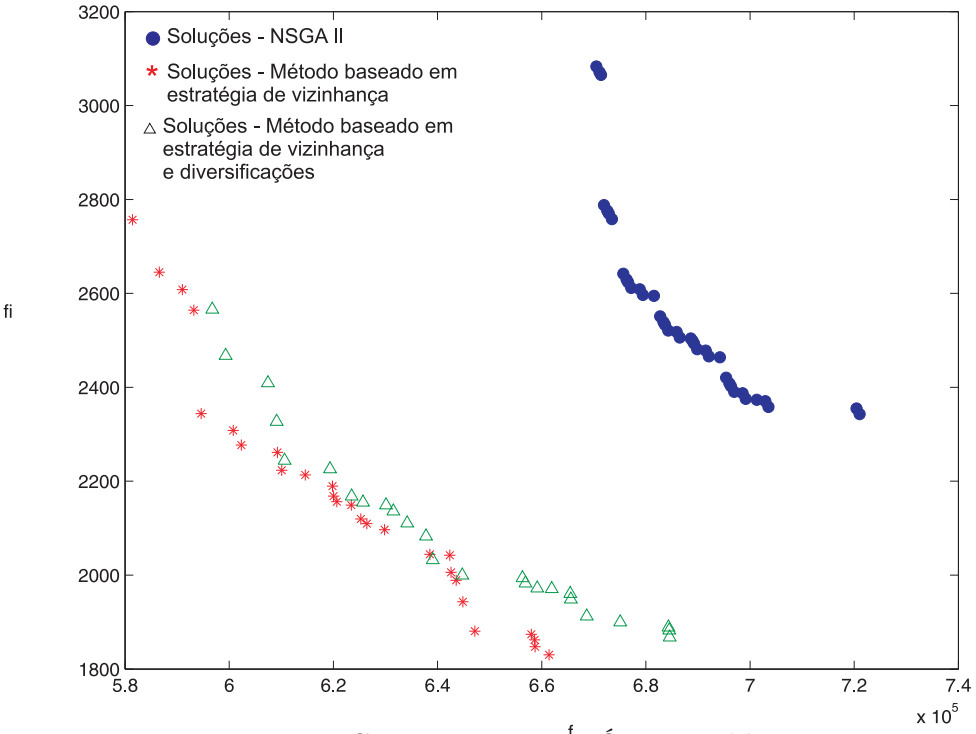

Figura 6.4: Conjuntos Pareto-Ótimos - bl06.

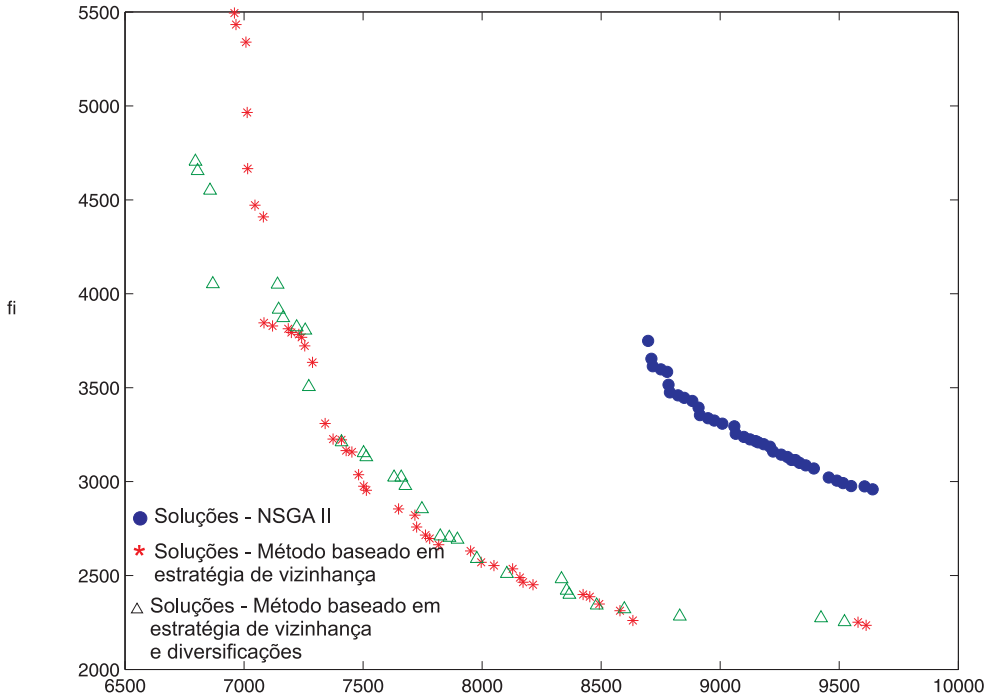

Figura 6.5: Conjuntos Pareto-Ótimos - bl07.

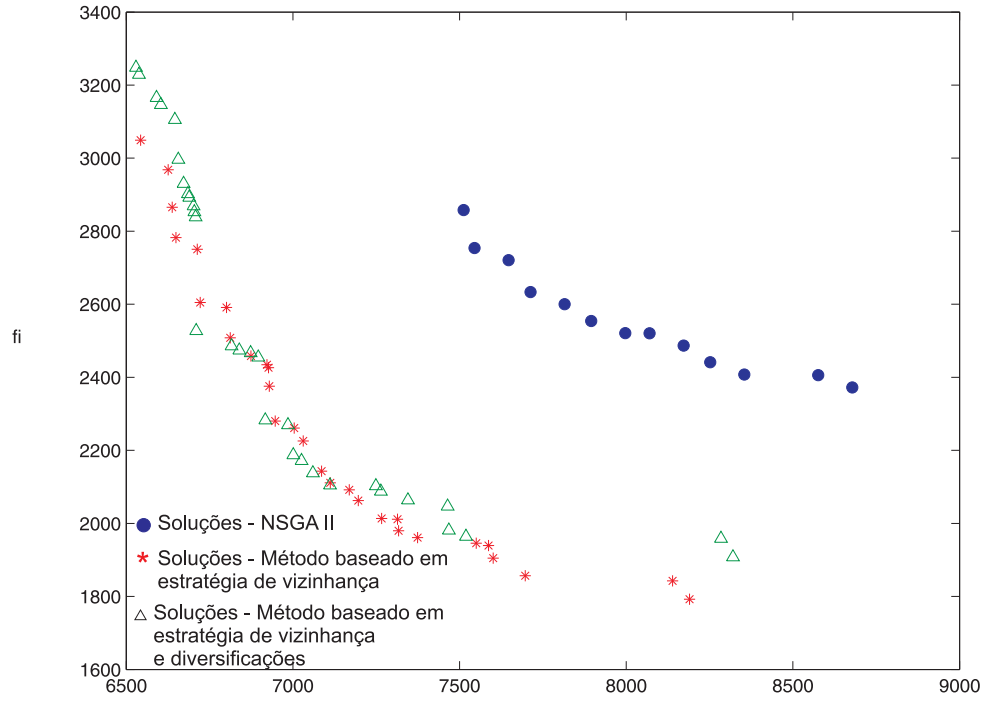

Figura 6.6: Conjuntos Pareto-Ótimos - bl08.

