Universidade Estadual de Campinas

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Departamento de Telemática

Algoritmo para resolução do problema de fluxo multiproduto fuzzy

Juliana Verga

Orientador: Prof. Dr. Akebo Yamakami

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para obtenção do título de **Mestre em Engenharia Elétrica**.

Banca Examinadora:

Prof. Dr. Takaaki Ohishi	FEEC / UNICAMP
Profa. Dra. Tatiane Regina Bonfim	FAC / Campinas

Campinas, SP

Agosto de 2009

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

Verga, Juliana
V586a Algoritmo para resolução do problema de fluxo multiproduto fuzzy
Juliana Verga – Campinas, SP:
[s.n.], 2009.

Orientador: Akebo Yamakami.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Teoria de grafos. 2. Números difusos.
3. Conjuntos difusos. 4. Algoritmos difusos.
I. Yamakami, Akebo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título

Título em Inglês: Algorithm for solving the fuzzy multicommodity flow problem
Palavras-chave em Inglês: Graph theory, Fuzzy numbers, Fuzzy sets, Fuzzy algorithms
Área de concentração: Automação
Titulação: Mestre em Engenharia Elétrica
Banca Examinadora: Takaaki Ohishi, Tatiane Regina Bonfim
Data da defesa: 28/08/2009
Programa de Pós Graduação: Engenharia Elétrica.

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidata: Juliana Verga

Data da Defesa: 28 de agosto de 2009

Título da Tese: "Algoritmo para Resolução do Problema de Fluxo Multiproduto Fuzzy"

Prof. Dr. Akebo Yamakami (Presidente): _____

Profa. Dra. Tatiane Regina Bonfim: _____

Prof. Dr. Takaaki Ohishi: _____

Resumo

A teoria dos grafos é comumente utilizada na área da engenharia para resolver problemas que podem ser representados na forma de redes. Dentre diversos problemas abordados, o problema de fluxo multiproduto é um dos que também podem ser modelados por grafos.

Este trabalho apresenta uma proposta de solução para o problema de fluxo multiproduto *fuzzy*. O problema foi modelado através de um grafo, cujos nós representam pontos de oferta e demanda de produtos, os quais trafegam pelos arcos da rede. O algoritmo proposto visa encontrar soluções factíveis e boas para o problema de fluxo multiproduto *fuzzy* em redes com incertezas nos custos e capacidades, contendo múltiplas origens e múltiplos destinos. As incertezas são modeladas por meio da teoria dos conjuntos *fuzzy*, que tem sido aplicada com sucesso em problemas com incertezas.

Palavras-chave: Fluxo Multiproduto *Fuzzy*, Números *Fuzzy*, Programação Matemática *Fuzzy*, Teoria de Grafos.

Abstract

The graph theory is commonly used in the area of engineering to solve problems that can be represented in the form of nets. Among several problems, the multicommodity flow problem is one that can be modeled by graphs.

This work presents an approach for solving the fuzzy multicommodity flow problem. The problem was modeled through a graph whose nodes represent points of supply and demand of commodities, which pass through arcs of the network. Our algorithm aims to find a set of good feasible solutions for the fuzzy multicommodity flow problem in networks with uncertainties in the costs and capacities, containing multiple origins and multiple destinations. The uncertainties are modeled by means of the fuzzy sets theory, which has been successfully applied to problems with uncertainties.

Keywords: Fuzzy Multicommodity Flow. Fuzzy Numbers. Fuzzy Mathematical Programming. Graph Theory.

Dedicatória

Aos meus pais, Inês e Luiz...

Ao meu namorado Wesley...

Agradecimentos

Agradeço a Deus em primeiro lugar, pelo dom da vida e por tudo o que tem me proporcionado.

Ao meu orientador, o professor Akebo Yamakami, pela preciosa orientação, oportunidade, paciência e motivação, os quais foram de extrema importância para a realização deste trabalho.

Aos meus pais, Inês e Luiz, um exemplo vivo de humildade, amor, perseverança, a quem eu devo tudo. A sua criação e educação me fizeram chegar até aqui.

Ao meu namorado Wesley, pelo amor, pelo carinho, pelas sugestões e por todos os incentivos. Sem você teria sido muito mais difícil. Juntos, não há sonhos que não possamos realizar.

Aos meus avós: Fortunato e Maria, por sempre me apoairem. Enfim, à toda minha família, por sempre me apoiarem e torcerem muito por mim.

A todos os colegas da FEEC, especialmente aos amigos que participaram diretamente comigo desta jornada: Jussara, Priscila, Márcia, Rosana, Elma, Ingrid e Tiago. Ao Fábio, pela valiosa ajuda.

Às minhas amigas Fabiana e Darlene. Aos amigos: Jair, Janete e Ariane. Enfim, a todos os meus amigos e amigas, os quais são indispensáveis em minha vida.

Aos professores do Departamento de Matemática da Unesp de Presidente Prudente.

À todos os professores e funcionários da Faculdade de Engenharia Elétrica - UNICAMP.

Ao CNPq, pelo apoio financeiro.

Aos membros da banca pelas valiosas correções e sugestões.

Lista de Figuras

2.1 Exemplo de uma função de pertinência triangular 9
2.2 Exemplo de uma função de pertinência trapezoidal 10

4.1 Rede exemplo 32
4.2 Funções de pertinência dos custos de p_1, p_2 e p_3 33
4.3 Capacidade do arco $(2, 5)$ 35
4.4 Fluxograma do algoritmo proposto 37

5.1 Rede com dois produtos 39
5.2 Rede com dois produtos, ambos com origem em dois nós 51
5.3 Rede óptica européia 54
5.4 Grafo da rede ferroviária 59

Lista de Tabelas

4.1 Capacidades dos arcos da Figura 4.1 34

5.1 Custos e capacidades da rede da Figura 5.1 39
5.2 Dados de cada produto 40
5.3 Fluxo final de cada produto 47
5.4 Envio de fluxo com a Opção 1 47
5.5 Fluxo final de cada produto com a Opção 1 47
5.6 Custo final de cada produto 48
5.7 Fluxo final de cada produto para $\alpha = 0,167$ 49
5.8 Envio de fluxo dos produtos 50
5.9 Fluxo final de cada produto sem a Opção 1 50
5.10 Custo final de cada produto 50
5.11 Dados de cada produto 51
5.12 Envio de fluxo dos produtos da rede da figura 5.2 52
5.13 Fluxo final de cada produto sem a Opção 1 referente a rede da figura 5.2 52
5.14 Custo final de cada produto 52
5.15 Fluxo final de cada produto para o caso clássico 53
5.16 Denominação dos nós da rede COST239 54
5.17 Dados da Rede COST239 55
5.18 Dados de cada produto 56
5.19 Envio de fluxo 56

5.20 Fluxo final do produto p_1 57
5.21 Fluxo final do produto p_2 57
5.22 Fluxo final do produto p_3 57
5.23 Fluxo total nos arcos 57
5.24 Fluxo total nos arcos 57
5.25 Custo final de cada produto 58
5.26 Fluxo final do produto p_1 para o caso clássico 58
5.27 Fluxo final do produto p_2 58
5.28 Fluxo final do produto p_3 para o caso clássico 58
5.29 Dados de cada produto 60
5.30 Dados da Rede Ferroviária 61
5.31 Envio de fluxo 62
5.32 Fluxo final de cada produto em cada arco 62
5.33 Fluxo final de cada produto em cada arco 63
5.34 Fluxo final de cada produto em cada arco 63
5.35 Fluxo total nos arcos 63
5.36 Fluxo total nos arcos 64
5.37 Fluxo total nos arcos 64
5.38 Custo final de cada produto 64
5.39 Fluxo final de cada produto em cada arco para o caso clássico 65
5.40 Fluxo final de cada produto em cada arco para o caso 65
5.41 Fluxo final de cada produto em cada arco 66

Lista de Símbolos

$G(N, A)$	-	Grafo G composto por N nós e A arcos
$\tilde{a} = (m, \alpha, \beta)$	-	número triangular *fuzzy*
$\tilde{a} = (m_1, m_2, \alpha, \beta)$	-	número trapezoidal *fuzzy*
$\tilde{a} \oplus \tilde{b}$	-	soma de números triangulares *fuzzy*
Poss	-	Medida de Possibilidade
PFCM	-	Problema de Fluxo de Custo Mínimo
$\tilde{c}_{ij}^{k}$	-	custo *fuzzy* do produto k no arco (i, j)

Sumário

1 Introdução **1**

2 Conceitos da Teoria dos Conjuntos *Fuzzy* **5**

2.1 Conjuntos *fuzzy* 5

2.1.1 Operações sobre conjuntos *fuzzy* 7

2.1.2 Relações sobre conjuntos *fuzzy* 7

2.2 Números *fuzzy* 8

2.3 Teoria de possibilidade 11

2.3.1 Relações de Ordem e Grafos *Fuzzy* 12

2.3.2 Grafos *Fuzzy* 15

3 Problemas de fluxo em redes *fuzzy* **16**

3.1 Problema de Caminhos Mínimos *Fuzzy* 16

3.1.1 Revisão bibliográfica 17

3.1.2 Formulação matemática do problema 19

3.1.3 Algoritmo proposto por Hernandes 20

3.2 Algoritmos de fluxo em redes *fuzzy* 22

3.2.1 Formulação do problema de fluxo de custo mínimo 24

3.2.2 Formulação do PFCM *fuzzy* 25

3.2.3 Algoritmo proposto por Hernandes para o PFCM *fuzzy* 25

4 Problema de Fluxo Multiproduto *Fuzzy* **28**

4.1 Formulação do problema de fluxo multiproduto 29

4.2 Algoritmo proposto 31

5 Testes Computacionais **38**

5.1 Instância 1 39

5.2 Instância 2 53

5.3 Instância 3 59

6 Conclusões e trabalhos futuros **67**

6.1 Conclusões 67

6.2 Perspectivas Futuras 68

6.3 Trabalhos aceitos 68

A **69**

Bibliografia **71**

Capítulo 1

Introdução

A teoria dos grafos é comumente utilizada para resolver problemas que podem ser representados na forma de redes [1, 2]. Ela fornece uma modelagem intuitiva de um problema, facilitando a implementação de algoritmos que auxiliam na obtenção da sua solução [1].

Acredita-se que a teoria dos grafos foi introduzida em 1736, quando Eüler propôs uma solução para o problema das Pontes de Köenisberg, apresentando uma condição necessária para percorrer um grafo sem repetir arcos e retornar ao ponto inicial. Entretanto, foi na segunda metade do século XX, que essa teoria teve um enorme crescimento, com inúmeros problemas e metodologias sendo propostos [1, 2].

Atualmente, problemas bem conhecidos como caminho mínimo, fluxo de custo mínimo, alocação, entre outros são muito estudados utilizando esta teoria. Nas obras de Gondran e Minoux [13], Goldbarg e Luna [12], Ahuja, Magnanti e Orlin [1], Bazaraa, Jarvis e Sherali [2] tais problemas são encontrados, bem como os vários algoritmos existentes para resolvê-los.

Geralmente, problemas reais têm associado vários parâmetros que são incertos, tais como: capacidade, custo e demanda. Ao trabalhar com parâmetros incertos, uma informação deixa de ser representada por um único valor e passa a ser representada por um conjunto. Assim, o uso da teoria dos conjuntos clássica torna-se inviável devido à sua ine-

ficiência no tratamento de informações incertas. No entanto, essas incertezas podem ser estudadas e modeladas de forma mais robusta utilizando a teoria dos conjuntos *fuzzy*.

A introdução à teoria dos conjuntos *fuzzy* foi proposta pelo matemático Lotfi A. Zadeh em 1965 [37]. Este foi o primeiro passo no sentido de se programar e armazenar conceitos vagos em computadores, tornando possível fazer cálculos com informações incertas, a exemplo do que faz o ser humano. Porém foi somente nas duas últimas décadas que houve uma intensificação nos estudos desta área, onde extensões das teorias que envolvem programação linear, não linear, mista e inteira foram abordadas neste ambiente impreciso.

O tratamento de incertezas em problemas de grafos e as principais definições de grafos *fuzzy* foram propostos por Rosenfeld em 1975 [29], marco inicial da teoria dos grafos *fuzzy*. Em contraste aos inúmeros trabalhos existentes utilizando a teoria de grafos clássica, o estudo dos grafos *fuzzy* ainda está na sua fase inicial, tanto na teoria quanto na prática. Isto se deve ao fato da teoria de grafos e os problemas associados a mesma serem extensos. Outro motivo se deve às diferentes abordagens que um problema de grafos *fuzzy* pode ter, pois os níveis de incerteza podem estar associados tanto na estrutura (nós e/ou arcos) quanto nos parâmetros (custo, capacidade, demanda) [33].

Quanto às soluções de problemas de grafos *fuzzy*, estas podem ser únicas ou um conjunto.

Segundo [1], grande parte dos problemas de grafos se mostram computacionalmente intratáveis, ou seja, apresentam complexidade muito alta, do tipo NP - completos, o que os torna de díficil resolução.

Considerando o vasto campo de aplicações da teria de grafos *fuzzy* e o estudo inicial desta teoria e de suas aplicações, neste trabalho é estudado o problema de fluxo multiproduto *fuzzy* e é proposto um algoritmo para resolvê-lo.

Os trabalhos iniciais que tratam o problema de fluxo multiproduto foram propostos por Ford e Fulkerson [9] e Hu [15], no início dos anos 60. Já para o problema de fluxo multiproduto *fuzzy* são encontrados, na literatura, poucos trabalhos. Dentre eles, podemos

citar o de Ghatee e Hashemi [11] no qual os autores propõem dois algoritmos para o problema de fluxo multiproduto *fuzzy*. O primeiro algoritmo utiliza caminhos mínimos fuzzy e k-caminhos para gerar os caminhos e encontrar o fluxo resolvendo um problema de fluxo de custo mínimo multiproduto clássico. No segundo algoritmo, a oferta e demanda são *fuzzy* e é empregado uma ordem total em números trapezoidais *fuzzy* para reduzir o problema de fluxo multiproduto *fuzzy* em quatro problemas de fluxo multiproduto clássicos. Um outro trabalho é o de Mendes [22], onde é resolvido um problema de transporte ferroviário multifluxo, sujeito a restrições de trecho, frota, balanceamento, associado a ida e a redistribuição dos vagões descarregados. O algoritmo proposto por Mendes é baseado em conceitos de μ-caminhos para tratar as incertezas nos custos, e as incertezas das capacidades são tratadas fazendo um mapeamento dos valores de pertinência dos arcos em relação ao limitante superior, para conceitos de μ-caminhos.

No trabalho de Hernandes [14] são propostos algoritmos para problemas conhecidos na teoria de grafos, porém pouco estudados quando há incerteza associada (caminho mínimo em grafos com estrutura crisp (conhecida) e parâmetros *fuzzy*) e também algoritmos mais abrangentes para problemas de fluxo monoproduto de custo mínimo com capacidades e custos *fuzzy*.

Objetivos

Neste trabalho é feito um estudo dos algoritmos propostos por Hernandes para os problemas de caminho mínimo *fuzzy* e fluxo monoproduto de custo mínimo *fuzzy*, e a partir desse estudo, é proposto um algoritmo para o problema de fluxo multiproduto de custo mínimo *fuzzy*. O objetivo deste algoritmo é encontrar soluções factíveis e boas para o problema de fluxo multiproduto *fuzzy* em redes com custos e capacidades incertas contendo múltiplas origens e múltiplos destinos.

Organização do trabalho

Este trabalho está organizado da seguinte forma:

- No Capítulo 2 são apresentados os conceitos básicos da teoria de conjuntos *fuzzy*;
- No Capítulo 3 são apresentados os algoritmos de caminhos mínimos *fuzzy* e algoritmos de fluxo em redes *fuzzy*. Neste capítulo é feita uma revisão bibliográfica dos trabalhos já existentes na literatura;
- No Capítulo 4 o problema de fluxo multiproduto *fuzzy* é conceituado, é apresentada sua formulação matemática e é proposto um algoritmo para resolvê-lo;
- No Capítulo 5 são apresentados os resultados e análises dos testes computacionais realizados;
- No Capítulo 6 são feitas as considerações finais deste trabalho e algumas propostas para trabalhos futuros.

Capítulo 2

Conceitos da Teoria dos Conjuntos *Fuzzy*

A teoria dos conjuntos *fuzzy* foi introduzida em 1965 pelo matemático Lotfi A. Zadeh [37] com intuito de tratar matematicamente termos lingüísticos, como "em torno de", "aproximadamente", entre outros. A teoria *fuzzy* consegue manipular e operar quantidades exatas e inexatas (quantificadores através de valores lingüísticos).

