Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Telemática

Computação Evolutiva Aplicada à Resolução do Problema da Árvore Geradora Mínima com Parâmetros Fuzzy

Tiago Agostinho de Almeida

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de **Mestre em Engenharia Elétrica**.

Orientador:
Prof. Dr. Akebo Yamakami
FEEC/UNICAMP

Co-orientadora:
Profa. Dra. Márcia Tomie Takahashi
PETROBRÁS/RJ

Banca Examinadora

Profa. Dra. Rosângela Ballini	IE / UNICAMP
Prof. Dr. Takaaki Ohishi	FEEC / UNICAMP
Profa. Dra. Tatiane Regina Bonfim	FAC / Indaiatuba

Julho de 2006
FEEC - UNICAMP

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

AL64c

Almeida, Tiago Agostinho de
Computação evolutiva aplicada à resolução do problema da árvore geradora mínima com parâmetros fuzzy / Tiago Agostinho de Almeida. – Campinas, SP: [s.n.], 2006.

Orientadores: Akebo Yamakami, Márcia Tomie Takahashi
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação

1. Pesquisa operacional. 2. Teoria dos grafos. 3. Conjuntos difusos. 4. Computação evolutiva. 5. Algoritmo genético. 6. Sistema imune - Simulação (Computadores). I. Yamakami, Akebo. II. Takahashi, Márcia Tomie. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Título em Inglês: Evolutionary computation applied to solve the minimum spanning tree problem with fuzzy parameters.

Palavras-chave em Inglês: Operational research, Graph theory, Fuzzy sets, Evolutionary computation, Genetic algorithms, Immunological system.

Área de concentração: Telecomunicações e Telemática

Titulação: Mestre em Engenharia Elétrica

Banca examinadora: Rosângela Ballini, Takaaki Ohishi e Tatiane Regina Bonfim.

Data da defesa: 28/07/2006

Resumo

Este trabalho propõe meta-heurísticas baseadas em técnicas da computação evolutiva, que visam encontrar um conjunto de árvores geradoras mínimas para problemas de grafos, que possuem incertezas em relação às informações associadas aos parâmetros.

Resolver problemas dessa natureza é um processo NP-Completo, pois envolve um número enorme de comparações. A fim de contornar essa complexidade, este trabalho propõe um algoritmo genético e um sistema imunológico artificial, capazes de explorar eficientemente o espaço de busca e de obter resultados satisfatórios, sem a necessidade de confrontar todas as soluções entre si.

Palavras-chave: Grafo *Fuzzy*, Árvore Geradora Mínima *Fuzzy*, Teoria dos Conjuntos *Fuzzy*, Computação Evolutiva, Algoritmo Genético, Sistema Imunológico Artificial.

Abstract

This work proposes heuristical approaches based on evolutionary computation, whose goal is to find a set of minimum spanning trees in graphs that contain uncertainties in their parameters.

These kind of problems is a NP-hard one, because it involves an enormous number of comparisons. In order to avoid this complexity, this work proposes a genetic algorithm and an artificial immune system, that explore efficiently the search space of solutions to looking for satisfactory results, without the necessity of comparing all possible solutions.

Keywords: Fuzzy Graph, Fuzzy Minimum Spanning Tree, Fuzzy Set Theory, Evolutionary Computation, Genetic Algorithm, Artificial Immune System.

"Não basta ensinar ao homem uma especialidade,porque se tornará assim uma máquina utilizável, mas não uma personalidade. É necessário que adquira um sentimento, um senso prático daquilo que vale a pena ser empreendido, daquilo que é belo, do que é moralmente correto. A não ser assim, ele se assemelhará, com seus conhecimentos profissionais, mais a um cão ensinado do que a uma criatura harmoniosamente desenvolvida. Deve aprender a compreender as motivações dos homens, suas quimeras e suas angústias, para determinar com exatidão seu lugar preciso em relação a seus próximos e à comunidade."

Albert Einstein

Agradecimentos

Ao meu orientador e co-orientadora, Profs. Akebo Yamakami e Márcia Tomie Takahashi, sou grato pela oportunidade e pela orientação cujos conselhos e sugestões foram muito valiosos tanto para este trabalho quanto para a minha vida.

Aos meus pais Jurandy e Sônia, por me ensinarem a dar valor às coisas mais preciosas que a vida nos oferece.

A minha esposa, Ana Carolina, pela dedicação, companheirismo, paciência e carinho.

Ao meu irmão Jurandy Jr., pelas correções e sugestões, que foram fundamentais para melhorar o texto.

Aos meus grandes amigos Paulo, Fabrício, Fábio e Vânia pelas sugestões e pelo apoio.

Aos demais colegas de pós-graduação, pelas críticas e sugestões.

A minha família pelo apoio durante esta jornada.

À Unicamp, pela infra-estrutura.

À CAPES, pelo apoio financeiro.

Finalmente, agradeço a Deus por todas as oportunidades que Ele me oferece.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$G : (N, A)$	-	Grafo G composto por N nós e A arestas
c_{ij}	-	Custo da aresta que conecta o nó i ao nó j
(c_i, c_m, c_s)	-	Custo triangular *fuzzy*: c_i representa o limitante inferior, c_m o custo modal e c_s o custo superior
T	-	Árvore Geradora
T^*	-	Árvore Geradora Mínima
$Poss$	-	Medida de Possibilidade
Sup	-	Valor supremo
AE	-	Algoritmo Exato
AG	-	Algoritmo Genético
SIA	-	Sistema Imunológico Artificial
AGM	-	Problema da Árvore Geradora Mínima

Capítulo 1

Introdução

A teoria dos grafos é comumente utilizada na área da engenharia para resolver problemas que podem ser representados na forma de redes (Ahuja et al. 1993, Bazaraa et al. 1990). Ela fornece uma modelagem prática e consistente de um problema, facilitando a implementação de algoritmos que auxiliam a obter a sua solução (Ahuja et al. 1993).

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 se percorrer um grafo sem repetir arestas e retornar ao ponto inicial. Entretanto, foi na segunda metade do século XX, que essa teoria teve um enorme crescimento, com inúmeros problemas sendo propostos (Ahuja et al. 1993, Bazaraa et al. 1990). Atualmente, problemas bem conhecidos como caminho mínimo, árvore geradora mínima, fluxo máximo, emparelhamento, entre outros são muito estudados por essa teoria (Goldbarg & Luna 2000).

O problema da árvore geradora mínima aparece em uma série de aplicações (Chunde 1996, Chang & Lee 1999, Raidl & Julstrom 2003), por exemplo, na instalação de linhas telefônicas (ou elétricas) entre um conjunto de localidades utilizando a infra-estrutura das rodovias com o menor uso de material. Outros casos, como análise de *clusters*, armazenamento de informações, entre outros, também podem ser resolvidos por essa modelagem, que possui eficientes algoritmos: *Kruskal*, *Prim* e *Sollin* (Ahuja et al. 1993, Goldbarg & Luna 2000).

Em problemas reais é comum encontrar-se incertezas nas informações associadas, pois muitas vezes os dados são imprecisos e os conceitos, como cheio e vazio, alto e baixo, quente e frio, perto e longe, etc, dependem do ponto de vista do observador. Dessa forma, as tomadas de decisões baseiam-se no conhecimento individual, sem ter certeza sobre essas informações (Gomide & Pedrycz 1998).

Ao trabalhar com dados incertos, uma informação deixa de ser representada por um único valor e passa a ser representada por um conjunto. Assim, o uso da teoria clássica dos conjuntos torna-se inviável devido a sua ineficiência no tratamento de informações imprecisas. Entretanto, essas incertezas podem ser estudadas e modeladas de forma mais robusta utilizando a teoria dos conjuntos nebulosos, também conhecida como teoria dos conjuntos *fuzzy* (Zadeh 1965, Dubois & Prade 1980, Gomide & Pedrycz 1998).

A teoria dos conjuntos nebulosos foi introduzida por Lotfi A. Zadeh em 1965 (Zadeh 1965). A partir de então, essa teoria passou a desempenhar um papel fundamental nas mais diversas áreas do processamento de informação, representando uma ponte de ligação entre o processamento simbólico e o numérico. Com ela é possível utilizar símbolos (termos lingüísticos) aos quais estão associadas semânticas bem definidas que, após serem convertidas em funções de pertinência de conjuntos nebulosos, possibilitam o seu processamento em forma numérica. Em sistemas baseados em regras nebulosas, as informações imprecisas e incompletas são tratadas de forma lingüística, em analogia às informações sensoriais recebidas pelos órgãos e interpretadas pelo cérebro (Gomide & Pedrycz 1998).

O tratamento de incertezas em problemas de grafos e as principais definições de grafos *fuzzy* foram propostos por Rosenfeld em 1975 (Rosenfeld 1975), 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 teoria quanto na prática. Isso se deve, as diferentes abordagens que um problema de grafos *fuzzy* pode ter, pois os níveis de incerteza podem estar associados tanto na estrutura (nós e/ou arestas) quanto nos parâmetros (custo, capacidade, demanda).

Segundo Ahuja et al. (1993), grande parte dos problemas de grafos, embora com número finito de possibilidades, se mostra computacionalmente intratável. Um dos motivos é que problemas desse tipo tornam-se mais complexos quanto maior o número de nós.

Em muitos casos, é praticamente impossível garantir a solução ótima global do problema e somente com a utilização de heurísticas é possível obter soluções aproximadas (Michalewicz 1996).

Atualmente, existe um grande número de heurísticas que são utilizadas para resolver diversos tipos de problemas (Bäck et al. 2000a, Castro 2001, Holland 1992). Muitas delas procuram simular processos evolutivos, por exemplo, os algoritmos genéticos e os sistemas

imunológicos artificiais. Essas heurísticas fazem parte de um grupo chamado computação evolutiva.

A computação evolutiva, área da computação que simula os mecanismos de evolução através de computadores, possibilitou o surgimento de uma nova filosofia de máquinas inteligentes. Esse conceito tem por base a capacidade de um sistema adaptar seu comportamento para atingir seus objetivos numa variedade de situações. Dessa, inteligência e evolução são dois conceitos intimamente relacionados.

A simulação de processos evolutivos biológicos permite o entendimento de como a evolução guia os seres vivos na direção de um maior nível de inteligência. A evolução, caracterizada por diferentes níveis de representação (gene, cromossomo, indivíduo, espécie, ecossistema), converge para soluções que apresentam um comportamento mais adaptado ao ambiente.

A modelagem do processo evolutivo emprega uma série de algoritmos de otimização baseados em regras simples e implementados como métodos de busca em superfícies de resposta de desempenho (*fitness surface*). O processo de otimização, inerente à seleção dos elementos melhores adaptados, aumenta a qualidade média das soluções obtidas ao longo das gerações. Entretanto, o grande potencial dos algoritmos baseados em computação evolutiva surge da integração da ampla exploração do espaço de busca com um processo de busca melhor localizada, denominado explotação. Essa integração resulta num alto grau de robustez, que permite aplicar a computação evolutiva em vários problemas práticos, nos quais outras estratégias de solução se mostram inócuas (Bäck et al. 2000a, Bäck et al. 2000b, Castro 2001).

Objetivo

Neste trabalho, é apresentado o problema da árvore geradora mínima com parâmetros *fuzzy* (ver Seção 2.2 do Capítulo 2), a complexidade do problema em questão (ver Seção 2.5 do Capítulo 2) e uma forma de resolvê-lo baseando-se em conceitos da teoria da possibilidade (Zadeh 1978) (ver Seção 2.4 do Capítulo 2). Também são propostos um método exato (Capítulo 3) que garante encontrar o melhor conjunto-solução para o problema, um algoritmo genético (Capitulo 4) e um sistema imunológico artificial (Capítulo 5) cujo objetivo é encontrar um conjunto-solução aproximado, visando contornar a questão da complexidade.

Revisão Bibliográfica

Os trabalhos mais relevantes encontrados na literatura que tratam do problema da árvore geradora mínima *fuzzy* são:

- Chunde (1996) estudou casos de grafos cuja estrutura é constituída por nós *crisp* (aqueles que não apresentam incertezas) e arestas associadas a graus de pertinência (ver Seção 2.3 no Capítulo 2). São propostos três algoritmos considerando algumas situações da rede, sem considerar o custo das arestas como números *fuzzy*.

- Chang & Lee (1999) analisaram os problemas da árvore geradora mínima, PERT e de caminho mínimo. Eles utilizaram algoritmos clássicos e propuseram um método (OERI), para o ordenamento de números *fuzzy* em problemas de redes. Este resolve o problema utilizando um algoritmo *crisp* para obter a solução.

- Takahashi (2004) desenvolveu um estudo sobre o problema da árvore geradora mínima com estrutura *crisp* e parâmetros *fuzzy*. Sua proposta baseia-se na teoria de possibilidade (Zadeh 1978) (ver Seção 2.4 no Capítulo 2), de forma a encontrar o conjunto-solução do problema. Para isso, a abordagem de Okada (2001) é utilizada com o objetivo de explorar a interatividade entre as arestas e assim, calcular o grau de possibilidade que cada uma delas tem de participar da solução.

- Em Almeida et al. (2005a) e Almeida et al. (2005b) foi realizado um estudo sobre o problema da árvore geradora mínima com parâmetros *fuzzy* e proposto um algoritmo genético com características adaptadas à obtenção do conjunto *fuzzy* de soluções. Nesses trabalhos, o grau de possibilidade é calculado utilizando a abordagem de Dubois & Prade (1980) de forma a encontrar o conjunto solução composto por árvores que tenham alto grau de possibilidade de serem as melhores soluções.

Organização

Para facilitar a leitura e a compreensão deste trabalho, ele foi estruturado da seguinte maneira:

- No Capítulo 2 é apresentada uma introdução sobre os principais assuntos que constituem a base teórica deste trabalho, a complexidade do problema e a escolha dos métodos de resolução adotados.

- Nos Capítulos 3, 4 e 5 são propostos um método exato, um algoritmo genético e um sistema imunológico artificial, respectivamente, que visam encontrar o conjunto-solução para o problema.

- No Capítulo 6 são apresentados os experimentos e os resultados obtidos para cada um dos métodos propostos.

- No Capítulo 7 são apresentadas as conclusões e propostas para trabalhos futuros.

Capítulo 2

Base Teórica

Neste capítulo são apresentados os principais conceitos teóricos necessários para a compreensão deste trabalho, a complexidade do problema, bem como a escolha dos métodos de resolução adotados. Ele está organizado da seguinte maneira:

- Na Seção 2.1 é feita uma revisão sobre a teoria dos grafos clássica;
- Na Seção 2.2 é feita uma revisão sobre o problema da árvore geradora mínima;
- Na Seção 2.3 são introduzidos os conceitos necessários sobre problemas de grafos *fuzzy* e da árvore geradora mínima *fuzzy*, e também são apresentados alguns tópicos da teoria dos conjuntos nebulosos que são necessários para a compreensão do problema e de sua resolução;
- Na Seção 2.4 é explicada a teoria da possibilidade utilizada como base para a resolução do problema;
- Na Seção 2.5 são apresentados os principais pontos de motivação para a resolução do problema, sua complexidade e a escolha dos métodos de resolução;

2.1 Noções sobre a Teoria de Grafos

Nesta Seção, apresentamos os conceitos básicos da teoria de grafos necessário para a compreensão deste trabalho. Para um estudo mais completo, sugerimos o Capítulo 2 de (Ahuja et al. 1993) e os capítulos subseqüentes com os problemas relativos a grafos. Outras referências utilizadas neste trabalho para os conceitos da teoria de grafos e os seus algoritmos

são: (Harary 1972), (Bazaraa et al. 1990), (Wilson & Watkins 1990) e (Goldbarg & Luna 2000).

Para facilitar o uso, os tópicos estão organizados em ordem alfabética.

Arco: um par ordenado (i, j) sendo i o nó de origem do arco e j o nó de destino. Neste caso, se a_{ij} é um arco , então $a_{ij} \neq a_{ji}$ (Figura 2.1).

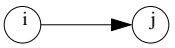

Figura 2.1: Exemplo de um arco (i, j)

Aresta: um par não ordenado $\{i, j\}$. Neste caso, se a_{ij} é uma aresta, temos necessariamente que $a_{ij} = a_{ji}$ (Figura 2.2).

Figura 2.2: Exemplo de uma aresta $\{i, j\}$

Árvore: um subgrafo conectado e sem ciclos.

Árvore geradora: seja um grafo $G = (N, A)$. Uma árvore geradora é um subgrafo gerador de G que não possui ciclos (Figura 2.3).

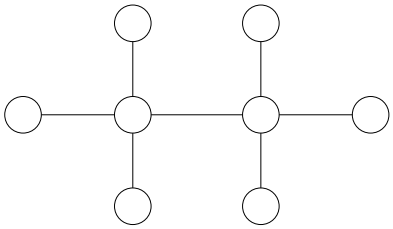

Figura 2.3: Exemplo de uma árvore geradora

Árvore geradora mínima: seja $G = (N, A)$ um grafo e $c_{ij} > 0$ os custos dos arcos (ou arestas). Uma árvore geradora mínima é uma árvore geradora, T, com menor custo total $C = \sum_{ij \in T} c_{ij}$. Veja Figura 2.17, na Seção 2.2 deste Capítulo.

Cadeia: uma seqüência não orientada de nós e arestas, sem repetições de nós (Figura 2.4).

Caminho: uma seqüência orientada de nós e arcos, sem repetições de nós (Figura 2.5).

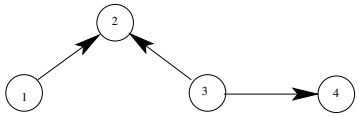

Figura 2.4: Exemplo de uma cadeia

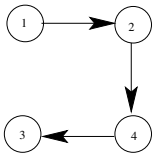

Figura 2.5: Exemplo de um caminho

Capacidade de um arco (ou aresta): são valores referentes a quantidade mínima (l_{ij}) e máxima (u_{ij}) de fluxo (x_{ij}) que pode ser transportada pelo arco (ou aresta).

Ciclo: uma cadeia fechada (Figura 2.6).

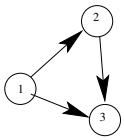

Figura 2.6: Exemplo de um ciclo em um grafo

Circuito: um caminho fechado (Figura 2.7).

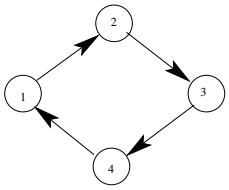

Figura 2.7: Exemplo de um circuito em um grafo

Clique: um grafo onde existe um arco conectando cada par de nós.

Componente: é um subgrafo conectado de G que não é subgrafo próprio de nenhum outro subgrafo de G.

Conjunto de corte $[X, \bar{X}]$**:** seja um grafo $G = (N, A)$. Seja $X \subset N$ e $\bar{X} = N - X$. $[X, \bar{X}] = \{(i, j) \in A | i \in X, j \in \bar{X}\}$, ou seja, um conjunto de corte é formado pelos arcos que conectam X a $\bar{X}$ (Figura 2.8).

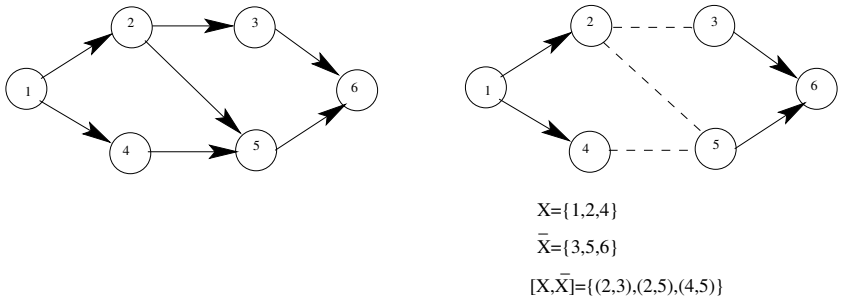

Figura 2.8: Exemplo de um conjunto de corte $[X, \bar{X}]$

Custo de um arco: para um problema de grafos é o valor associado ao escolher uma aresta (ou arco) para a solução de um problema. Se um fluxo estiver associado ao grafo, é o valor associado para se passar uma unidade de fluxo nesta aresta (ou arco), desde que não viole a capacidade deste.

Floresta: um grafo desconexo sendo que todas as suas componentes são árvores (Figura 2.9).

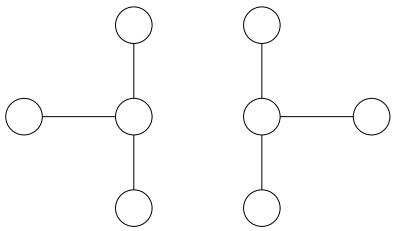

Figura 2.9: Exemplo de floresta

Fluxo: é uma maneira de se enviar objetos, pessoas, etc. sobre arestas (ou arcos) de um nós para outro do grafo. Associamos números aos nós e arcos (ou arestas) de um grafo, normalmente designando uma capacidade (mínima e máxima) destes nós e arcos (ou arestas).

Grafo: ou rede. Um par de conjuntos N e A, não vazios, finitos sendo que N é o conjunto de nós e A é o conjunto de arcos (ou arestas) tal que para $(i, j) \in A$ (ou $\{i, j\} \in A$), então $i, j \in N$.

OBS: Alguns autores não fazem distinção entre estes dois conceitos. Outros admitem que o conceito de rede pressupõe a presença de fluxo.

Grafo bipartido: seja um grafo $G = (N, A)$. O grafo é bipartido se $N = N_1 \bigcup N_2$ tal que $N_1 \bigcap N_2 = \emptyset$, $|N_1| = |N_2| = n$ e para $(i, j) \in A$, temos que $i \in N_1$ e $j \in N_2$ (Figura

2.10).

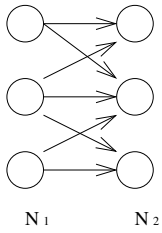

Figura 2.10: Exemplo de um grafo bipartido

Grafo completo: um grafo onde existe um arco (ou aresta) conectado cada par de nós do grafo. Neste caso, temos para um grafo com n nós, $C_n^2 = \frac{n(n-1)}{2}$ arcos (ou arestas) (Figura 2.11).