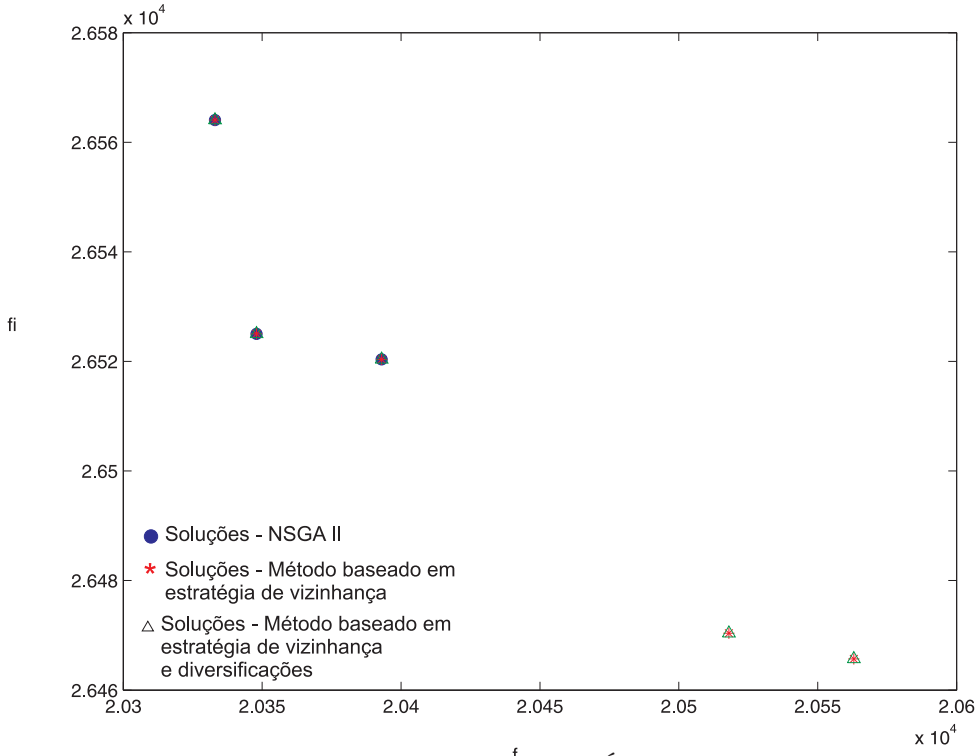

Figura 6.7: Conjuntos Pareto-Ótimos - bs04.

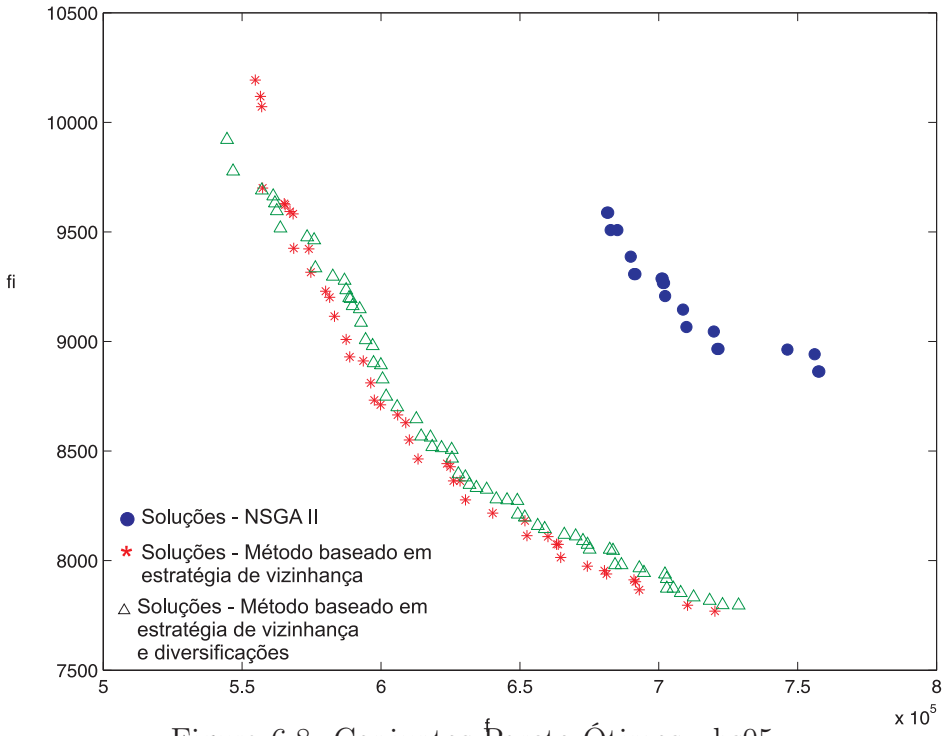

Figura 6.8: Conjuntos Pareto-Ótimos - bs05.