Neste capítulo são introduzidos os conceitos básicos da teoria dos conjuntos *fuzzy*. As obras de Dubois e Prade [8] e Pedrycz e Gomide [28] podem ajudar no aprofundamento dos estudos desta teoria.

2.1 Conjuntos *fuzzy*

Conjunto é uma maneira de organizar, resumir e generalizar o conhecimento sobre objetos. Assim, trabalha-se com uma dicotomia sobre um objeto pertencer ou não a um conjunto. Esta dicotomia pode ser representada matematicamente por uma função característica:

$$f_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} \tag{2.1}$$

Um conjunto *fuzzy* A em X é definido por uma função de pertinência μ_A que associa cada ponto de X a um número real no intervalo $[0, 1]$, com o valor de μ_A em x representando o grau de pertinência de x em A. Assim, quanto mais próximo o valor de $f_A(x)$ estiver da unidade, maior o grau de pertinência de x em A.

Quando houver a necessidade de diferenciar entre conjuntos clássicos e conjuntos *fuzzy*, o conjunto com função característica com dois valores (0 ou 1), será chamado de conjunto clássico (ordinário ou crisp) ou simplesmente conjunto.

Definição 2.1.1. *Seja X o conjunto universo. Um conjunto fuzzy A é normal se e somente se $\exists\ x \in X$ tal que $\mu_A(x) = 1$.*

Caso não exista um valor x tal que o valor supremo (altura) da função de pertinência seja igual a um ($\sup(\mu_A(x)) \neq 1, \forall x \in A$), então A é um conjunto subnormal.

Definição 2.1.2. *O α-corte de um conjunto fuzzy A é definido por:*

$$[\tilde{a}]_\alpha = \{x \mid \mu_{\tilde{a}}(x) \geq \alpha\}$$

Definição 2.1.3. *O suporte de um conjunto fuzzy A é definido por:*

$$\text{supp}(A) = \{x \in X \mid \mu_A(x) > 0\}$$

ou seja, o suporte é formado pelos elementos que possuem graus de pertinência não-nulos.

Definição 2.1.4. *Um conjunto fuzzy A é convexo se sua função de pertinência é tal que:*

$$\mu_A[\lambda x_1 + (1 - \lambda)x_2] \geq \min[\mu_A(x_1), \mu_A(x_2)]$$

para quaisquer $x_1, x_2 \in X$ e $\lambda \in [0, 1]$.

Definição 2.1.5. *O núcleo de um conjunto fuzzy A é definido por:*

$$\text{nucleo}(A) = \{x \in X \mid \mu_A(x) = 1\}$$

2.1.1 Operações sobre conjuntos *fuzzy*

As operações básicas realizadas em conjuntos clássicos também são definidas para conjuntos *fuzzy*. Com base nisso, Zadeh [37] definiu, a partir da função de pertinência tais operações básicas.

Definição 2.1.6. *A união de dois conjuntos fuzzy A e B é definida por:*

$$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\} = \mu_A(x) \vee \mu_B(x)$$

Definição 2.1.7. *A intersecção de dois conjuntos fuzzy A e B é definida por:*

$$\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\} = \mu_A(x) \wedge \mu_B(x)$$

Definição 2.1.8. *O complemento de um conjunto fuzzy A é definido por:*

$$\mu_{\tilde{A}}(x) = 1 - \mu_A(x)$$

A intersecção e a união podem ser identificadas pela conjunção (E) e pela disjunção (OU), assim, estas operações podem ser representadas pelos operadores $\wedge$ e $\vee$ [28].

2.1.2 Relações sobre conjuntos *fuzzy*

Seja X um conjunto. Um subconjunto *fuzzy* S de X possui uma função $\sigma : S \to [0, 1]$ que associa elementos $x \in S$ a um grau de pertinência, $0 \leq \sigma \leq 1$. Similarmente, uma relação *fuzzy* sobre X é um subconjunto *fuzzy* de $X \times X$, que possui uma função $\mu : S \times X \to [0, 1]$ que associa cada par ordenado de elementos (x, y) a um grau de pertinência $0 \leq \mu(x, y) \leq 1$. Nos casos em que σ e μ podem assumir valores 0 e 1, eles se tornam funções características de um subconjunto ordinário de X e uma relação ordinária sobre X, respectivamente.

Se $T \subseteq X$ é um subconjunto de X e $R \subseteq X \times X$ é uma relação sobre X, então R é uma relação sobre T desde que $(x, y) \in R$ implique que $x \in T$ e $y \in T$, $\forall\, x, y$. Sejam τ e ρ

funções características. Então esta condição pode ser definida como:

$$R(x,y) = 1 \Rightarrow T(x) = T(y) = 1 \quad , \forall\, x, y \in X$$

Pode-se associar estas condições às funções de pertinência:

$$\rho(x,y) \leq \tau(x) \wedge \tau(y) \quad , \forall x, y \in S$$

onde $\wedge$ significa ínfimo.

Para o caso geral, onde S é um subconjunto *fuzzy* de X e σ sua função de pertinência, sendo R uma relação *fuzzy* sobre X, diz-se que R é uma relação *fuzzy* sobre S se:

$$R(x,y) \leq S(x) \wedge S(y) \ , \forall x, y \in S$$

Em outras palavras, para R ser uma relação *fuzzy* sobre S, é necessário que o grau de pertinência de um par de elementos nunca exceda o grau de pertinência dos próprios elementos. Por exemplo, se os nós de um grafo forem os elementos e os arcos forem os pares, é necessário que a pertinência de um arco nunca exceda as pertinências de seus nós.

2.2 Números *fuzzy*

Definição 2.2.1. *Um número triangular fuzzy é um conjunto fuzzy convexo e normal cuja função de pertinência é representada por uma função contínua, onde um único elemento possui grau de pertinência igual a um ($\mu_A(x) = 1$ para exatamente um único $x \in X$).*

Definição 2.2.2. *Um número fuzzy denotado por $\tilde{a} = (m, \alpha, \beta)$, é definido pela seguinte função de pertinência:*

$$\mu_{\tilde{a}}(x) = \begin{cases} 0, & x \leq m - \alpha \\ \frac{x-(m-\alpha)}{\alpha}, & m - \alpha < x < m \\ 1, & x = m \\ \frac{(m+\beta)-x}{\beta}, & m < x < m + \beta \\ 0, & x \geq m + \beta \end{cases} \tag{2.2}$$

A este número dá-se o nome de número triangular *fuzzy*, onde m : valor modal, α : espalhamento à esquerda, β : espalhamento à direita.

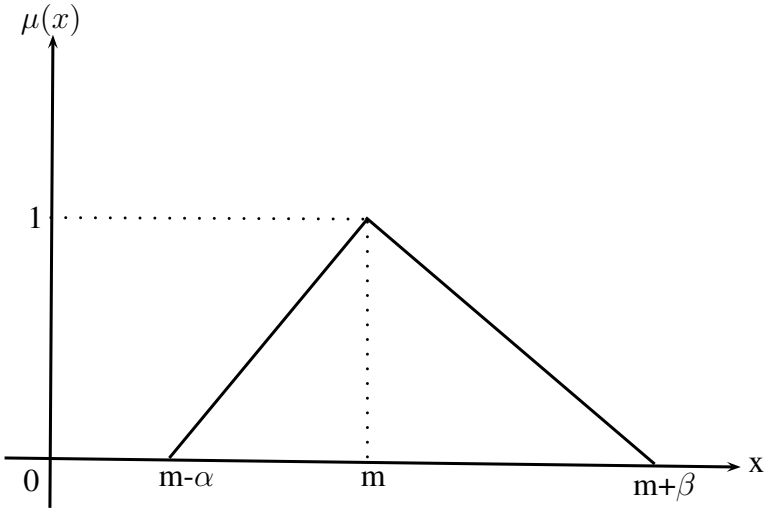

Figura 2.1: Exemplo de uma função de pertinência triangular

Os valores $(m - \alpha)$ e $(m + \beta)$ são chamados de limitante inferior e superior, respectivamente.

Definição 2.2.3. *Um número trapezoidal fuzzy denotado por* $\tilde{a} = (m_1, m_2, \alpha, \beta)$ *possui sua função de pertinência* $\mu_{\tilde{a}}(x)$*, definida por:*

$$\mu_{\tilde{a}}(x) = \begin{cases} 0, & x \leq m_1 - \alpha \\ \frac{x-(m_1-\alpha)}{\alpha}, & m_1 - \alpha < x < m_1 \\ 1, & m_1 \leq x \leq m_2 \\ \frac{(m_2+\beta)-x}{\beta}, & m_2 < x < m_2 + \beta \\ 0, & x \geq m_2 + \beta \end{cases} \tag{2.3}$$

onde m_1 : *extremo inferior do valor modal,* m_2 : *extremo superior do valor modal,* α : *espalhamento à esquerda,* β : *espalhamento à direita.*

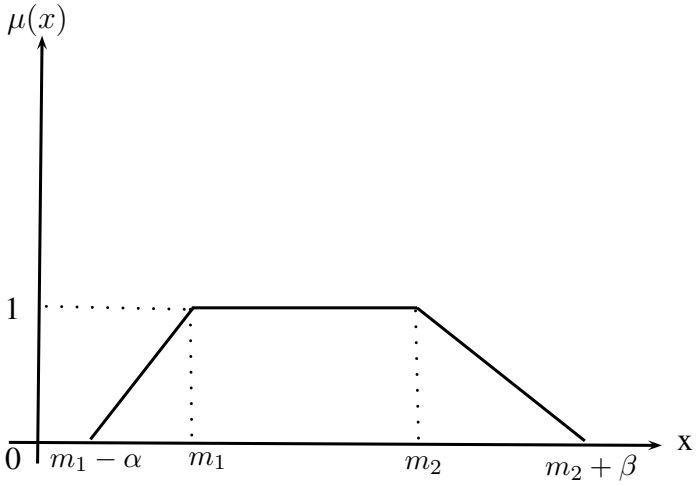

Figura 2.2: Exemplo de uma função de pertinência trapezoidal

Definição 2.2.4. *Sejam* $\tilde{a} = (m_1, \alpha_1, \beta_1)$ *e* $\tilde{b} = (m_2, \alpha_2, \beta_2)$ *dois números triangulares fuzzy, então a soma fuzzy é dada por:*

$$\tilde{a} \oplus \tilde{b} = (m_1 + m_2, \alpha_1 + \alpha_2, \beta_1 + \beta_2)$$

A soma de números trapezoidais *fuzzy* ocorre de maneira similar.

Definição 2.2.5. *Valor modal é o valor* $x \in [m-\alpha, m+\beta]$ *para o qual a função de pertinência tem valor máximo, ou seja,* 1.

2.3 Teoria de possibilidade

Seja $G = (N, A)$ um grafo com custo $\tilde{c} = \tilde{\{}c_{ij}\}$ associado aos seus arcos. Sejam dois subgrafos T^1 e T^2, $T^1 \neq T^2$. O grau de possibilidade de T^2 ter custo menor que T^1 é dado por [26]:

$$\tilde{w} = Poss\left(\sum_{(i,j)\in T^2} \tilde{c_{ij}} \leq \sum_{(i,j)\in T^1} \tilde{c_{ij}}\right) = \sup \min_x \{\mu_{T^1}(x), \mu_{T^2}(x)\} \tag{2.4}$$

onde:

- Poss: medida de Possibilidade;
- $\mu_{T^1}(x)$ e $\mu_{T^2}(x)$ são as funções de pertinência dos custos totais dos subgrafos T^1 e T^2, respectivamente;
- $\sup \min$: o valor supremo do mínimo (intersecção), ou seja, o maior grau de pertinência que pode ser obtido do conjunto resultante da intersecção das funções de pertinência $\mu_{T^1}(x)$ e $\mu_{T^2}(x)$.

Esta equação também é estudada por Dubois e Prade [8].

Para encontrar uma solução *fuzzy* utilizando a teoria de possibilidade, é preciso encontrar todas as soluções que possuem algum grau de possibilidade de ser a solução ótima e comparar estas soluções para obter o grau de possibilidade de cada uma [26]. O grau de possibilidade de T ser a solução ótima é dado por:

$$D_T = \min_{T^k \in \tau} \left\{ Poss\left(\sum_{(i,j)\in T} \tilde{c_{ij}} \leq \sum_{(i,j)\in T^k} \tilde{c_{ij}}\right)\right\} \tag{2.5}$$

onde τ é o conjunto de todas as soluções.

O grau de possibilidade do arco (i, j) pertencer à solução ótima é dado por [26]:

$$D_{ij} = \max_{T^k | (i,j)\in T^k} \{D_{T^k}\} \tag{2.6}$$

Isso torna o problema de difícil resolução pois, além de ter que enumerar todas as soluções, a comparação entre elas torna o problema NP-completo.

2.3.1 Relações de Ordem e Grafos *Fuzzy*

Na próxima seção será abordado o problema de caminho mínimo *fuzzy*, onde é necessária a comparação entre números *fuzzy*, que é feita utilizando as relações de ordem descritas a seguir.

1. **Relação de Okada e Soper [27]**

   O algoritmo de Okada e Soper tem como finalidade encontrar os caminhos não dominados, isto é, Pareto-ótimos de uma rede. Nesse algoritmo utilizou-se números triangulares *fuzzy* e foi determinado o seguinte critério de dominância *fuzzy*:

   Definição 2.3.1. *Sejam* $\tilde{a} = (m_1, \alpha_1, \beta_1)$ *e* $\tilde{b} = (m_2, \alpha_2, \beta_2)$ *dois números triangulares fuzzy, então* $\tilde{a} \prec \tilde{b}$ *(*$\tilde{a}$ *domina* $\tilde{b}$*) se e somente se* $m_1 \leq m_2$, $(m_1 - \alpha_1) \leq (m_2 - \alpha_2)$, $(m_1 + \beta_1) \leq (m_2 + \beta_2)$ *e* $\tilde{a} \neq \tilde{b}$.

2. **Primeiro índice de Yager [34, 35, 36]**

   Este índice é chamado de centróide e é definido por:

   $$f(\tilde{a}) = \frac{\int_0^1 \alpha \tilde{a_\alpha} d\alpha}{\int_0^1 \tilde{a_\alpha} d\alpha}$$

3. **Índice de Liou e Wang [21]**

   Em [21] foi proposto um método de defuzzificação que trabalha com as áreas de espalhamento dos números triangulares *fuzzy* (espalhamento à direita e espalhamento à esquerda). O índice proposto é definido a seguir.

Definição 2.3.2. *Dado um número fuzzy* $\tilde{a} = (m, \alpha, \beta)$*, o índice de Liou e Wang é dado por:*

$$LW^{\lambda}(\tilde{a}) = \lambda S_D(\tilde{a}) + (1 - \lambda) S_E(\tilde{a})$$

onde:

- S_D *é a área do espalhamento a direita;*
- S_E *é a área do espalhamento a esquerda;*
- $\lambda \in [0, 1]$ *é o grau que reflete o otimismo e o pessimismo do decisor.*

4. **Índice de García e Lamata [10]**

García e Lamata propuseram um índice que, além de trabalhar com o grau de otimismo, semelhante ao índice de Liou e Wang, inclui um grau de modalidade do usuário.

$$I(\tilde{a}) = (1 - \delta)[\lambda S_D(\tilde{a}) + (1 - \lambda) S_E(\tilde{a})] + \delta_m$$

onde:

- S_D e S_E estão definidos em Liou e Wang [21];
- δ, $\lambda \in [0, 1]$ são os graus de modalidade e otimismo do decisor, respectivamente.

5. **Índice de aceitabilidade de Nayeem e Pal [23]**

Este índice é definido para números triangulares *fuzzy*, sendo uma extensão do índice proposto por Sengupta e Pal [30] para números intervalares. Tal índice é definido a seguir.

Definição 2.3.3. *Dados dois números triangulares fuzzy* $\tilde{a} = (m_1, \alpha_1, \beta_1)$ *e* $\tilde{b} = (m_2, \alpha_2, \beta_2)$*,* $\tilde{a}$ *domina* $\tilde{b}$*, se e somente se:*

$$A(\tilde{a} \prec \tilde{b}) = \frac{m_2 - m_1}{\beta_1 + \alpha_2} \geq 1$$

Se $0 < A(\tilde{a} \prec \tilde{b}) < 1$ então $\tilde{a}$ domina $\tilde{b}$ parcialmente.