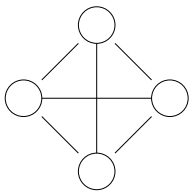

Figura 2.11: Exemplo de grafo completo

Grafo conectado: um grafo (ou rede) é conectado se sempre existe um caminho (ou cadeia) ligando qualquer par de nós. Caso contrário o grafo é dito desconectado. Um grafo é dito fortemente conectado se existe um caminho ligando dois nós quaisquer. Se existe uma cadeia ligando dois nós quaisquer, então o grafo é dito fracamente conectado.

Figura 2.12: Exemplo de grafo fortemente e fracamente conectado

Grafo misto: um grafo $G = (N, A)$ onde A é um conjunto de arcos e arestas (Figura 2.13).

Grafo não orientado: um grafo $G = (N, A)$ onde A é um conjunto de arestas. Veja Figura 2.11.

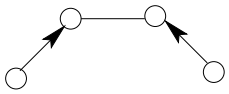

Figura 2.13: Exemplo de um grafo misto

Grafo orientado (digrafo): um grafo $G = (N, A)$ onde A é um conjunto de arcos. Veja Figura 2.14.

Grau de um nó: se refere ao número de arcos (ou arestas) incidentes no referido nó. Na Figura 2.11, todos os nós tem o mesmo grau 3.

Laço: um arco (ou aresta) em que os nós de origem e destino do arco são os mesmo, ou seja $(i, j),\ i = j$ (Figura 2.14).

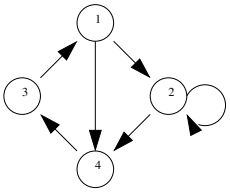

Figura 2.14: Exemplo de um laço em um grafo orientado

Nó: também conhecidos como vértices ou pontos. Um dos conjuntos que caracterizam um grafo.

Nó-folha: ou nó terminal. Nó com grau 0 ou 1. Na Figura 2.15, os nós s e t são nós terminais, ou nós folhas.

Nó-interno: qualquer nó que não seja um nó-folha.

Nó origem: em um problema de fluxos em redes, um nó de origem é um nó capacitado. Se b_i é o parâmetro referente a capacidade do nó i, então $b_i > 0$. Está associado ao depósito, à matéria-prima, etc. Na Figura 2.15, o nó s é um nó destino.

Nó destino: em um problema de fluxos em redes, um nó de destino é um nó capacitado. Se b_i é o parâmetro referente a capacidade do nó i, então $b_i < 0$. Está associado ao consumidor, a demanda, etc. Na Figura 2.15, o nó t é um nó destino.

Nó de transbordo: em um problema de fluxos em redes, um nó de transbordo é um nó não capacitado. Se b_i é o parâmetro referente a capacidade do nó i, então $b_i = 0$. Na Figura 2.15, os nós 1, 2, 3 e 4 são nós de transbordo (Figura 2.15).

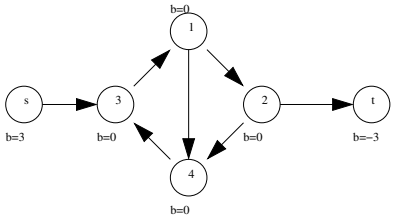

Figura 2.15: Exemplo de nós capacitados

Rede: vide grafo.

Subgrafo: $G' = (N', A')$ do grafo $G = (N, A)$ é um sugrafo se $N' \subseteq N$ e $A' \subseteq A$.

Subgrafo gerador: é um subgrafo onde $N' \equiv N$ (Figura 2.16).

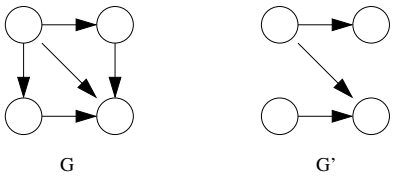

Figura 2.16: Exemplo de um sugrafo gerador

2.2 Árvore Geradora Mínima

Dado um grafo $G : (N, A)$ conexo e não-direcionado formado por N nós e A arestas, uma árvore geradora T de G é um subgrafo com $N - 1$ arestas que conecta todos os nós de G sem formar ciclos. Um único grafo pode ter muitas árvores geradoras diferentes. A cada aresta, geralmente, está associado um "peso" $c_{(ij)}$ que representa o "custo" para se ir de um determinado nó i a outro nó j. Uma árvore geradora mínima T^* de G (Figura 2.17) é a árvore geradora cuja somatória dos pesos das arestas $c(T^*) = \sum_{ij \in T^*} c_{ij}$ é menor que a de qualquer outra árvore geradora de G (Ahuja et al. 1993, Bazaraa et al. 1990, Goldbarg & Luna 2000).

Um exemplo prático de aplicação da árvore geradora mínima pode ser dado pelo seguinte caso: uma companhia de TV a cabo precisa instalar cabos de transmissão em uma nova vizinhança. Sabendo que é preciso enterrar os cabos ao longo de determinados trajetos, é

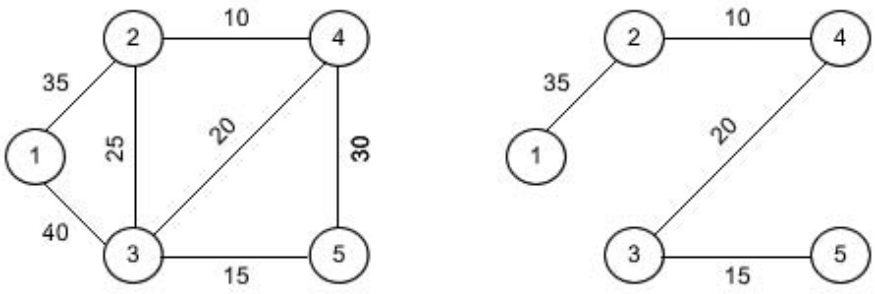

Figura 2.17: Grafo G e sua árvore geradora mínima T^*

possível obter um grafo que representa as casas que são conectadas por esses trajetos. Alguns caminhos podem ser mais custosos que outros, porque eles são mais longos, ou por requererem que o cabo seja enterrado mais ao fundo, então, esses trajetos seriam representados por arestas com pesos maiores. Uma árvore geradora desse grafo seria um subconjunto de caminhos que conecta todas as casas sem formar ciclos. Podem haver diversas árvores geradoras possíveis, mas a árvore geradora mínima (T^*) seria aquela com menor custo total.

O problema da árvore geradora mínima aparece em uma série de aplicações, ou como um subproblema, entre elas estão: instalação de linhas telefônicas ou elétricas entre um conjunto de localidades, análise de *clusters*, armazenamento de informações, etc.

O primeiro algoritmo para encontrar a árvore geradora mínima foi desenvolvido pelo cientista Otakar Boruvka em 1926. Sua finalidade era encontrar uma cobertura elétrica eficiente para a cidade da Boêmia. Atualmente, os algoritmos de Prim e Kruskal também são igualmente utilizados, e todos eles resolvem o problema em tempo polinomial (Ahuja et al. 1993).

2.3 A Questão da Incerteza

Em problemas reais é comum encontrarmos incertezas associadas às informações dos problemas. Muitas vezes, dados que envolvem conceitos como perto e longe, quente e frio, alto e baixo, etc; são imprecisos e dependem do ponto de vista de cada observador.

Em problemas de grafos, as incertezas podem estar presentes em diversos níveis: na estrutura do grafo (quando não é possível determinar com certeza se uma aresta está conectada em um nó, ou até mesmo, se aquela aresta existe), nos parâmetros (quando não se

sabe exatamente quanto vale o peso das arestas) e os dois casos ocorrendo ao mesmo tempo (Takahashi 2004).

Um exemplo de grafo e de árvore geradora mínima com parâmetros *fuzzy* pode ser visto na Figura 2.18. As incertezas relacionadas aos parâmetros podem ser de várias naturezas, como por exemplo: não se sabe exatamente qual deve ser a profundidade para a instalação do cabo de TV entre as casas 2 e 4, mas acredita-se que será **em torno de** 10 metros, e assim, é possível associar uma função de pertinência para este custo.

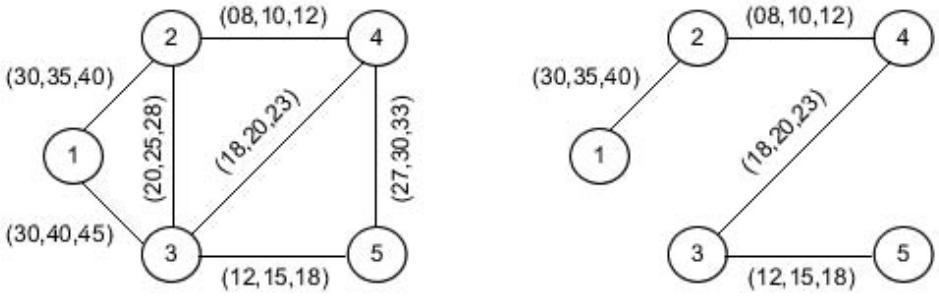

Figura 2.18: Grafo G e sua árvore geradora mínima T^* com parâmetros *fuzzy*

Para o grafo *crisp* da Figura 2.17 foi encontrada uma árvore geradora mínima com custo total igual a 80. Entretanto, para o grafo com parâmetros *fuzzy* da Figura 2.18 foi obtida uma árvore geradora mínima com custo *fuzzy* igual a (68,80,93) que está associado à função de pertinência representada pela Figura 2.19.

Figura 2.19: Função de pertinência associada ao custo da T^* do grafo da Figura 2.18

Os custos associados às arestas para o grafo da Figura 2.18 são representados por *números fuzzy* (ver Definição 2.4 na Seção 2.3.1). Neste caso, como pode ser observado na função de

pertinência da Figura 2.19, os números *fuzzy* estão representados por uma função triangular, e assim, estes podem ser definidos como "números triangulares *fuzzy*"do tipo (c_i, c_m, c_s), no qual c_i equivale ao custo (limitante) inferior, c_m ao custo modal, aquele com maior grau de pertinência, e c_s ao custo (limitante) superior. Existem várias formas de expressar uma função de pertinência, sendo que, as mais utilizadas na literatura são as funções: Gaussianas, Triangulares e Trapezoidais (Gomide & Pedrycz 1998).

Considere as árvores geradoras mostradas na Figura 2.20:

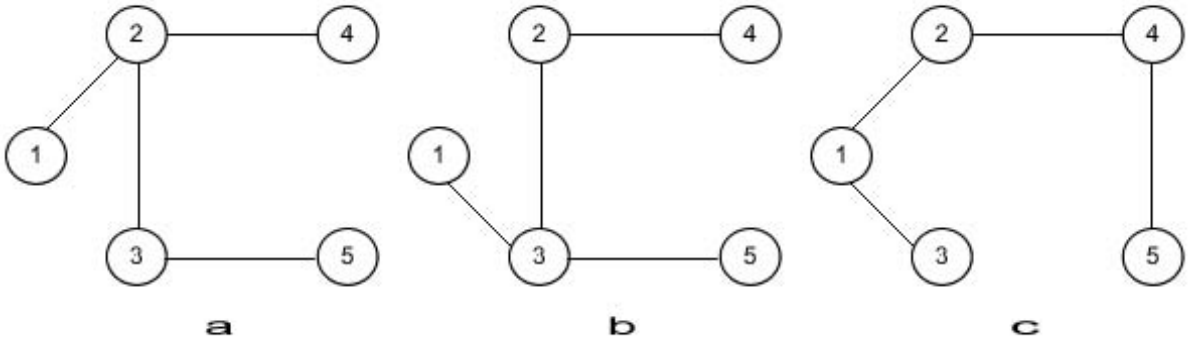

Figura 2.20: Árvores geradoras obtidas do grafo da Figura 2.18

a árvore (a) tem custo (70,85,98), (b) tem custo (70,90,103) e (c) (95,115,130). Suas funções de pertinência podem ser vistas na Figura 2.21

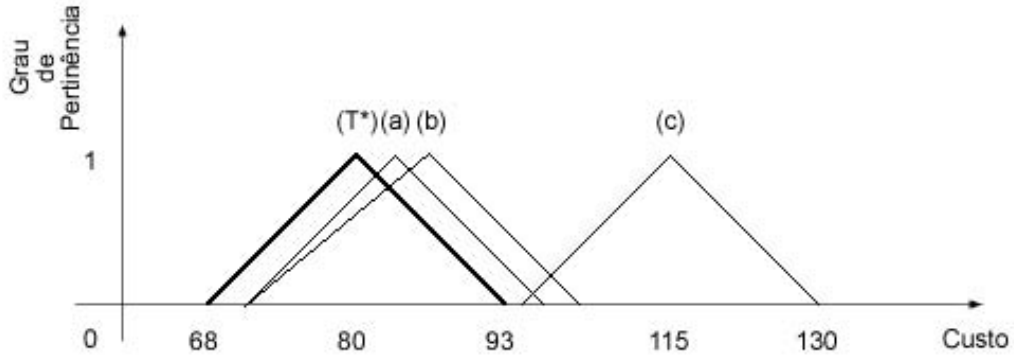

Figura 2.21: Comparação dos custos *fuzzy* das árvores da Figura 2.20

Observando a Figura 2.21, percebe-se que não há possibilidade da árvore geradora (c) ser menor que a solução previamente encontrada na Figura 2.18 (T^*), pois o menor valor possível

que o custo total de (c) pode assumir (custo limitante inferior) é maior que o maior custo que T^* pode obter (limitante superior). Já para as árvores (a) e (b) existe uma possibilidade de serem menores que a solução encontrada, pois existe pelo menos um custo em (a) e em (b) que podem ser menores ou iguais ao custo limitante superior de T^*. Assim, dizemos que existe um **grau de possibilidade** de uma árvore pertencer ao conjunto solução do problema. Blue et al. (2002) entende este valor como sendo o grau de pertinência da solução no conjunto *fuzzy* de soluções. Com isso, é evidente que uma boa solução para o problema da árvore geradora mínima com parâmetros *fuzzy* deve ser composta pelo conjunto de árvores que possuem alto grau de possibilidade de serem a árvore geradora com menor custo total.

Ao trabalhar com informações incertas, uma unidade deixa de ser representada por um número e passa a ser representada por um conjunto de valores, de forma que, o uso da teoria clássica dos conjuntos torna-se inviável pela sua ineficiência no tratamento de informações imprecisas. Entretanto, essas incertezas podem ser modeladas e tratadas de uma forma mais robusta utilizando a teoria dos conjuntos nebulosos (Zadeh 1965, Dubois & Prade 1980, Gomide & Pedrycz 1998).

Os conjuntos nebulosos são uma extensão da teoria clássica dos conjuntos. Enquanto que nesta os conjuntos são denominados *crisp* e um dado elemento do universo de discurso $X \subseteq \mathbb{R}^n$ pode pertencer ou não pertencer ao referido conjunto, na teoria dos conjuntos nebulosos existe um grau de pertinência $\mu_A(x)$ caracterizado por uma função de pertinência $\mu_A : X \rightarrow [0, 1]$ associado a cada elemento de um determinado conjunto. Ele pode ser interpretado como sendo o grau de pertinência dos elementos do conjunto X em relação ao conjunto A, ou seja, o quanto é possível um elemento x de X pertencer ao conjunto A. Desta forma, o conjunto A é caracterizado como um conjunto nebuloso.

Por meio de uma função de pertinência é possível associar os elementos do domínio a seus graus de pertinência, assim como mostrado na Figura 2.19.

2.3.1 Conceitos Básicos sobre Conjuntos *Fuzzy*

Definição 2.1 *Um conjunto fuzzy A é normal se sua função de pertinência (μ_A) possui pelo menos um valor 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 2.2 *Sendo X o universo de discurso, o suporte de um conjunto fuzzy A é dado por:*

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

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

Definição 2.3 *Um conjunto fuzzy A é convexo se a 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 e $x_2 \in X$ e $\lambda \in [0,1]$.

Definição 2.4 *Um número fuzzy é um conjunto fuzzy convexo e normal cuja função de pertinência é representada por uma função contínua, onde apenas um único elemento possui grau de pertinência igual a um ($\mu_A(x) = 1$ para exatamente um único $x \in X$).*

2.4 Teoria da Possibilidade

Seja um grafo $G : (N, A)$ no qual a cada aresta $ij \in A$ está associado um custo representado por um número *fuzzy* $c_{ij} \in \mathbb{R}^n$. Considere duas árvores geradoras T^1 e T^2, $T^2 > T^1$. O grau de possibilidade de T^2 ser menor do que T^1 pode ser obtido pela equação definida por Dubois & Prade (1980) (Figura 2.22) :

$$w = Poss(\sum_{ij \in T^2} c_{ij} \leq \sum_{ij \in T^1} c_{ij}) = \sup_{u \leq v} min\{\mu_{T^1}(u), \mu_{T^2}(v)\} \tag{2.1}$$

sendo:

- $Poss$: a medida de Possibilidade;
- μ_{T^1} e μ_{T^2}: as funções de pertinência dos custos totais das árvores T^1 e T^2, respectivamente;
- *supmin*: 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 μ_{T^1} e μ_{T^2};

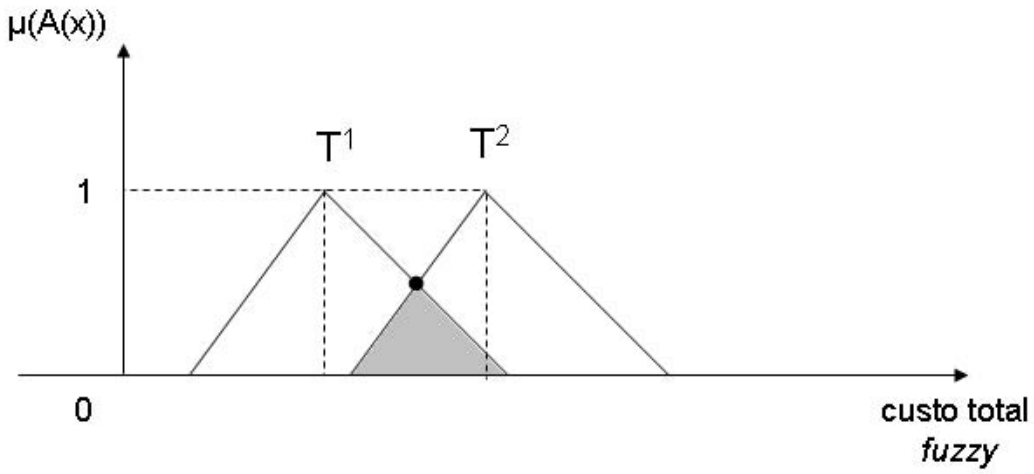

Figura 2.22: Grau de Possibilidade entre dois números fuzzy

Considerando T^* como a árvore geradora com maior grau de possibilidade de ser a mínima, para encontrar o conjunto das árvores que possuem alto grau de possibilidade de ter custo menor ou igual ao de T^* é necessário encontrar todas as árvores geradoras $T^k \in T$, e compará-las com T^*. Fazendo isto, obtém-se o grau de possibilidade dado pela equação (Okada 2001):

$$D_T = \min_{T^k \in T} \{Poss(\sum_{ij \in T^k} c_{ij} \leqslant \sum_{ij \in T^*} c_{ij})\} \tag{2.2}$$

Isso torna o problema de difícil solução, pois além de ter que enumerar todas as soluções, a comparação entre elas torna o problema NP-Completo (Takahashi 2004).

Proposição 2.1 *De Takahashi (2004) temos que, dados dois números triangulares fuzzy, normalizados* $a = (a_i, a_m, a_s)$ *e* $b = (b_i, b_m, b_s)$. *Temos que se* $a_m \leq b_m$, *então* $Poss(a \leq b) = 1$.

Prova: utilizando a definição do grau de possibilidade entre dois números *fuzzy* dado pela Equação 2.1:

$$Poss(a \leq b) = \sup_{T^k \in T} \{min\{\mu_a(u), \mu_b(v)\}\}$$

Supondo que $u = a_m$ e $v = b_m$, então, se $a_m \leq b_m$, temos que $u \leq v$. Sendo os números *fuzzy* normalizados:

$$\mu_a(u) = \mu_b(v) = 1 \Rightarrow min\{\mu_a(u), \mu_b(v)\} = 1$$

Portanto, $Poss(a \leq b) = 1$. ■

Desta forma, se for encontrada a árvore com menor valor modal, será encontrada a árvore geradora com maior grau de possibilidade de ser a árvore geradora mínima *fuzzy*. Esta pode ser obtida pelo grafo *crisp* G_m, com custo das arestas igual a $c'_{ij} = (c_m)_{ij}$.

Proposição 2.2 *De Takahashi (2004) temos que, T^0 é uma árvore geradora mínima do grafo crisp G_m, com os valores modais (c_m) de G sendo os custos associados às arestas do grafo crisp. Seja a árvore geradora fuzzy T^*, que possui as mesmas arestas de T^0, com custo igual a $c^* = \sum_{ij \in T^*} (c_m)_{ij}$. Se T^* for a solução ótima do problema, então $w = Poss(\sum_{ij \in T^*} c_{ij} \leq \sum_{ij \in T^k} c_{ij}) = 1$, sendo T^k qualquer árvore geradora fuzzy de G.*

Prova: Suponha que exista um T^k com custo total c^k tal que $w = Poss(\sum_{ij \in T^*} c_{ij} \leq \sum_{ij \in T^k} c_{ij}) < 1$. Pela Proposição 2.1, temos que $c^* > c^k$. Porém, como c^* é o menor valor encontrado para o grafo G_m, temos que $c^* \leq c^k$, gerando uma contradição. Portanto:

$$w = Poss(\sum_{ij \in T^*} c_{ij} \leq \sum_{ij \in T^k} c_{ij}) = 1$$

■

2.5 Motivação

Encontrar a árvore geradora mínima é um problema cujos métodos envolvem busca em um espaço que cresce exponencialmente conforme se aumenta o número de nós. Além disso, quando se trabalha com grafos cujas arestas estão associadas a custos *fuzzy* a complexidade aumenta, pois o fato de ter que escolher os subconjuntos de arestas a serem retiradas e os grupos que podem substituir cada um dos subconjuntos, faz com que o problema seja combinatorial (ver Seção 2.4).