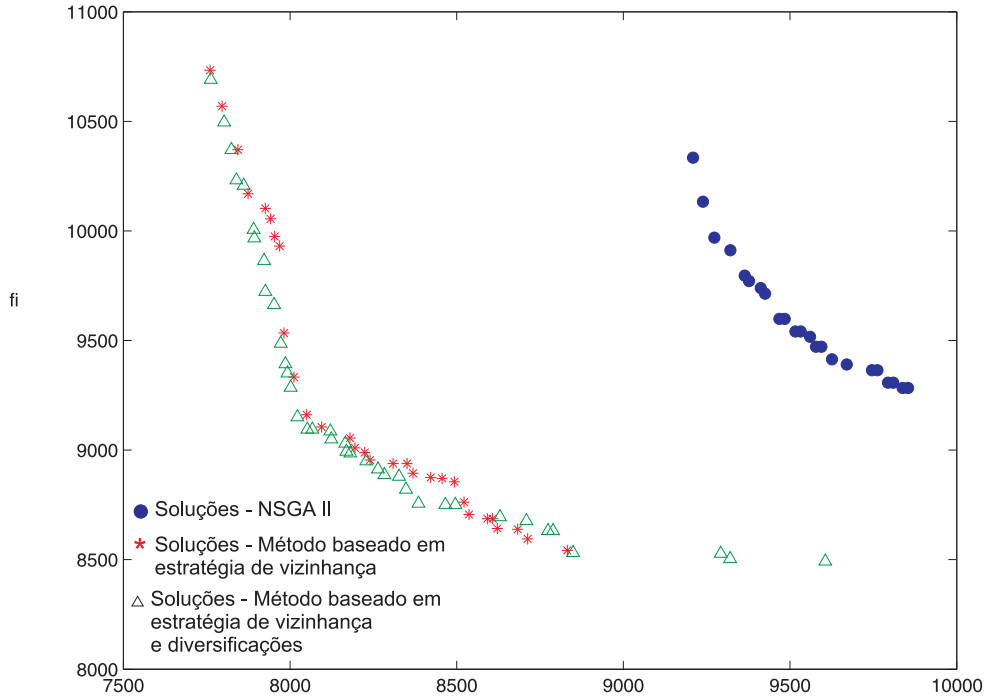

Figura 6.9: Conjuntos Pareto-Ótimos - bs07.

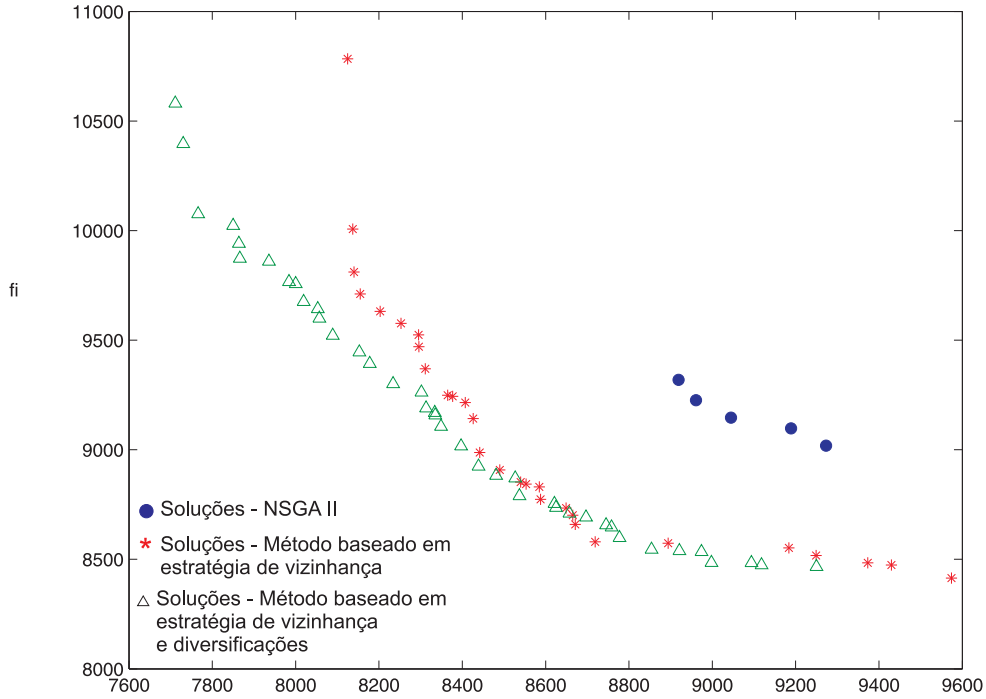

Figura 6.10: Conjuntos Pareto-Ótimos - bs08.

6.6.1 Outros Resultados

A fim de se obter conjuntos Pareto-Ótimos para mais instâncias, foi realizada uma pequena modificação que atinge o processo de busca local proposto no AG, com

o intuito de gerar mais soluções iniciais factíveis para o NSGA II. Foi implementada uma análise de frequência, de forma que o método faz uma contagem da quantidade de vezes que uma determinada coluna é alterada na solução, quando a mesma é trocada. Assim, a escolha por análise de frequência escolhe aleatoriamente uma coluna entre as 30% menos alteradas. Também foi implementado um método que guarda os índices dos arcos violados e, em seguida, escolhe aleatoriamente um desses arcos violados. Após escolher um arco violado, o método escolhe aleatoriamente um produto que passe por este arco violado, selecionando, assim, uma nova coluna e buscando uma nova rota para o produto escolhido. Este é um método de análise de violação. Portanto, a busca local desses novos resultados procede da seguinte maneira: até 40% do valor **itermax**, a escolha é feita de forma aleatória; de 40% a 80% de **itermax** é realizada a análise de frequência e, finalmente, de 80% até **itermax**, a escolha é feita com base na análise de violação. As Figuras 6.11 - 6.13 apresentam os resultados obtidos.

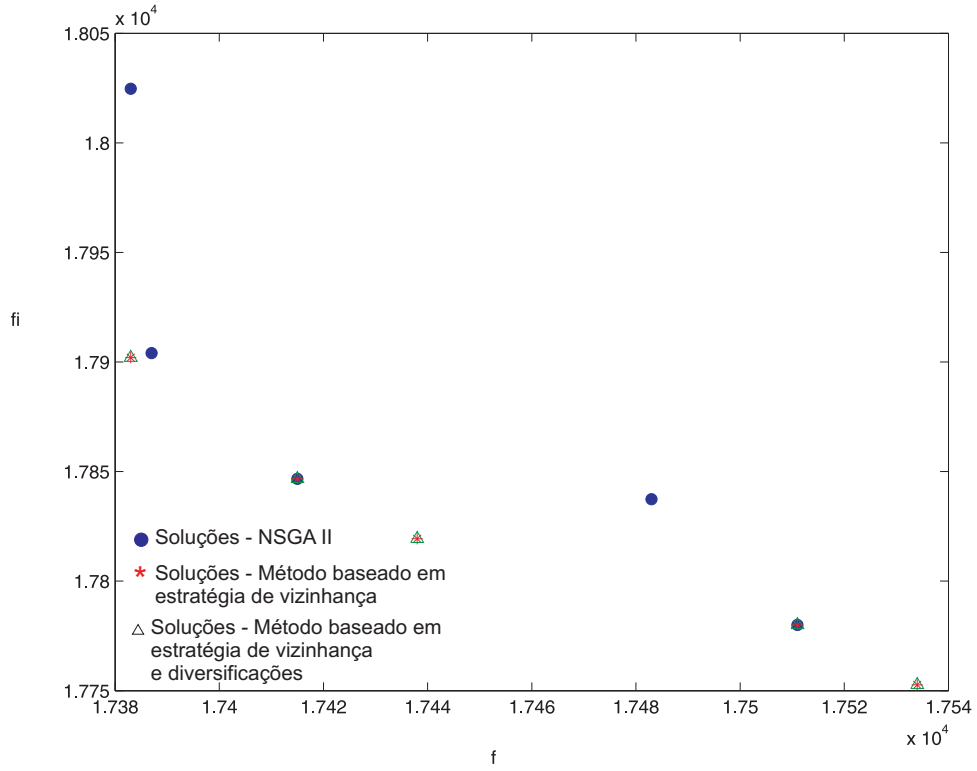

Figura 6.11: Conjuntos Pareto-Ótimos - bl03.