6. **Possibilidade de Dubois e Prade [8]**

   O índice de possibilidade de Dubois e Prade é definido por:

   Definição 2.3.4. *Dados dois números triangulares fuzzy,* $\tilde{a} = (m_1, \alpha_1, \beta_1)$ *e* $\tilde{b} = (m_2, \alpha_2, \beta_2)$*,* $\tilde{a}$ *domina* $\tilde{b}$*, se e somente se:*

$$Poss(\tilde{a} < \tilde{b}) > Poss(\tilde{b} < \tilde{a})$$

   Vale lembrar que a medida de possibilidade de Dubois e Prade está definida com mais detalhes na seção 2.3.

Analisando as relações de ordem citadas anteriormente, observamos que:

- Se o usuário desejar um conjunto de solução de caminhos não dominados e não somente um caminho, é interessante utilizar a relação de ordem de Okada e Soper descrita na definição 1.
- Caso esteja interessado em trabalhar com índice de defuzzificação que utilize o valor médio (centróide), deverá utilizar o primeiro índice de Yager descrito na definição 2.
- Se estiver interessado em trabalhar com índices de defuzzificação que utilizam os espalhamentos à esquerda ou direita, abordando otimismo ou pessimismo, é interessante utilizar o índice de Liou e Wang descrito na definição 3.
- Caso o interesse seja em índices que utilizam a modalidade, abordando otimismo ou pessimismo, é interessante utilizar o índice de García e Lamata descrito na definição 4.
- Se o interesse estiver no valor modal, recomenda-se o índice de Nayeem e Pal descrito na definição 5.
- Caso deseja trabalhar com medida de possibilidade, utiliza-se a de Dubois e Prade descrita na definição 6.

2.3.2 Grafos *Fuzzy*

Um grafo *fuzzy* $G = (\sigma, \mu)$ é um par de funções $\sigma : S \to [0, 1]$ e $\mu : S \times S \to [0, 1]$, onde para todo $x, y \in S$ temos $\mu(x, y) \leq \sigma(x) \wedge \sigma(y)$.

Um grafo *fuzzy* $H = (\tau, \nu)$ é um subgrafo *fuzzy* de G se:

$$\tau(x) \leq \sigma(x) \ \forall x \in S$$

e

$$\nu(x, y) \leq \mu(x, y) \ \ , \forall x, y \in S$$

Neste trabalho não trataremos esse tipo de problema.

Capítulo 3

Problemas de fluxo em redes *fuzzy*

Neste capítulo apresentaremos o problema de caminho mínimo *fuzzy* e o problema de fluxo monoproduto de custo mínimo *fuzzy*.

3.1 Problema de Caminhos Mínimos *Fuzzy*

Seja $G = (N, A)$ um grafo direcionado, onde N é o conjunto de nós e A é o conjunto de arcos. Um *caminho* entre dois nós é uma sequência alternada de nós e arcos tendo como pontos iniciais e finais um nó. Assim, o *custo* de um caminho é a soma dos custos dos arcos pertencentes a este caminho. Geralmente existe mais de um caminho entre dois nós específicos, então o problema de caminhos mínimos consiste em encontrar o caminho com menor custo dentre todos os possíveis [14].

Este problema é um dos mais importantes e mais estudados na teoria de grafos, pois aparece em diversas aplicações, tais como: transporte, telecomunicações, etc.

O problema de caminhos mínimos *fuzzy* é de alta complexidade computacional, uma vez que os custos finais dos caminhos são números *fuzzy*, o que torna difícil a comparação entre os mesmos, e às vezes é impossível determinar o menor caminho. Embora este assunto tenha um enorme campo de aplicações, o estudo considerando incertezas ainda encontra-se

na fase inicial.

3.1.1 Revisão bibliográfica

Em problemas com a estrutura e parâmetros bem definidos (crisp), existem algoritmos que encontram o caminho de menor custo (ou comprimento) de forma eficiente [1]. Contudo, é comum encontrarmos problemas onde os parâmetros e/ou a própria estrutura possuem incertezas. Por exemplo, em um roteiro de viagem, o tempo de translado de uma cidade (nó) i a uma cidade j pode ser considerado um parâmetro sujeito a incertezas, tais como: congestionamento, condições da rodovia, condições climáticas etc. Portanto, um problema pode ter diferentes níveis de incerteza:

- Problemas onde a estrutura do grafo é bem definida (grafo crisp) e os parâmetros associados possuem incertezas. Estas incertezas podem estar associadas a parâmetros tais como: custo, capacidade e tempo. Esses parâmetros são modelados como números *fuzzy*;

- Problemas envolvendo um conjunto finito de itens cuja estrutura associada não é clara e é modelada como um conjunto de nós e arcos *fuzzy*, ou seja, grafo *fuzzy*.

O primeiro caso é o mais estudado, existindo assim, uma literatura significativa sobre o mesmo.

Dentre os principais trabalhos da literatura destacam-se: Dubois e Prade [8], Klein [17], Okada e Gen [24], Li, Gen e Ida [18], Lin e Chern [19], Chang e Lee [6], Okada e Soper [27], Okada [26], Nayeem e Pal [23], Chuang e Kung [7], Ji e Shao [16] e Blue et al. [4]. Os trabalhos de Takahashi [33] e Hernandes [14] também tratam deste assunto.

Uma das principais referências tratando este assunto é o trabalho de Dubois e Prade [8], no qual é feita uma extensão de dois algoritmos clássicos do problema do caminho mínimo (algoritmos de Floyd e de Ford-Bellman) para a teoria *fuzzy*. Porém ambos os

algoritmos podem retornar custos sem um caminho associado [33], problema este que foi contornado por Klein [17] com o uso de dominância de conjuntos *fuzzy*.

Os trabalhos de Okada e Gen [24, 25] tratam de um caso especial de problemas de caminhos mínimos *fuzzy*, no qual foi feita uma generalização do algoritmo de Djikstra, mas com o uso de intervalos no lugar de comprimento dos arcos e conceito de ordenação parcial dos mesmos.

Lin e Chern [19] propõe um algoritmo para encontrar arcos vitais em uma rede. Arcos vitais são aqueles em que a remoção dos mesmos resulta em um aumento da distância mínima.

Li, Gen e Ida [18] apresentam um algoritmo para resolver problema de caminho mínimo *fuzzy* utilizando redes neurais. Para tanto, o problema de caminho mínimo *fuzzy* é transformado em um problema crisp.

Chang e Lee [6] utilizam um método de ordenação de números *fuzzy* que associa a cada número *fuzzy* um valor crisp e através deste resolve um problema clássico associado.

Okada e Soper [27] introduziram o conceito de dominância de caminhos e definiram uma relação de ordem entre números *fuzzy*. No algoritmo proposto, a solução do problema não é o caminho mínimo com maior grau de pertinência, mas sim um conjunto solução *fuzzy*, onde cada elemento é um caminho não dominado.

Considerando a idéia de encontrar um conjunto solução, Okada [26] analisou a interatividade entre caminhos *fuzzy* e introduziu o conceito de grau de possibilidade de um arco pertencer a um caminho mínimo. Ele definiu também um novo índice de comparação entre a soma de números *fuzzy* considerando interatividade entre números *fuzzy*.

Blue et al. [4] apresentam uma taxionomia para grafos fuzzy e propõem alguns algoritmos para os problemas de caminho mínimo e corte mínimo.

Em Chuang e Kung [7] é proposto um algoritmo para o problema de caminhos mínimos discretos *fuzzy*. Um método de comprimento mínimo discreto *fuzzy* é proposto para encontrar o comprimento mínimo *fuzzy* e medidas de similaridade são utilizadas para

encontrar o caminho mínimo.

Nayeem e Pal [23] propõem um algoritmo utilizando o índice de aceitabilidade proposto por Sengupta e Pal [30], onde o usuário escolhe, de acordo com seu ponto de vista (otimista/pessimista) o melhor caminho. Mas esta escolha só é feita quando, na solução final, houver mais de um caminho entre dois nós, sendo que isso só ocorrerá se mais de um caminho possuir os mesmos valores modais, pois, caso contrário, sempre haverá um caminho mínimo.

No recente trabalho de Ji, Iwamura e Shao [16] são propostos os conceitos de: caminho mínimo esperado, o caminho α-mínimo e o caminho mínimo absoluto e três tipos de modelos são formulados. Para resolver esses modelos é proposto um algoritmo inteligente híbrido integrando simulação e algoritmos genéticos.

Uma desvantagem comum em todos os trabalhos citados nesta seção é o fato de lidarem somente com parâmetros positivos.

Para contornar este problema, Hernandes [14], propôs um algoritmo baseado no algoritmo clássico de Ford-Moore-Bellman [3]. Em tal algoritmo, para a construção do conjunto solução, foi utilizado o conceito de dominância de caminhos de Okada e Soper [27].

Neste trabalho utilizamos o algoritmo proposto por Hernandes [14] para encontrar o conjunto de caminhos não-dominados entre cada par de nó origem-destino, para posteriormente encontrar as soluções factíveis para o problema de fluxo multiproduto *fuzzy* com múltiplas origens-múltiplos destinos.

3.1.2 Formulação matemática do problema

Seja $G = (N, A)$ um grafo direcionado, onde N é o conjunto de nós e A é o conjunto de arcos e cada arco é denotado por um par ordenado $(i, j) \in A$, $i, j \in N$. Considere também que há um único arco direcionado (i, j) de i para j e que há pelo menos um caminho em $G = (N, A)$ entre s e t, onde s é o nó origem e t é o nó destino. O problema de caminho mínimo *fuzzy* é formulado da seguinte maneira [26]:

$$\begin{aligned} \min \ \ & \tilde{f}(x) = \sum_{(i,j)\in A} \tilde{c_{ij}} x_{ij} \\ \text{s.a} \ & \sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1, & i = s \\ 0, & i \neq s,\, t\,(i = 1, \ldots, r) \\ -1, & i = t \end{cases} \end{aligned} \tag{3.1}$$

onde r: número de nós e $\sum$ refere-se à soma de números *fuzzy*.

O significado de min na equação 3.1 é ambíguo, pois depende do índice de ordenação de números *fuzzy* utilizado e a solução ótima pode não ser obtida.

3.1.3 Algoritmo proposto por Hernandes

Como dito anteriormente, o algoritmo proposto por Hernandes [14] é baseado no algoritmo clássico de Ford-Moore-Bellman [3]. O algoritmo proposto é iterativo, tendo como critério de parada o número de iterações ou a não alteração dos custos de todos os caminhos encontrados na iteração anterior com relação a iteração atual. Em tal algoritmo, para a construção do conjunto solução, foi utilizado o conceito de dominância de caminhos de Okada e Soper [27] descrita na seção 2.3.1, então cada caminho recebe um rótulo (etiqueta) para que este seja construído no final do algoritmo, visto que a relação de Okada e Soper pode apresentar, entre dois nós, mais de um caminho não dominado.

Notações utilizadas no algoritmo

N: conjunto dos nós;

it: contador de iterações;

$(m + \beta)^i$ limitante superior do custo do nó i;

$\tilde{l}_{ji}$: custo do arco (j, i);

$\tilde{c}^{it}_{(i,k)}$: custo do caminho entre os nós 1 e i, com rótulo k na iteração it;

na: número de arcos;

r: número de nós;

$M = \sum_{i=1}^{na} \mid (m+\beta)^i \mid$, valor grande que substitui ∞ do algoritmo clássico de Ford-Moore-Bellman;

Γ_i^{-1}: conjunto dos nós predecessores do nó i.

Algoritmo

Passo 1: Inicialização

- $\tilde{c}^0_{(1,1)} = (0,0,0)$
- $\tilde{c}^0_{(j,1)} = (M+1, M, M+2) \;\; j = 2,3,\ldots,r$
- $it \leftarrow 1$.

Passo 2: Determinação dos caminhos e verificação da dominância

- $\tilde{c}^{it}_{(1,1)} = (0,0,0)$
- $\forall j \in \Gamma_i^{-1} \;\; i = 2,3,\ldots,r$, faça:
- $\tilde{c}^{it}_{(i,k_1)} = \tilde{c}^{it-1}_{(j,k_2)} \oplus \tilde{l}_{ji}$
- Verificação da dominância entre as etiquetas do nó i, utilizando a relação de ordem de Okada e Soper:
  - Se $\tilde{c}^{it}_{(i,m)} \succ \tilde{c}^{it}_{(i,n)}$ então elimine a m-ésima etiqueta
  - Se $\tilde{c}^{it}_{(i,m)} \prec \tilde{c}^{it}_{(i,n)}$ então elimine a n-ésima etiqueta

Passo 3: Critério de parada

- Se $(\tilde{c}^{it}_{(i,k_1)} = \tilde{c}^{it-1}_{(i,k_1)} \;\; \forall i \in N)$ ou $it = r$ faça:
  - Se $it = r$ ou $\tilde{c}^{it}_{(i,k_1)} = \tilde{c}^{it-1}_{(i,k_1)}$ vá para o passo 5;
  - Senão vá para o passo 4.
- Senão: $it \leftarrow it + 1$, vá para o passo 2.

Passo 4: Composição dos caminhos

- Encontre todos os caminhos não-dominados entre os nós 1 e i.

Passo 5: Termine a execução do algoritmo.

Observação 3.1.1. *O algoritmo foi proposto para números triangulares, mas pode ser adaptado para números trapezoidais sem grande dificuldade.*

Convergência e Complexidade

O algoritmo de Ford-Moore-Bellman examina todos os nós até que não seja possível encontrar melhorias [12], dessa forma, aceita arcos com custos negativos. Como o algoritmo proposto é baseado no algoritmo de Ford-Moore-Bellman, a estrutura é mantida.

A análise de convergência do algoritmo proposto é semelhante a do algoritmo de Ford-Moore-Bellman, ou seja, no caso de não existir circuito negativo, o algoritmo converge em no máximo $r-1$ iterações, onde r é o número de nós.

A complexidade do algoritmo proposto é $O(r^2V_{max}^2)$ [14].

Observação 3.1.2. *O algoritmo proposto por Hernandes [14] pode ser aplicado em redes com custos quaisquer e possui complexidade computacional* ($O(r^2V_{max}^2)$) *enquanto o algoritmo de Okada e Soper [27] possui complexidade* ($O(r^3V_{max}^2)$) *[14].*

3.2 Algoritmos de fluxo em redes *fuzzy*

O problema de fluxo de custo mínimo (PFCM) também é um problema muito estudado na teria de grafos. Este problema consiste em atender, com custo mínimo, a demanda em uma rede, dada a oferta de recursos e as restrições de capacidade dos arcos. O problema de caminho mínimo descrito na seção 3.1 é um caso especial do PFCM.

Geralmente, existem restrições de fluxo associadas a cada arco (mínimo e máximo) e é associado um custo referente ao trânsito em cada arco, por unidade de fluxo (c_{ij}).

Este problema tem aplicações em diversar áreas, tais como: transportes, comunicações, distribuição em redes de TV a cabo, entre outros [1].

Na literatura são encontrados poucos artigos que abordam o problema de fluxo de custo mínimo *fuzzy*, onde as incertezas apresentadas estão nos custos e/ou nas capacidades. O principal trabalho é o de Shih e Lee [31], no qual os autores fazem uma adaptação do método Húngaro para encontrar uma solução para o problema de fluxo de custo mínimo *fuzzy* em uma rede com custos e capacidades *fuzzy*.Neste método é utilizada programação possibilística para problemas com múltiplos níveis, no qual o problema resultante a ser resolvido é crisp. No trabalho de Liou e Kao [20] é utilizado o índice de defuzzificação de Yager, transformando o custo *fuzzy* em crisp, e após é aplicado um algoritmo clássico para encontrar a solução ótima do problema.

O trabalho de Takahashi [33] é um dos mais recentes, no qual são propostos três algoritmos para o PFCM *fuzzy*: um considerando os custos *fuzzy*, outro considerando as capacidades *fuzzy* e o último considerando os custos e as capacidades *fuzzy*. Para o primeiro caso, o algoritmo de Takahashi é baseado em α-cortes, onde um problema crisp é resolvido para cada valor de α. No segundo caso, afim de encontrar as soluções factíveis, as capacidades dos arcos foram utilizadas para desviar o fluxo para rotas alternativas, com objetivo de enumerar todas as soluções possíveis. O terceiro algoritmo é a união dos dois primeiros, mas só pode ser aplicado em problemas de pequeno porte visto que é necessário enumerar as várias soluções possíveis.