O número de árvores geradoras em um grafo completo $G : (N, A)$ é da ordem de m^{m-2}, $m = |N|$ (Raidl & Julstrom 2003). Por exemplo, de um grafo completo com 50 nós é possível obter $50^{48} \cong 3,55 \times 10^{81}$ árvores. É inviável utilizar uma busca exaustiva para encontrar a árvore de menor custo, pois supondo que 1000 computadores analisem 1000 árvores por segundo cada um, levariam aproximadamente 10^{68} anos para analisar todas as árvores possíveis (estima-se que o universo tenha 10^{13} anos). Este é um espaço de busca grande por qualquer parâmetro, mesmo para um grafo de 50 nós, que é considerado pequeno se comparado a casos reais como o Brasil, por exemplo, que possui mais de 5000 cidades.

Considerando que a solução pode ser composta pelo conjunto de árvores com alto grau de possibilidade de serem menores ou iguais a T^*, e que ainda não existe nenhum método eficiente ou específico para solucionar este tipo de problema, propomos heurísticas para encontrar uma solução aproximada satisfatória tentando contornar a questão da complexidade.

Entre as possíveis heurísticas, escolhemos aquelas que imitam processos naturais em busca de equilíbrio, em geral otimizado: algoritmos genéticos e sistemas imunológicos artificiais.

Capítulo 3

O Método Exato

Neste capítulo será apresentado um algoritmo proposto por Takahashi (2004), que possibilita encontrar o conjunto de árvores geradoras com maior grau de possibilidade de ser a árvore geradora mínima de um grafo com parâmetros *fuzzy*. Foi necessário realizar várias modificações para adequá-lo ao nosso método de resolução, pois em sua versão original, ele foi proposto para encontrar o grau de possibilidade que cada aresta do grafo tem de pertencer à árvore geradora mínima *fuzzy*.

3.1 Introdução

Para encontrar as melhores soluções (árvores) de um grafo constituído por uma estrutura *crisp* e arestas associadas a custos *fuzzy* do tipo (c_i, c_m, c_s) utilizando a Teoria de Possibilidade (Zadeh 1978) (ver Seção 2.4 no Capítulo 2) é necessário:

1. Encontrar a árvore geradora mínima *fuzzy* com grau máximo de possibilidade (T^*);

2. Encontrar uma solução parcial, ou seja, um conjunto de árvores no qual cada uma possui um grau de possibilidade de ser a árvore geradora mínima *fuzzy* em relação à solução de grau máximo (T^*) previamente encontrada.

3.2 Encontrar a árvore geradora mínima com o maior grau de possibilidade

Utilizando as Proposições 2.1 e 2.2 do Capítulo 2 podemos determinar qual é a árvore geradora mínima com maior grau de possibilidade. Segundo essas proposições, com a comparação dos valores modais (c_m) de dois números *fuzzy*, é possível verificar qual deles tem grau

de possibilidade igual a 1. Então, o número *fuzzy* com menor valor modal deve ser associado a solução com maior grau de possibilidade. Portanto, aplicando um método conhecido para se encontrar a árvore geradora mínima, como Kruskal, Prim, Boruvka, etc (Ahuja et al. 1993); utilizando o grafo *crisp* com os custos das arestas associados aos valores modais (c_m) dos custos *fuzzy*, encontramos a árvore geradora com maior grau de possibilidade de ser a árvore geradora mínima (T^*).

3.3 Encontrar o conjunto de árvores que tenham os melhores graus de possibilidade

Para encontrar o conjunto composto pelas melhores árvores geradoras com parâmetros *fuzzy* é necessário construir todas as árvores geradoras possíveis e compará-las uma a uma com T^*. Enumerar todas as árvores torna o problema não polinomial e faz-se necessário o desenvolvimento de técnicas que promovam uma redução no espaço de busca a fim de evitar comparações desnecessárias.

Para contornar esta questão, a árvore T^* será utilizada para gerar outras árvores. Com isso, deve-se privilegiar a construção de árvores com grau de possibilidade maior do que zero, pois árvores que possuem partes em comum costumam ter maiores graus de possibilidade (Okada 2001).

As novas árvores geradoras podem ser obtidas substituindo as arestas pertencentes à T^* por arestas do grafo $G : (N, A)$ que fazem parte do conjunto de corte da aresta retirada. Portanto, se retirarmos k arestas ($k = 1, ..., n-1$ sendo $n = |N|$) de T^*, podemos encontrar outras árvores geradoras que possivelmente tenham grau de possibilidade não nulo, pois estas possuem arestas em comum com T^*. Quando $k = n-1$, todas as arestas de T^* serão retiradas e então, resolvemos o problema com o novo grafo $G' = G - T^*$ e continuamos a enumeração a partir de T', comparando as novas árvores com T^* até a enumeração completa das árvores, quando $|A| < |N| - 1$.

A construção das árvores pode ser orientada de forma que o descarte das árvores que não farão parte do conjunto solução (grau de possibilidade nula) seja facilitado. Usando o teste de corte proposto por Blue et al. (2002) para o caso do caminho mínimo, é conveniente descartar as árvores que apresentam as seguintes características:

$$sup\{supp\{c_{T^*}\} < inf\{supp\{c_{T^k}\}\}, \forall T^k \in \top$$

isto é, se o valor supremo (sup) do conjunto suporte ($supp$) do custo total *fuzzy* de T^* (c_{T^*}) for menor que o valor ínfimo (inf) do conjunto suporte do custo total *fuzzy* da árvore T_k pertencente ao conjunto $\top$ de todas as árvores geradoras de G, então a árvore (T_k) deve ser descartada pois não há nenhuma possibilidade dela ser menor ou igual a T^*, e portanto, dizemos que o seu grau de possibilidade é igual a zero ($Poss(T_k \leq T^*) = 0$).

Para que o descarte seja feito de modo mais eficiente, no momento em que o número de arestas for igual a $k = |N| - 1$, a próxima árvore geradora a ser a base para a formação de outras será obtida a partir do grafo G'_i associado aos custos formados pelos limitantes inferiores dos custos *fuzzy* ($c'_i = inf\{supp\{c_{ij}\}\}$). A árvore geradora T'_i obtida do grafo G'_i será comparada a T^*. Caso esta árvore seja descartada pelo critério de corte dado acima, então todas as árvores geradoras que ainda não foram enumeradas também serão descartadas pois possuem grau nulo de possibilidade.

3.4 Algoritmo Exato

O Algoritmo Exato para resolver o problema da Árvore Geradora Mínima com Custos *Fuzzy* está dividido em dois procedimentos (vide Algoritmo 3.4.1).

Em geral, o Procedimento 1 é responsável por retirar arestas do grafo para formar novas árvores geradoras (T), podendo estas compartilharem arestas com T^* ou não.

No Procedimento 2 é calculado o grau de possibilidade da árvore geradora T ser menor ou igual a árvore geradora mínima *fuzzy* (T^*).

3.4.1 Exemplo de Funcionamento

Segue abaixo, um exemplo de funcionamento do Algoritmo Exato para encontrar o conjunto de árvores geradoras mínimas *fuzzy* para o grafo da Figura 2.18 da Seção 2.3 do Capítulo 2 (Tabela 3.1).

Etapa 1: encontrar o grafo *crisp* (Tabela 3.2) com custos das arestas associados aos valores modais dos custos *fuzzy* do grafo original (Tabela 3.1).

Algoritmo 3.4.1 Algoritmo Exato

Entrada: grafo $G : (N, A)$, sendo N o conjunto de nós e A o conjunto de arestas. Com custos c_{ij} representados por números *fuzzy* do tipo (c_i, c_m, c_s) associados a cada aresta do grafo significando o custo para se percorrer do nó i ao nó j.

Saída: conjunto solução composto pelas árvores que possuem maiores graus de possibilidade de serem a árvore geradora mínima *fuzzy*.

Procedimento 1:

Passo 0: (inicializar os parâmetros) encontrar a árvore geradora mínima *fuzzy* T^* associada a solução do grafo *crisp* G_c com custo *crisp* $c^* = c_m$. Sendo $c_{T^*} = \sum_{(ij)\in T^*} c_{ij}$ o custo total *fuzzy* de T^*, faça $\delta = sup\{supp\{c_{T^*}\}\}$.

Passo 1: (encontrar árvores com arestas coincidentes a T^*) para $k = 1$ até $k = |N| - 2$, faça:
- Para k arestas da árvore T^* (estas k arestas $\in R'$, conjunto das arestas retiradas da árvore) execute o Procedimento 2.

Passo 2: (encontrar árvores sem arestas coincidentes com T^*) faça:
1. $A_1 \leftarrow A - T^*$. Seja $G_1 : (N, A_1)$ o grafo sem as arestas de T^* e com custos *fuzzy* $\overline{c}_{ij}$ do tipo $(\overline{c}_i, \overline{c}_m, \overline{c}_s)$ associados as arestas.
2. Encontre T_1 através da resolução do grafo *crisp* G_1 com custo $c_{ij} = inf\{supp\{\overline{c_m}\}\}$.
3. Se não houver mais árvores geradoras ou se $inf\{supp\{c_{T_1}\}\} > \delta$, então FIM. Caso contrário, para $k = 1$ até $|N| - 2$ faça:
- Para cada k arestas da árvore T_1 (estas k arestas $\in R'$), execute o procedimento 2.

Procedimento 2: (construir as árvores geradoras e calcular o grau de possibilidade)

Passo 0: (inicializar os parâmetros) sejam k, T,c_{T^*} e R' dados, calcule c_T.

Passo 1: (comparação com a árvore geradora mínima *fuzzy* T^*) para cada k aresta pertencente ao conjunto de corte $(s, t) \in [S, \overline{S}]$ (estas k arestas $\in R''$) que não formam ciclo com $(T^* - R')$, faça:

- Calcule $c'' \leftarrow c_T - \sum_{(ij)\in R'} c_{ij} + \sum_{(i,j)\in R''} c_{ij}$.
- Se $inf\{c''\} > \delta$, então descarte esta árvore. Caso contrário, calcule o grau de possibilidade $w \leftarrow Poss(c'' \leq c_{T^*})$.

Passo 2: caso não existam mais árvores geradoras a serem formadas, volte ao Passo 2 do Procedimento 1.

Arestas	**Custos *Fuzzy***
(1,2)	[30,35,40]
(1,3)	[30,40,45]
(2,3)	[20,25,28]
(2,4)	[08,10,12]
(3,4)	[18,20,23]
(3,5)	[12,15,18]
(4,5)	[27,30,33]

Tabela 3.1: Grafo com parâmetros *fuzzy*

Arestas	**Custos *Fuzzy***
(1,2)	35
(1,3)	40
(2,3)	25
(2,4)	10
(3,4)	20
(3,5)	15
(4,5)	30

Tabela 3.2: Grafo *crisp* com os custos associados aos valores modais

Etapa 2: encontrar a árvore geradora mínima associada ao grafo *crisp* (Tabela 3.2). Utilizando um algoritmo específico qualquer, como Kruskal, por exemplo, a solução encontrada será $T^* = (1,2),(2,4),(3,4),(3,5)$ com custo total igual a 80.

Etapa 3: calcular δ. Sendo o custo total *fuzzy* $c_{T^*} = [68, 80, 93]$ associado à árvore geradora mínima T^*. Então, $\delta = sup\{supp\{c_{T^*}\}\} = 93$. Esse valor representa o limitante superior do custo de T^*, e é utilizado para eliminar árvores com grau de possibilidade igual a zero.

Etapa 4: construir novas árvores retirando k arestas de T^* e calcular a possibilidade.

$k = 1$:

1. sai a aresta (1,2). Candidatas = (1,3).

- Nova árvore $T^1 = (1,3),(2,4),(3,4),(3,5)$. $c_{T^1} = [68, 85, 98]$.

2. sai a aresta (2,4). Candidatas = (1,3),(2,3).

- Nova árvore $T^2 = (1,2),(1,3),(3,4),(3,5)$. $c_{T^2} = [90, 110, 126]$.

- Nova árvore $T^3 = (1,2),(2,3),(3,4),(3,5)$. $c_{T^3} = [80, 95, 109]$.

3. sai a aresta (3,4). Candidatas = (1,3),(2,3),(3,5).

- Nova árvore $T^4 = (1,2),(1,3),(2,4),(3,5)$. $c_{T^4} = [80, 100, 115]$.

- Nova árvore $T^5 = (1,2),(2,3),(2,4),(3,5)$. $c_{T^5} = [70, 85, 98]$.

- Nova árvore $T^6 = (1,2),(2,4),(3,5),(4,5)$. $c_{T^6} = [77, 90, 103]$.

4. sai a aresta (3,5). Candidatas = (4,5).

- Nova árvore $T^7 = (1,2),(2,4),(3,4),(4,5)$. $c_{T^7} = [83, 95, 108]$.

Até aqui, todas as árvores geradas satisfazem o critério de corte $inf\{c''\} > \delta$, onde c'' corresponde ao custo total *fuzzy* de cada nova árvore gerada pela retirada de arestas. Isso significa que existe um ponto de intersecção entre o custo total de T^* (c_{T^*}) e o custo total de cada árvore gerada (c''), e portanto, o grau de possibilidade dessas árvores é diferente de zero.

$k = 2$:

1. saem as arestas (1,2),(2,4). Candidatas = (1,3),(2,3).

- Nova árvore $T^8 = (1,3),(2,3),(3,4),(3,5)$. $c_{T^8} = [80, 95, 109]$.

2. saem as arestas (1,2),(3,4). Candidatas = (1,3),(2,3),(4,5).

- Nova árvore $T^9 = (1,3),(2,3),(2,4),(3,5)$. $c_{T^9} = [70, 90, 103]$.

- Nova árvore $T^{10} = (1,3),(2,4),(3,5),(4,5)$. $c_{T^{10}} = [77, 95, 108]$.

3. saem as arestas (1,2),(3,5). Candidatas = (1,3),(2,3),(4,5).

- Nova árvore $T^{11} = (1,3),(2,4),(3,4),(4,5)$. $c_{T^{11}} = [83, 100, 113]$.

4. saem as arestas (2,4),(3,4). Candidatas = (1,3),(2,3),(4,5).

- Nova árvore $T^{12} = (1,2),(1,3),(3,5),(4,5)$. $c_{T^{12}} = [99, 120, 126]$. $inf\{c''\} > \delta = 99 > 93$, portanto $Poss(T^{12} \leq T^*) = 0$.

- Nova árvore $T^{13} = (1,2),(2,3),(3,5),(4,5)$. $c_{T^{13}} = [89, 105, 109]$.

5. saem as arestas (2,4),(3,5). Candidatas = (1,3),(2,3),(4,5).

- Nova árvore $T^{14} = (1,2),(1,3),(3,4),(4,5)$. $c_{T^{14}} = [105, 125, 126]$. $inf\{c''\} > \delta = 105 > 93$, portanto $Poss(T^{14} \leq T^*) = 0$.

- Nova árvore $T^{15} = (1,2),(2,3),(3,5),(4,5)$. $c_{T^{15}} = [95, 110, 124]$. $inf\{c''\} > \delta = 95 > 93$, portanto $Poss(T^{15} \leq T^*) = 0$.

6. saem as arestas (3,4),(3,5). Candidatas = (1,3),(2,3),(4,5).

- Nova árvore $T^{16} = (1,2),(1,3),(2,4),(4,5)$. $c_{T^{16}} = [100, 115, 130]$. $inf\{c''\} > \delta = 100 > 93$, portanto $Poss(T^{16} \leq T^*) = 0$.

- Nova árvore $T^{17} = (1,2),(2,3),(2,4),(4,5)$. $c_{T^{17}} = [85, 100, 113]$.

$k = 3$

1. saem as arestas (2,4),(3,4),(3,5). Candidatas = (1,3),(2,3),(4,5).

- Não é possível construir uma nova árvore.

2. saem as arestas (1,2),(3,4),(3,5). Candidatas = (1,3),(2,3),(4,5).

- Nova árvore $T^{18} = (1,3),(2,3),(3,4),(4,5)$. $c_{T^{18}} = [85, 105, 118]$.

3. saem as arestas (1,2),(2,4),(3,5). Candidatas = (1,3),(2,3),(4,5).

- Nova árvore $T^{19} = (1,3),(2,3),(3,4),(4,5)$. $c_{T^{19}} = [95, 115, 129]$. $inf\{c''\} > \delta = 95 > 93$, portanto $Poss(T^{19} \leq T^*) = 0$.

4. saem as arestas (1,2),(2,4),(3,4). Candidatas = (1,3),(2,3),(4,5).

- Nova árvore $T^{20} = (1,3),(2,3),(3,5),(4,5)$. $c_{T^{20}} = [89, 110, 124]$.

$k = |N| - 1$, então $G' = G - T^* = (1,3),(2,3),(4,5)$.

Como $|A'| = 3$, $|N| = 5$ e $|A'| < |N| - 1$, então **FIM**.

Caso $|A'| > |N| - 1$, então seria encontrada uma nova T^* baseada no novo grafo G' e todo o processo seria repetido.

O cálculo do grau de possibilidade é realizado logo após a geração de cada árvore geradora. As n melhores árvores são armazenadas e devolvidas ao decisor após o término do programa.

Mesmo com os testes de corte previstos no algoritmo, a enumeração das árvores que farão parte da solução ainda torna o problema de difícil solução. A necessidade de escolher os subconjuntos de arestas a serem retirados e quais grupos podem substituir cada um destes subconjuntos faz com que o problema ainda seja combinatorial.

O espaço de busca de soluções para este tipo de problema é grande por qualquer parâmetro, pois o número de árvores geradoras em um grafo completo é da ordem de m^{m-2}, $m = |N|$ (Raidl & Julstrom 2003). Portanto, se for observado que a solução é composta pelo conjuntos de árvores com maior grau de possibilidade, pode-se propor heurísticas para encontrar uma solução aproximada satisfatória para o problema. Neste contexto, um algoritmo genético (ver Capítulo 4) e um sistema imunológico artificial (ver Capítulo 5) são propostos como uma alternativa para tentar contornar a questão da complexidade.

Capítulo 4

Algoritmo Genético

Neste capítulo, inicialmente, são fornecidas algumas noções básicas sobre os algoritmos genéticos na Seção 4.1. A intenção aqui não é a de produzir um texto completo sobre o assunto, mas apenas disponibilizar informações sobre os principais conceitos utilizados nesta dissertação, de modo a facilitar a compreensão do método proposto. Para um estudo mais aprofundado sobre algoritmos genéticos, consultar Holland (1992), Michalewicz (1996), Bäck et al. (2000a) e Bäck et al. (2000b).

Na Seção 4.2 é proposto um algoritmo genético com características adaptadas à resolução do problema da árvore geradora mínima com parâmetros *fuzzy* (Almeida et al. 2005a, Almeida et al. 2005b) cujo principal objetivo é encontrar um conjunto solução analisando eficientemente o espaço de busca de soluções de maneira a contornar a complexidade do problema.

4.1 Noções Básicas

Os algoritmos genéticos foram desenvolvidos inicialmente por Holland nos anos 60 para estudar formalmente os fenômenos de adaptação, naturais ou artificiais, com o propósito de importar estes mecanismos de adaptação para ambientes computacionais (Holland 1992). Nesta época, os cientistas da computação já estudavam sistemas evolutivos com o objetivo e utilizá-los como uma ferramenta de otimização em problemas de engenharia.

Um algoritmo genético é considerado um método fraco (um método genérico aplicado em um ambiente genérico) que deve ser utilizado apenas quando métodos fortes (métodos genéricos em ambientes específicos) e/ou específicos (métodos específicos em ambientes específicos) falham. Embora não garantam eficiência total na obtenção da solução, geralmente garantem uma boa aproximação. Outra característica é que os algoritmos genéticos operam em proble-

mas não-lineares, não estacionários e problemas com explosão combinatória de candidatos a solução, com uma grande vantagem em relação a outras técnicas (Michalewicz 1996).

As regras de evolução são simples: as espécies evoluem aleatoriamente (via operadores de mutação, cruzamento e outros), estando sujeitas à seleção natural sob recursos limitados. Os indivíduos mais adaptados sobrevivem e se reproduzem, propagando o seu material genético para as próximas gerações.

Os principais componentes de um algoritmo genético são:

Alelos: dois genes que se encontram nas mesmas posições de dois cromossomos e são responsáveis por uma determinada característica.

Codificação: consiste na determinação da estrutura do cromossomo. Os algoritmos genéticos clássicos adotam a codificação binária, isto é, cada gene (*bit*) pode assumir valores 0 ou 1. Outras codificações são a inteira e a real. Respectivamente, nestas codificações valores inteiros ou reais são associados a cada *bit*. No caso da codificação mista, mais de um tipo de codificação devem estar combinados em um cromossomo.

Cromossomo: estrutura nucleoprotéica formada por uma cadeia de DNA, sendo a base física dos genes nucleares, os quais estão dispostos linearmente. Em algoritmos genéticos, um cromossomo haplóide (apenas uma cadeia de DNA representa o cromossomo) geralmente corresponde a uma cadeia de bits que representa um candidato à solução do problema (vide Figura 4.1).

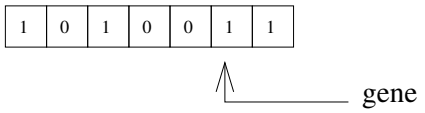

Figura 4.1: Exemplo de cromossomo com codificação binária

Fenótipo: é a manifestação do genótipo no comportamento, fisiologia e morfologia do indivíduo, como um produto de sua interação com o ambiente.