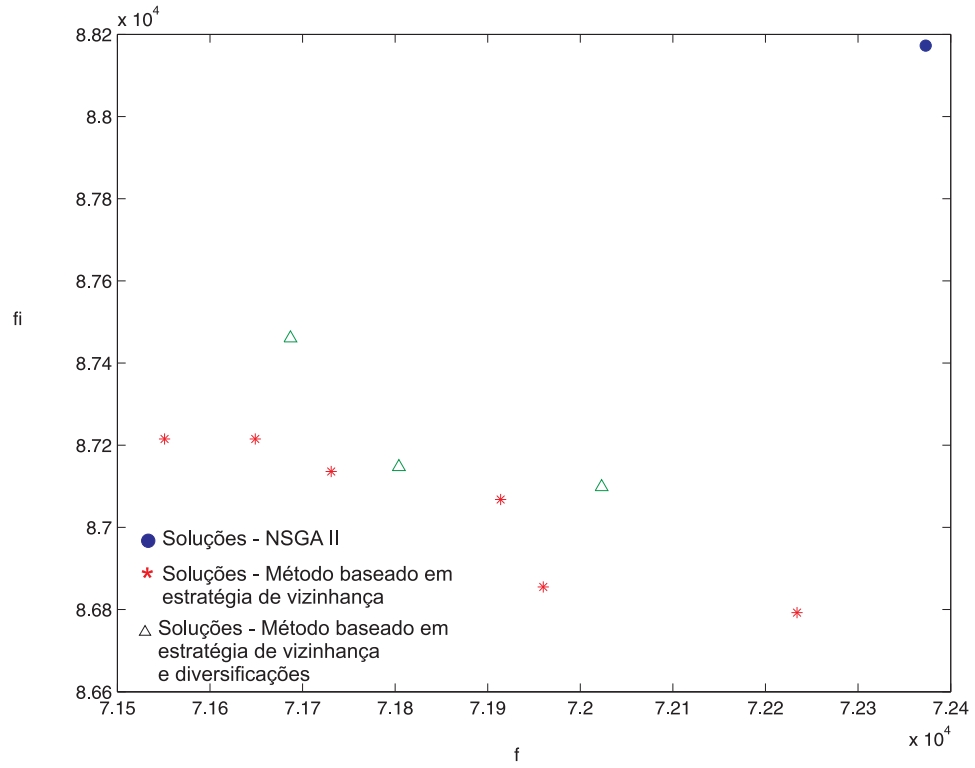

Figura 6.12: Conjuntos Pareto-Ótimos - bl11.

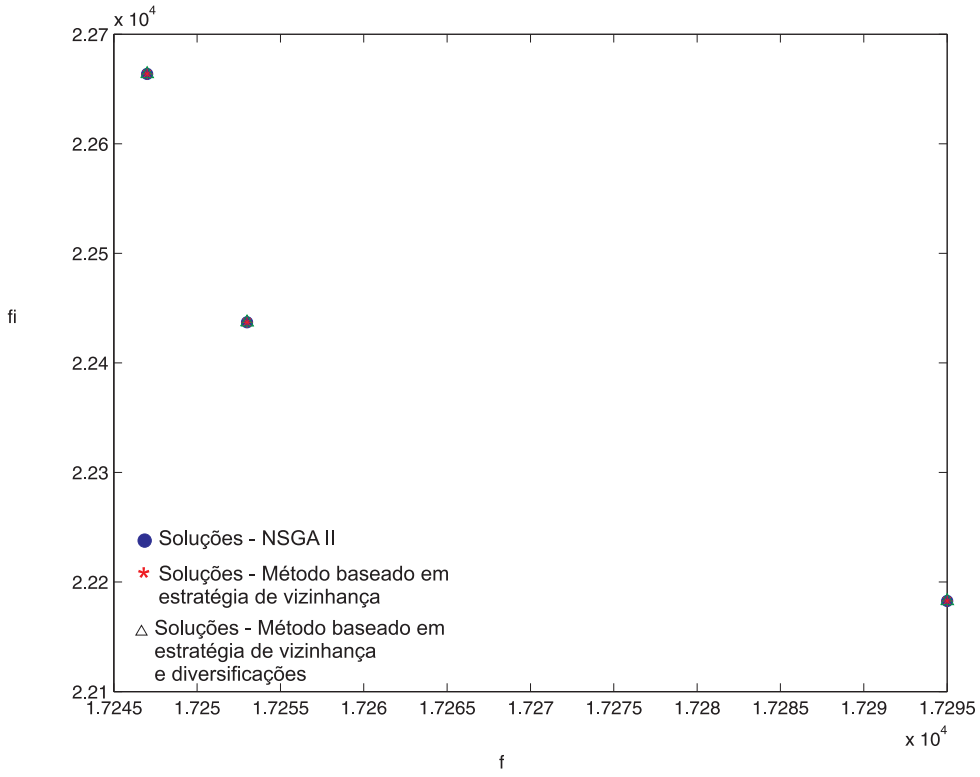

Figura 6.13: Conjuntos Pareto-Ótimos - bs03.

6.7 Conclusões

O método foi capaz de solucionar algumas das instâncias menores, que são as instâncias *bl01 - bl08* e *bs01 - bs08*. O principal problema do método consiste na geração das soluções iniciais. O AG não foi capaz de gerar boas soluções iniciais na maior parte das instâncias testadas. Conclui-se que o NSGA II obteve êxito, pois sempre que partiu de um conjunto de soluções iniciais factíveis, foi capaz de gerar Conjuntos Pareto-Ótimos e somente para a instância *bl04* o conjunto não está bem distribuído. Nas demais instâncias, os conjuntos foram obtidos com as características esperadas, principalmente pelo fato da função de custo ser inteira e, portanto, não pode-se afirmar que trata-se de um problema convexo. É possível observar que os objetivos tratados são conflitantes, pois, em todos os casos, uma redução do custo implicou num maior congestionamento na rede.