Hernandes [14] também tratou deste problema. Em seu trabalho, ele propôs alguns algoritmos para resolver o PFCM *fuzzy* com custos e capacidades incertas. O primeiro algoritmo proposto é uma adaptação do método do Big-M, transformando o problema *fuzzy* em um problema clássico. Em tal algoritmo, as incertezas nos custos e nas capacidades são tratadas de formas independentes. Para as capacidades são aplicados α-cortes e depois encontradas as soluções com grau de pertinência iguais a α. Os custos são abordados de duas maneiras, na primeira é aplicado o primeiro índice de defuzzificação de Yager [36]; na

segunda é utilizado o valor modal.

Os demais algoritmos de Hernandes [14] propostos trabalham com o problema *fuzzy*, onde quatro algoritmos foram propostos. O primeiro para o PFCM *fuzzy* com um nó origem e um nó destino, o segundo para um nó origem e múltiplos nós destinos, o terceiro, para múltiplos nós origens e um nó destino e o último, para múltiplos nós origens e múltiplos nós destinos. Os quatro algoritmos citados são diferentes dos existentes na literatura e são os únicos, que se tem conhecimento, que tratam do PFCM *fuzzy* sem transformá-lo num problema crisp.

3.2.1 Formulação do problema de fluxo de custo mínimo

Seja uma rede direcionada $G = (N, A)$ com custo c_{ij} associado a cada unidade de fluxo, x_{ij} , que passa pelo arco $(i, j) \in A$. Para cada arco (i, j), temos uma capacidade mínima l_{ij} e máxima u_{ij} de fluxo, e para cada nó tem-se um parâmetro b_i, tal que:

- $b_i > 0$, se i for um nó gerador (nó origem);
- $b_i = 0$, se i for um nó de passagem;
- $b_i < 0$, se i for um nó consumidor (nó destino).

Considera-se que $\sum_{i \in N} b_i = 0$, ou seja, a quantidade total de demanda é igual a quantidade de matéria prima disponível nos nós geradores.

O PFCM clássico é formulado da seguinte forma:

$$\begin{aligned} \min \quad & z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \text{s.a} \quad & \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = b_i, \quad \forall i \in N \\ & l_{ij} \leq x_{ij} \leq u_{ij}, \quad (i, j) \in A \end{aligned} \tag{3.2}$$

no qual a primeira equação de 3.2 é a função objetivo do problema (minimizar o custo total do fluxo na rede), a segunda é referente a conservação de fluxo em cada nó e a terceira é a restrição de capacidade dos arcos. Vamos considerar, sem perda de generalidade, que $l_{ij} = 0$.

Existem vários algoritmos desenvolvidos para a resolução do PFCM. Entre os mais conhecidos estão: o método simplex para redes e o algoritmo *Out-of-Kilter*, que estão descritos em [2].

3.2.2 Formulação do PFCM *fuzzy*

Seja um PFCM com custos e capacidades incertos. Este problema é formulado da seguinte forma:

$$\begin{aligned} \min \quad & z = \sum_{i=1}^{n}\sum_{j=1}^{n} \tilde{c}_{ij} x_{ij} \\ \text{s.a} \quad & \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = b_i, \quad \forall i \in N \\ & \tilde{l}_{ij} \leq x_{ij} \leq \tilde{u}_{ij}, \quad (i,j) \in A \end{aligned} \tag{3.3}$$

onde: $\tilde{c}_{ij}$, $\tilde{l}_{ij}$ e $\tilde{u}_{ij}$ são, respectivamente, o custo *fuzzy* e os limitantes inferior e superior da capacidade.

Sem perda de generalidade, os custos são números triangulares *fuzzy*, escritos na forma: (m, α, β) e as capacidades são números trapezoidais *fuzzy* escritos na forma: $(m_1 - \alpha, m_1, m_2, m_2 + \beta)$, onde $\alpha = m_1 = 0$ e denotamos $\underline{\alpha} = m_1 - \alpha$ e $\overline{\alpha} = m_2 + \beta$. Logo a capacidade é escrita da seguinte forma: $(\underline{\alpha}, m_1, m_2, \overline{\alpha})$.

3.2.3 Algoritmo proposto por Hernandes para o PFCM *fuzzy*

Como dito anteriormente, Hernandes [14] propôs quatro algoritmos para o PFCM *fuzzy*. Apresentaremos a seguir o algoritmo para o PFCM *fuzzy* com múltiplas origens-múltiplos destinos. Tal algoritmo utiliza o índice de possibilidade de Dubois e Prade descrito na seção 2.3.1, para ordenação dos custos dos caminhos não dominados. Esta ordenação é

feita da seguinte forma: após determinar os caminhos não dominados para cada par de nós origem-destino, através dessa medida de possibilidade, verifica-se qual a possibilidade de cada caminho ter custo menor em relação aos demais. Depois verifica-se também a pertinência das capacidades de cada caminho.

Algoritmo

Passo 1: Encontrar os caminhos mínimos não dominados para cada par de nós origem-destino, com possibilidade de serem mínimos ($\geq \alpha$).

Passo 2: Ordenar todos os caminhos p_k:
μ_{cam}=$Poss\{p_k$ ser o melhor caminho$\}$= min$\{\mu_{custo}, \mu_{capac}\}$ onde:

- $\mu_{custo} = Poss\{p_k$ ser mínimo$\}$.
- $\mu_{capac} = \min_{(i,j)\in p_k}\{\mu_{capac(i,j)}\} = Poss\{$fluxo passar no arco$(i,j)$ através de $p_k\}$.

Passo 3: Envio de fluxo:

- Enviar fluxo para o melhor caminho (de acordo com μ_{p_k}) até que: $\mu_{p_k} = \mu_{p_{k+1}}$. Caso $\mu_{p_k} = \mu_{p_{k+1}}$, preencher este caminho até o caminho com pertinência diferente.
- Atualizar $\mu_{p_k} \forall p_k$ e reordená-lo baseado em: $\mu_{p_k} = \min\{\mu_{custo_{p_k}}, \mu_{capac_{p_k}}\}$.
- Se o fluxo necessário ao nó destino do primeiro caminho ordenado já foi satisfeito, eliminar todos os caminhos não dominados que tenham tal nó como nó destino e voltar ao Passo 2 do algoritmo. Senão, voltar a primeira etapa deste passo.

Passo 4: Critério de parada.

- Se existe fluxo a transitar vá para o Passo 3.
- Senão $\Rightarrow$ fim.

O algoritmo descrito acima possui a vantagem de abordar as incertezas em dois parâmetros (custos e capacidades), podendo ser aplicável a problemas de grande porte [14]. Como dito anteriormente, este algoritmo resolve o problema de fluxo monoproduto de custo mínimo *fuzzy*. No capítulo a seguir, apresentamos o problema de fluxo multiproduto de custo mínimo *fuzzy*, que é uma generalização do problema descrito neste capítulo.

Capítulo 4

Problema de Fluxo Multiproduto *Fuzzy*

O problema de fluxo multiproduto também é estudado em grafos. Este problema surge quando vários produtos compartilham os arcos em uma rede e competem pela capacidade dos mesmos. O objetivo é determinar o fluxo dos produtos em cada arco, a um custo mínimo, de modo a atender as restrições de capacidade dos arcos e as restrições de conservação de fluxo.

As restrições de capacidade dos arcos limitam o fluxo dos produtos, de modo que em nenhum arco trafegue uma quantidade de produtos superior à capacidade suportada por ele. Já as restrições de conservação de fluxo gerenciam o fluxo dos produtos pelos arcos da rede, que saem de um ponto de oferta e chegam em um ponto de demanda. Cada produto pode ser transportado de um ou vários nós origens para um ou vários nós destinos da rede (grafo) [5].

Geralmente é associado um custo referente ao trânsito em cada arco, por unidade de fluxo. Estes custos podem ser os mesmos para todos os produtos ou podem ser diferentes para cada produto. Neste trabalho, propomos um algoritmo para abordar problemas de fluxo multiproduto.

Os problemas de fluxo multiproduto, de um modo geral, têm recebido muita atenção devido a aplicabilidade na resolução de problemas atuais presentes nas mais diversas áreas, como transportes, telecomunicações, entre outras. A dificuldade prática de se resolver esse tipo de problema aumenta de forma muito rápida à medida que o mesmo cresce em número de variáveis e, principalmente, em relação ao número de produtos. Aplicações em telecomunicações, por exemplo, tipicamente conduzem a instâncias com um grande número de produtos. Resolver tais instâncias torna-se, portanto, um grande desafio [5].

Um fato que tem contribuído para o aumento de interesse no estudo deste problema é que, apesar de o problema multiproduto ser modelado como um problema de programação matemática genérico, de um modo geral, as instâncias que surgem são muito grandes. Problemas com um pequeno número de arcos, nós e produtos geram modelos com um grande número de variáveis e restrições, o que torna inviável a utilização de métodos de programação matemática genéricos [5].

Este capítulo está organizado da seguinte forma: na seção 1 é apresentada a formulação matemática do problema de fluxo multiproduto *fuzzy*. O algoritmo proposto está na seção 2.

4.1 Formulação do problema de fluxo multiproduto

Seja $G = (N, A)$ um grafo, onde N é o conjunto de nós e A é o conjunto de arcos. Cada arco é denotado por (i, j) onde $i, j \in N$. O problema de fluxo multiproduto *fuzzy* é formulado como o seguinte problema de programação linear:

$$\min \ z = \sum_{k=1}^{K} \sum_{(i,j)\in A} \tilde{c}_{ij}^{k} x_{ij}^{k}$$

$$\text{s.a} \begin{cases} \sum\limits_{j:(i,j)\in A} x_{ij}^{k} - \sum\limits_{j:(j,i)\in A} x_{ji}^{k} = b_i^k, \forall i \in N\,, \forall k = 1, \ldots, K \\ \tilde{l}_{ij} \leq \sum\limits_{k=1}^{K} x_{ij}^{k} \leq \tilde{u}_{ij}, \ \ \forall (i,j) \in A \end{cases} \tag{4.1}$$

onde:

- x_{ij}^{k} é o fluxo do produto k que percorre o arco (i, j);
- $\tilde{c}_{ij}^{k}$ é o custo unitário (o mesmo para todos os produtos) *fuzzy* do produto k associado a cada unidade de fluxo x_{ij} que percorre o arco (i, j);
- $\tilde{l}_{ij}$ é o limitante inferior *fuzzy* da capacidade do arco (i, j);
- $\tilde{u}_{ij}$ é o limitante superior *fuzzy* da capacidade do arco (i, j);
- K é o número total de produtos;
- b_i^k é a oferta ou demanda do produto k:
  - Se $b_i^k > 0$, i é nó gerador do produto k
  - Se $b_i^k < 0$, i é nó consumidor do produto k
  - Se $b_i^k = 0$, i é nó de passagem do produto k

Sem perda de generalidade, os custos são números triangulares *fuzzy*, escritos na forma: (m, α, β) e as capacidades são números trapezoidais *fuzzy* escritos na forma: $(m_1 - \alpha, m_1, m_2, m_2 + \beta)$, onde $\alpha = m_1 = 0$ e denotamos $\underline{\alpha} = m_1 - \alpha$ e $\overline{\alpha} = m_2 + \beta$. Logo a capacidade é escrita da seguinte forma: $(\underline{\alpha}, m_1, m_2, \overline{\alpha})$.

4.2 Algoritmo proposto

Nesta seção apresentamos o algoritmo proposto e uma explicação detalhada sobre o mesmo.

- Passo 1: Encontrar os caminhos mínimos não dominados para cada par de nós origem-destino de cada produto;

  Para encontrar o conjunto de todos os caminhos não dominados para cada par de nós origem-destino de cada produto, utilizamos o algoritmo proposto por Hernandes, descrito na seção 3.1.3. Tal algoritmo é baseado no algoritmo clássico de Ford-Moore-Bellman [3]. Trata-se de um algoritmo iterativo, tendo como critério de parada o número de iterações ou a não alteração dos custos de todos os caminhos encontrados na iteração anterior com relação à iteração atual. Desta forma, são encontrados todos os caminhos não-dominados entre os nós origem e destino de cada produto aplicando a relação de ordem de Okada e Soper [27], descrita na seção 2.3.1, para descartar os caminhos dominados. Note que é necessário identificar os caminhos não-dominados encontrados para cada produto, de forma que, ao ordenarmos e enviarmos fluxo, sabemos qual é o produto em questão.

- Passo 2: Ordenar todos os caminhos p_k:

  μ_{cam}=$Poss\{p_k$ ser o melhor caminho$\}$= $\min\{\mu_{custo}, \mu_{capac}\}$, onde:

  1. $\mu_{custo} = Poss\{p_k \text{ ser mínimo}\}$
  2. $\mu_{capac} = \min_{(i,j)\in p_k}\{\mu_{capac}(i,j)\} = Poss\{\text{fluxo passar no arco } (i,j) \text{ através de } p_k\}$

  Para fazer a ordenação dos caminhos:

  – Calcular μ_{custo} para cada caminho, ou seja, verificar através da medida de possibilidade de Dubois e Prade, qual a possibilidade de cada caminho ter o custo menor em relação aos demais;

- Selecionar os caminhos não dominados que satisfazem a pertinência mínima para o fluxo transitar na rede (α);
- Calcular μ_{capac} para cada caminho, ou seja, verificar a possibilidade dos arcos de determinado caminho pertencerem à solução ótima;
- Calcular a pertinência de cada caminho: $\mu_{cam} = \min\{\mu_{custo}, \mu_{capac}\}$;
- Colocar os caminhos em ordem decrescente de acordo com μ_{cam}. Se houver empate, usar o valor modal dos custos dos caminhos: escolher aquele com menor custo. Se ainda empatar, assumir a ordem que aparece.

Exemplo ilustrativo: Ordenação dos caminhos

Dados três caminhos entre os nós 1 e 6 da Figura 4.1, $p_1 : 1 \rightarrow 2 \rightarrow 4 \rightarrow 6$, $p_2 : 1 \rightarrow 3 \rightarrow 4 \rightarrow 6$ e $p_3 : 1 \rightarrow 2 \rightarrow 5 \rightarrow 6$, com respectivos custos $(234,\ 75,\ 15)$, $(222,\ 62,\ 13)$ e $(195,\ 18,\ 61)$, ordená-los utilizando a medida de possibilidade de Dubois e Prade.

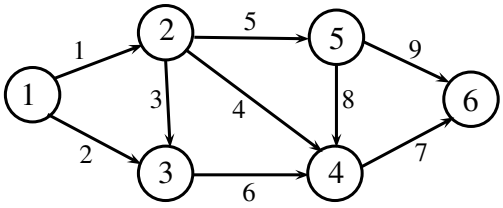

Figura 4.1: Rede exemplo

Na figura 4.2, temos as funções de pertinência dos custos dos caminhos p_1, p_2 e p_3.

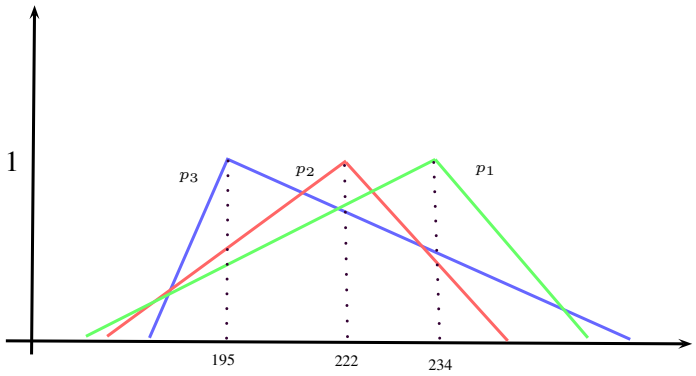

Figura 4.2: Funções de pertinência dos custos de p_1, p_2 e p_3

Solução:

- $\text{Poss}(p_1 < p_2) = 0,8636, \text{Poss}(p_1 < p_3) = 0,7132 \rightarrow \text{Poss}(p_1 \text{ ser mínimo}) = 0,7132 \rightarrow \mu_{custo}(p_1) = 0,7132$
- $\text{Poss}(p_2 < p_1) = 1, \text{Poss}(p_2 < p_3) = 0,7805 \rightarrow \text{Poss}(p_2 \text{ ser mínimo}) = 0,7805 \rightarrow \mu_{custo}(p_2) = 0,7805$
- $\text{Poss}(p_3 < p_1) = 1, \text{Poss}(p_3 < p_1) = 1 \rightarrow \text{Poss}(p_3 \text{ ser mínimo}) = 1 \rightarrow \mu_{custo}(p_3) = 1$

Ordenação dos caminhos: p_3, p_2 e p_1.