Função de Avaliação (*Fitness*): medida que avalia a capacidade de adaptação e de reprodução de um indivíduo.

Genes: blocos funcionais de DNA, os quais codificam uma proteína específica. No caso do algoritmo genético, um gene corresponde a um único *bit*, ou então a um pequeno bloco de *bits* adjacentes que codificam um elemento particular da solução candidata.

Geração: em algoritmos genéticos, uma iteração que consiste: seleção para reprodução, a reprodução e a seleção para a nova população.

Genoma: conjunto de todos os cromossomos que compõe o material genético do organismo.

Genótipo: representa o conjunto específico de genes do genoma. Neste caso, indivíduos com o mesmo genoma são ditos terem o mesmo genótipo.

Mutação: uma perturbação na informação contida em um determinado gene. Dentre os operadores de mutação temos:

Mutação Aleatória: escolhe-se um indivíduo, sorteia-se a porcentagem de genes que serão trocados e sorteia-se um valor para os genes, dentro do intervalo permitido pelo alelo (Figura 4.2).

Figura 4.2: Exemplo de mutação simples

Mutação Indutiva: semelhante à mutação aleatória, só que o valor a ser sorteado para o alelo é somado ao valor atual do alelo, não substituído (Figura 4.3).

Figura 4.3: Exemplo de mutação indutiva em um cromossomo com codificação real

População: conjunto de indivíduos (cromossomos) que evoluem por meio de operadores genéticos.

Cruzamento (*Crossover*): consiste na troca aleatória de material genético entre dois cromossomos. Existem alguns tipos de cruzamento, como:

***Crossover* Simples (de um ponto):** escolhe-se dois indivíduos da população, determina-se o ponto de corte e efetua-se a troca das duas partes posteriores ao ponto de corte (Figura 4.4).

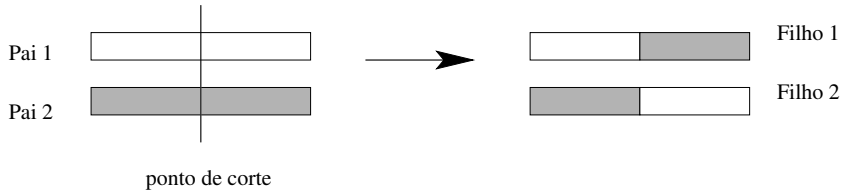

Figura 4.4: Exemplo de *crossover* de um ponto

***Crossover* de dois pontos:** escolhe-se dois indivíduos da população e determina dois pontos de corte para efetuar o *crossover* trocando as partes do cromossomo entre os pontos de corte (Figura 4.5).

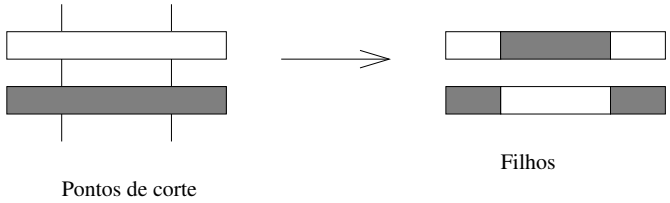

Figura 4.5: Exemplo de *crossover* de dois pontos

***Crossover* Uniforme:** escolhe-se dois indivíduos da população e efetua-se o *crossover* trocando alguns alelos entre os dois indivíduos (Figura 4.6).

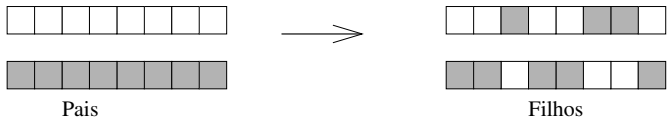

Figura 4.6: Exemplo de *crossover* uniforme

***Crossover OX*:** escolhe-se dois indivíduos da população, copia-se um trecho de um dos indivíduos e o restante do cromossomo do novo indivíduo é preenchido com as informações do outro indivíduo, na mesma sequência evitando valores repetidos nos genes (Figura 4.7).

Reprodução: pode ser sexuada ou assexuada. No caso assexuado, a informação genética é transmitida sem alterações aos novos indivíduos, salvo mutações ocorridas durante a

Figura 4.7: Exemplo de *crossover OX* com codificação inteira

reprodução. Na reprodução sexuada, a informação de dois indivíduos pais são combinadas para a formação de um novo indivíduo.

Seleção: geralmente a seleção é utilizada para escolher os indivíduos para a próxima geração. Os mecanismos mais comuns para a seleção são:

Elitista: são selecionados uma porcentagem dos melhores indivíduos da população intermediária.

Bi-classista: são selecionados os $P\%$ melhores indivíduos e os $(100 - P)\%$ piores indivíduos.

Torneio: dois indivíduos são escolhidos aleatoriamente. Um número aleatório r, entre 0 e 1 é gerado. Se $r < k$ (k um parâmetro dado) o melhor dos indivíduos é escolhido. Senão o outro é escolhido.

Roleta: uma técnica de seleção, em que a probabilidade de um cromossomo ser escolhido é diretamente proporcional ao seu valor de *fitness*.

4.1.1 Esboço de um Algoritmo Genético Clássico

Aqui é apresentado um esquema básico dos algoritmos genéticos mais encontrados na literatura (Figura 4.8). Para cada problema, as funções e as características do algoritmo são adaptadas: como gerar os indivíduos da população inicial e os valores dos parâmetros associados; operadores de mutação e cruzamento; as taxas de mutação e cruzamento, porcentagem de seleção de indivíduos para reprodução e para a nova geração, etc.

Passo 0: criar a população inicial, com K indivíduos, segundo o critério adotado para cada problema. Seja P a população. Calcule o valor do *fitness* de cada indivíduo.

Passo 1: para $I = 1$ até GER faça:

1. Selecione uma porcentagem dos indivíduos de P para cruzamento, proceda a operação e aloque os novos indivíduos na população intermediária (P_i).
2. Selecione uma porcentagem dos indivíduos de P para a mutação, proceda a operação e aloque os indivíduos na P_i.
3. Selecione indivíduos de P e sorteie os restantes dos indivíduos que devem completar a P_i.
4. Faça $P \leftarrow P_i$. Calcule o valor do *fitness* de cada indivíduo de P.

Passo 2: devolva P ao decisor.

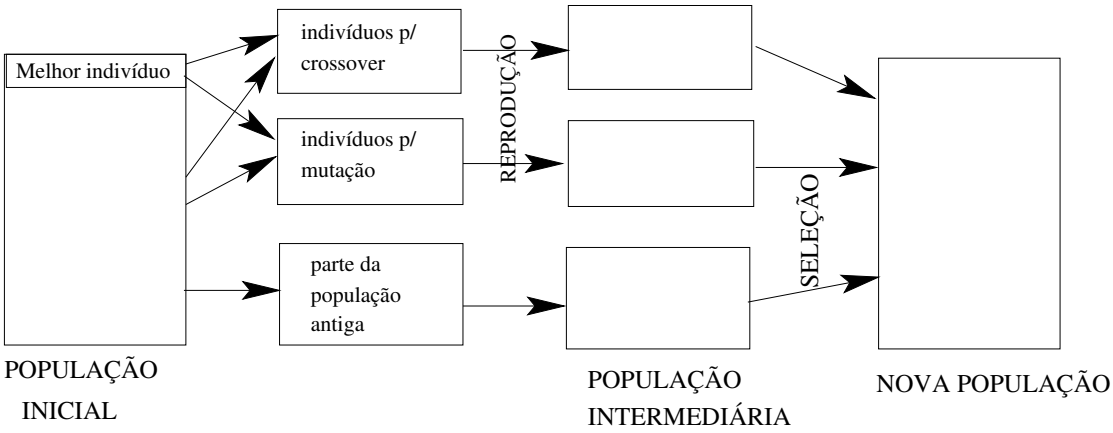

Figura 4.8: Esquema de um algoritmo genético clássico

4.2 Algoritmo Genético para Resolver o Problema da AGM com Parâmetros *Fuzzy*

Seja dado um grafo $G : (N, A)$ com estrutura *crisp* e arestas com custos representados por números *fuzzy* associados, deseja-se encontrar o conjunto (T) composto pelas árvores geradoras que contenham os maiores graus de possibilidade de terem seus custos menores ou iguais ao custo da árvore geradora mínima de maior grau de possibilidade (T^*).

Este problema possui as seguintes características:

- Um conjunto *fuzzy* de soluções em que cada árvore geradora possui um **grau de possibilidade** de ser melhor ou igual a T^*;

- Necessidade de avaliar uma grande quantidade de árvores geradoras para a construção do conjunto *fuzzy* de soluções.

Estas características nos levaram a desenvolver um algoritmo genético nos moldes propostos por Holland (1992) de forma a explorar eficientemente o espaço de busca de soluções, visando encontrar um conjunto solução para o problema e procurando contornar a questão da complexidade.

Numa visão geral, a população é formada por indivíduos (árvores geradoras) que evoluem por meio de operadores de cruzamento e mutação, que geram novos indivíduos que herdam determinadas características dos indivíduos-pai. Procurando um conjunto de soluções, a medida de de adaptação (*fitness*) é utilizada de forma a privilegiar a reprodução dos indivíduos mais adaptados ao ambiente, nesse caso, aqueles que tiverem maiores graus de possibilidade.

4.2.1 Estrutura do Algoritmo Genético

Representação: cada cromossomo representa uma árvore geradora, com seus genes representando as arestas do grafo. Seja um grafo $G : (N, A)$ e $m = |N|$, uma árvore geradora possui $m - 1$ arestas que conectam todos os nós sem formar ciclos. Assim, cada cromossomo possui $m - 1$ genes, como mostrado na Figura 4.9.

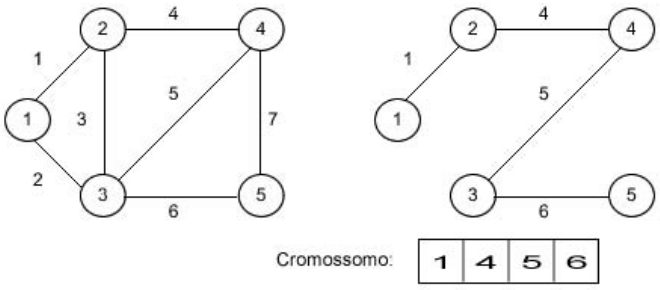

Figura 4.9: Exemplo de um cromossomo representando uma árvore

Inicialização: a inicialização da população é feita por meio de uma busca ao redor da árvore-base (T^*) da seguinte maneira: algumas arestas de T^* são selecionadas e substituídas por outras arestas do grafo para formar uma nova árvore, e assim, esse processo é repetido até que a população inicial seja totalmente preenchida. Esse procedimento é utilizado para aumentar o desempenho do algoritmo, pois segundo Okada (2001) árvores que compartilham um maior número de arestas de T^* tendem a possuírem maiores graus de possibilidade. Desta

forma, geralmente a população inicial é preenchida com árvores que possuem alto grau de possibilidade e a convergência do algoritmo é acelerada.

Cruzamento (*Crossover*): dois pais são unidos, gerando um subgrafo do grafo original. Em seguida, o operador de inicialização é aplicado sobre esse subgrafo para gerar um novo indivíduo (Figura 4.10). Como esse resultado é aleatório, o novo indivíduo tem chance de possuir arestas de um dos pais ou de ambos. Para aproveitar esse caráter aleatório, foi inserida uma busca local que repete a operação de cruzamento n vezes, guardando o melhor indivíduo encontrado. Isso aumenta a *explotação* do espaço de busca.

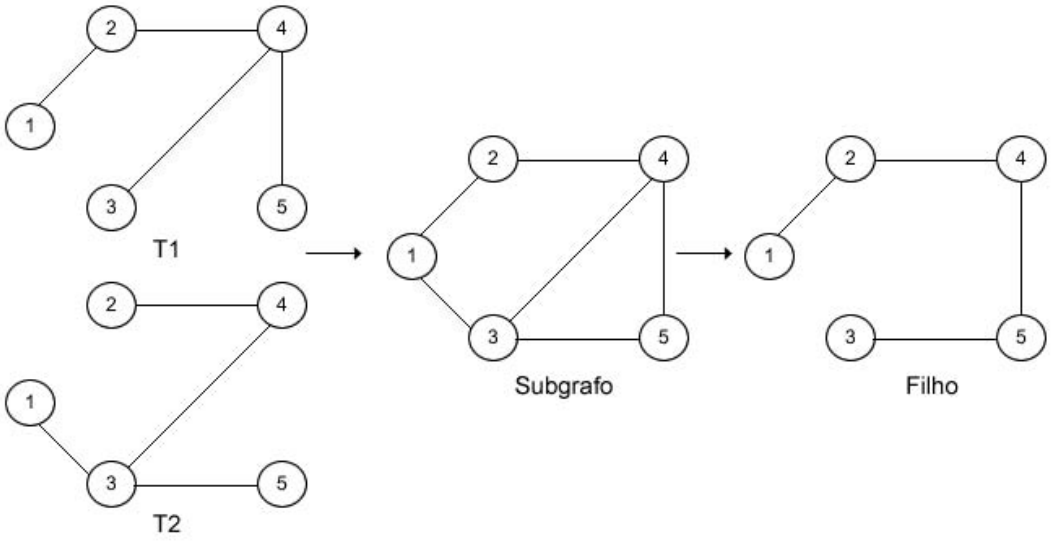

Figura 4.10: Exemplo de cruzamento entre duas árvores T_1 e T_2

Medida de Adaptação (*Fitness*): o valor de *fitness* de um indivíduo i é dado pelo grau de possibilidade da árvore em questão (T^k) ter seu custo menor ou igual ao custo da árvore geradora mínima de maior grau de possibilidade (T^*), conforme descrito na Seção 2.4 no Capítulo 2 e dado pela Equação 4.1.

$$fitness(i) = Poss(\sum_{ij \in T^k} cij \leq \sum_{ij \in T^*} cij) \tag{4.1}$$

Mutação: é feita sorteando-se n arestas do indivíduo e trocando-as por outras arestas que pertençam ao conjunto de corte (Figura 4.11).

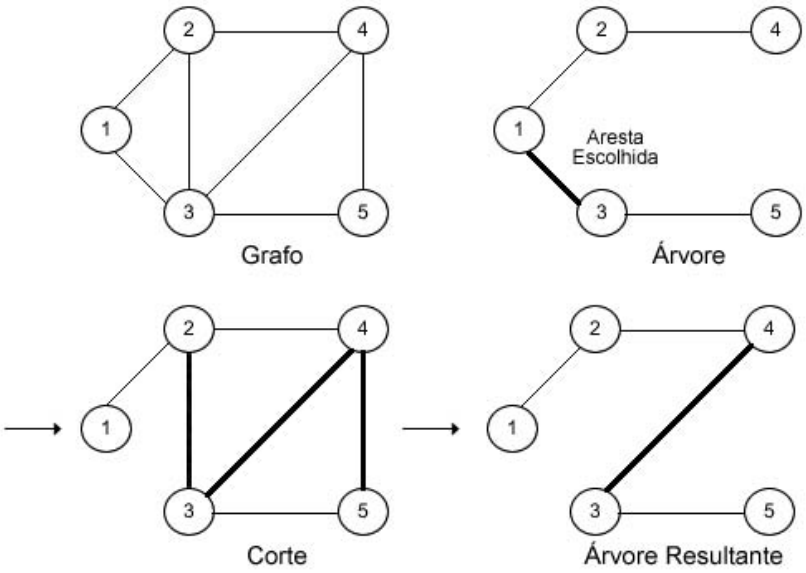

Figura 4.11: Exemplo de mutação em uma árvore

Diversidade Populacional: a chance de obter boas soluções aumenta em uma população de indivíduos mais diversificados. Para analisar o comportamento da diversidade da população, foi definida a seguinte medida: "*Diversidade Populacional* é a razão entre o número de indivíduos únicos (aqueles que não se repetem na população) e o tamanho da população".

Variabilidade Genética: segundo Michalewicz (1996) a variabilidade de uma população cai conforme o *fitness* aumenta. Isso ocorre porque a chance dos indivíduos gerarem descendentes é maior para aqueles que possuem *fitnesses* maiores. Com isso, o algoritmo utiliza os seguintes mecanismos para injetar variabilidade na população:

- *Seleção Elitista +:* é preservada uma certa porcentagem dos melhores indivíduos;
- *Seleção Elitista -:* é preservada uma certa porcentagem dos piores indivíduos;
- *Imigração:* em cada geração, é injetada uma porcentagem de novos indivíduos, gerados pelo mesmo mecanismo utilizado para inicializar a população.

4.2.2 Algoritmo Genético

Algoritmo 4.2.1 Algoritmo Genético para resolver o problema da Árvore Geradora Mínima com Custos *Fuzzy*

Entradas:
- grafo: **G**:(N, A), sendo N o conjunto de nós e A o conjunto de arestas. Com custos c_{ij} representados por números *fuzzy* do tipo (c_i, c_m, c_s) associados a cada aresta do grafo significando o custo para se percorrer do nó i ao nó j;
- número máximo de gerações: **gen**;
- tempo limite para processar o algoritmo: **tp**;
- tamanho da população: **tampop**;
- quantidade de indivíduos que serão armazenados no conjunto solução: **q**;
- taxa de cruzamento: **txc**;
- taxa de mutação: **txm**;
- taxa de melhores indivíduos a preservar: **txmind**;
- taxa de piores indivíduos a preservar: **txpind**.

Saída:
- conjunto solução **S** composto pelas **q** melhores árvores encontradas, seus graus de possibilidade (*fitness*) e seus custos totais *fuzzy*.

Procedimento 1: (inicializar os parâmetros) crie a população **pop** com **tampop** indivíduos. Calcule o valor de *fitness* de cada indivíduo de **pop** e armazene em **S** os **q** melhores indivíduos.

Procedimento 2: (corpo do algoritmo genético)
$i \leftarrow 1$;
enquanto $i \leq$ **gen ou** tempo $\leq$ **tp faça**:
1. selecione aleatóriamente **txc**% de **pop** e aplique cruzamento nesses indivíduos. Aloque os novos indivíduos na população intermediária **Pi**;
2. selecione aleatóriamente **txm**% de **pop** e aplique mutação nesses indivíduos. Aloque os novos indivíduos na população intermediária **Pi**;
3. selecione **txmind**% dos melhores indivíduos de **pop** e **txpind**% dos piores indivíduos de **pop** para preservação. Aloque esses indivíduos na população intermediária **Pi**;
4. **enquanto** o número de indivíduos de **Pi** $<$ **tampop**, complete a população **Pi** com novos indivíduos imigrantes;
5. **pop** $\leftarrow$ **Pi**;
6. calcule o *fitness* de cada indivíduo de **pop**;
7. atualize o conjunto solução **S**;
8. $i \leftarrow i + 1$

Procedimento 3: (finalização)
devolva **S** ao decisor;

Capítulo 5

Sistema Imunológico Artificial

5.1 Introdução

A simulação de processos evolutivos naturais objetivando resolver problemas de explosão combinatória e multimodais demonstrou ser uma estratégia eficaz e robusta. A otimização do comportamento de um sistema obtida através da simulação de processos evolutivos representa uma abordagem poderosa para aprendizagem de máquina e estudo de fenômenos auto-organizados. A implementação computacional destes métodos de simulação da evolução chamada computação evolutiva, possibilita a determinação de soluções para várias classes de problemas (Bäck et al. 2000a). Além dessas, outras linhas de pesquisa ainda mais recentes têm surgido como novos paradigmas de computação.

Os sistemas imunológicos artificiais (SIA), que surgiram a partir de tentativas de modelar e aplicar princípios imunológicos no desenvolvimento de novas ferramentas computacionais, já vêm sendo utilizados em diversas áreas, como reconhecimento de padrões, detecção de falhas e anomalias, segurança computacional, otimização, controle, robótica, *scheduling*, análise de dados, aprendizagem de máquina, dentre outras (Dasgupta 1998).

Nas áreas de engenharia e computação, tem surgido um forte interesse pelo estudo dos sistemas imunológicos devido, principalmente, à sua capacidade de processamento de informação. Sob uma perspectiva de engenharia, existem diversas características do sistema imunológico que podem ser destacadas:

- **Unicidade:** cada animal possui o seu próprio sistema imunológico, com suas capacidades e vulnerabilidades particulares;
- **Reconhecimento de padrões internos e externos ao sistema:** as células e mo-

léculas que não pertencem ao organismo são reconhecidas e eliminadas pelo sistema imunológico;

- **Detecção de anomalia:** o sistema imunológico pode detectar e reagir a agentes patogênicos (causadores de anomalias) a que o organismo nunca havia sido exposto anteriormente;
- **Detecção imperfeita (tolerância a ruídos):** um reconhecimento perfeito não é necessário para que o sistema imunológico reaja contra um elemento causador de patologia (patógeno);
- **Diversidade:** existe uma quantidade limitada de células e moléculas no sistema imunológico que são utilizadas para se obter o reconhecimento de um número praticamente infinito de elementos, incluindo aqueles sintetizados em laboratório;
- **Aprendizagem por reforço:** a cada encontro com o mesmo patógeno, o sistema imunológico melhora a qualidade de sua resposta;
- **Memória:** os componentes do sistema imunológico bem sucedidos no reconhecimento e combate às patologias são armazenados para uma resposta futura mais intensa e efetiva.

5.2 Definição

Segundo Dasgupta (1998), os sistemas imunológicos artificiais são mecanismos computacionais compostos por metodologias inteligentes, inspiradas no sistema imunológico biológico, para a solução de problemas do mundo real.