Para as instâncias *bl01*, *bl02* e *bl04*, pode-se afirmar que os melhoramentos foram desnecessários. Para as outras instâncias, pode-se perceber que tais métodos foram significativos. É importante ressaltar que as demais instâncias, para as quais os melhoramentos obtiveram êxito, possuem *32* nós, *320* arcos e *48* produtos.

Não se pode afirmar, ainda, qual dos dois melhoramentos gera os melhores conjuntos, pois não houve dominância de um método sobre o outro em todas ou na maior parte das instâncias testadas.

No caso dos outros resultados, para os quais foram utilizadas análises de frequência e de violação, também não foram gerados quantidades de soluções iniciais satisfatórias, o que motiva a busca de novas metodologias aplicadas à geração de tais soluções.

Não foram feitas comparações com outros resultados pelo fato de não terem sido encontrados trabalhos relacionados na literatura.

A principal contribuição desse capítulo está na apresentação de um método multiobjetivo, considerando o balanceamento de uma rede multifluxo e, também, na implementação de um método evolutivo multiobjetivo, o NSGA II, para a solução do Problema Multiobjetivo de Fluxo Multiproduto. Com isso, busca-se incentivar o acréscimo de alguma função de balanceamento no problema, pois muitos problemas reais são modelados por meio de um problema de fluxo multiproduto e, normalmente, o balanceamento da rede não é tratado.

Capítulo 7

Conclusões Finais e Trabalhos Futuros

Esta dissertação apresenta um estudo da solução, utilizando metaheurísticas, do Problema de Fluxo Multiproduto e do Problema Multiobjetivo de Fluxo Multiproduto. Para o primeiro problema, uma formulação mono-objetivo, foram propostas e desenvolvidas as metaheurísticas *Iterated Local Search* e Algoritmo Genético e uma formulação híbrida utilizando as duas metaheurísticas. Os resultados obtidos foram de boa qualidade, quando comparados a outros métodos existentes na literatura baseados em métodos exatos, como os apresentadas em (Alvelos, 2005), utilizando decomposição. Em muitas instâncias, o algoritmo proposto AG + ILS superou os resultados obtidos por meio de decomposição baseada no problema da mochila e, também os resultados obtidos por um algoritmo de partição e geração de colunas com cortes, baseado em decomposição por caminhos. Foi obtido um resultado inédito para a instância *bl10* e o método AG + ILS se mostrou eficiente para $93,75\%$ das instâncias testadas.

O segundo problema é uma formulação multiobjetivo de um problema de balanceamento de rede. A proposta de uma metodologia multi-objetivo, considerando as funções de custo e de balanceamento, é relevante, pois em muitos casos é importante o estudo do congestionamento da rede, como mostra (Buriol, 2003). O método proposto para a abordagem multi-objetivo encontrou soluções candidatas a Pareto-ótimas somente em instâncias menores, pois, nestes casos, o método proposto para gerar soluções iniciais se mostrou eficiente. Para as instâncias maiores, o método não foi capaz de gerar soluções iniciais factíveis. Deve-se ressaltar que não foram encontradas referências similares, na literatura pesquisada, para esta formulação de problema multiobjetivo.

Durante o desenvolvimento deste trabalho, foram publicados, em anais de eventos, oito artigos completos, um resumo e um resumo expandido, sendo que as principais publicações são (Mourão et al., 2008c), (Mourão et al., 2008a), (Mourão et al., 2008b), (Mourão et al., 2008d), (Mourão et al., 2009b) e (Mourão et al., 2009a).

Em (Mourão et al., 2008c), é aplicado o AG e, também, o ILS, ao problema de Fluxo Multiproduto Mono-objetivo. Foram testadas instâncias geradas aleatoriamente e os resultados obtidos foram bons, pois estão próximos aos resultados encontrados por meio de métodos exatos e, em uma das instâncias testadas, foi obtido um resultado inédito. Em (Mourão et al., 2008a) foi utilizado somente o

método ILS e os resultados foram satisfatórios quando comparados a resultados obtidos por métodos exatos. Em (Mourão et al., 2008b) foi aplicado um método evolutivo e os resultados foram bons somente para instâncias menores. (Mourão et al., 2008d) compara resultados obtidos somente por meio de métodos heurísticos. (Mourão et al., 2009b) apresenta o caso multiobjetivo do problema, considerando a função de balanceamento da rede multiproduto. A metodologia utilizada consiste em algoritmos evolutivos, utilizados para geração de soluções iniciais e, também, para a obtenção de soluções candidatas a ótimas de Pareto. Por fim, em (Mourão et al., 2009a) é apresentado um método evolutivo para geração de soluções iniciais e para a obtenção do conjunto de soluções candidatas, além de apresentar uma metodologia que busca melhorar o conjunto de soluções candidatas. Os resultados obtidos por meio deste método de melhoramento foram bons, pois em muitos casos foram encontradas soluções que dominam as soluções candidatas, obtendo um novo conjunto de Pareto.

Como propostas de trabalhos futuros, podem ser citados:

- estudo de outras variações do problema, em particular o problema inteiro e o problema linear;
- aplicação de outros métodos evolutivos multiobjetivos, como o SPEA2;
- adaptação de heurísticas, tradicionalmente utilizadas em problemas mono-objetivos, para o problema multiobjetivo, como o ILS;
- implementação de um método mais eficiente aplicado à geração de soluções iniciais factíveis para o problema multiobjetivo.

Apêndice A

Características das Instâncias Testadas

Tabela A.1: Características das instâncias - blxx - 01

Inst.	#N	#A	#P	C_Max	T_Custo	CM	DM	CM / DM
bl01	32	96	48	1000	2	76.13	15.56	4.89
bl02	32	96	48	1000	1	72.72	16.33	4.45
bl03	32	96	48	10	2	69.29	15.54	4.46
bl04	32	96	48	10	1	53.58	15.50	3.46
bl05	32	320	48	1000	2	109.14	15.63	6.99
bl06	32	320	48	1000	1	106.77	16.75	6.37
bl07	32	320	48	10	2	100.08	15.33	6.53
bl08	32	320	48	10	1	100.10	16.06	6.23
bl09	32	96	192	1000	2	104.11	15.43	6.75
bl10	32	96	192	1000	1	120.09	15.37	7.81
bl11	32	96	192	10	2	109.70	109.70	1.00
bl12	32	96	192	10	1	114.17	15.96	7.15
bl13	32	320	192	1000	2	53.72	15.44	3.48
bl14	32	320	192	1000	1	56.49	15.46	3.65
bl15	32	320	192	10	2	55.47	15.52	3.58
bl16	32	320	192	10	1	52.68	15.23	3.46
bl17	32	96	320	1000	2	197.58	15.38	12.85
bl18	32	96	320	1000	1	186.82	15.46	12.08
bl19	32	96	320	10	2	184.91	15.25	12.13
bl20	32	96	320	10	1	182.13	15.21	11.97
bl21	32	320	320	1000	2	51.06	15.24	3.35
bl22	32	320	320	1000	1	58.20	15.51	3.75
bl23	32	320	320	10	2	58.77	15.15	3.88
bl24	32	320	320	10	1	57.23	15.33	3.73