- Passo 3: Envio de fluxo :
  1. Enviar fluxo para o primeiro caminho ordenado até alcançar a próxima pertinência diferente desse caminho;
  2. Se o fluxo necessário ao nó destino do caminho em questão já foi satisfeito, eliminar todos os caminhos não dominados do produto em questão que tenham tal nó como nó destino;

3. Atualizar μ_{capac}, $\forall p_k$.

Para enviar fluxo pelos caminhos de acordo com a ordenação feita:

- Verificar se existe oferta no nó origem do melhor caminho, caso não haja, ir para o próximo melhor caminho com oferta;
- Encontrar a próxima pertinência diferente (se houver) a do melhor caminho (denotaremos por μ_{fluxo}) para calcular o fluxo a ser enviado nos arcos. Caso este seja zero e a pertinência do melhor caminho for diferente de zero, usamos ela própria.

Exemplo ilustrativo: Pertinência das capacidades dos caminhos

Considerando que são percorridos $8,439$ unidades de fluxo nos arcos do caminho p_3, calcular a pertinência da capacidade de p_3 (μ_{capac}).

Tabela 4.1: Capacidades dos arcos da Figura 4.1

Arcos	Capacidades
1	(0, 0, 8, 10)
2	(0, 0, 12, 16)
3	(0, 0, 20, 22)
4	(0, 0, 11, 18)
5	(0, 0, 16, 18)
6	(0, 0, 10, 15)
7	(0, 0, 15, 20)
8	(0, 0, 16, 18)
9	(0, 0, 23, 27)

Solução:

Analisando as capacidades do arcos de p_3 temos:

- O arco $(1,2)$ tem capacidade (0, 0, 8, 10), calculando a pertinência desse arco temos:

$$\mu(1,2) = \frac{(m+\beta) - x}{\beta} = \frac{10 - 8,439}{2} = 0,7805$$

A fórmula acima vem da definição de número trapeziodal *fuzzy* descrito na seção 2.2.

– O arco $(2,5)$ tem capacidade $(0,\ 0,\ 16,\ 18)$, como a quantidade de fluxo percorrida nesse arco é menor que 16, a pertinência desse arco é 1.

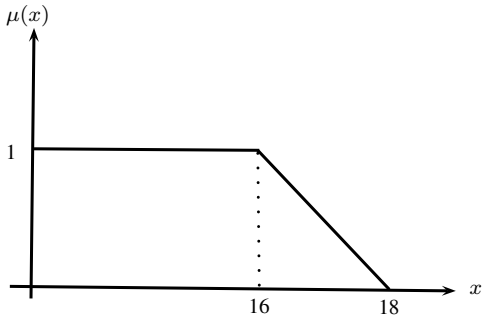

Figura 4.3: Capacidade do arco $(2,5)$

– Arco $(5,6)$, idem ao arco $(2,5)$.

– Portanto, $\mu_{capac}(p_3) = \min\{0,7805,\ 1,\ 1\} = 0,7805$

Observação 4.2.1. *Conforme as demandas forem atendidas eliminamos os caminhos correspondentes, podendo ocasionar a sobra de somente um caminho, neste caso, relaxamos as pertinências dos arcos até o limitante superior. Fazemos o mesmo quando temos vários caminhos e somente no último há oferta no nó origem.*

– Para calcular o fluxo a ser enviado, primeiramente calculamos o fluxo permitido em cada arco do caminho em questão, usando μ_{fluxo}. A seguir, calculamos a diferença entre o fluxo permitido e o fluxo já existente para cada arco do caminho e tomamos o mínimo.

– Se $\mu_{fluxo} = \mu_{cam}$, então pode acontecer de $\mu_{cam} = \mu_{capac}$ e portanto o fluxo permitido será igual ao fluxo presente no arco, logo não enviamos fluxo algum. Neste caso, utilizamos o próximo melhor caminho para enviar fluxo ou a seguinte opção:

Opção 1. Recalcular o fluxo da seguinte forma:

$$\min_{(i,j)\in p_k} \left\{ \frac{\overline{\alpha} \text{ do arco}(i,j) - \text{fluxo no arco}(i,j)}{2} \right\}.$$

onde $\overline{\alpha}$ é o limitante superior do arco (i, j).

Dessa forma, enviaremos uma certa quantidade de fluxo, a qual garante que não alcançará o limitante superior de nenhum arco do caminho p_k.

- Passo 4: Critério de parada.

  1. Se existir fluxo a transitar ir para o Passo 2.
  2. Senão ⇒ fim.

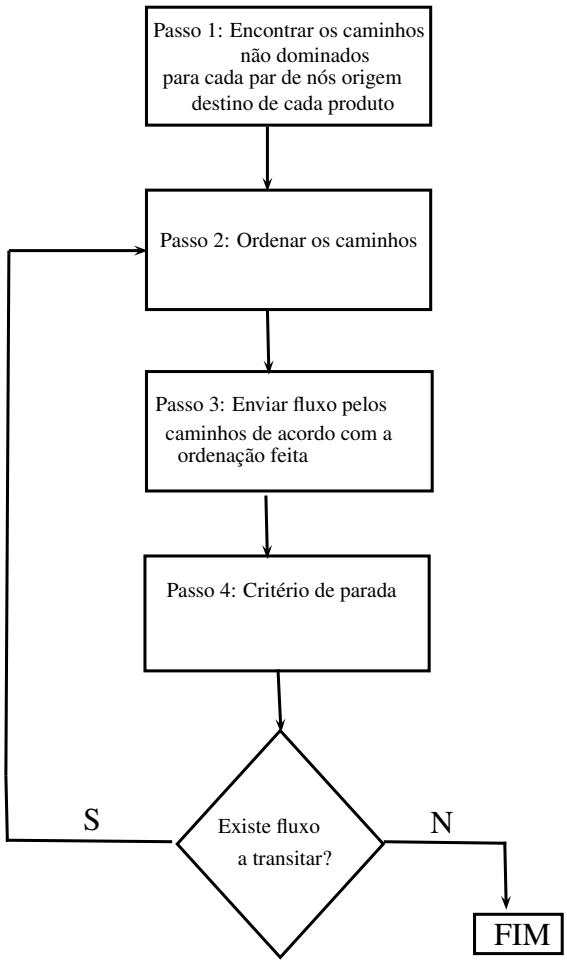

Figura 4.4: Fluxograma do algoritmo proposto

No próximo capítulo apresentamos os testes computacionais feitos utilizando o algoritmo proposto neste capítulo.

Capítulo 5

Testes Computacionais

Neste capítulo apresentaremos os testes computacionais realizados em algumas instâncias com o algoritmo prosposto no capítulo anterior, implementado em Matlab 7.0.1 e executado em uma plataforma Intel Core Duo, 1.73 GHz e 2 Gb de memória RAM, com sistema operacional Windons Vista Business.

A primeira instância é apresentada na seção 5.1. É uma rede pequena contendo seis nós e nove arcos, com finalidade didática pois permite uma análise detalhada.

A segunda instância é apresentada na seção 5.2, maior que a primeira, com finalidade de mostrar a funcionalidade e eficiência do algoritmo proposto.

A terceira instância é apresentada na seção 5.3. É uma rede ferroviária, um pouco maior que as duas instâncias apresentadas anteriormente.

A análise dos resultados será feita baseando-se no fluxo final de cada produto, custo final de cada produto e na pertinência final. O custo final é calculado através da multiplicação dos custos dos arcos com a quantidade de fluxo percorrido em cada arco. A pertinência final é o valor mínimo dentre as pertinências dos arcos.

5.1 Instância 1

Caso 1: O primeiro exemplo foi resolvido utilizando o grafo apresentado na Figura 5.1, representado pelos dados da Tabela 5.1. Consideramos dois produtos, os quais estão descritos na Tabela 5.2.

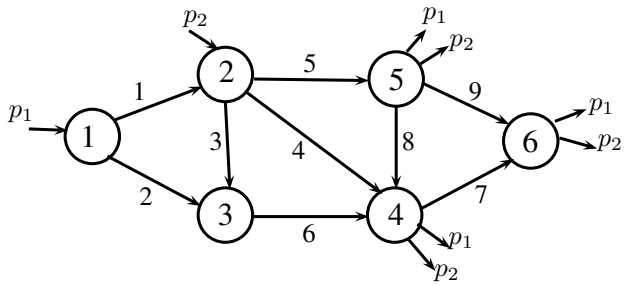

Figura 5.1: Rede com dois produtos

Tabela 5.1: Custos e capacidades da rede da Figura 5.1

Arcos	Custos	Capacidades
1	(2, 1, 1)	(0, 0, 3, 4)
2	(3, 2, 1)	(0, 0, 1,5, 2,5)
3	(3, 3, 2)	(0 0 2 3)
4	(6, 4, 2)	(0, 0, 2, 4)
5	(5, 1, 1)	(0, 0, 7, 8)
6	(5, 2, 1)	(0, 0, 3, 4)
7	(2, 1, 1)	(0, 0, 4, 6)
8	(5, 4, 2)	(0, 0, 4, 5)
9	(3, 1, 1)	(0, 0, 2, 3)

Tabela 5.2: Dados de cada produto

produto	origem	destinos	oferta	demanda
p_1	1	{4, 5, 6}	5	{2,3333; 1,3333; 1,3333}
p_2	2	{4, 5, 6}	2	{0,6667; 0,6667; 0,6667}

Aplicando o algoritmo proposto passo a passo tem-se:

1^a iteração

Passo 1: Seja $\alpha = 0$. Encontrar os caminhos não-dominados para cada par de nós origem-destino de cada produto. Caminhos encontrados para p_1:

- Caminho 1 (cam_1): $1 \rightarrow 2 \rightarrow 4$ com custo (8, 5, 3).
- Caminho 2 (cam_2): $1 \rightarrow 3 \rightarrow 4$ com custo (8, 4, 2).
- Caminho 3 (cam_3): $1 \rightarrow 2 \rightarrow 5$ com custo (7, 2, 2).
- Caminho 4 (cam_4): $1 \rightarrow 2 \rightarrow 4 \rightarrow 6$ com custo (10, 6, 4).
- Caminho 5 (cam_5): $1 \rightarrow 3 \rightarrow 4 \rightarrow 6$ com custo (10, 5, 3).
- Caminho 6 (cam_6): $1 \rightarrow 2 \rightarrow 5 \rightarrow 6$ com custo (10, 3, 3).

Caminhos encontrados para p_2:

- Caminho 7 (cam_7): $2 \rightarrow 4$ com custo (6, 4, 2).
- Caminho 8 (cam_8): $2 \rightarrow 5$ com custo (5, 1, 1).
- Caminho 9 (cam_9): $2 \rightarrow 4 \rightarrow 6$ com custo (8, 5, 3).
- Caminho 10 (cam_{10}): $2 \rightarrow 5 \rightarrow 6$ com custo (8, 2, 2).

Passo 2: Ordenação dos caminhos não-dominados de acordo com

$$\mu_{cam} = \min\{\mu_{custo}, \mu_{capac}\}$$

- cam_8 com $\mu_{cam} = \min\{1,0000, 1,0000\} = 1,0000$;
- cam_7 com $\mu_{cam} = \min\{0,8000, 1,0000\} = 0,8000$;
- cam_1 com $\mu_{cam} = \min\{0,5000, 1,0000\} = 0,5000$;
- cam_9 com $\mu_{cam} = \min\{0,5000, 1,0000\} = 0,5000$;
- cam_2 com $\mu_{cam} = \min\{0,4000, 1,0000\} = 0,4000$;
- cam_3com $\mu_{cam} = \min\{0,3333, 1,0000\} = 0,3333$;
- cam_4 com $\mu_{cam} = \min\{0,2857, 1,0000\} = 0,2857$;
- cam_5 com $\mu_{cam} = \min\{0,1667, 1,0000\} = 0,1667$;
- cam_{10} com $\mu_{cam} = \min\{0, 1,0000\} = 0$;
- cam_6 com $\mu_{cam} = \min\{0, 1,0000\} = 0$.

Passo 3: Enviar fluxo para o primeiro caminho ordenado até alcançar a próxima pertinência diferente à desse caminho.

Relaxar as capacidades dos arcos de cam_8 até 0,8000 (pertinência do segundo melhor caminho).

Fluxo de p_2 ao longo dos arcos é (0 0 0 0 0,6667 0 0 0 0). Como o fluxo de p_2 necessário ao nó destino de cam_8 foi satisfeito, eliminamos tal caminho.

Atualizando μ_{capac} para os caminhos restantes temos que: $\mu_{capac} = 1 \,\forall\, cam$.

Passo 4: Ainda há fluxo a transitar.

2^a iteração:

Passo 2: Ordenação.

- cam_7 com $\mu_{cam} = \min\{0,8000, 1,0000\} = 0,8000$;
- cam_1 com $\mu_{cam} = \min\{0,5000, 1,0000\} = 0,5000$;

- cam_9 com $\mu_{cam} = \min\{0,5000, 1,0000\} = 0,5000$;
- cam_2 com $\mu_{cam} = \min\{0,4000, 1,0000\} = 0,4000$;
- cam_3com $\mu_{cam} = \min\{0,3333, 1,0000\} = 0,3333$;
- cam_4 com $\mu_{cam} = \min\{0,2857, 1,0000\} = 0,2857$;
- cam_5 com $\mu_{cam} = \min\{0,1667, 1,0000\} = 0,1667$;
- cam_{10} com $\mu_{cam} = \min\{0, 1,0000\} = 0$;
- cam_6 com $\mu_{cam} = \min\{0, 1,0000\} = 0$.

Passo 3: Envio de fluxo.

Relaxar as capacidades dos arcos de cam_7 até 0,5000 (pertinência do segundo melhor caminho).

Fluxo de p_2 ao longo dos arcos é (0 0 0 0,6667 0,6667 0 0 0 0).

Como o fluxo de p_2 necessário ao nó destino de cam_7 foi satisfeito, eliminamos tal caminho.

Atualizando μ_{capac} para os caminhos restantes temos que: $\mu_{capac} = 1 \,\forall\, cam$.

Passo 4: Ainda há fluxo a transitar.

3^a iteração:

Passo 2: Ordenação.

- cam_1 com $\mu_{cam} = \min\{0,5000, 1,0000\} = 0,5000$;
- cam_9 com $\mu_{cam} = \min\{0,5000, 1,0000\} = 0,5000$;
- cam_2 com $\mu_{cam} = \min\{0,4000, 1,0000\} = 0,4000$;
- cam_3com $\mu_{cam} = \min\{0,3333, 1,0000\} = 0,3333$;
- cam_4 com $\mu_{cam} = \min\{0,2857, 1,0000\} = 0,2857$;

- cam_5 com $\mu_{cam} = \min\{0,1667, 1,0000\} = 0,1667$;
- cam_{10} com $\mu_{cam} = \min\{0, 1, 0000\} = 0$;
- cam_6 com $\mu_{cam} = \min\{0, 1, 0000\} = 0$.

Passo 3: Envio de fluxo.

Relaxar as capacidades dos arcos de cam_1 até 0,4000, pois a pertinência de cam_9 é igual a de cam_1.

Fluxo de p_1 e p_2 ao longo dos arcos é, respectivamente: (2,3333 0 0 2,3333 0 0 0 0 0) e (0 0 0 0,6667 0,6667 0 0 0 0).

Como o fluxo de p_1 necessário ao nó destino de cam_1 foi satisfeito, eliminamos cam_1 e cam_2.

Atualizando μ_{capac} para os caminhos restantes temos que: $\mu_{capac} = 1 \,\forall\, cam$.

Passo 4: Ainda há fluxo a transitar.

4^a iteração:

Passo 2: Ordenação.

- cam_9 com $\mu_{cam} = \min\{0,5000, 1,0000\} = 0,5000$;
- cam_3com $\mu_{cam} = \min\{0,3333, 1,0000\} = 0,3333$;
- cam_4 com $\mu_{cam} = \min\{0,2857, 1.0000\} = 0,2857$;
- cam_5 com $\mu_{cam} = \min\{0,1667, 1,0000\} = 0,1667$;
- cam_{10} com $\mu_{cam} = \min\{0, 1, 0000\} = 0$;
- cam_6 com $\mu_{cam} = \min\{0, 1, 0000\} = 0$.

Passo 3: Envio de fluxo.

Relaxar as capacidades dos arcos de cam_9 até 0,3333 (pertinência do segundo caminho ordenado).

Fluxo de p_1 e p_2 ao longo dos arcos é, respectivamente: (2,3333 0 0 2,3333 0 0 0 0 0) e (0 0 0 1,0000 0,6667 0 0,3333 0 0).

Atualizando μ_{capac} para os caminhos restantes temos que: $\mu_{capac}(cam_9) = 0,3333$, $\mu_{capac}(cam_3) = \mu_{capac}(cam_4) = \mu_{capac}(cam_5) = \mu_{capac}(cam_{10}) = \mu_{capac}(cam_6) = 1,0000$.