5.3 Princípios Fundamentais dos Sistemas Imunológicos

Para facilitar a compreensão da estrutura e do funcionamento do sistema imunológico artificial serão apresentados, resumidamente, como age o sistema imunológico nos animais vertebrados e os princípios da seleção clonal. Informações referentes aos elementos que constituem o sistema, bem como definições sobre os termos biológicos utilizados podem ser encontrados nas referências: Castro (2001), Dreher (1995), Janeway et al. (2000) e Timmis (2000).

5.3.1 Composição e Funcionamento do Sistema Imunológico

Quando um sistema (corpo) detecta a presença de um antígeno (patógeno), ocorrem duas formas de respostas imunológicas dadas pelos seguintes sistemas (Figura 5.1):

- **Sistema Imune Inato:** é a primeira linha de defesa do organismo. Ele é formado por células fagocitárias, como macrófagos e neutrófilos, além de fatores solúveis como o complemento e algumas enzimas. As células do sistema imune inato desempenham um papel crucial na iniciação e posterior direcionamento das respostas imunes adaptativas, principalmente devido ao fato de que as respostas adaptativas demoram um período de tempo (da ordem de dias) para exercer seus efeitos. Portanto, a resposta imune inata apresenta um papel muito importante no controle das infecções durante esse tempo;

- **Sistema Imune Adaptativo:** os linfócitos (produzidos na medula óssea e no timo) são as principais células que compõe esse sistema, presentes apenas nos animais vertebrados, eles desempenham um papel crucial no desencadeamento e posterior regulação das respostas imunes inatas. Cada "linfócito virgem" que penetra na corrente circulatória é portador de receptores de um antígeno com uma única especificidade. Os linfócitos sofrem, então, um processo parecido com a seleção natural durante a vida do indivíduo: somente aqueles que encontram um antígeno com o qual seu receptor pode interagir serão ativados para proliferar e se diferenciar em outros tipos de células. Após a ligação do anticorpo de superfície ao antígeno, a célula é ativada para proliferar e produzir uma numerosa prole, conhecida como clone. Essas células secretam anticorpos com uma especificidade idêntica à do receptor de superfície. Este princípio recebeu o nome de *teoria da seleção clonal*, e constitui a parte central da imunidade adaptativa.

5.4 Princípio da Seleção Clonal

Uma vez que cada célula apresenta um padrão (forma) distinto de receptor antigênico, o número de linfócitos que pode se ligar a um determinado antígeno é restrito. A fim de produzir células efetoras específicas em quantidade suficiente para combater uma infecção, um linfócito ativado deve se proliferar antes que sua prole se diferencie em células efetoras.

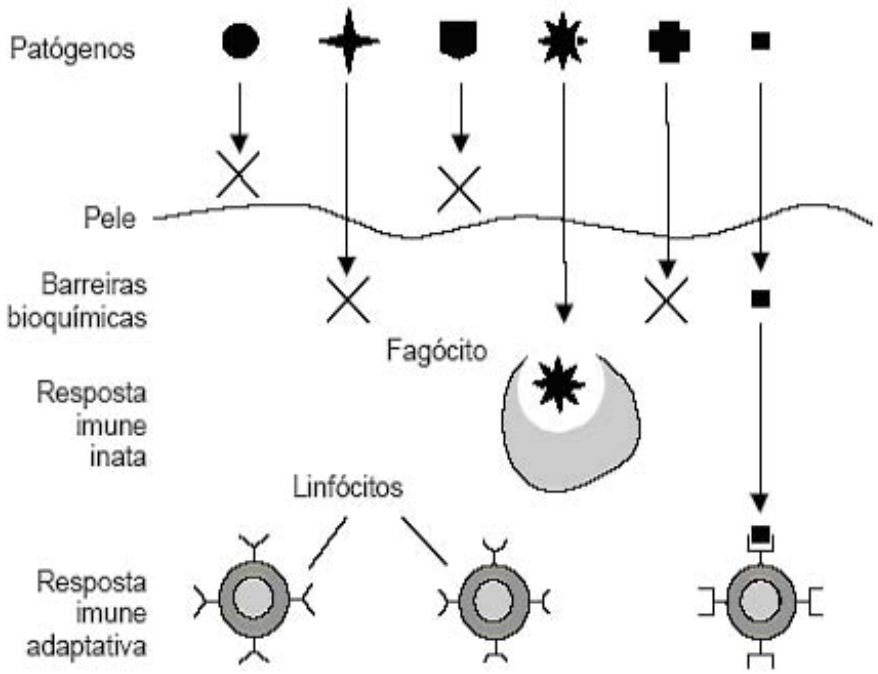

Figura 5.1: Estrutura multicamadas do sistema imunológico

O princípio de seleção clonal estabelece que apenas aquela célula capaz de reconhecer um determinado estímulo antigênico irá se proliferar, sendo, portanto, selecionada em detrimento de outras.

Quando um animal é exposto a um antígeno, uma subpopulação de linfócitos responde através da produção de anticorpos. Cada célula secreta um único tipo de anticorpo, que é relativamente específico para o antígeno. Através da ligação do antígeno com o receptor do linfócito e, dado um segundo sinal (ou sinal co-estimulatório) de células acessórias como a célula T_H, um antígeno estimula o linfócito a se proliferar (dividir) e transformar-se em uma célula terminal capaz de secretar anticorpos em altas taxas. Estas células são chamadas de plasmócitos.

Os linfócitos, além de se proliferar e diferenciar em plasmócitos, também podem se diferenciar em linfócitos de memória, caracterizadas por longos períodos de vida. Esses linfócitos de memória provavelmente não produzem anticorpos, mas quando re-expostas ao mesmo estímulo antigênico começam a se diferenciar em plasmócitos capazes de produzir anticorpos pré-selecionados pelo antígeno específico que estimulou a resposta primária.

5.5 Implementação de um Sistema Imunológico Artificial

Assim como no algoritmo genético (ver Seção 4.2 no Capítulo 4), de um grafo com estrutura *crisp* e parâmetros associados a custos representados por números *fuzzy*, deseja-se encontrar um conjuntos de árvores geradoras (T) que tenham alto grau de possibilidade de terem seus custos menores ou iguais ao custo da árvore geradora mínima (T^*) associada ao grafo com parâmetros *crisp* (G_c).

Devido às características inerentes a este tipo de problema,desenvolvemos um sistema imunológico artificial nos moldes propostos por Dasgupta (1998) e Castro (2001) para tentar encontrar um conjunto-solução tentando contornar a complexidade do problema. A população é formada por anticorpos (árvores geradoras) que geram novos indivíduos herdeiros das características dos anticorpos-pai (clones) que evoluem por meio de operadores de maturação. Procurando um subconjunto *fuzzy* de soluções, a medida de afinidade (*fitness*) é utilizada de forma a privilegiar a reprodução dos indivíduos mais adaptados ao ambiente (princípio de seleção clonal - Seção 5.4), ou seja, somente aqueles anticorpos que encontrarem um antígeno com o qual seus receptores poderão interagir serão ativados para proliferarem e diferenciarem em células efetoras. Neste caso, a medida de afinidade de cada indivíduo é dada pelo grau de possibilidade da árvore geradora (T) ter custo menor ou igual ao da árvore geradora mínima *fuzzy* (T^*).

5.5.1 A Estrutura do Sistema Imunológico Artificial

Representação da população de anticorpos (Ab): como descrito no modelo de representação utilizado no algoritmo genético (ver Seção 4.2 no Capítulo 4), de um grafo $G : (N, A)$, sendo $m = |N|$, uma árvore geradora possui $m - 1$ arestas que conectam todos os nós sem formar ciclos (Ahuja et al. 1993, Bazaraa et al. 1990). Assim, cada anticorpo é representado por um vetor com $m - 1$ posições sendo que cada uma delas representa uma aresta do grafo. Desta forma, cada anticorpo representa uma árvore geradora, como mostrado na Figura 5.2.

1. Inicialização: semelhante ao procedimento utilizado pelo algoritmo genético, a inicialização da população (**Ab**) é feita por meio de uma busca ao redor da árvore-base (T^*) da seguinte maneira: algumas arestas de T^* são selecionadas e substituídas por outras arestas

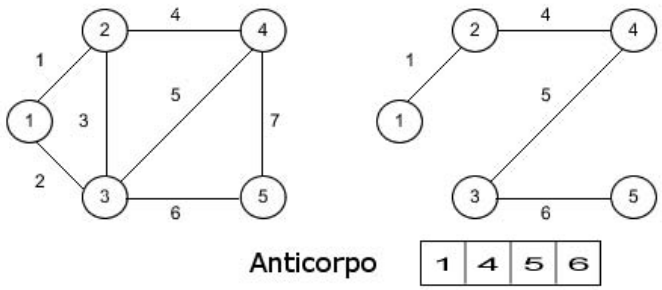

Figura 5.2: Exemplo de um anticorpo representando uma árvore

do grafo para formar uma nova árvore, e assim, esse processo é repetido até que a população inicial seja totalmente preenchida. Esse procedimento é utilizado para aumentar o desempenho do algoritmo, pois segundo Okada (2001) árvores que compartilham um maior número de arestas de T^* tendem a possuir maiores graus de possibilidade. Desta forma, geralmente a população inicial é preenchida com árvores que possuem alto grau de possibilidade e a convergência do algoritmo é acelerada.

2. Clonagem: a clonagem da população é feita gerando-se n clones idênticos de cada indivíduo da população. Esses anticorpos clonados irão formar a população **C** de clones.

3. Maturação: a população **C** de clones sofre um processo de maturação de afinidade ("*mutação*") gerando uma nova população **C*** de clones maturados. Esse processo ocorre sorteando-se um número aleatório de arestas que compõe o indivíduo e trocando-as por outras que pertençam ao conjunto de corte, como na Figura 5.3.

4. Medida de Afinidade (*Fitness*): assim como o operador de mutação utilizado pelo algoritmo genético, o grau de afinidade de um indivíduo (i) é dado pelo grau de possibilidade da árvore (anticorpo) em questão (T^k) ter seu custo total menor ou igual ao custo da árvore geradora mínima de maior grau de possibilidade (T^* - anticorpo com maior grau de afinidade), conforme descrito na Seção 2.4 no Capítulo 2 e dado pela Equação 5.1.

$$afinidade(i) = Poss(\sum_{ij \in T^k} cij \leq \sum_{ij \in T^*} cij) \tag{5.1}$$

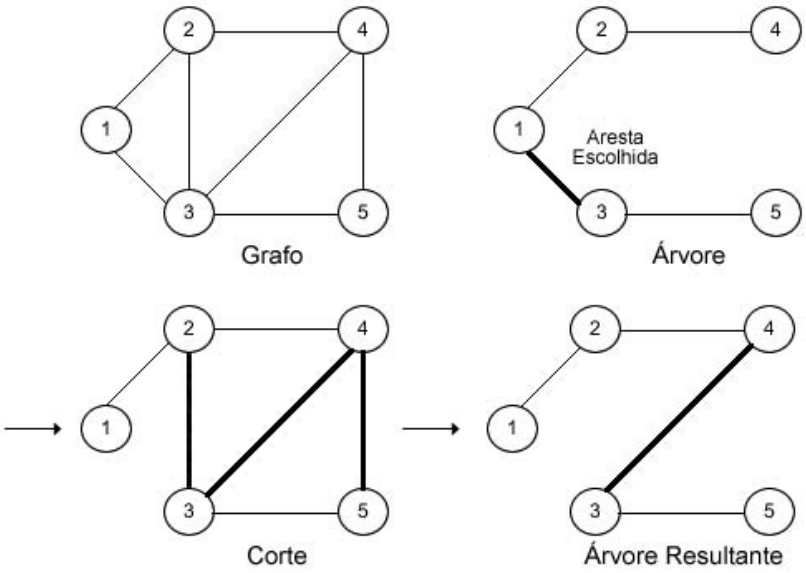

Figura 5.3: Exemplo de maturação de um anticorpo

5. Seleção: o processo de seleção dos clones foi inspirado nos princípios da seleção clonal (ver Seção 5.4). Portanto, o método utilizado é o elitista (Michalewicz 1996), pois é necessário privilegiar o clone com maior grau de afinidade. Assim, o clone com maior *fitness* é selecionado entre os indivíduos da população de clones e comparado com o seu anticorpo-pai e apenas o melhor entre os dois é preservado na população.

6. Diversidade Populacional: assim como descrito no algoritmo genético, a chance de obter boas soluções aumenta em uma população de indivíduos mais diversificados. Para analisar o comportamento da diversidade da população, foi definida a seguinte medida: "*Diversidade Populacional* é a razão entre o número de indivíduos únicos (aqueles que não se repetem na população) e o tamanho da população".

Uma possível injeção de diversidade pode ocorrer em determinados momentos, tais como:

- Quando a variação da média de *fitness* for menor que uma determinada taxa;
- Quando a diversidade populacional for menor que um determinado limiar.

Quando um desses fatores ocorrer significa que a população está se tornando homogênea prejudicando o desempenho de algoritmo. Neste caso, há a necessidade de se fazer uma injeção de diversidade. Isso pode ser feito de diversas maneiras, algumas delas são:

- Cruzando elementos da população para gerar novos anticorpos;

- Aplicar mutação em indivíduos semelhantes;
- Retirar uma parcela de anticorpos da população e preenchê-la com novos indivíduos.

5.5.2 Sistema Imunológico Artificial

Para facilitar a compreensão do Algoritmo 5.5.1, considere o diagrama de blocos da Figura 5.4.

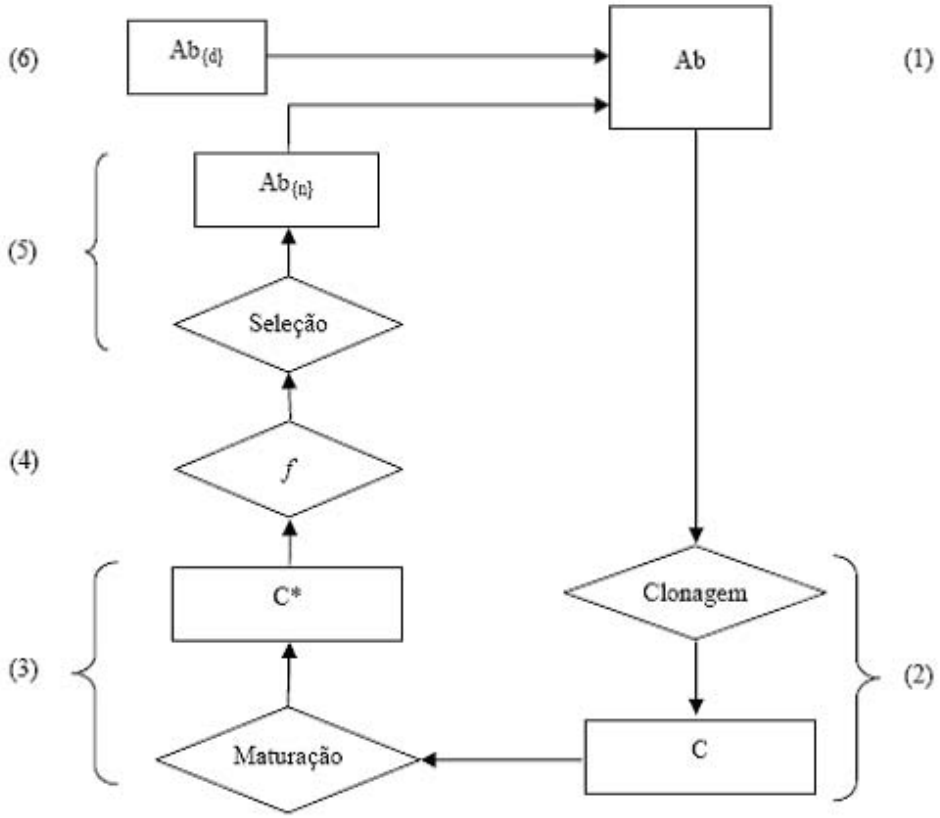

Figura 5.4: Diagrama de blocos do sistema imunológico artificial

Algoritmo 5.5.1 Sistema Imunológico Artificial para resolver o problema da Árvore Geradora Mínima com Custos *Fuzzy*

Entradas:
- grafo: **G**:(N, A), sendo N o conjunto de nós e A o conjunto de arestas. Com custos c_{ij} representados por números *fuzzy* do tipo (c_i, c_m, c_s) associados a cada aresta do grafo significando o custo para se percorrer do nó i ao nó j;
- número máximo de gerações: **gen**;
- tempo limite para processar o algoritmo: **tp**;
- tamanho da população: **tampop**;
- limiar para injeção de diversidade populacional: **l**;
- grau de afinidade mínimo entre o anticorpo e o antígeno necessário para gerar descendentes: **gam**;
- quantidade de indivíduos que serão armazenados no conjunto solução: **q**;
- número de clones a serem gerados por cada anticorpo-pai em cada geração: **nc**;

Saída:
- conjunto solução **S** composto pelos **q** melhores anticorpos encontrados, seus graus de afinidade (*fitness*) e seus custos totais *fuzzy*.

Procedimento 1: (inicializar os parâmetros) criar a população **Ab** com **tampop** anticorpos. Calcular o grau de afinidade (*fitness*) de cada um deles e armazenar em **S** os **q** melhores.

Procedimento 2: (corpo do Sistema Imunológico Artificial)
$i \leftarrow 1$;
enquanto $i \leq$ **gen ou** tempo $\leq$ **tp faça**:
1. selecionar os anticorpos da população **Ab** que tenham grau de afinidade maior ou igual a **gam** e aloque-os na população intermediária **Ci**;
2. clonar todos os indivíduos de **Ci**, no qual cada anticorpo deve gerar **nc** clones idênticos. Aloque esses clones na população de clones **C**;
3. maturar todos os clones de **C** gerando uma população de clones maturados **C***;
4. calcular o grau de afinidade (*fitness*) dos clones maturados de **C***;
5. selecionar os clones maturados de **C*** que possuam grau de afinidade maior que os de seus anticorpos-pai. Aloque esses clones selecionados na população $\mathbf{Ab}_{\{n\}}$;
6. substituir os anticorpos-pai da população **Ab** pelos seus respectivos clones maturados de $\mathbf{Ab}_{\{n\}}$. Calcular a diversidade da nova população. **Se** diversidade $<$ **l então** aloque os anticorpos idênticos de **Ab** em $\mathbf{Ab}_{\{d\}}$, aplique maturação nesses indivíduos e devolva-os em **Ab**;
7. atualizar o conjunto solução **S**;
8. $i \leftarrow i + 1$

Procedimento 3: (finalização)
devolver **S** ao decisor;

Capítulo 6

Experimentos e Resultados

Neste Capítulo, alguns grafos foram utilizados para avaliar o desempenho do algoritmo genético (AG) e do sistema imunológico artificial (SIA).

Para cada grafo foram realizadas 10 rodadas, sendo que, em cada uma delas obtivemos como resultado um conjunto solução composto pelas n melhores árvores encontradas em ordem crescente de grau de possibilidade (*fitness*). O conjunto solução final é formado pelos valores máximos de cada linha dos conjuntos soluções encontrados em cada rodada, ou seja, avalia-se a primeira linha dos 10 conjuntos soluções e a árvore com maior *fitness* desta linha é armazenada na primeira linha do conjunto solução final e assim, sucessivamente, até a última linha dos conjuntos soluções. Fazendo desta maneira, no final do processo obteremos um conjunto solução composto pelos melhores indivíduos de cada posição, por exemplo, o indivíduo (i) (linha i) da solução final será o melhor indivíduo da i-ésima linha dentre todas as soluções obtidas em cada rodada.

Devido ao fato do sistema imunológico artificial trabalhar com múltiplas populações (clones), ele consome mais tempo de processamento que o algoritmo genético para realizar cada experimento, mediante a igualdade dos parâmetros de número de gerações e tamanho da população. Assim sendo, cada experimento realizado pelo algoritmo genético foi testado utilizando o mesmo tempo de processamento consumido pelo sistema imunológico artificial.

6.1 Parâmetros

Os seguintes parâmetros foram calibrados e utilizados para todos os experimentos:

6.1.1 Parâmetros do Sistema Imunológico Artificial

- Número máximo de gerações (critério de parada): 50;
- Tamanho da população de anticorpos: 50;
- Grau de afinidade mínimo para geração de clones: 50%;
- Número de clones gerados por cada anticorpo: 10;
- Taxa mínima (limiar) de diversidade populacional: 50%;
- Número de melhores anticorpos armazenados no conjunto solução: 10.

6.1.2 Parâmetros do Algoritmo Genético

- Tempo utilizado pelo SIA (critério de parada)
- Tamanho da população de cromossomos: 50;
- Taxa de cruzamento: 50%;
- Taxa de mutação: 3%;
- Preservação Bi-classista: o melhor e o pior indivíduos;
- Número de melhores cromossomos armazenados no conjunto solução: 10.

6.2 Plataforma

Todos os algoritmos foram implementados em MatLab 7, e todos os testes foram realizados em um microcomputador com processador modelo Athlon 2200 com 2 Ghz e 256 Mb de memória.

6.3 Resultados

Nesta Seção são apresentados os resultados obtidos em cada teste realizado para cada grafo, tanto para o AG (ver Seção 4.2 no Capítulo 4) quanto para o SIA (ver Seção 5.5 no Capítulo 5).

Com o intuito de obter uma boa avaliação, diversos tipos diferentes de grafos foram utilizados para os testes, tais como: grafos esparsos, densos, completos, alguns com uma única árvore geradora mínima, outros com muitas árvores geradoras mínima.

A diversidade da população também foi avaliada em cada geração tanto para o algoritmo genético quanto para o sistema imunológico artificial. Por meio dos gráficos gerados para cada instância é possível avaliar o desempenho da manutenção da diversidade para ambos os métodos.

A maioria das instâncias foi obtida de duas bases de dados disponíveis na internet:

1. **TSPLIB:** trata-se de um repositório público de instâncias para o problema do caixeiro viajante na internet onde, além das instâncias, consta também o ótimo global para algumas delas. Ele está disponível na internet em: http://www.iwr.uni-heidelberg.de/groups/comopt/soft/TSPLIB95/TSPLIB.html.