Tabela A.2: Características das instâncias - blxx - 02

Inst.	#P / #A	#A / #N	#P / #N
bl01	0.50	3	1.5
bl02	0.50	3	1.5
bl03	0.50	3	1.5
bl04	0.50	3	1.5
bl05	0.15	10	1.5
bl06	0.15	10	1.5
bl07	0.15	10	1.5
bl08	0.15	10	1.5
bl09	2.00	3	6
bl10	2.00	3	6
bl11	2.00	3	6
bl12	2.00	3	6
bl13	0.60	10	6
bl14	0.60	10	6
bl15	0.60	10	6
bl16	0.60	10	6
bl17	3.33	3	10
bl18	3.33	3	10
bl19	3.33	3	10
bl20	3.33	3	10
bl21	1.00	10	10
bl22	1.00	10	10
bl23	1.00	10	10
bl24	1.00	10	10

Tabela A.3: Características das instâncias - bsxx - 01

Inst.	#N	#A	#P	C_Max	T_Custo	CM	DM	CM / DM
bs01	32	96	48	1000	2	29.75	15.06	1.98
bs02	32	96	48	1000	1	30.81	14.75	2.09
bs03	32	96	48	10	2	33.35	15.96	2.09
bs04	32	96	48	10	1	39.76	15.65	2.54
bs05	32	320	48	1000	2	21.62	15.23	1.42
bs06	32	320	48	1000	1	21.36	15.46	1.38
bs07	32	320	48	10	2	21.58	15.52	1.39
bs08	32	320	48	10	1	22.25	15.42	1.44
bs09	32	96	192	1000	2	101.22	15.42	6.56
bs10	32	96	192	1000	1	122.88	15.69	7.83
bs11	32	96	192	10	2	108.24	15.36	7.05
bs12	32	96	192	10	1	114.13	15.42	7.40
bs13	32	320	192	1000	2	33.18	15.41	2.15
bs14	32	320	192	1000	1	33.73	15.61	2.16
bs15	32	320	192	10	2	32.33	15.56	2.08
bs16	32	320	192	10	1	31.83	15.15	2.10
bs17	32	96	320	1000	2	180.00	15.65	11.50
bs18	32	96	320	1000	1	188.95	15.59	12.12
bs19	32	96	320	10	2	170.02	15.40	11.04
bs20	32	96	320	10	1	179.33	15.49	11.58
bs21	32	320	320	1000	2	49.37	15.53	3.18
bs22	32	320	320	1000	1	48.01	15.49	3.10
bs23	32	320	320	10	2	50.03	15.66	3.20
bs24	32	320	320	10	1	51.63	15.50	3.33

Tabela A.4: Características das instâncias - bsxx - 02

Inst.	#P / #A	#A / #N	#P / #N
bs01	0.50	3	1.5
bs02	0.50	3	1.5
bs03	0.50	3	1.5
bs04	0.50	3	1.5
bs05	0.15	10	1.5
bs06	0.15	10	1.5
bs07	0.15	10	1.5
bs08	0.15	10	1.5
bs09	2.00	3	6
bs10	2.00	3	6
bs11	2.00	3	6
bs12	2.00	3	6
bs13	0.60	10	6
bs14	0.60	10	6
bs15	0.60	10	6
bs16	0.60	10	6
bs17	3.33	3	10
bs18	3.33	3	10
bs19	3.33	3	10
bs20	3.33	3	10
bs21	1.00	10	10
bs22	1.00	10	10
bs23	1.00	10	10
bs24	1.00	10	10

Referências Bibliográficas

Ahuja, R.K.; Magnanti, T.L. e Orlin, J.B. (1993). *Network Flows*. Prentice Hall.

Alvelos, F. P. *Branch-And-Price and Multicommodity Flows*. PhD thesis, Universidade do Minho, (2005).

Arantes, M.B.; Oliveira, G.T.S. e Saramago, S.F.P. maio(2006). Evolução diferencial aplicada à solução de alguns problemas de engenharia da produção. *FAMAT em Revista*, v. 1, n. 6, p. 48–61.

Avila, S. L. *Otimização multiobjetivo e análise de sensibilidade para concepção de dispositivos*. Tese de Doutorado, UFSC, (2006).

Bauer, K.; Fisher, T.; Krumke, S.O.; Gerhardt, K.; Westphal, S. e Merz, P. (2008). Improved construction heuristics and iterated local search for the routing and wavelength assignment problem. Springer-Verlag,, editor, *8th European Conference on Evolutionary Computation in Combinatorial Optmisation*, p. 1–12, (2008).

Bazaraa, M.S.; Jarvis, J.J. e Shetyy, C.M. (1990). *Linear Programming and Network Flows*. Wiley.

Becceneri, J.C. (2007). Metaheurísticas e otimização. r.t. lac, inpe. *Computers and Operations Research*.

Bocanegra, S.; Santos, M.A. e Campos, F.F. (2000). Uma proposta de solução para o problema não linear de fluxo multiproduto utilizando pontos interiores. *III SECICOM*, p. 69–73, (2000).

Buriol, L. S. *Roteamento do Tráfego na Internet: algoritmos para projeto e operação de redes com protocolo OSPF*. Tese de Doutorado, UNICAMP, (2003).

Buriol, L.S.; França, P.M.; Resende, M.C.G. e Ribeiro, C.C. (2003). Otimizando o roteamento do tráfego na internet. *XXXV Simpósio Brasileiro de Pesquisa Operacional*, (2003).

Calvete, H.I. e Mateo, P.M. (1995). An approach for the network flow problem with mutiple objectives. *Computers and Operations Research*, v. 22, n. 9, p. 971–983.