Passo 4: Ainda há fluxo a transitar.

5^a iteração:

Passo 2: Ordenação.

- cam_3 com $\mu_{cam} = \min\{0,5000, 0,3333\} = 0,3333$;
- cam_9com $\mu_{cam} = \min\{0,3333, 1,0000\} = 0,3333$;
- cam_4 com $\mu_{cam} = \min\{0,2857, 1,0000\} = 0,2857$;
- cam_5 com $\mu_{cam} = \min\{0,1667, 1,0000\} = 0,1667$;
- cam_{10} com $\mu_{cam} = \min\{0, 1,0000\} = 0$;
- cam_6 com $\mu_{cam} = \min\{0, 1,0000\} = 0$.

Passo 3: Envio de fluxo.

Relaxar as capacidades dos arcos de cam_3 até 0,2857, pois a pertinência de cam_9 é igual a de cam_3.

Fluxo de p_1 e p_2 ao longo dos arcos é, respectivamente: (3,6667 0 0 2,3333 1,3333 0 0 0 0) e (0 0 0 1,0000 0,6667 0 0,3333 0 0).

Como o fluxo de p_1 necessário ao nó destino de cam_3 foi satisfeito eliminamos tal caminho.

Atualizando μ_{capac} para os caminhos restantes temos que: $\mu_{capac}(cam_3) = 0,3333$ $\mu_{capac}(cam_4) = \mu_{capac}(cam_5) = \mu_{capac}(cam_{10}) = \mu_{capac}(cam_6) = 1,0000$.

Passo 4: Ainda há fluxo a transitar.

6^a iteração:

Passo 2: Ordenação.

- cam_9 com $\mu_{cam} = \min\{0,5000, 0,3333\} = 0,3333$;
- cam_4 com $\mu_{cam} = \min\{0,2857, 1,0000\} = 0,2857$;
- cam_5 com $\mu_{cam} = \min\{0,1667, 1,0000\} = 0,1667$;
- cam_{10} com $\mu_{cam} = \min\{0, 1,0000\} = 0$;
- cam_6 com $\mu_{cam} = \min\{0, 1,0000\} = 0$.

Passo 3: Envio de fluxo.

Relaxar as capacidades dos arcos de cam_9 até 0,2857 (pertinência do segundo melhor caminho ordenado).

Fluxo de p_1 e p_2 ao longo dos arcos é, respectivamente: (3,6667 0 0 2,3333 1,3333 0 0 0 0) e (0 0 0 1,0952 0,6667 0 0,4286 0 0).

Atualizando μ_{capac} para os caminhos restantes temos que: $\mu_{capac}(cam_3) = 0,2857$ $\mu_{capac}(cam_4) = \mu_{capac}(cam_5) = \mu_{capac}(cam_{10}) = \mu_{capac}(cam_6) = 1,0000$.

Passo 4: Ainda há fluxo a transitar.

7^a iteração:

Passo 2: Ordenação.

- cam_9 com $\mu_{cam} = \min\{0,5000, 0,2857\} = 0,2857$;
- cam_4 com $\mu_{cam} = \min\{0,2857, 1,0000\} = 0,2857$;
- cam_5 com $\mu_{cam} = \min\{0,1667, 1,0000\} = 0,1667$;
- cam_{10} com $\mu_{cam} = \min\{0, 1,0000\} = 0$;
- cam_6 com $\mu_{cam} = \min\{0, 1,0000\} = 0$.

Passo 3: Envio de fluxo.

Relaxar as capacidades dos arcos de cam_9 até 0,1667, pois a pertinência de cam_4 é igual a de cam_9.

Fluxo de p_1 e p_2 ao longo dos arcos é, respectivamente: (3,6667 0 0 2,3333 1,3333 0 0 0 0) e (0 0 0 1,3333 0,6667 0 0,6667 0 0).

Como o fluxo de p_2 necessário ao nó destino de cam_9 foi satisfeito eliminamos cam_9 e cam_{10}.

Atualizando μ_{capac} para os caminhos restantes temos que: $\mu_{capac}(cam_4) = 0,1667$ $\mu_{capac}(cam_5) = \mu_{capac}(cam_6) = 1,0000$.

Passo 4: Ainda há fluxo a transitar.

8^a iteração:

Passo 2: Ordenação.

- cam_4 com $\mu_{cam} = \min\{0,2857, 0,1667\} = 0,1667$;
- cam_5 com $\mu_{cam} = \min\{0,1667, 1,0000\} = 0,1667$;
- cam_6 com $\mu_{cam} = \min\{0, 1,0000\} = 0$.

Passo 3: Envio de fluxo.

Relaxar as capacidades dos arcos de cam_3 até 0,1667, pois a próxima pertinência é zero.

Aqui, o fluxo a ser enviado por cam_4 é zero, daí utilizamos o próximo melhor caminho ordenado, neste caso, cam_5.

Fluxo de p_1 e p_2 ao longo dos arcos é, respectivamente: (3,6667 1,3333 0 2,3333 1,3333 1,3333 1,3333 0 0) e (0 0 0 1,3333 0,6667 0 0,6667 0 0).

Como o fluxo de p_1 necessário ao nó destino de cam_5 foi satisfeito eliminamos cam_4, cam_5 e cam_6.

Passo 4: Não há mais fluxo a transitar: fim.

O fluxo final dos produtos p_1 e p_2 em cada arco, segue na Tabela 5.3.

Tabela 5.3: Fluxo final de cada produto

arcos	(1,2)	(1,3)	(2,3)	(2,4)	(2,5)	(3,4)	(4,6)	(5,4)	(5,6)
fluxo de p_1	3,6667	1,3333	0	2,3333	1,3333	1,3333	1,3333	0	0
fluxo de p_2	0	0	0	1,3333	0,6667	0	0,6667	0	0
fluxo total	3,6667	1,3333	0	3,6667	2,0000	1,3333	2,0000	0	0

Podemos observar, na 8^a iteração, que não foi enviado fluxo do produto p_1 por cam_4. Neste caso, podemos usar a Opção 1, citada anteriormente, enviando fluxo pelo melhor caminho segundo a ordenação feita conforme a Tabela 5.4.

Tabela 5.4: Envio de fluxo com a Opção 1

caminho	produto	fluxo enviado
$1 \rightarrow 2 \rightarrow 4 \rightarrow 6$	p_1	0,1667
$1 \rightarrow 3 \rightarrow 4 \rightarrow 6$	p_1	1,1667

A Tabela 5.5 mostra o fluxo final de cada produto com a Opção 1.

Tabela 5.5: Fluxo final de cada produto com a Opção 1

arcos	(1,2)	(1,3)	(2,3)	(2,4)	(2,5)	(3,4)	(4,6)	(5,4)	(5,6)
fluxo de p_1	3,8333	1,1667	0	2,5000	1,3333	1,1667	1,3333	0	0
fluxo de p_2	0	0	0	1,3333	0,6667	0	0,6667	0	0
fluxo total	3,8333	1,1667	0	3,8333	2,0000	1,1667	2,0000	0	0

Podemos observar na Tabela 5.3, que no arco (1,2) tem 3,6667 de fluxo e no arco (1,3) tem 1,3333. Isso se deve ao fato do arco (1,2) pertencer aos melhores caminhos segundo a ordenação feita. Para p_1, os arcos (1,2) e (2,4) foram utilizados para atender toda a demanda do nó 4 (enviando 2,3333 pelo caminho $1 \rightarrow 2 \rightarrow 4$) com custo (8, 5, 3). De outra forma custaria (10, 6, 4), correspondente ao caminho $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ ou (12, 6, 4), correspondente ao caminho $1 \rightarrow 2 \rightarrow 5 \rightarrow 4$ ou (8, 4, 2), correspondente ao caminho $1 \rightarrow 3 \rightarrow 4$. Para p_2, o arco (2,4) foi utilizado para atender toda demanda do nó 4 (enviando 0,6667 por $2 \rightarrow 4$) com custo (6, 4, 2), enquanto que as outras opções custariam (8, 5, 3) correspondente ao caminho $2 \rightarrow 3 \rightarrow 4$ ou (10, 5,3) correspondente ao caminho $2 \rightarrow 5 \rightarrow 4$. Portanto, os caminhos utilizados foram boas opções.

Neste exemplo, a opção de usar o melhor caminho (Opção 1), não trouxe vantagens. O custo final total sem o uso da Opção 1 foi $(54,0000,\ 27,6667,\ 17,6667)$ e a pertinência $0,1667$, referente ao arco $(2,4)$; enquanto que, com o uso da Opção 1 o custo foi $(54,0000,\ 27,8333,\ 17,8333)$ e a pertinência $0,0833$ devido ao fluxo no arco $(2,4)$ ter aumentado. A tabela 5.6 ilustra o custo final de cada produto sem a Opção 1.

O tempo de processamento foi de 7 segundos.

Tabela 5.6: Custo final de cada produto

custo final	produto
(41,3333, 21,0000, 13,6667)	p_1
(12,6667, 6,6667, 4,0000)	p_2

Foram realizados testes com diferentes valores de α. Observamos que, para α variando de 0 até $0,166$, a solução final (custo, fluxo nos arcos, pertinência) foi a mesma e a demanda foi atendida. Para $\alpha = 0,167$ a demanda de $p_1(b_6^1 = -1,3333)$ do nó 6 não é atendida, como mostra a Tabela 5.7.

Tabela 5.7: Fluxo final de cada produto para $\alpha = 0,167$

arcos	(1,2)	(1,3)	(2,3)	(2,4)	(2,5)	(3,4)	(4,6)	(5,4)	(5,6)
fluxo de p_1	4,0000	0	0	2,6667	1,3333	0	0,3333	0	0
fluxo de p_2	0	0	0	1,3333	0,6667	0	0,6667	0	0
fluxo total	4,0000	0	0	4,0000	2,0000	0	2,0000	0	0

Podemos observar na Tabela 5.7, que os arcos $(1,2)$ e $(2,4)$ estão no limitante superior, isso se dá porque ao utilizarmos $\alpha = 0,167$, o caminho $1 \rightarrow 3 \rightarrow 4 \rightarrow 6$ foi eliminado ($\mu_{custo} = 0,16667$), daí o fluxo de p_1 do nó 1 ao nó 6 passou somente pelo caminho $1 \rightarrow 2 \rightarrow 4 \rightarrow 6$.

Os caminhos $2 \rightarrow 5 \rightarrow 6$ e $1 \rightarrow 2 \rightarrow 5 \rightarrow 6$, ambos com $\mu_{custo} = 0$, também foram eliminados, mas esta eliminação não inteferiu no resultado final, porque tais caminhos não foram utilizados na solução.

Agora consideramos p_1 com origem no nó 2 e destinos os nós 4, 5 e 6, p_2 com origem no nó 1 e destinos os nós 4, 5 e 6. As ofertas e as demandas de cada produto são: $b_2^1 = 5$, $b_1^2 = 2$, $b_4^1 = -2,3333$, $b_4^2 = -0,6667$, $b_5^1 = -1,3333$, $b_5^2 = -0,6667$, $b_6^1 = -1,3333$ e $b_6^2 = -0,6667$.

O algoritmo atendeu todas as demandas em dez etapas utilizando sete caminhos, dos dez caminhos mínimos não-dominados.

O envio de fluxo ocorreu segundo a Tabela 5.8.

O fluxo final dos produtos p_1 e p_2 em cada arco, segue na Tabela 5.9.

Tabela 5.8: Envio de fluxo dos produtos

etapa	caminho	produto	fluxo enviado	demanda atendida
1	$2 \rightarrow 5$	p_1	1,3333	1,3333
2	$2 \rightarrow 4$	p_1	2,3333	2,3333
3	$2 \rightarrow 4 \rightarrow 6$	p_1	0,8667	0,8667
4	$2 \rightarrow 4 \rightarrow 6$	p_1	0,1333	1,0000
5	$1 \rightarrow 3 \rightarrow 4$	p_2	0,6667	0,6667
6	$1 \rightarrow 2 \rightarrow 5$	p_2	0,6667	0,6667
7	$2 \rightarrow 4 \rightarrow 6$	p_1	0,0952	1,0953
8	$2 \rightarrow 4 \rightarrow 6$	p_1	0,2380	1,3333
9	$1 \rightarrow 2 \rightarrow 4 \rightarrow 6$	p_2	0,0001	0,0001
10	$1 \rightarrow 3 \rightarrow 4 \rightarrow 6$	p_2	0,6665	0,6666

Tabela 5.9: Fluxo final de cada produto sem a Opção 1

arcos	(1,2)	(1,3)	(2,3)	(2,4)	(2,5)	(3,4)	(4,6)	(5,4)	(5,6)
fluxo de p_1	0	0	0	2,3333	1,3333	0	1,3333	0	0
fluxo de p_2	0,6668	1,3333	0	1,3333	0,6667	1,3333	0,6667	0	0
fluxo total	0,6668	1,3333	0	3,6667	2,0000	1,3333	2,0000	0	0

Tabela 5.10: Custo final de cada produto

custo final	produto
(31,3327, 17,3330, 9,9998)	p_1
(16,6665, 7,3333, 4,6667)	p_2

O custo final total foi $(47,9992,\ 24,6663,\ 14,6665)$ e a pertinência final foi $0,1667$ referente ao arco $(2,4)$. O tempo de processamento foi 2,75 segundos. Neste caso, utilizando a Opção 1, os resultados obtidos foram os mesmos.

Caso 2: Na figura 5.2, temos p_1 e p_2 com origem nos nós 1 e 2 e os mesmos

destinos considerados anteriormente. Os dados dos produtos seguem na Tabela 5.11.

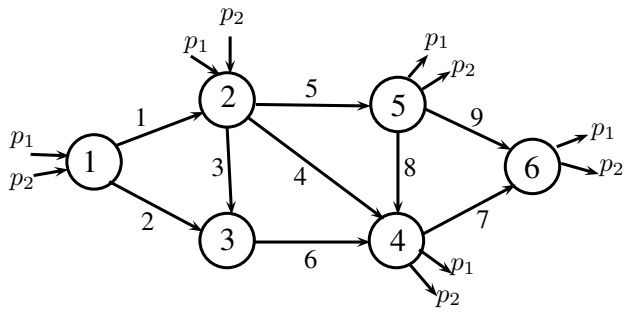

Figura 5.2: Rede com dois produtos, ambos com origem em dois nós

Tabela 5.11: Dados de cada produto

produto	origens	destinos	oferta	demanda
p_1	{1, 2}	{4, 5, 6}	{2, 3}	{2,3333; 1,3333; 1,3333}
p_2	{1, 2}	{4, 5, 6}	{1, 1}	{0,6667; 0,6667; 0,6667}

O algoritmo atendeu todas as demandas em nove etapas utilizando cinco caminhos, dos vinte caminhos mínimos não-dominados (dez para p_1 e dez para p_2).

O envio de fluxo ocorreu segundo a Tabela 5.12.

Tabela 5.12: Envio de fluxo dos produtos da rede da figura 5.2

etapa	caminho	produto	fluxo enviado	demanda atendida
1	$2 \rightarrow 5$	p_1	1,3333	1,3333
2	$2 \rightarrow 5$	p_2	0,6667	0,6667
3	$2 \rightarrow 4$	p_1	1,6667	1,6667
4	$2 \rightarrow 4$	p_2	0,3333	0,3333
5	$1 \rightarrow 2 \rightarrow 4$	p_1	0,6666	2,3333
6	$1 \rightarrow 2 \rightarrow 4$	p_2	0,3334	0,6667
7	$2 \rightarrow 4 \rightarrow 6$	p_1	0,6667	0,6667
8	$1 \rightarrow 3 \rightarrow 4 \rightarrow 6$	p_1	0,6666	1,3333
9	$1 \rightarrow 3 \rightarrow 4 \rightarrow 6$	p_2	0,6667	0,6667

O fluxo final dos produtos p_1 e p_2 em cada arco, segue na Tabela 5.13.

Tabela 5.13: Fluxo final de cada produto sem a Opção 1 referente a rede da figura 5.2

arcos	(1,2)	(1,3)	(2,3)	(2,4)	(2,5)	(3,4)	(4,6)	(5,4)	(5,6)
fluxo de p_1	1,3333	0,6667	0	3,0000	1,3333	0,6667	1,3333	0	0
fluxo de p_2	0,3334	0,6666	0	0,6667	0,6667	0,6666	0,6667	0	0
fluxo total	1,6667	1,3333	0	3,6667	2,0000	1,3333	2,0000	0	0

Tabela 5.14: Custo final de cada produto

custo final	produto
(35,3325, 18,6663, 11,3331)	p_1
(14,6665, 6,9999, 4,3333)	p_2

O custo final total é $(49,9990,\ 25,6662,\ 15,6664)$. A pertinência final é $0,16667$ referente ao arco $(2,4)$. O tempo de processamento foi 1,9680 segundos. Novamente, utilizando a Opção 1, os resultados foram os mesmos.