2. **OR-LIBRARY:** é parecido com o TSPLIB, entretanto, neste caso, trata-se de um repositório público de instâncias para uma grande quantidade de problemas de otimização, além do problema do caixeiro viajante (Beasley 1990).

6.3.1 Grafo Itália

O grafo Itália (Figura 6.1) foi encontrado em Ali et al. (2000) e representa uma rede óptica de telecomunicações interligando diversas cidades italianas. As distâncias entre elas estão em km e são representadas por números triangulares *fuzzy* do tipo (c_i, c_m, c_s), sendo que, c_m corresponde ao valor modal da distância, c_i e c_s correspondem ao limitante inferior - desvio de 10% para menos - e limitante superior - desvio de 10% para mais, respectivamente.

Este grafo é composto por 21 nós e 35 arestas. Ele possui cerca de $2,2 \times 10^{11}$ árvores geradoras possíveis e, para um grafo completo com o mesmo número de nós teríamos aproximadamente $1,3 \times 10^{25}$ possíveis soluções (Almeida et al. 2005a).

Os resultados obtidos pelas heurísticas estão exibidos na Tabela 6.1.

Para avaliar o desempenho dos algoritmos foi necessário implementar e analisar os resultados (Tabela 6.2) obtidos pelo método exato (ver Capítulo 3).

O comportamento da diversidade populacional e a convergência do algoritmo tanto para o AG quanto para o SIA, estão exibidos nos gráficos das Figuras 6.2 e 6.3, respectivamente.

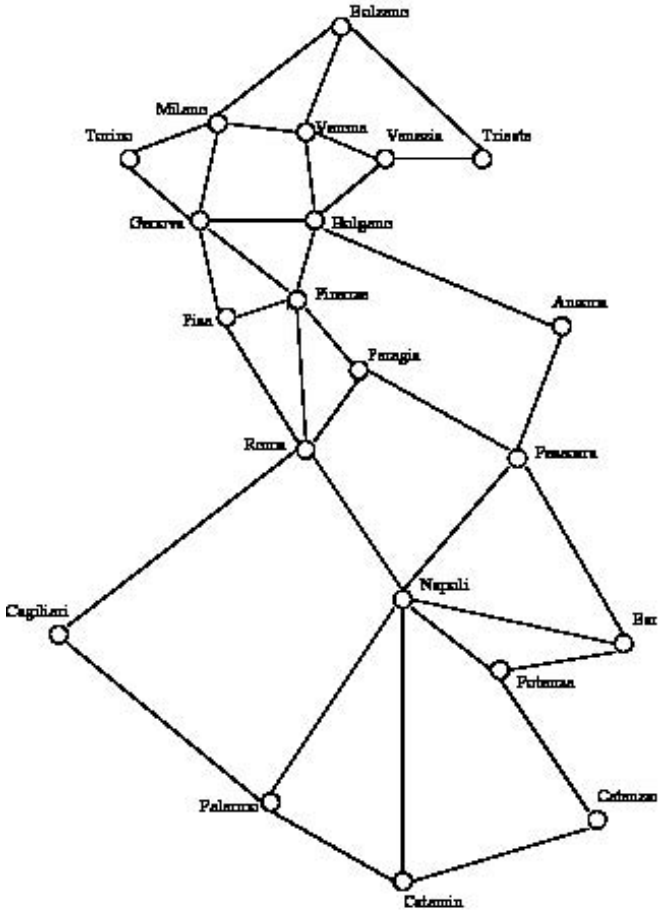

Figura 6.1: Grafo Itália - 21 nós e 35 arestas

Num. Sol.	**AG *Fitness***	**AG Custo *Fuzzy***	**SIA *Fitness***	**SIA Custo *Fuzzy***
1	0,9814	[2406,00 2675,00 2944,00]	0,9814	[2406,00 2675,00 2933,00]
2	0,9814	[2405,00 2675,00 2944,00]	0,9814	[2406,00 2675,00 2944,00]
3	0,9814	[2405,00 2675,00 2945,00]	0,9814	[2405,00 2675,00 2944,00]
4	0,9907	[2401,00 2670,00 2939,00]	0,9907	[2401,00 2670,00 2939,00]
5	0,9907	[2401,00 2670,00 2939,00]	0,9907	[2401,00 2670,00 2939,00]
6	0,9907	[2401,00 2670,00 2939,00]	0,9907	[2401,00 2670,00 2939,00]
7	0,9907	[2401,00 2670,00 2939,00]	0,9907	[2401,00 2670,00 2939,00]
8	0,9907	[2401,00 2670,00 2939,00]	0,9907	[2401,00 2670,00 2939,00]
9	1,0000	[2397,00 2665,00 2933,00]	1,0000	[2397,00 2665,00 2933,00]
10	1,0000	[2397,00 2665,00 2933,00]	1,0000	[2397,00 2665,00 2933,00]

Tabela 6.1: Resultados obtidos pelo AG e pelo SIA para o grafo Itália

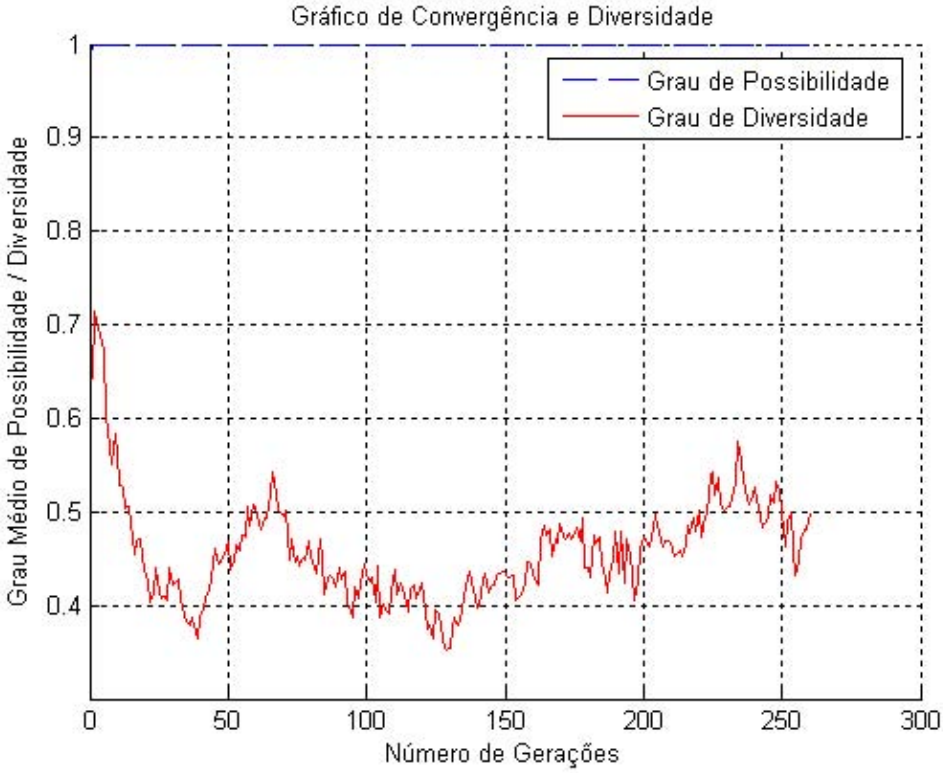

Figura 6.2: Gráfico de diversidade e convergência obtido pelo AG para o grafo Itália

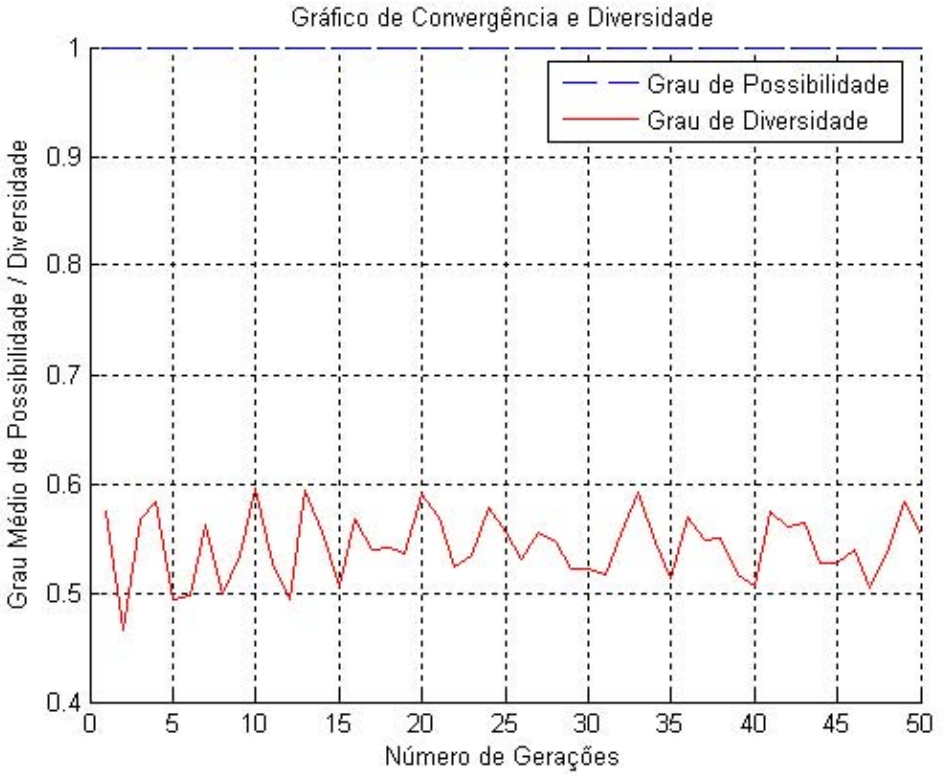

Figura 6.3: Gráfico de diversidade e convergência obtido pelo SIA para o grafo Itália

Quanto a diversidade da população, apesar de ambas heurísticas conseguirem obter resultados parecidos, o SIA (Figura 6.3) obteve um melhor rendimento, conseguindo manter, em média, 55% de indivíduos únicos em cada geração, enquanto que o AG (Figura 6.2) manteve 45%, em média.

Num.	***Fitness***	**Custo *Fuzzy***
1	0,9814	[2406,00 2675,00 2944,00]
2	0,9814	[2405,00 2675,00 2944,00]
3	0,9814	[2405,00 2675,00 2945,00]
4	0,9907	[2401,00 2670,00 2939,00]
5	0,9907	[2401,00 2670,00 2939,00]
6	0,9907	[2401,00 2670,00 2939,00]
7	0,9907	[2401,00 2670,00 2939,00]
8	0,9907	[2401,00 2670,00 2939,00]
9	1,0000	[2397,00 2665,00 2933,00]
10	1,0000	[2397,00 2665,00 2933,00]

Tabela 6.2: Resultados obtidos pelo método exato para o grafo Itália

Enquanto que o método exato consumiu exatamente 45 horas, 14 minutos e 12 segundos (quase dois dias completos, ou ainda, exatos 162.852 segundos) para concluir o experimento, o AG e o SIA consumiram apenas 21 segundos, em média, para realizar cada rodada de cada teste. Neste tempo, enquanto o SIA executou 50 gerações, o AG executou 260.

Apesar das heurísticas terem consumido bem menos tempo (na ordem de, aproximadamente, $1,3 \times 10^{-4}$) que o método exato e terem avaliadas poucas soluções-candidatas (limitante superior de 25.000 candidatos para o SIA e 13.000 para o AG) dentro do enorme espaço de busca composto por aproximadamente $2,2 \times 10^{11}$ soluções, a qualidade dos resultados obtidos por elas é tão boa quanto ao conjunto solução obtido pelo método exato. Isso demonstra que os algoritmos propostos são adequados para resolver este tipo de problema.

Uma vez comprovada a eficiência das heurísticas propostas, optamos por não utilizar o método exato para encontrar as soluções para os experimentos seguintes. Isto porque os grafos utilizados nos próximos experimentos são bem mais complexos que o grafo Itália, sendo intratáveis pelo método exato, levando-se em conta que este levaria semanas, meses ou anos para encontrar os conjuntos soluções de cada instância.

6.3.2 Grafo Bays

Este grafo foi extraído da base de dados TSPLIB e representa todas as possíveis conexões entre 29 cidades do estado da Bavária - Alemanha, com as respectivas distâncias geográficas entre elas. Assim como no grafo Itália, os custos são representados por números triangulares *fuzzy* do tipo (c_i, c_m, c_s), no qual, c_m corresponde ao valor modal da distância de uma cidade a outra e desvio de 10% para os limitantes inferiores (c_i) e superiores (c_s).

Este é um grafo completo, formado por 29 nós e 406 arestas. Ele possui em torno de $3,05 \times 10^{39}$ candidatos a solução e a sua principal característica é conter diversas árvores com grau de possibilidade igual a 1. Este fato influencia diretamente na diversidade da população, aumentando a possibilidade dela se tornar homogênea com passar das gerações, pois os indivíduos mais adaptados ao ambiente tendem a sobreviver em todas as gerações, além de possuírem chances maiores de gerar descendentes.

Os resultados obtidos estão exibidos na Tabela 6.3.

Num. Sol.	**AG *Fitness***	**AG Custo *Fuzzy***	**SIA *Fitness***	**SIA Custo *Fuzzy***
1	0,9936	[1403,10 1559,00 1714,90]	0,9712	[1409,40 1566,00 1722,60]
2	0,9968	[1402,20 1558,00 1713,80]	0,9872	[1404,90 1561,00 1717,10]
3	0,9968	[1402,20 1558,00 1713,80]	1,0000	[1401,30 1557,00 1712,70]
4	0,9968	[1402,20 1558,00 1713,80]	1,0000	[1401,30 1557,00 1712,70]
5	1,0000	[1401,30 1557,00 1712,70]	1,0000	[1401,30 1557,00 1712,70]
6	1,0000	[1401,30 1557,00 1712,70]	1,0000	[1401,30 1557,00 1712,70]
7	1,0000	[1401,30 1557,00 1712,70]	1,0000	[1401,30 1557,00 1712,70]
8	1,0000	[1401,30 1557,00 1712,70]	1,0000	[1401,30 1557,00 1712,70]
9	1,0000	[1401,30 1557,00 1712,70]	1,0000	[1401,30 1557,00 1712,70]
10	1,0000	[1401,30 1557,00 1712,70]	1,0000	[1401,30 1557,00 1712,70]

Tabela 6.3: Resultados obtidos pelo AG e pelo SIA para o grafo Bays

Ambos os algoritmos conseguiram encontrar boas soluções (árvores), pois todas elas possuem alto grau de possibilidade de serem melhores que T^*. Entretanto, por uma pequena diferença, o SIA obteve um desempenho melhor. Enquanto este conseguiu encontrar 8 árvores com grau máximo de possibilidade, o AG encontrou 6. Esta diferença pode ser explicada pelo fato do SIA trabalhar com múltiplas populações (clones), o que faz com que ele consiga manter a diversidade da população (Figura 6.5) de maneira mais eficaz que o AG (Figura 6.4), permitindo-lhe obter um melhor desempenho em problemas multimodais.

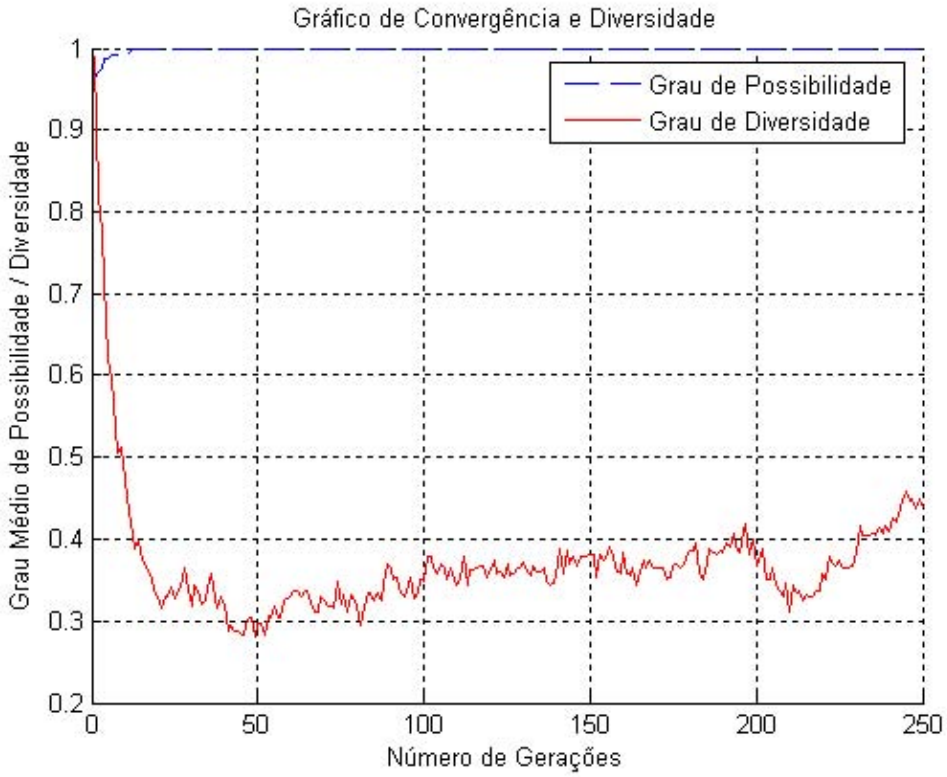

Figura 6.4: Gráfico de diversidade e convergência obtido pelo AG para o grafo Bays

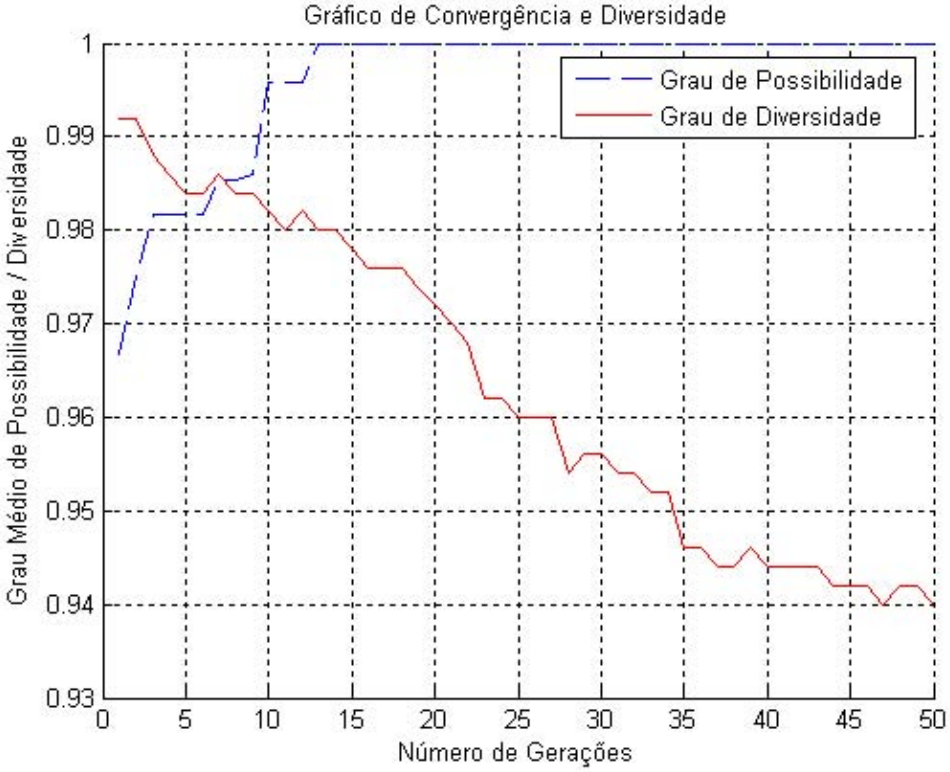

Figura 6.5: Gráfico de diversidade e convergência obtido pelo SIA para o grafo Bays

Em relação à diversidade populacional, o SIA (Figura 6.5) conseguiu manter a população mais heterogênea que o AG (Figura 6.4) durante as gerações, pois enquanto este conseguiu manter, em média, 38% de indivíduos únicos na população em cada geração, o SIA conseguiu um número significativo de 96%.

O tempo consumido em cada rodada, tanto para o AG quanto para o SIA, foi de 21 segundos, em média. Neste tempo, enquanto o SIA executou 50 gerações, o AG executou 250. O número de indivíduos avaliados pelos algoritmos tem como limitante superior 25.000 para SIA e 12.500 para o AG.

6.3.3 Grafo Swiss

Este grafo também foi encontrado na base de dados TSPLIB e representa todas as possibilidades de interligar 42 cidades suíças. As distâncias entre elas são representadas por números triangulares *fuzzy* do tipo (c_i, c_m, c_s), no qual, c_m corresponde ao valor modal da distância de uma cidade a outra e desvio de 10% para os limitantes inferiores (c_i) e superiores (c_s).

Trata-se de um grafo completo, constituído por 42 nós e 861 arestas. Ele possui aproximadamente $8,51 \times 10^{64}$ soluções possíveis.

Os resultados obtidos estão exibidos na Tabela 6.4.