Carrano, E.G.; Soares, L.A.E.; Takahashi, R.H.C.; Saldanha, R.R. e Neto, O.N. (2006). Eletric distribution network multiobjective design using a problem-specific genetic algorithm. *IEEE Transactions on Power Delivery*, v. 21, n. 2, p. 995–1005.

Castro, J. (2003). Solving difficult multicommodity problems with a specialized interior-point algorithm. Research, Operations, editor, *Annals of Operations Research*, p. 35–48, (2003).

Castro, R.E. *Otimização de estruturas com multi-objetivos via algoritmos genéticos*. Tese de Doutorado, UFRJ, (2001).

Chekuri, C.; Khanna, S. e Shepherd, F.B. (2004). The all-or-nothing multicommodity flow problem. *Thirty-sixth annual ACM symposium on Theory of computing*, p. 156–165, (2004).

Collette, Y. e Siarry, P. (2003). *Multiobjective Optimization: principles ande case studies*. Springer.

Costa, W.E.; Gouvea, E.F. e Goldbarg, M.C. (2003). Oimização combinatória para expansão de redes de distribuição de gás. *II Congresso Brasileiro de P&D em Petróleo e Gás*, p. 1–6, (2003).

Das, S.; Abraham, A.; Chakraborty, U. K. e Konar, A. June(2009). Multiple objective minimum cost flow problems: A review. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, v. 13, NO. 3, JUNE 2009, n. 3, p. 526–553.

Deb, K.; Agrawal, S.; Pratab, A. e Meyarivan, T. Apr.(2002). A fast and elitist multi-objective genetic algorithm: Nsga-ii. v. 6, n. 2, p. 182–197.

Dorigo, M. *Optimization, learning and natural algorithms*. PhD thesis, Politecnico di Milano, (1992).

Dorigo, M. e Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, v. 1, n. 1, p. 53–66.

Dorigo, M.; Maniezzo, V. e A., Colorni. (1996). Ant system: Optimization by a colony of cooperatings agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, v. 26, n. 1, p. 1–13.

Dorigo, M. e Stützle, T. (2002). The ant colony optimization metaheuristic: Algorithms, applications, and advances. Glover, F. e Kochenberger, G., editors, *Handbook of Metaheuristics*, p. 251–285. Kluwer Academic Publishers, (2002).

Erickson, M.; Resende, M.C.G. e Pardalos, P.M. (2002). A genetic algorithm for the weight setting problem in ospf routing. *Combinatorial Optimization*, v. 6, n. 3, p. 299–333.

Ericsson, M.; Resende, M.G.C. e Pardalos, P.M. (2002). A genetic algorithm for the weight setting problem in ospf routing. *Journal of Combinatorial Optimization*, v. 6, p. 299–333.

Fonseca, C.M. e Fleming, P.J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. *Fifth International Conference on Genetic Algorithms*, p. 416–423, (1993).

Fortz, B.; Rexford, J. e Thorup, M. (2002). Traffic engineering with traditional ip routing protocols. *IEEE Communications Magazine*, v. 40, p. 118–124.

Fortz, B. e Thorup, M. (2000)a. Increasing internet capacity using local search. Technical Report IS-MG 2000/21, Université Libre de Bruxelles. URL `http://smg.ulb.ac.be/Preprints/Fortz00_21.html`.

Fortz, B. e Thorup, M. (2000)b. Internet traffic engineering by optimizing ospf weights. *in Proc. IEEE INFOCOM*, p. 519–528, (2000)b.

Frangioni, A. e Gendron, B. (2007). 0-1 reformulations of the multicommodity capacitated network design problem. *CIRRELT*, v. 29, p. 1–19.

Fulkerson, L.R. e Ford, D.R. (1962). Flows in networks. Princeton University, USA.

Garcia, V.J. *Metaheurísticas multiobjetivo para o problema de restauração do serviço em redes de distribuição de energia elétrica*. Tese de Doutorado, UNICAMP, (2005).

Garey, M.R. e Johson, D.S. (1979). *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

Goffin, J.L.; Gondzio, J.; Sarkissian, R. e Vial, J.P. (1995). Solving nonlinear multicommodity flow problems by the analytic center cutting plane method. Relatório técnico, McGill University.

Gonçalves, E.N. (2007). *Tópicos especiais em otimização multiobjetivo - notas de aula*. Cefet-MG.

Guardia, L.E.T. e Mello, J.C.C.B.S. (2007). Experiências computacionais com modelos de fluxo para múltiplos produtos com funções de custo não lineares e não separáveis. *Investigação Operacional*, v. 27, p. 53–65.

Hamacher, H.W.; Pedersen, C.R. e Ruzika, S. (2007). Multiple objective minimum cost flow problems: A review. *European Journal of Operational Research*, v. , n. 176, p. 1404–1422.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. The University of Michigan Press.

Hu, T. C. (1969). *Integer Programming and Network Flows*. Addison-Wesley.

Hu, T.C. (1963). Multicommodity network flows. *Operations Research*, v. 11, p. 344–360.

Jones, Kim L.; Lustig, Irvin; Farvolden, Judith M. e Powell, Warren B. (1993). Multicommodity network flows: The impact of formulation on decomposition. *Math. Program.*, v. 62, p. 95–117.

Kennedy, J. e Eberhart, R. (1995). Particle swarm optimization. *IEEE International Conference on Neural Network*, p. 1942–1948, (1995).

Lampinen, J. e Zelinka, I. (2000). On stagnation of the differential evolution algorithm. *MENDEL 2000, 6th International Mendel Conference on Soft Computing*, p. 1–9, (2000).

Lim, C. e Smith, J.C. (2007). Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Transactions*, v. 39, p. 15–26.

Lobato, F.S.; Jr., V. Steffen; Lopes, L.C.O. e Murata, V.V. (2007). Evolução diferencial aplicada à solução de problemas de controle ótimo com restrições de desigualdade. *X Encontro de Modelagem Computacional*, p. 1–10, (2007).

Lourenço, H.R.; Martin, O.C. e Stützle, T. (2002). Iterated local search. *Handbook of metaheuristics*, v. 7, p. 321–353.

Machado, E.C.M.N. (2006). Operação de redes de escoamentos de petróleo utilizando algoritmo genético multi-objetivo. Master's thesis, Universidade Federal de Campina Grande.

Mccallum, J.C.J. (1977). A generalized upper bounding approach to communications network planning problems. *Networks*, v. 7, n. 1, p. 1–23.

Medeiros, J.A.C.C. *Exame de Partículas como Ferramenta de Otimização em Problemas Complexos de Engenharia Nuclear*. Tese de Doutorado, UFRJ, (2005).