Pensando em termos de uma fábrica, podemos concluir que é mais vantajoso produzir p_1 na fábrica 2 (nó 2) e p_2 na fábrica 1 (nó 1). A produção de p_1 e p_2 em ambas as fábricas (nós 1 e 2) também é uma boa opção.

Para resolver o problema crisp, ou seja, quando os custos e capacidades são números reais, utilizamos o solver do Xpress, que é um pacote de resolução de problemas de programação linear que resolve através do simplex. Os custos são os valores modais dos custos da Tabela 5.1 e as capacidades, o limitante superior das capacidades da Tabela 5.1. Os resultados seguem na Tabela 5.15.

Tabela 5.15: Fluxo final de cada produto para o caso clássico

arcos	(1,2)	(1,3)	(2,3)	(2,4)	(2,5)	(3,4)	(4,6)	(5,4)	(5,6)
fluxo de p_1	4	1	0	1.3333	2,6667	1	0	0	1,3333
fluxo de p_2	0	0	0	0.6667	1.3333	0	0,6667	0	0.6667
fluxo total	4	1	0	3	4	1	0,6667	0	2

Os custos finais de p_1 e p_2 foram respectivamente, $41,3333$ e $12,6667$. O custo final total foi 54.

Analisando as soluções do problema crisp e do problema *fuzzy*, observamos que as mesmas são próximas. Isso demonstra o bom funcionamento do algoritmo proposto.

5.2 Instância 2

O segundo exemplo foi resolvido utilizando a Rede Óptica Européia COST239 [32] apresentada pela Figura 5.3. As capacidades foram adaptadas para números *fuzzy*. Os custos, a definição dos nós e arcos foram retirados de Hernandes [14].

Os nós da rede estão definidos na Tabela 5.16.

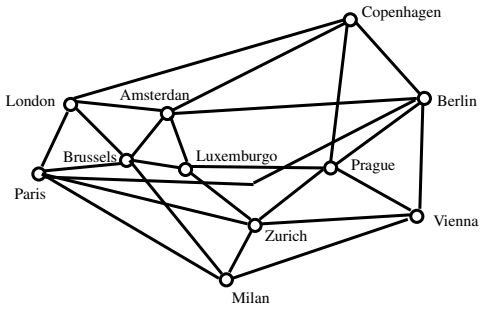

Figura 5.3: Rede óptica européia

Tabela 5.16: Denominação dos nós da rede COST239

Nós	Cidades	Nós	Cidades
1	Paris	7	Amsterdan
2	Milan	8	Luxemburgo
3	Zurich	9	Brussels
4	Prague	10	London
5	Vienna	11	Copenhagen
6	Berlin		

Os arcos, custos e capacidades estão definidos na Tabela 5.17.

Neste problema consideramos os arcos com duplo sentido sendo que, os arcos de 26 a 50 correspondem aos arcos de 1 a 25 em sentido contrário.

Tabela 5.17: Dados da Rede COST239

Arco	Origem → Destino	Custos	Capacidades
1	1 → 2	(820, 20, 20)	(0, 0, 10, 12)
2	1 → 3	(361, 11, 9)	(0, 0, 2,5, 4,5)
3	1 → 6	(677, 27, 6)	(0, 0, 10, 12)
4	1 → 9	(300, 10, 50)	(0, 0, 4,0, 6,5)
5	1 → 10	(450, 30, 20)	(0, 0, 2,5, 3,5)
6	2 → 3	(186, 6, 7)	(0, 0, 30, 34)
7	2 → 5	(510, 15, 15)	(0, 0, 20, 23)
8	2 → 9	(930, 30, 30)	(0, 0, 30, 34)
9	3 → 4	(667, 17, 196)	(0, 0, 10, 12)
10	3 → 5	(748, 18, 22)	(0, 0, 10, 12)
11	3 → 8	(443, 18, 22)	(0, 0, 2,5, 3,5)
12	4 → 5	(199, 9, 11)	(0, 0, 20, 23)
13	4 → 6	(340, 30, 20)	(0, 0, 10, 12)
14	4 → 11	(740, 30, 30)	(0, 0, 10, 12)
15	5 → 6	(660, 50, 30)	(0, 0, 30, 34)
16	6 → 11	(242, 12, 18)	(0, 0, 10, 12)
17	7 → 6	(410, 20, 30)	(0, 0, 20, 23)
18	7 → 11	(472, 22, 18)	(0, 0, 10, 12)
19	8 → 4	(730, 20, 5)	(0, 0, 2,5, 3,5)
20	8 → 7	(242, 12, 13)	(0, 0, 2,5, 3,5)
21	9 → 8	(137, 7, 8)	(0, 0, 2,5, 3,5)
22	9 → 7	(130, 10, 20)	(0, 0, 10, 12)
23	9 → 10	(242, 12, 18)	(0, 0, 2,5, 3,5)
24	10 → 7	(342, 12, 8)	(0, 0, 2,5, 3,5)
25	10 → 11	(1310, 60, 120)	(0, 0, 2,5, 3,5)

Consideramos três produtos, os quais estão descritos na Tabela 5.18 (em $Gbits/s$).

Tabela 5.18: Dados de cada produto

produto	origem	destinos	oferta	demanda
p_1	1	{3, 6, 11}	6	{0,5; 2,5; 3}
p_2	11	{1, 5, 10}	5	{2; 0,5; 2,5}
p_3	1	{4, 7, 9}	7	{4; 1; 2}

Para $\alpha = 0$, o algoritmo atendeu todas as demandas em nove etapas utilizando nove caminhos, dos onze caminhos mínimos não-dominados. O tempo de processamento foi de 2,5380 segundos.

O envio de fluxo ocorreu segundo a Tabela 5.19.

Tabela 5.19: Envio de fluxo

etapa	caminho	produto	fluxo enviado	demanda atendida
1	$1 \rightarrow 9$	p_3	2,0	2,0
2	$1 \rightarrow 3$	p_1	0,5	0,5
3	$1 \rightarrow 9 \rightarrow 7$	p_3	1,0	1,0
4	$1 \rightarrow 6$	p_1	2,5	2,5
5	$11 \rightarrow 6 \rightarrow 4 \rightarrow 5$	p_2	0,5	0,5
6	$11 \rightarrow 7 \rightarrow 10$	p_2	2,5	2,5
7	$1 \rightarrow 9 \rightarrow 7 \rightarrow 11$	p_1	3,0	3,0
8	$11 \rightarrow 7 \rightarrow 9 \rightarrow 1$	p_2	2,0	2,0
9	$1 \rightarrow 6 \rightarrow 4$	p_3	4,0	4,0

Podemos observar na Tabela 5.19 que para cada par de nós origem-destino de cada produto, as demandas foram atendidas usando apenas um caminho. As Tabelas 5.20, 5.21 e 5.22, ilustram o fluxo final dos produtos p_1, p_2 e p_3, respectivamente e, as Tabelas 5.23 e 5.24, o fluxo total nos arcos.

Tabela 5.20: Fluxo final do produto p_1

arcos	2	3	4	18	22
fluxo de p_1	0,5000	2,5000	3,0000	3,0000	3,0000

Tabela 5.21: Fluxo final do produto p_2

arcos	12	29	38	41	43	47	49
fluxo de p_2	0,5000	2,0000	0,5000	0,5000	4,5000	2,0000	2,5000

Tabela 5.22: Fluxo final do produto p_3

arcos	3	4	22	38
fluxo de p_3	4,0000	3,0000	1,0000	4,0000

Tabela 5.23: Fluxo total nos arcos

arcos	2	3	4	12	18	22
fluxo total	0,5000	6,5000	6.0000	0,5000	3,0000	4,0000

Tabela 5.24: Fluxo total nos arcos

arcos	29	38	41	43	47	49
fluxo total	2,0000	4,0000	0,5000	4,5000	4,0000	2,50000

O custo final total foi $10^4 \cdot (1,2425,\ 0,0592,\ 0,0655)$ e a pertinência $0,2000$, referente ao arco $(1,9)$.

Neste exemplo, utilizando a Opção 1, os resultados foram os mesmos que os aqui apresentados (sem a Opção 1).

A Tabela 5.25 ilustra o custo final de cada produto.

Tabela 5.25: Custo final de cada produto

custo final	produto
$10^3 \cdot (4,5790,\ 0,1919,\ 0,2835)$	p_1
$10^3 \cdot (2,7475,\ 0,1250,\ 0,0975)$	p_2
$10^3 \cdot (5,0980,\ 0,2680,\ 0,2740)$	p_3

Para o problema crisp, ou seja, quando os custos e capacidades são números reais, utilizando o solver do Xpress, obtivemos os resultados apresentados nas Tabelas 5.26, 5.27 e 5.28. Os custos são os valores modais dos custos da Tabela 5.17 e as capacidades, os valores modais das capacidades da Tabela 5.17.

Tabela 5.26: Fluxo final do produto p_1 para o caso clássico

arcos	2	3	4	18	22
fluxo de p_1	0,5000	2,5000	3,0000	3,0000	3,0000

Tabela 5.27: Fluxo final do produto p_2

arcos	12	29	38	41	43	47	49
fluxo de p_2	0,5000	2,0000	0,5000	0,5000	4,5000	2,0000	2,5000

Tabela 5.28: Fluxo final do produto p_3 para o caso clássico

arcos	3	4	22	38
fluxo de p_3	4,0000	3,0000	1,0000	4,0000

Os custos finais de p_1, p_2 e p_3 foram respectivamente, 4579, $2747,5$ e 5098. O custo final total foi 12.425.

5.3 Instância 3

Neste exemplo, aplica-se o algoritmo numa rede maior, constituída de 15 nós e 34 arcos e transportando 8 produtos. A rede em questão é mostrada na Figura 5.4.

Este exemplo representa um sistema ferroviário de transporte de carga multiproduto. Os dados aqui descritos foram tirados de [22].

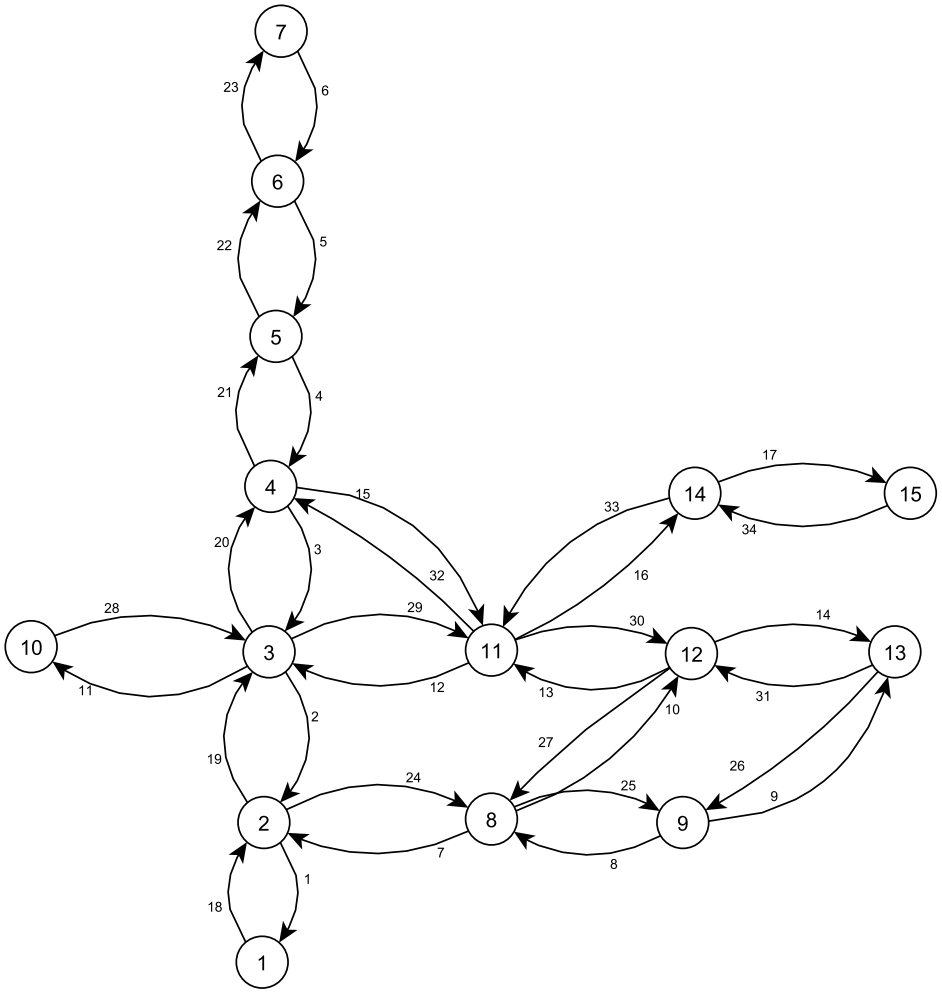

Figura 5.4: Grafo da rede ferroviária

Os oito produtos estão descritos na Tabela 5.29 (em toneladas).

Tabela 5.29: Dados de cada produto

produto	origem	destinos	oferta	demanda
p_1	4	{7, 13, 15}	3000	{1000; 1000; 1000}
p_2	5	{1}	1000	{1000}
p_3	15	{3}	1000	{1000}
p_4	12	{10}	1000	{1000}
p_5	9	{1}	2000	{2000}
p_6	6	{1}	2000	{2000}
p_7	14	{1}	1000	{1000}
p_8	1	{6, 9, 15}	3000	{1000; 1000; 1000}

Os arcos, custos e capacidades estão definidos na Tabela 5.30. Neste problema, consideramos os arcos com duplo sentido sendo que, os arcos de 18 a 34 correspondem, respectivamente, aos arcos de 1 a 17 em sentido contrário.

Tabela 5.30: Dados da Rede Ferroviária

Arco	Origem → Destino	Custos	Capacidades
1	1 → 2	(2, 0,2, 0,2)	(0, 0, 2200, 6500)
2	2 → 3	(2, 0,2, 0,2)	(0, 0, 2200, 4500)
3	3 → 4	(2, 0,2, 0,2)	(0, 0, 2200, 3500)
4	4 → 5	(2, 0,2, 0,2)	(0, 0, 2200, 3500)
5	5 → 6	(2, 0,2, 0,2)	(0, 0, 2200, 2425)
6	6 → 7	(2, 0,2, 0,2)	(0, 0, 2200, 2425)
7	2 → 8	(2, 0,2, 0,2)	(0, 0, 2200, 2425)
8	8 → 9	(2, 0,2, 0,2)	(0, 0, 2200, 2425)
9	9 → 13	(2, 0,2, 0,2)	(0, 0, 2200, 2425)
10	8 → 12	(2, 0,2, 0,2)	(0, 0, 2200, 2425)
11	10 → 3	(2, 0,2, 0,2)	(0, 0, 2200, 2425)
12	3 → 11	(2, 0,2, 0,2)	(0, 0, 2200, 4000)
13	11 → 12	(2, 0,2, 0,2)	(0, 0, 2200, 2425)
14	12 → 13	(2, 0,2, 0,2)	(0, 0, 2200, 2425)
15	4 → 11	(2, 0,2, 0,2)	(0, 0, 2200, 2425)
16	11 → 14	(2, 0,2, 0,2)	(0, 0, 2200, 2425)
17	14 → 15	(2, 0,2, 0,2)	(0, 0, 2200, 2425)

Para $\alpha = 0$, o algoritmo atendeu todas as demandas em doze etapas utilizando os doze caminhos mínimos não-dominados encontrados. O tempo de processamento foi de 3,9960 segundos.

O envio de fluxo ocorreu segundo a Tabela 5.31.

Podemos observar na Tabela 5.31 que, para cada par de nós origem-destino de cada produto, as demandas foram atendidas usando apenas um caminho. As Tabelas 5.32, 5.33 e 5.34, ilustram o fluxo final dos produtos em cada arco.

Tabela 5.31: Envio de fluxo

etapa	caminho	produto	fluxo enviado	demanda atendida
1	$4 \rightarrow 5 \rightarrow 6 \rightarrow 7$	p_1	1000	1000
2	$4 \rightarrow 11 \rightarrow 12 \rightarrow 13$	p_1	1000	1000
3	$4 \rightarrow 11 \rightarrow 14 \rightarrow 15$	p_1	1000	1000
4	$15 \rightarrow 14 \rightarrow 11 \rightarrow 3$	p_3	1000	1000
5	$12 \rightarrow 11 \rightarrow 3 \rightarrow 10$	p_4	1000	1000
6	$9 \rightarrow 8 \rightarrow 2 \rightarrow 1$	p_5	2000	2000
7	$1 \rightarrow 2 \rightarrow 8 \rightarrow 9$	p_8	1000	1000
8	$5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$	p_2	1000	1000
9	$14 \rightarrow 11 \rightarrow 3 \rightarrow 2 \rightarrow 1$	p_7	1000	1000
10	$6 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$	p_6	2000	2000
11	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$	p_8	1000	1000
12	$1 \rightarrow 2 \rightarrow 3 \rightarrow 11 \rightarrow 14 \rightarrow 15$	p_8	1000	1000

Tabela 5.32: Fluxo final de cada produto em cada arco

arcos	1	2	3	4	5	6	7	8	12	13
fluxo de p_1	0	0	0	1000	1000	1000	0	0	0	1000
fluxo de p_2	0	0	0	0	0	0	0	0	0	0
fluxo de p_3	0	0	0	0	0	0	0	0	0	0
fluxo de p_4	0	0	0	0	0	0	0	0	0	0
fluxo de p_5	0	0	0	0	0	0	0	0	0	0
fluxo de p_6	0	0	0	0	0	0	0	0	0	0
fluxo de p_7	0	0	0	0	0	0	0	0	0	0
fluxo de p_8	3000	2000	1000	1000	1000	0	1000	1000	1000	0

Tabela 5.33: Fluxo final de cada produto em cada arco

arcos	14	15	16	17	18	19	20	21
fluxo de p_1	1000	2000	1000	1000	0	0	0	0
fluxo de p_2	0	0	0	0	1000	1000	1000	1000
fluxo de p_3	0	0	0	0	0	0	0	0
fluxo de p_4	0	0	0	0	0	0	0	0
fluxo de p_5	0	0	0	0	2000	0	0	0
fluxo de p_6	0	0	0	0	2000	2000	2000	2000
fluxo de p_7	0	0	0	0	1000	1000	0	0
fluxo de p_8	0	0	1000	1000	0	0	0	0

Tabela 5.34: Fluxo final de cada produto em cada arco

arcos	22	24	25	28	29	30	33	34
fluxo de p_1	0	0	0	0	0	0	0	0
fluxo de p_2	0	0	0	0	0	0	0	0
fluxo de p_3	0	0	0	0	1000	0	1000	1000
fluxo de p_4	0	0	0	1000	1000	1000	0	0
fluxo de p_5	0	2000	2000	0	0	0	0	0
fluxo de p_6	2000	0	0	0	0	0	0	0
fluxo de p_7	0	0	0	0	1000	0	1000	0
fluxo de p_8	0	0	0	0	0	0	0	0

As Tabelas 5.35, 5.36 e 5.37 ilustram o fluxo total nos arcos.

Tabela 5.35: Fluxo total nos arcos

arcos	1	2	3	4	5	6	7	8	12
fluxo total	3000	2000	1000	2000	2000	1000	1000	1000	1000

Tabela 5.36: Fluxo total nos arcos

arcos	13	14	15	16	17	18	19	20	21
fluxo total	1000	1000	2000	2000	2000	6000	4000	3000	3000

Tabela 5.37: Fluxo total nos arcos

arcos	22	24	25	28	29	30	33	34
fluxo total	2000	2000	2000	1000	3000	1000	2000	1000

A Tabela 5.38 ilustra o custo final de cada produto.

Tabela 5.38: Custo final de cada produto

custo final	produto
(18000, 1800, 1800)	p_1
(8000, 800, 800)	p_2
(6000, 600, 600)	p_3
(6000, 600, 600)	p_4
(12000, 1200, 1200)	p_5
(20000, 2000, 2000)	p_6
(8000, 800, 800)	p_7
(26000, 2600, 2600)	p_8

O custo final total foi (104000, 10400, 10400) e a pertinência $0,1163$, referente ao arco $(2,1)$.

Neste exemplo, com o uso da Opção 1, os resultados foram os mesmos, isso porque o fluxo a ser enviado não deu zero nenhuma vez.

Este exemplo, teve como objetivo mostrar que o algoritmo proposto é viável para problemas de médio porte e é eficiente computacionalmente, pois seu tempo de execução é pequeno.

Novamente, utilizamos o solver do Xpress para resolver o problema crisp associado a este problema, ou seja, quando os custos e capacidades são números reais. Os custos são os valores modais dos custos da Tabela 5.30 e as capacidades, os limitantes superiores das capacidades da Tabela 5.30. Os resultados obtidos são apresentados nas Tabelas 5.39, 5.40 e 5.41.

Tabela 5.39: Fluxo final de cada produto em cada arco para o caso clássico

arcos	1	2	3	4	5	6	7	8	12	13
fluxo de p_1	0	0	0	1000	1000	1000	0	0	0	1000
fluxo de p_2	0	0	0	0	0	0	0	0	0	0
fluxo de p_3	0	0	0	0	0	0	0	0	0	0
fluxo de p_4	0	0	0	0	0	0	0	0	0	0
fluxo de p_5	0	0	0	0	0	0	0	0	0	0
fluxo de p_6	0	0	0	0	0	0	0	0	0	0
fluxo de p_7	0	0	0	0	0	0	0	0	0	0
fluxo de p_8	3000	2000	1000	1000	1000	0	1000	1000	1000	0

Tabela 5.40: Fluxo final de cada produto em cada arco para o caso

arcos	14	15	16	17	18	19	20	21
fluxo de p_1	1000	2000	1000	1000	0	0	0	0
fluxo de p_2	0	0	0	0	1000	1000	1000	1000
fluxo de p_3	0	0	0	0	0	0	0	0
fluxo de p_4	0	0	0	0	0	0	0	0
fluxo de p_5	0	0	0	0	2000	0	0	0
fluxo de p_6	0	0	0	0	2000	2000	2000	2000
fluxo de p_7	0	0	0	0	1000	1000	0	0
fluxo de p_8	0	0	1000	1000	0	0	0	0

Tabela 5.41: Fluxo final de cada produto em cada arco

arcos	22	24	25	28	29	30	33	34
fluxo de p_1	0	0	0	0	0	0	0	0
fluxo de p_2	0	0	0	0	0	0	0	0
fluxo de p_3	0	0	0	0	1000	0	1000	1000
fluxo de p_4	0	0	0	1000	1000	1000	0	0
fluxo de p_5	0	2000	2000	0	0	0	0	0
fluxo de p_6	2000	0	0	0	0	0	0	0
fluxo de p_7	0	0	0	0	1000	0	1000	0
fluxo de p_8	0	0	0	0	0	0	0	0

O custo final total foi 103997.

Capítulo 6

Conclusões e trabalhos futuros

6.1 Conclusões

Neste trabalho apresentamos um algoritmo para a resolução do problema de fluxo multiproduto *fuzzy* com incertezas em dois parâmetros (custos e capacidades) e possui a vantagem de ser aplicável a problemas de grande porte. A análise dos resultados dos testes computacionais mostra que o algoritmo proposto apresentou resultados satisfatórios. A comparação com o problema resolvido na forma crisp mostra que os resultados, de fato, foram bons. Cabe ressaltar que o algoritmo proposto comporta problemas de fluxo multiproduto com custos diferentes por produto por arco.

O uso da teoria *fuzzy* para tratar as incertezas nos custos e nas capacidades tornou o algoritmo mais aderente à realidade e promissor para a resolução do problema apresentado, pois resolvemos o problema na forma *fuzzy*, sem transformá-lo na forma crisp para resolvê-lo.

Nos exemplos apresentados nas Seções 5.1 e 5.2 temos valores de demanda que não são inteiros: no primeiro exemplo consideramos que as unidades dos produtos podem ser particionadas para destinos diferentes. Já no segundo exemplo, a medida de tráfego foi dada em termos de Gbit/s [32].

Os resultados obtidos são bastantes satisfatórios, comprovando que o uso da teoria

fuzzy no tratamento de parâmetros incertos é realmente promissor para obter boas soluções em diversos problemas, dentre eles, o de fluxo multiproduto de custo mínimo que é abordado neste trabalho.

6.2 Perspectivas Futuras

Apresentamos algumas sugestões para trabalhos futuros:

• Tratar a integralidade, ou seja, trabalhar apenas com valores inteiros nas ofertas e demandas dos produtos.

• Estudar métodos utilizando técnicas de decomposição para problemas de grande porte.

6.3 Trabalhos aceitos

Durante esse período foram aceitos dois trabalhos para apresentação:

- (Verga, J., Ciappina, J.R. e Yamakami, A.) Algoritmo para resolução do problema de fluxo multiproduto *fuzzy*, aceito no XLI Simpósio Brasileiro de Pesquisa Operacional, em setembro de 2009;

- (Ciappina, J.R., Verga, J. e Yamakami, A.) Problema de fluxo multiproduto *fuzzy*: proposta de um algoritmo, aceito no $XXXII$ Congresso Nacional de Matemática Aplicada e Computacional, em setembro de 2009.

No primeiro, consideramos custos iguais nos arcos para todos os produtos, e no segundo, consideramos custos diferentes nos arcos para cada produto.

Apêndice A

Algoritmo de Ford Moore Bellman

O algoritmo de Ford-Moore-Bellman, assim denominado em homenagem ao trabalho simultâneo desses pesquisadores [3], mas publicado em épocas diferentes, abre mão do fechamento de um nó a cada iteração, e examina todos os nós até que não seja mais possível melhorias, podendo, com isso, aceitar arestas negativas. O critério de parada está associado a não modificação de todos os rótulos em uma iteração. A idéia básica é de que, se um caminho de um nó i para um nó j contém k arcos, um caminho melhor de i para j conterá, no máximo, $k + 1$ arcos.

Notações utilizadas do algoritmo

V : conjunto dos nós;

it: contador de iterações;

r: número de nós;

l_{ji}: custo do arco (j, i);

c_i^{it}: custo do caminho entre os nós 1 e i na iteraçãoo it;

Γ_i^{-1}: conjunto dos nós predecessores de i.

Algoritmo

Passo 1: Inicialização das variáveis

1. $c_1^0 = 0$
2. $c_i^0 = \infty$, $i = 1, 2, 3, \ldots, r$
3. $it \leftarrow 1$

Passo 2: Determinação dos caminhos das próximas iterações

1. $c_1^{it} = 0$;
2. $\forall i \in V$, $i = 1, 2, 3, \ldots, r$, faça:

$$c_i^{it} = \min(c_i^{it-1}, \min_{j \in \Gamma_i^{-1}} (c_j^{it-1} + l_{ji}))$$

Passo 3: Critério de Parada

- Se $c_i^{it} = c_i^{it-1}$, $\forall i \in V \Rightarrow$ FIM
- Senão:
  1. Se $it \leq r - 1 \Rightarrow$ Passo 2
  2. Senão existe um circuito com custo negativo $\Rightarrow$ FIM

Referências Bibliográficas

[1] T.L. Ahuja, R.K. Magnanti. *Network Flows*. Prentice Hall, Philadelphia, PA, USA, 1993.

[2] J. Bazaraa, M. Jarvis and H.F. Sherali. *Linear Programming and Network Flows*. John Wiley, New York, 1990.

[3] R. E. Bellman. On a routing problem. *Quarterly Applied Mathematics*, 16:87–90, 1958.

[4] B. Blue, M. Bush and J. Puckett. Unified approach to fuzzy graph problems. *Fuzzy Sets and Systems*, 125(3):355–368, 2002.

[5] R.J. Chagas. Uma aplicação de métodos aproximados a problemas de fluxo multiproduto inteiro. Tese de mestrado, Dissertação de Mestrado, CEFET–MG, Outubro, 2005.

[6] E. Chang, P. Lee. Fuzzy decision networks and deconvolution. *Computers and Mathematics with Applications*, 37(11/12):53–63, 1999.

[7] J.-Y. Chuang, T.-N. Kung. A new algorithm for the discrete fuzzy shortest path problem in a network. *Applied Mathematics and Computation*, 174:660–668, 2006.

[8] H. Dubois, D. Prade. *Fuzzy Sets and Systems: Teory and Applications*. Academic Press, INC, New York, 1980.

[9] D.R. Ford, L.R. Fulkerson. *Flows in networks*. Princeton University Press, New York, 1962.

[10] M.T. García, M.S. Lamata. The fuzzy sets in maintenance process. In *Proceedings of the European Society for Fuzzy Logic and Technology*, 2005.

[11] S.M. Ghatee, M. Hashemi. Some concepts of the fuzzy multicommodity flow problem and their application in fuzzy network design. *Mathematical and Computer Modelling*, 33:344–360, 2008.

[12] H.P.L. Goldbarg, M.C. Luna. *Otimização Combinatória e Programação Linear*. Editora Campus, Rio de Janeiro, 2000.

[13] M. Gondran, M. Minoux. *Graphs and Algorithms*. John Wiley and Sons, New York, 1984.

[14] Hernandes, F. Algoritmos para Problemas de Grafos com Incertezas. Tese de doutorado, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, Fevereiro 2007.

[15] T.C. Hu. Multicommodity network flows. *Operations Research*, 11:344–360, 1962.

[16] Iwamura K. Ji, X. and Z. Shao. New models for shortest path problem with fuzzy arc lengths. *Applied Mathematical Modelling*, 31:259–269, 2007.

[17] C.M. Klein. Fuzzy shortest paths. *Fuzzy Sets and Systems*, 39:27–41, 1990.

[18] Gen M. Ida-K. Li, Y. Solving fuzzy shortest path problems by neural networks. *Computers and Engineering*, 31(3/4):861–865, Junho 1996.

[19] M.S. Lin, C. Chern. The fuzzy shortest path problem and its most vital arcs. *Fuzzy Sets and Systems*, 58:343–353, 1993.

[20] C. Liou, S.T. Kao. Network flow problems with fuzzy arc lengths. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 34(1):765–769, 2004.

[21] M.-J. Liou, T.-S. Wang. Ranking fuzzy numbers with integral interval. *Fuzzy Sets and Systems*, 50:247–255, 1992.

[22] Mendes, R.R. Programação Matemática Fuzzy Aplicada a um Problema de Transporte Multiproduto em Ferrovias. Tese de mestrado, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, Setembro 1999.

[23] M. Nayeem, S.M.A. Paul. Shortest path problem on a network with imprecise edge weight. *Fuzzy Optimization and Decision Making*, 4:293–312, 2005.

[24] M. Okada, S. Gen. Order relation between intervals and its application to shortest path problem. *Computers & industrial engineering*, 25(7):147–150, 1993.

[25] M. Okada, S. Gen. Fuzzy multiple choice knapsack problem. *Fuzzy Sets and Systems*, 67(1):71–80, 1994.

[26] S. Okada. Fuzzy shortest path problems incorporating interactivity among paths. *Fuzzy Sets and Systems*, 142(3):335–357, 2004.

[27] T. Okada, S. Soper. A shortest path problem on a network with fuzzy arc lengths. *Fuzzy Sets and Systems*, 109:129–140, 2000.

[28] F. Pedrycz, W. Gomide. *An Introduction to Fuzzy Sets: Analysis and Design*. MIT PRESS, London, 1998.

[29] A. Rosenfeld. Fuzzy graphs, fuzzy sets and their applications. In *Proceeding of US-Japan Sem., University of California, Berkeley,CA*, 1975.

[30] T. K. Sengupta, A. Pal. On comparing interval numbers. *European Journal of Operational Research*, 127:29–43, 2000.

[31] E.S. Shis, H.S Lee. Fuzzy multi-level minimum cost flow problems. *Fuzzy Sets and Systems*, 107:159–176, 1999.

[32] L.G. Sinclair, M.C. Tan. Wavelength assignment between the central nodes of the cost239 european optical network. In *11th UK Performance Engineering Workshop, Liverpool*, 1995.

[33] Takahashi, M. T. Contribuições ao Estudo de Grafos Fuzzy: Teoria e Algoritmos. Tese de doutorado, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, Maio 2004.

[34] R.R. Yager. Ranking fuzzy subsets over the unit interval. *Proceedings of the CDC*, pages 1435–1437, 1978.

[35] R.R. Yager. On choosing between fuzzy subsets. *Kybernects*, 9:151–154, 1980.

[36] R.R. Yager. A procedure for ordering fuzzy subsets of the unit interval. *Informations Sciences*, 24:143–161, 1981.

[37] L. Zadeh. Fuzzy sets. *Journal of Information and Control*, 8:338–353, 1965.