Num. Sol.	**AG *Fitness***	**AG Custo *Fuzzy***	**SIA *Fitness***	**SIA Custo *Fuzzy***
1	0,9815	[974,70 1083,00 1191,30]	0,9539	[980,10 1089,00 1197,90]
2	0,9815	[974,70 1083,00 1191,30]	0,9631	[978,30 1087,00 1195,70]
3	0,9815	[974,70 1083,00 1191,30]	0,9631	[978,30 1087,00 1195,70]
4	0,9861	[973,80 1082,00 1190,20]	0,9677	[977,40 1086,00 1194,60]
5	0,9861	[973,80 1082,00 1190,20]	0,9723	[976,50 1085,00 1193,50]
6	0,9907	[972,90 1081,00 1189,10]	0,9769	[975,60 1084,00 1192,40]
7	0,9954	[972,00 1080,00 1188,00]	0,9907	[972,90 1081,00 1189,10]
8	1,0000	[971,10 1079,00 1186,90]	1,0000	[971,10 1079,00 1186,90]
9	1,0000	[971,10 1079,00 1186,90]	1,0000	[971,10 1079,00 1186,90]
10	1,0000	[971,10 1079,00 1186,90]	1,0000	[971,10 1079,00 1186,90]

Tabela 6.4: Resultados obtidos pelo AG e pelo SIA para o grafo Swiss

Neste caso, apesar de ambos os métodos encontrarem 3 árvores geradoras com grau máximo de possibilidade de serem melhores ou iguais a T^*, o AG conseguiu um melhor desempenho que SIA, pois as demais soluções encontradas por este possuem custos superiores que

as soluções encontradas pelo AG.

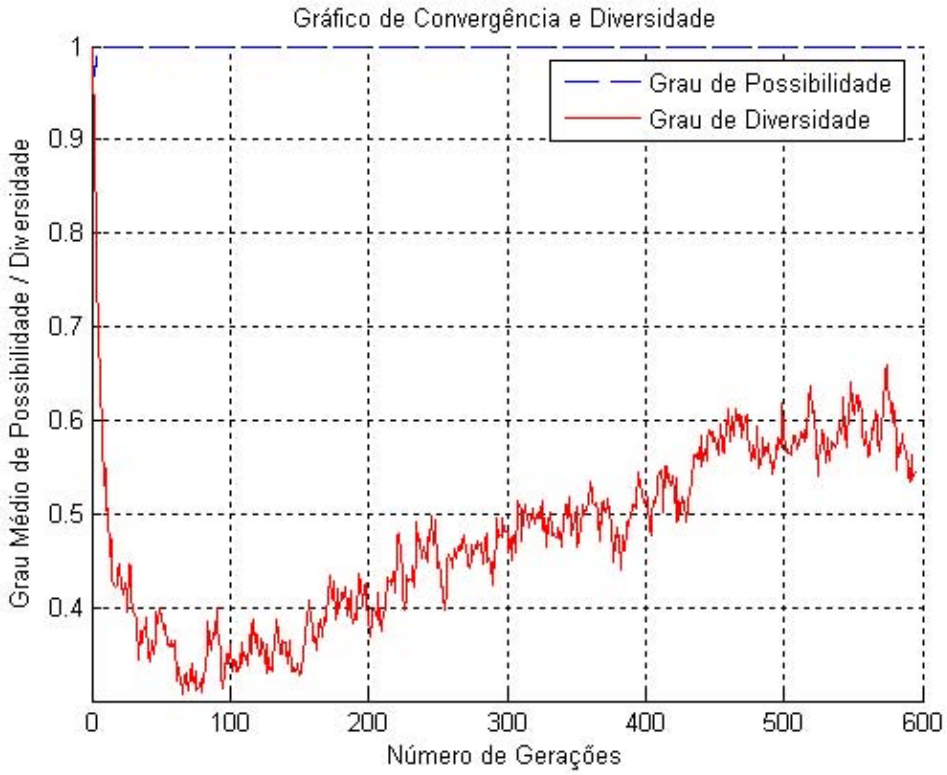

Figura 6.6: Gráfico de diversidade e convergência obtido pelo AG para o grafo Swiss

Em relação à manutenção da diversidade populacional, o SIA (gráfico da Figura 6.7) obteve desempenho melhor que o AG (gráfico da Figura 6.6), pois enquanto este conseguiu manter, em média, 50% de indivíduos únicos na população em cada geração, o SIA conseguiu manter 97%.

O tempo consumido em cada rodada, tanto para o AG quanto para o SIA, foi de 55 segundos, em média. Neste tempo, enquanto o SIA executou 50 gerações, o AG executou 595.

O número de indivíduos avaliados pelos algoritmos tem como limitante superior 25.000 para SIA e 29.750 para o AG.

6.3.4 Grafo Berlim

O grafo Berlim, disponível no repositório TSPLIB, representa todas as ligações possíveis entre 52 pontos de referência da capital alemã. As distâncias entre eles também são representadas por números triangulares *fuzzy* do tipo (c_i, c_m, c_s), no qual, c_m corresponde ao valor

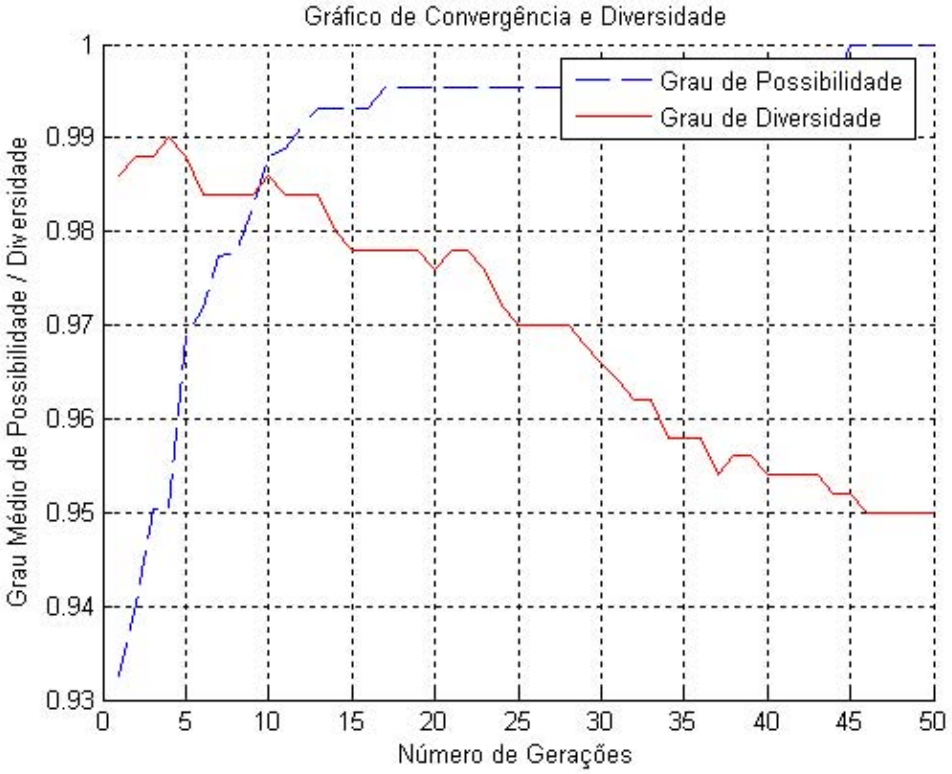

Figura 6.7: Gráfico de diversidade e convergência obtido pelo SIA para o grafo Swiss

modal da distância de um ponto a outro e desvio de 10% para os limitantes inferiores (c_i) e superiores (c_s).

Este é um grafo completo com 52 nós e 1326 arestas, totalizando em torno de $6,81 \times 10^{85}$ possíveis soluções.

Os resultados obtidos estão exibidos na Tabela 6.5.

Num. Sol.	**AG *Fitness***	**AG Custo *Fuzzy***	**SIA *Fitness***	**SIA Custo *Fuzzy***
1	0,9833	[5491,70 6101,80 6712,10]	0,9551	[5522,60 6136,30 6750,00]
2	0,9837	[5491,20 6101,30 6711,50]	0,9626	[5514,30 6127,10 6739,80]
3	0,9873	[5487,20 6096,90 6706,70]	0,9649	[5511,70 6124,20 6736,60]
4	0,9879	[5486,40 6096,10 6705,70]	0,9682	[5508,10 6120,20 6732,30]
5	0,9888	[5485,50 6095,10 6704,60]	0,9704	[5505,80 6117,50 6729,30]
6	0,9952	[5478,50 6087,30 6696,00]	0,9845	[5490,30 6100,30 6710,40]
7	0,9960	[5477,70 6086,30 6695,00]	0,9923	[5481,70 6090,80 6700,00]
8	0,9961	[5477,50 6086,20 6694,80]	0,9973	[5476,20 6084,70 6693,20]
9	0,9985	[5474,80 6083,20 6691,50]	0,9985	[5474,80 6083,20 6691,50]
10	1,0000	[5473,20 6081,40 6689,60]	1,0000	[5473,20 6081,40 6689,60]

Tabela 6.5: Resultados obtidos pelo AG e pelo SIA para o grafo Berlim

Ambos os algoritmos obtiveram desempenhos semelhantes, pois todas as soluções (árvores) encontradas possuem alto grau de possibilidade de serem melhores que T^*. Porém, como o AG tende a realizar uma busca mais concentrada em torno do ótimo, neste caso, ele conseguiu obter, em média de grau de possibilidade, um conjunto solução melhor que o SIA.

O tempo consumido em cada rodada, tanto para o AG quanto para o SIA, foi de 1 minuto e 25 segundos, em média. Neste tempo, enquanto o SIA executou 50 gerações, o AG executou 850.

O número de indivíduos avaliados pelos algoritmos tem como limitante superior 25.000 para SIA e 42.500 para o AG.

O comportamento da diversidade populacional e a convergência do algoritmo tanto para o AG quanto para o SIA, estão exibidos nos gráficos das Figuras 6.8 e 6.9, respectivamente.

Em relação à manutenção da diversidade da população, o SIA (Figura 6.9) novamente foi mais eficiente que o AG (Figura 6.8), pois enquanto este conseguiu manter, em média, 47% de indivíduos únicos na população em cada geração, o SIA conseguiu manter 98%.

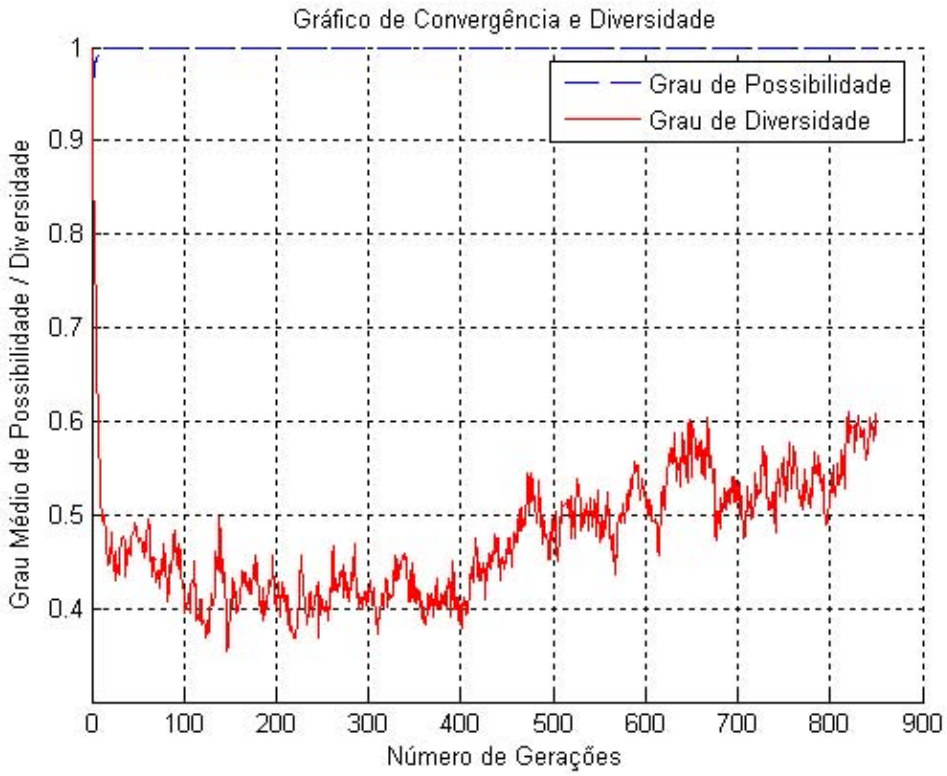

Figura 6.8: Gráfico de diversidade e convergência obtido pelo AG para o grafo Berlim

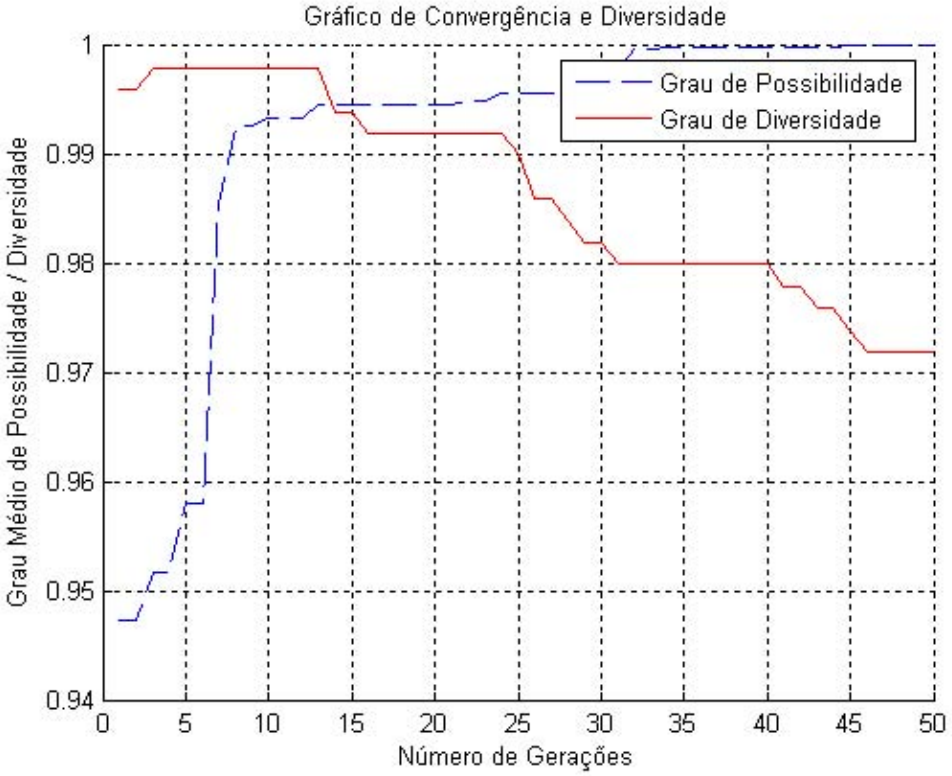

Figura 6.9: Gráfico de diversidade e convergência obtido pelo SIA para o grafo Berlin

6.3.5 Grafo USA Map

Em Okada (2001) foi encontrado este grafo no qual cada aresta representa o tempo (aproximado) consumido para se deslocar entre duas cidades (nós) norte-americanas (Figura 6.10). O tempo (custo) em questão é representado por um número triangular *fuzzy* do tipo (c_i, c_m, c_s), no qual, c_m corresponde ao valor modal do tempo gasto para se deslocar de uma cidade a outra, e desvio de 10% para os limitantes inferiores (c_i) e superiores (c_s).

Este grafo é composto por 70 nós e 210 arestas e um grafo completo com o mesmo número de nós possui em torno de $2,93 \times 10^{125}$ possíveis soluções. A sua principal característica é possuir uma única árvore geradora mínima (T^*) e muitas outras árvores com custos aproximados (Almeida et al. 2005b).

Em relação aos resultados (Tabela 6.6), os dois métodos propostos foram eficientes e conseguiram encontrar bons conjuntos soluções. Entretanto, o AG obteve um conjunto solução melhor, em média de grau de possibilidade, que o obtido pelo SIA. Isso se deve ao fato, desse grafo possuir apenas uma árvore com grau de possibilidade igual a 1 (T^*), e as demais árvores com melhores graus de possibilidade estarem ao seu redor, ou seja, elas compartilham várias

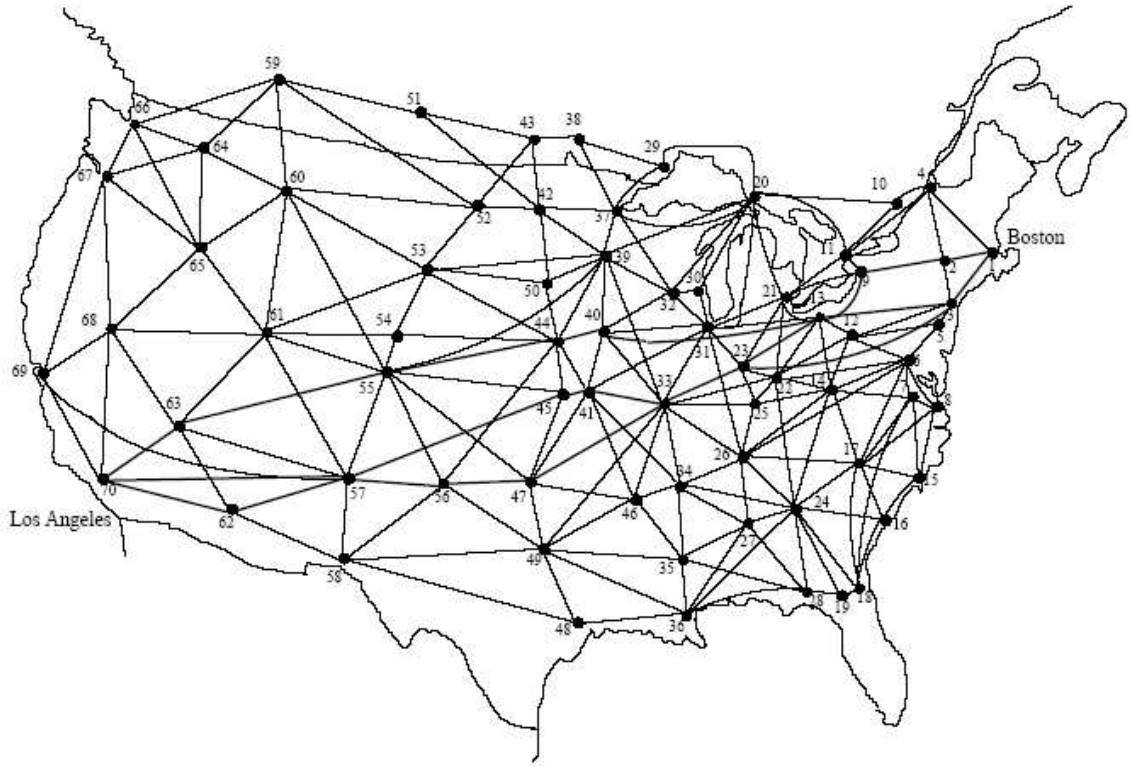

Figura 6.10: Grafo Usa Map - 70 nós e 210 arestas

Num. Sol.	**AG *Fitness***	**AG Custo *Fuzzy***	**SIA *Fitness***	**SIA Custo *Fuzzy***
1	0,9946	[1508,20 1676,10 1844,00]	0,9914	[1509,20 1677,20 1845,20]
2	0,9949	[1508,10 1676,00 1843,90]	0,9934	[1508,60 1676,50 1844,40]
3	0,9952	[1508,10 1675,90 1843,70]	0,9946	[1508,30 1676,10 1843,90]
4	0,9958	[1507,80 1675,70 1843,60]	0,9952	[1508,10 1675,90 1843,70]
5	0,9961	[1507,70 1675,60 1843,50]	0,9958	[1507,80 1675,70 1843,60]
6	0,9964	[1507,60 1675,50 1843,40]	0,9961	[1507,70 1675,60 1843,50]
7	0,9982	[1507,10 1674,90 1842,70]	0,9964	[1507,60 1675,50 1843,40]
8	0,9988	[1506,90 1674,70 1842,50]	0,9982	[1507,10 1674,90 1842,70]
9	0,9997	[1506,70 1674,40 1842,10]	0,9997	[1506,70 1674,40 1842,10]
10	1,0000	[1506,60 1674,30 1842,00]	1,0000	[1506,60 1674,30 1842,00]

Tabela 6.6: Resultados obtidos pelo AG e pelo SIA para o USA Map

arestas entre si, e também, porque o AG realiza uma busca mais concentrada ao redor de T^*.

O tempo consumido em cada rodada, para cada um dos algoritmos, foi de 4 minutos, em média. Neste tempo, enquanto o SIA executou 50 gerações, o AG executou 2065.

O número de indivíduos avaliados pelos algoritmos tem como limitante superior 25.000

para SIA e 103.250 para o AG.

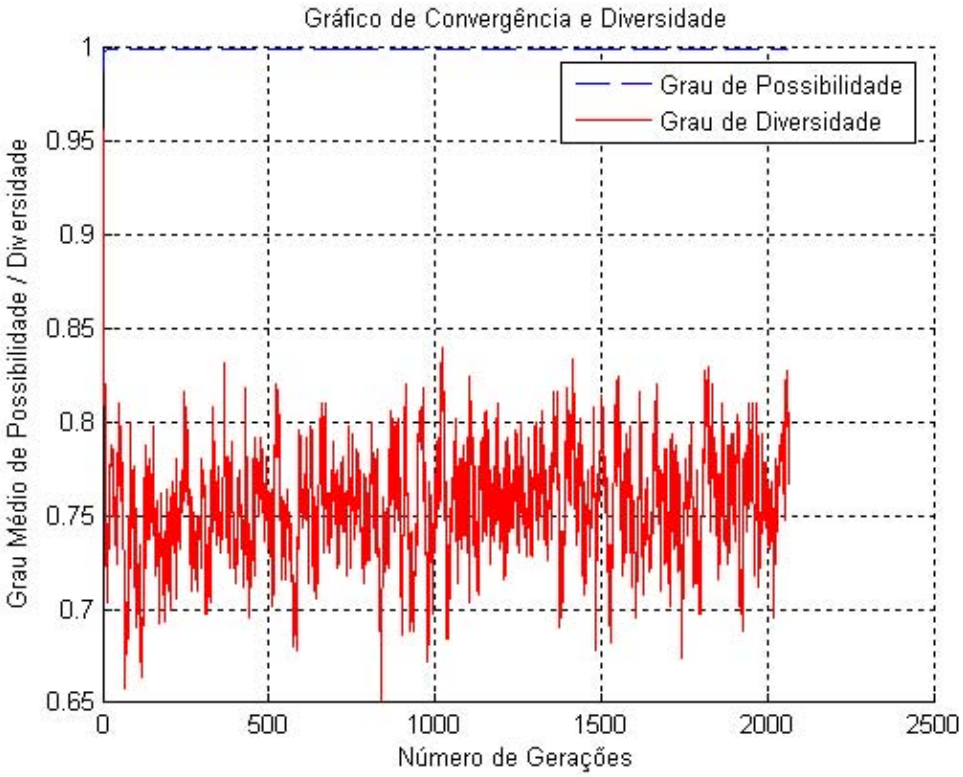

Figura 6.11: Gráfico de diversidade e convergência obtido pelo AG para o grafo USA Map

Em relação ao controle da diversidade populacional, o AG (gráfico da Figura 6.11) obteve desempenho melhor que o SIA (gráfico da Figura 6.12), pois enquanto este conseguiu manter, em média, 67% de indivíduos únicos na população em cada geração, o AG manteve 75%.

6.3.6 Grafo 274

Este grafo foi retirado da base de dados OR-LIBRARY. A sua principal característica é possuir muitas árvores geradoras mínimas (T^*), o que faz com que ele seja uma boa instância para avaliar o desempenho dos algoritmos para resolver problemas multimodais.

O custo de cada aresta também foi estabelecido como um número triangular *fuzzy* do tipo (c_i, c_m, c_s), no qual, c_m corresponde ao valor modal, $c_i = c_m - 0,1 \times c_m - a$ equivale ao limitante inferior e $c_s = c_m + 0,1 \times c_m + a$ o limitante superior, onde a é um número aleatório entre 1 e 5 ($a \in [1, 5]$).

Esta instância é constituída por 50 nós e 274 arestas. Um grafo completo com o mesmo número de nós possui aproximadamente $3,55 \times 10^{81}$ árvores geradoras.

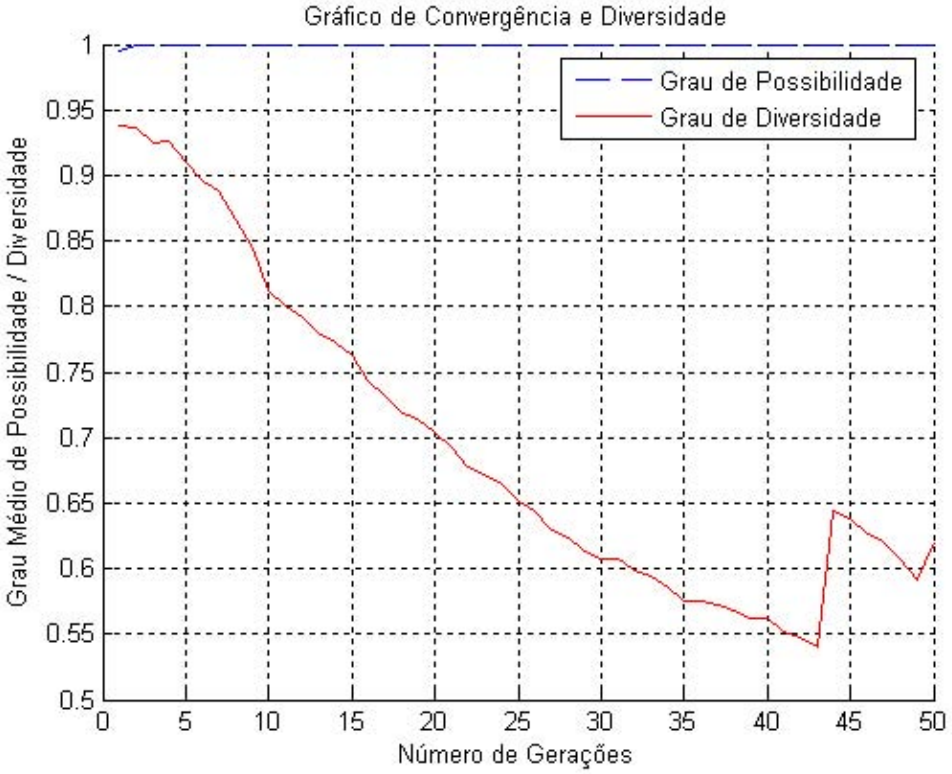

Figura 6.12: Gráfico de diversidade e convergência obtido pelo SIA para o grafo USA Map

Os resultados obtidos estão exibidos na Tabela 6.7.

Num. Sol.	**AG *Fitness***	**AG Custo *Fuzzy***	**SIA *Fitness***	**SIA Custo *Fuzzy***
1	0,9981	[1821,00 2086,00 2351,00]	1,0000	[1820,00 2085,00 2352,00]
2	0,9981	[1822,00 2086,00 2353,00]	1,0000	[1820,00 2085,00 2352,00]
3	0,9981	[1822,00 2086,00 2352,00]	1,0000	[1818,00 2085,00 2352,00]
4	0,9981	[1821,00 2086,00 2353,00]	1,0000	[1821,00 2085,00 2350,00]
5	0,9981	[1820,00 2086,00 2353,00]	1,0000	[1819,00 2085,00 2353,00]
6	1,0000	[1821,00 2085,00 2352,00]	1,0000	[1820,00 2085,00 2352,00]
7	1,0000	[1821,00 2085,00 2348,00]	1,0000	[1821,00 2085,00 2352,00]
8	1,0000	[1821,00 2085,00 2350,00]	1,0000	[1821,00 2085,00 2350,00]
9	1,0000	[1819,00 2085,00 2350,00]	1,0000	[1820,00 2085,00 2351,00]
10	1,0000	[1822,00 2085,00 2350,00]	1,0000	[1820,00 2085,00 2350,00]

Tabela 6.7: Resultados obtidos pelo AG e pelo SIA para o grafo 274

Em relação aos resultados, a característica multimodal do problema exerce forte influência para que SIA encontre soluções melhores que o AG. Enquanto que este conseguiu encontrar apenas 5 soluções (árvores) com grau de possibilidade igual a 1, o SIA encontrou 10, ou seja, todos os indivíduos presentes no conjunto solução encontrados pelo SIA possuem grau

máximo de possibilidade de serem melhores ou iguais a T^*.

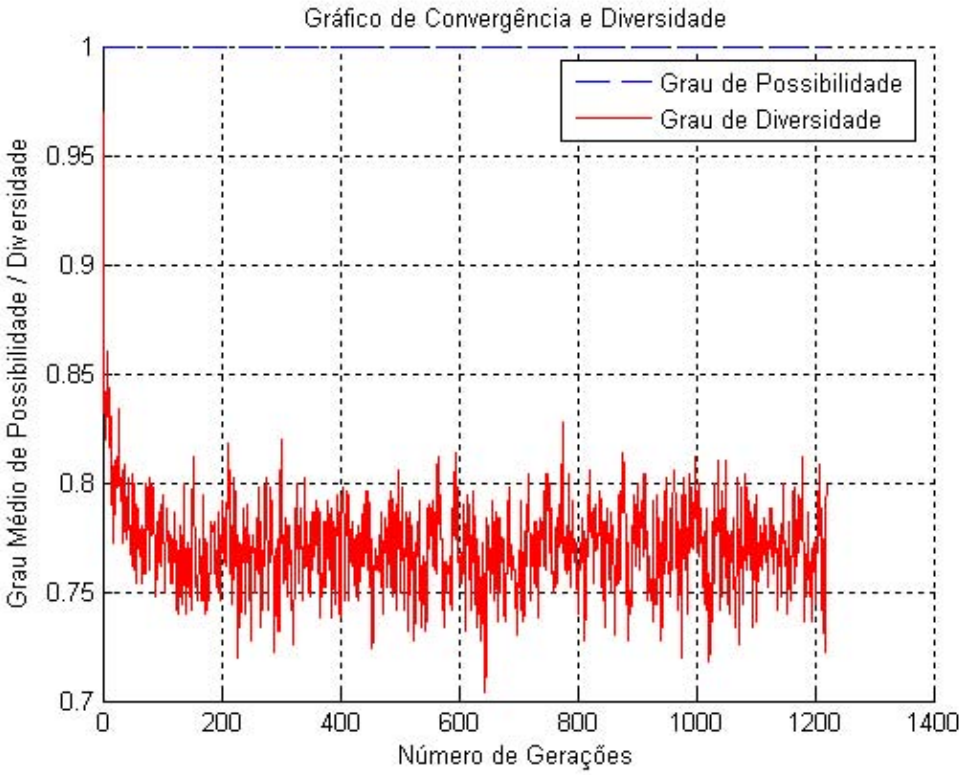

Figura 6.13: Gráfico de diversidade e convergência obtido pelo AG para o grafo 274

O tempo consumido em cada rodada, tanto para o AG quanto para o SIA, foi de 2 minutos, em média, sendo que enquanto o SIA executou 50 gerações, o AG executou 1220.

O número de indivíduos avaliados pelos algoritmos tem como limitante superior 25.000 para SIA e 61.000 para o AG.

Em relação ao controle da diversidade populacional, o SIA (Figura 6.14) conseguiu manter a população mais heterogênea que o AG (Figura 6.13) durante as gerações, pois enquanto este conseguiu manter, em média, 77% de indivíduos únicos na população em cada geração, o SIA conseguiu manter 83%. Isto se deve ao fato do SIA trabalhar com múltiplas populações (clones), que possibilita manter a diversidade da população de maneira mais eficaz que o AG.

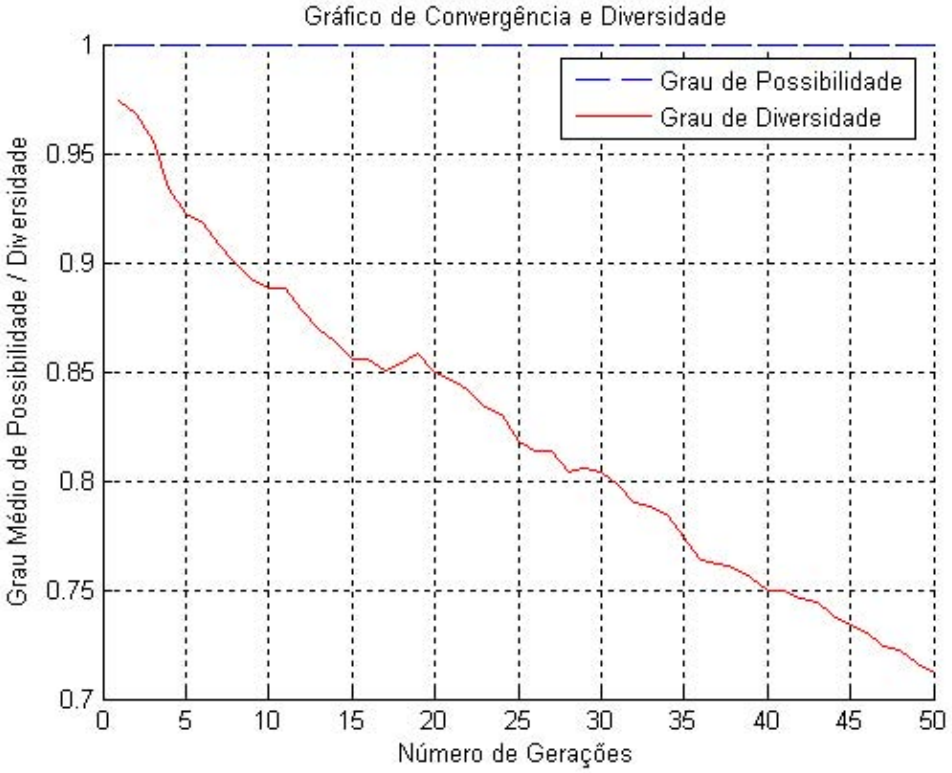

Figura 6.14: Gráfico de diversidade e convergência obtido pelo SIA para o grafo 274

Capítulo 7

Conclusões e Perspectivas Futuras

Uma grande quantidade de problemas no mundo pode ser representada na forma de redes utilizando a teoria dos grafos. Esta abordagem permite obter uma melhor manipulação das entidades envolvidas, de forma que a implementação de métodos para solucioná-los torna-se mais simples e intuitiva. O problema de encontrar a árvore geradora mínima em um grafo é bastante estudado dentro desta teoria, pois o mesmo possui diversas aplicações práticas.

Freqüentemente, em casos reais, as informações associadas aos problemas, como: custo, capacidade, tempo, etc; possuem valores incertos (ou imprecisos), fazendo com que a teoria clássica dos conjuntos torne-se ineficaz para representá-los. Assim, é necessário utilizar uma representação mais robusta ,que pode ser obtida por meio da teoria dos conjuntos *fuzzy*.

Neste trabalho, foi estudado o problema de encontrar a árvore geradora mínima em grafos que possuem incertezas associadas às arestas, e a teoria de possibilidade (Zadeh 1978) foi utilizada para encontrar o conjunto solução composto pelas árvores que possuem alto grau de possibilidade de serem as melhores soluções. A necessidade de enumerar e comparar todas as soluções entre si faz com que o problema seja NP-Completo e, portanto, ele é intratável por meio de buscas exaustivas ou por métodos exatos.

A simulação de processos evolutivos biológicos permite o entendimento de como a evolução guia os seres vivos na direção de um maior nível de inteligência. A evolução converge ao final do processo para soluções que apresentam um comportamento mais adaptado ao ambiente. A modelagem do processo evolutivo emprega uma série de algoritmos de otimização baseados em regras simples e implementados como métodos de busca em superfícies de resposta de desempenho. O processo de otimização, inerente à seleção dos elementos mais bem adaptados, melhora a qualidade média das soluções obtidas ao longo das gerações. Entretanto, o grande

potencial dos algoritmos baseados em computação evolutiva vem da integração da ampla exploração do espaço de busca com um processo de busca mais localizada. Essa integração resulta num alto grau de robustez, que permite a aplicação em vários problemas práticos.

Baseando-se em técnicas da computação evolutiva, um algoritmo genético (Almeida et al. 2005a, Almeida et al. 2005b) e um sistema imunológico artificial foram propostos para resolver o problema da árvore geradora mínima com parâmetros *fuzzy*, cujo objetivo é explorar o espaço de busca de soluções de forma eficiente visando encontrar um conjunto-solução aproximado, de forma a contornar a questão da complexidade.

Enquanto que o algoritmo genético evolui os candidatos buscando soluções ao redor dos indivíduos mais adaptados ao ambiente, o sistema imunológico artificial evolui múltiplas populações (clones). Em geral, o sistema imunológico artificial consegue manter um melhor controle sobre a diversidade da população, possibilitando-o obter maior desempenho quando aplicado em grafos que possuem diversas árvores com o mesmo custo total *fuzzy* de T^*. Entretanto, quando as melhores árvores se encontram ao redor de T^* (muitas vezes, com a variação de uma única aresta) o algoritmo genético consegue ser mais eficiente.

O tempo consumido pelas meta-heurísticas (segundos) é bem menor que o tempo consumido pelo método exato (dias) e os resultados obtidos são bastante satisfatórios, provando que a implementação de técnicas da computação evolutiva é realmente promissora para resolver problemas de explosão combinatória, como o caso do problema abordado neste trabalho.

Perspectivas Futuras

A implementação de uma busca local dentro do sistema imunológico artificial, logo após a maturação da população de clones, pode fazer com que o processo de explotação do espaço de busca seja otimizado, possibilitando obter resultados melhores.

Novas técnicas de seleção e manutenção de diversidade podem ser adaptadas ao algoritmo genético de forma a melhorar a qualidade dos resultados. Além disso, ao invés da população de imigrantes que deve preencher a população entre uma geração e outra ser formada por elementos encontrados ao redor de T^*, pode-se gerá-los aleatoriamente de forma a aumentar a variabilidade genética entre os cromossomos em cada geração.

A utilização de outras técnicas da computação evolutiva como otimização por colônias de formigas, algoritmos co-evolutivos, ou ainda a implementação de algoritmos meméticos

podem ser métodos eficientes para resolver o problema estudado.

Devido aos bons resultados obtidos pela aplicação dessas técnicas para resolver o problema da árvore geradora mínima com parâmetros *fuzzy*, pode-se propor a implementação das mesmas para resolver outros problemas de grafos, como é o caso do problema de encontrar os k-caminhos mínimos *fuzzy*.

Referências Bibliográficas

Ahuja, R. K., Magnati, T. L. & Orlin, J. B. (1993), *Network flows : Theory, Algorithms, and Applications*, 1 edn, Prentice-Hall.

Ali, M., Ramamurthy, B. & Deogun, J. S. (2000), 'Routing and wavelength assignment with power considerations in optical networks', *Computer Networks* **32**(5), 539–555.

Almeida, T. A., Prado, F. M. S., Sousa, V. N., Takahashi, M. T. & Yamakami, A. (2005a), A genetic algorithm to solve the minimum spanning tree problem with fuzzy parameters using possibility measure, *in* 'Fuzzy Information Processing Society, 2005. NAFIPS 2005. Annual Meeting of the North American', IEEE, MI, USA, pp. 627–632.

Almeida, T. A., Takahashi, M. T. & Yamakami, A. (2005b), 'An evolutionary approach to solve minimum spanning tree problem with fuzzy parameters', *CIMCA* **2**, 203–208.

Bazaraa, M. S., Jarvis, J. J. & Sherali, H. F. (1990), *Linear Programming and Network Flows*, 2 edn, John Wiley & Sons, NY, USA.

Bäck, T., Fogel, D. B. & Michalewicz, Z. (2000a), *Evolutionary Computation 1: Basic Algorithms and Operators*, 1 edn, Institute of Physics Publishing, Bristol, UK.

Bäck, T., Fogel, D. B. & Michalewicz, Z. (2000b), *Evolutionary Computation 2: Advanced Algorithms and Operators*, 1 edn, Institute of Physics Publishing, Bristol, UK.

Beasley, J. E. (1990), 'Or-library: Distributing test problems by electronic mail', *Jornal of the Operational Research Society* **41**(11), 1069–1072. `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.

Blue, M. P., Bush, B. W. & Puckett, J. (2002), 'Unified approach to fuzzy graph problems', *Fuzzy Sets and Systems* **125**(3), 355–368.

Castro, L. N. (2001), Engenharia Imunológica: Desenvolvimento e Aplicação de Ferramentas Computacionais Inspiradas em Sistemas Imunológicos Artificiais, PhD thesis, Universidade Estadual de Campinas - UNICAMP, Campinas, SP, Brasil. `www.dca.fee.unicamp.br/~vonzuben/research/lnunes_dout/index.html`.

Chang, P. T. & Lee, E. S. (1999), 'Fuzzy decision networks and deconvolution', *Computers and Mathematics with Applications* **37**(11), 53–63.

Chunde, Y. (1996), 'On the optimization problem of spanning tree in fuzzy network', *The Journal of China Universities of Posts and Telecommunications* **3**(2), 22–25.

Dasgupta, D. (1998), *Artificial Immune Systems and Their Applications*, 1 edn, Springer-Verlag, Berlin, Germany.

Dreher, H. (1995), *The Immune Power Personality*, Penguin Books, USA.

Dubois, D. & Prade, H. (1980), *Fuzzy Sets and Systems*, Academic Press, NY, USA.

Goldbarg, M. C. & Luna, H. P. (2000), *Otimização Combinatória e Programação Linear: Modelos e Algoritmos*, 1 edn, Campus, RJ, Brasil.

Gomide, F. & Pedrycz, W. (1998), *An Introduction to Fuzzy Sets: Analysis and Design*, Complex Adaptive Systems, 1 edn, MIT Press, MA, USA.

Harary, F. (1972), *Graph Theory*, Addison-Wesley, MA, USA.

Holland, J. H. (1992), *Adaptation in Natural and Artificial Systems*, 2 edn, MIT Press, MA, USA.

Janeway, C. A., Travers, P., Walport, M. & Capra, J. D. (2000), *Imunobiologia: O Sistema Imunológico na Saúde e na Doença*, 4 edn, Artmed, Porto Alegre, RS, Brasil.

Michalewicz, Z. (1996), *Genetic Algorithms + Data Structures = Evolution Programs*, 3 edn, Springer-Verlag and Heidelberg GmbH & Co. K, Berlin, Germany.

Okada, S. (2001), Interactions among paths in fuzzy shortest path problems, *in* 'Joint 9th IFSA World Congress and 20th NAFIPS International Conference', Vol. 1, Vancouver, BC, Canada, pp. 41–46.

Raidl, G. R. & Julstrom, B. A. (2003), 'Edge-sets: An effective evolutionary coding of spanning trees', *IEEE Transactions on Evolutionary Computation* **7**(3), 225–239.

Rosenfeld, A. (1975), Fuzzy graphs, fuzzy sets and their applications to cognitive and decision processes, *in* L. A. Zadeh, K. S. Fu & M. Shimura, eds, 'Proceeding of U.S.-Japan Sem., University of California, Berkeley, CA, 1974', Academic Press, New York, pp. 77–95.

Takahashi, M. T. (2004), Contribuições ao Estudo de Grafos Fuzzy: Teoria e Algoritmos, PhD thesis, Universidade Estadual de Campinas - UNICAMP, Campinas, SP, Brasil.

Timmis, J. (2000), Artificial Immune Systems: A Novel Data Analysis Technique Inspired by the Immune Network Theory, PhD thesis, Department of Computer Science, University of Wales, Aberystwyth. Ceredigion. Wales. `http://www.cs.kent.ac.uk/pubs/2000/1102`.

Wilson, R. & Watkins, J. (1990), *Graphs: An Introductory Approach: A First Course In Discrete Mathematics*, John Wiley, NY, USA.

Zadeh, L. A. (1965), 'Fuzzy sets', *Information and Control* **8**(3), 338–353.

Zadeh, L. A. (1978), 'Fuzzy sets as a basis for a theory of possibility', *Fuzzy Sets and Systems* **1**(1), 3–28.