Mendes, R.R. *Programação Matemática Fuzzy Aplicada a um Problema de Transporte Multiproduto em Ferrovias*. Tese de Doutorado, UNICAMP, (1999).

Michalewicz, Z.; Vignaux, G.A. e Hobbs, M. (1991). A nonstandard genetic algorithm for the nonlinear transportation problem. *ORSA J. Comput.*, v. 3, p. 307–316.

Miettenen, K. (1999). *NonLinear Multiobjective Optimization*. Kluwer Academic Publishers.

Milidiu, R. L.; Pessoa, A. A.; Braconi, V.; Laber, E. S. e Rey, P. A. (2001). Um algoritmo grasp para o problema de transporte de derivados de petróleo em oleodutos. *XXXIII Simpósio Brasileiro De Pesquisa Operacional*, p. 237–246, (2001).

Mourão, F.P.; Souza, S.R.; Carrano, E.G. e Souza, M.J.F. (2009)a. Um algoritmo evolutivo multiobjetivo duas fases com melhoramento do conjunto pareto-Ótimo aplicado ao problema de fluxo multiproduto com balanceamento da rede. *Simpósio Brasileiro de Automação Inteligente*, (2009)a.

Mourão, F.P.; Souza, S.R.; Carrano, E.G. e Souza, M.J.F. (2009)b. Um algoritmo evolutivo multiobjetivo em duas fases aplicado ao problema de fluxo multiproduto binário com balanceamento da rede. *Congresso Nacional de Matemática Aplicada e Computacional*, (2009)b.

Mourão, F.P.; Souza, S.R. e Silva, C.A. (2008)a. Aplicação da metaheurística iterated local search à solução do problema de fluxo multiproduto inteiro. *Simpósio Brasileiro de Pesquisa Operacional*, (2008)a.

Mourão, F.P.; Souza, S.R. e Silva, C.A. (2008)b. Aplicação de algoritmo evolucionário populacional ao problema de fluxo multiproduto inteiro. *Congresso Nacional de Matemática Aplicada e Computacional*, (2008)b.

Mourão, F.P.; Souza, S.R. e Silva, C.A. (2008)c. Uma aplicação do algoritmo genético e metaheurística iterated local search a problemas de fluxo multiproduto. *Simpósio de Pesquisa Operacional e Logística da Marinha*, (2008)c.

Mourão, F.P.; Souza, S.R. e Silva, C.A. (2008)d. Uma comparação de métodos heurísticos aplicados ao problema de fluxo multiproduto. *Encontro de Modelagem Computacional*, (2008)d.

Mustafa, A. e Goh, M. (1998). Finding integer efficient solutions for bicriteria and tricriteria network flow problems using dinas. *Computers and Operations Research*, v. 25, p. 139–157.

Nikolova, M. (2001). An approach for finding pareto optimal solutions of the multicriteria network flow problem. *Cybernetics and Information Technologies*, v. 2, p. 56–62.

Nikolova, M. (2003). A classification based approach for finding pareto optimal solutions of the multicriteria network flow. *Cybernetics and Information Technologies*, v. 3, p. 11–17.

Noda, A.S. e Martin, C.G. (2000). The biobjective minimum cost flow problem. *European journal of operation research*, v. , p. 591–600.

Noda, A.S. e Martin, C.G. (2001). An algorithm for the biobjective integer minimum cost flow problem. *Computers and Operations Research*, v. 28, p. 139–156.

Ozdaglar, A.E. e Bertsekas, D.P. (2003). Optimal solution of integer multicommodity flow with application in optical networks. *Symposium on Global Optimization*, (2003).

Park, S.; Kim, D. e Lee, K. (2002). An integer programming approach to the path selection problems. *International Network Optimization Conference*. Universidade de Lisboa, (2002).

Schaffer, J.D. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, (1984).

Shaffer, R. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, (1984).

Shan, Y. S. *A Dynamic Multicommodity Network Flow Model for Real Time Optimal Rail Freight Car Management*. PhD thesis, Princeton University, (2007).

Sheng, S.; Dechen, Z. e Xiaofei, X. (2006). Genetic algorithm for the transportation problem with discontinuous piecewise linear cost function. *International Journal of Computer Science and Network Security*, v. 6, n. 7, p. 182–189.

Silva, A.F. e Oliveira, A. C. De. (2006). Algoritmos genéticos: alguns experimentos com os operadores de cruzamento ("crossover") para o problema do caixeiro viajante assimétrico. *XXVI Encontro Nacional de Engenharia da Produção*, p. 1–9, (2006).

Silva, C.A. (2007). Uma abordagem de problemas de fluxo multiproduto inteiro via métodos heurísticos. Master's thesis, Centro Federal de Educação Tecnológica de Minas Gerais.

Souza, M. J. F. (2007). *Inteligência Computacional Para Otimização, Disponível Em http://www.Decom.Ufop.Br/Prof/Marcone*. Universidade Federal de Ouro Preto.

Srinivas, N. e Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, p. 221–248, (1994).

Storn, R. e Price, K. março(1995). Differential evolution: a simple and effcient adaptive scheme for global optimization over continuous spaces. Relatório técnico, International Computer Science Institute.

Syswerda, G. Apr.(1989). Uniform crossover in genetic algorithms. J.D.„ editor, *Proceedings of the Third International Conference on Genetic Algorithms*, p. 2–9. Morgan Kaufmann Publishers, Apr.(1989).

Vesterstrøm, J. e Thomson, R. June(2004). A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. Press, IEEE, editor, *Proc. 6th Congr. Evol. Comput. (CEC-2004)*, volume 2, p. 1980–1987, Piscataway, NJ.

Von Zuben, F. J. V. (2000). *Computação Evolutiva: Uma Abordagem Pragmática*. UNICAMP.

Wille, E.C.G.; Mellia, M.; Leonardi, E. e Marsan, M.A. (2005). Topological design of survivable ip networks using metaheuristic approaches. lecture notes in computer science. *Lecture Notes in Computer Science*, v. 3375, p. 191–206.

Zitzler, E.; Deb, K. e Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput*, v. 8, n. 2, p. 173–195.

Zitzler, E.; Laumanns, M. e Thiele, L. (2001). Spea 2: Improving the strength pareto evolutionary algorithm. Relatório técnico, Swiss Federal Institute of Technology (ETH).

Zitzler, E. e Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: The strength pareto approach. Relatório técnico, Swiss Federal Institute of Technology (ETH).

Ziviane, N. (2004). *Projeto de algoritmos, com implementações em PASCAL e C*. Thomson, 2 edição.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo