**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**

Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial

---

**DISSERTAÇÃO**

apresentada à UTFPR
para obtenção do grau de

**MESTRE EM CIÊNCIAS**

por

**MARCOS HIDEO MARUO**

---

# PROJETO AUTOMÁTICO DE SISTEMAS NEBULOSOS UTILIZANDO ALGORITMOS GENÉTICOS AUTO-ADAPTATIVOS

---

Banca Examinadora:

Presidente e Orientadora:

| | |
|---|---|
| **PROF.ª DR.ª MYRIAM REGATTIERI DELGADO** | **UTFPR** |

Examinadores:

| | |
|---|---|
| **PROF. DR. LEANDRO DOS SANTOS COELHO** | **PUC-PR** |
| **PROF. DR. LEANDRO MAGATÃO** | **UTFPR** |
| **PROF.ª DR.ª LÚCIA VALÉRIA RAMOS DE ARRUDA** | **UTFPR** |

Curitiba, agosto de 2006

Ficha catalográfica elaborada pela Biblioteca da UTFPR – Campus Curitiba

M389p Maruo, Marcos Hideo

Projeto automático de sistemas nebulosos utilizando algoritmos genéticos auto-adaptativos / Marcos Hideo Maruo. Curitiba. UTFPR, 2006

XIII, 137 f. : il.; 30 cm

Orientadora: Profª. Drª. Myriam Regattieri Delgado

Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial. Curitiba, 2006

Bibliografia: f. 127 – 138

1. Algoritmos genéticos. 2. Automação de sistemas. 3. Sistemas nebulosos. I. Delgado, Myriam Regattieri, orient. II. Universidade Tecnológica Federal do Paraná. Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial. III. Título.

CDD: 511.8

**Marcos Hideo Maruo**

# *Projeto automático de sistemas nebulosos utilizando algoritmos genéticos auto-adaptativos*

Dissertação apresentada ao programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná como requisito parcial para a obtenção do grau de Mestre em Ciências
Área de Concentração: Informática Industrial

Orientadora Myriam Regattieri De Biase Delgado

Universidade Tecnológica Federal do Paraná - UTFPR
Programa de Pós graduação em Engenharia Elétrica e Informática Industrial - CPGEI

Curitiba

2006

*“I accept chaos, I’m not sure whether it accepts me.”*
*Robert Zimmerman (aka Bob Dylan)*

*“Doubt is not a pleasant condition, but certainty is absurd.”*
*(Voltaire)*

*“I don’t have all the answers. I don’t know how to jolt myself into seeing what each moment could become. But I do know one thing: the solution doesn’t involve watering down my every little idea and creative impulse for the sake of someday easing my fit into a mold. It doesn’t involve tempering my life to better fit someone’s expectations. It doesn’t involve constantly holding back for fear of shaking things up.”*
*xkcd web comics*

*Dedico esta dissertação a meu pai (In memorian) pelo exemplo de luta,*
*à minha mãe pelo apoio.*

# *Agradecimentos*

O presente trabalho foi realizado com o apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq - Brasil.

Este trabalho também recebeu o financiamento da Fundação Araucária/Funcefet-PR 4758-018/2004.

Dedico meus sinceros agradecimentos:

– à Profa. Myriam Regattieri Delgado, pela orientação e incentivo;

– à equipe do Laboratório de Bioinformática da UTFPR;

– à equipe do Laboratório de Sistemas Distribuídos da UTFPR;

– ao Prof. Dr. Heitor Silvério Lopes, Prof. Dr. Richard Demo Souza e Prof. Dr. Luiz Nacamura Júnior pelo apoio sempre manifestado;

– à estação de trabalho Sparc e Douglas, "a cadeira", pela companhia no laboratório;

– a todos os colegas do Mestrado e Doutorado da UTFPR.

# *Sumário*

# *Lista de símbolos*

| | | |
|---|---|---|
| $\Upsilon$ | Número de itens no problema da mochila, | p. 106 |
| $\mathcal{G}$ | gramática para a geração dos termos, | p. 23 |
| $\mathcal{M}$ | regra de associação entre rótulos e conjuntos nebulosos, | p. 23 |
| $\mathcal{P}$ | Partição nebulosa, | p. 23 |
| $\mathcal{T}$ | conjunto de termos lingüísticos, | p. 23 |
| $\mathcal{X}$ | nome da variável lingüística, | p. 23 |
| $\delta^0$ | Conjunto inicial dos parâmetros estratégicos dos operadores, | p. 36 |
| $\eta$ | Grau de ativação de uma regra nebulosa, | p. 29 |
| $\gamma$ | $\gamma$-completude, | p. 64 |
| $\hat{y}(\mathbf{x}_p)$ | Saída estimada do sistema nebuloso, | p. 64 |
| $\kappa$ | $\kappa$-sobreposição, | p. 64 |
| $\lambda$ | Número de descendentes, | p. 36 |
| $\mathbf{K}$ | Lucro do objeto alocado na mochila, | p. 106 |
| $\mathbf{Q}$ | Número de itens alocados na mochila, | p. 106 |
| $\mathbf{R}$ | Total de restrições, | p. 106 |
| $\mathbf{W}$ | Peso ou volume o objeto na mochila, | p. 106 |
| $\mathbf{c}$ | Limite da restrição, | p. 106 |
| $\mathbf{h}$ | Cromossomo, | p. 41 |
| $\mathbf{s}$ | norma-s, | p. 17 |
| $\mathbf{t}$ | norma-t, | p. 15 |
| $\mathbf{y}$ | Vetor de saídas de treinamento, | p. 78 |
| $\mu$ | Função de pertinência, | p. 9 |
| $\omega$ | Média da distribuição normal, | p. 114 |
| $\rho$ | Operador de recombinação, | p. 43 |
| $\sigma$ | Desv. pad. da distribuição normal, | p. 114 |
| $\mathbf{x}_p$ | Entrada do sistema nebuloso, | p. 64 |
| $\mathbb{N}$ | Conjunto dos números naturais, | p. 37 |

| | | |
|---|---|---|
| $\mathbb{R}$ | Conjunto dos número reais, | p. 37 |
| $\mathbb{R}^+$ | Conjunto dos número reais positivos, | p. 37 |
| $\mathbb{X}$ | Universo de discurso, | p. 9 |
| $\varphi$ | Tamanho da população, | p. 36 |
| $\xi$ | Operador de mutação, | p. 45 |
| $\zeta$ | Mecanismo de seleção, | p. 47 |
| $\mathbb{R}^n$ | Espaço euclidiano n-dimensional, | p. 96 |
| $C$ | Complemento nebuloso, | p. 15 |
| $D$ | Número de dimensões consideradas nas restrições do problema da mochila, | p. 106 |
| $F$ | Função de avaliação, | p. 37 |
| $f$ | semântica da regra nebulosa, | p. 24 |
| $g$ | Conseqüente de Takagi-Sugeno, | p. 31 |
| $G$ | Função de reprodução, | p. 37 |
| $h$ | Indivíduo, | p. 36 |
| $I$ | Alfabeto do gene, | p. 36 |
| $l$ | Tamanho do cromossomo, | p. 36 |
| $M$ | Número de mochilas, | p. 106 |
| $m$ | Número de regras nebulosas, | p. 29 |
| $N$ | Número de pares de treinamento, | p. 78 |
| $Nec$ | Medida de necessidade entre conjuntos nebulosos, | p. 19 |
| $O$ | Conjunto de descendentes, | p. 48 |
| $P^0$ | População inicial, | p. 36 |
| $p_c$ | Probabilidade de recombinação, | p. 44 |
| $p_m$ | Probabilidade de mutação, | p. 46 |
| $Poss$ | Medida de possibilidade entre conjuntos nebulosos, | p. 19 |
| $Q$ | Tipo de conseqüente de Takagi-Sugeno, | p. 77 |
| $T$ | Número de gerações, | p. 46 |
| $U$ | Função de atualização, | p. 37 |
| $y_p$ | Saída desejada, | p. 64 |

# *Lista de figuras*

# *Lista de tabelas*

# *Lista de abreviaturas e siglas*

| | | |
|---|---|---|
| AG | Algoritmos Genéticos, | p. 2 |
| CE | Computação Evolutiva, | p. 1 |
| CF | Computação Flexível, | p. 1 |
| EQM | Erro quadrático médio, | p. 64 |
| FRBS | *Fuzzy Rule-based systems*, | p. 2 |
| GFRBS | *Genetic fuzzy rule-based systems*, | p. 3 |
| GFS | *Genetic Fuzzy Systems,* | p. 3 |
| IC | Inteligência Computacional, | p. 1 |
| LD | Linearmente dependentes, | p. 79 |
| MLP | Perceptron de múltiplas camadas, | p. 93 |
| norma-s | Co-norma triangular, | p. 17 |
| norma-t | Norma triangular, | p. 15 |
| NP | Não-polinomial, | p. 89 |
| P | Polinomial, | p. 88 |
| sGA | *standard Genetic Algorithm,* | p. 49 |
| TSK | Takagi-Sugeno-Kang, | p. 31 |

# *Resumo*

Abordagens híbridas baseadas em sistemas nebulosos com aprendizado através de algoritmos genéticos são também denominadas de sistemas genético-nebulosos. Neste caso, os algoritmos genéticos são usados para resolver o problema do projeto automático de sistemas nebulosos. Entretanto, a questão da definição dos parâmetros evolutivos do algoritmo genético persiste. Esta dissertação propõe uma metodologia para o controle dos parâmetros evolutivos para um sistema genético-nebuloso. A abordagem do sistema genético-nebuloso que está sendo utilizada é inspirada em um modelo anterior, no qual diferentes populações codificam soluções parciais do problema do projeto automático de sistemas nebulosos, populações estas organizadas em quatro níveis hierárquicos que co-evoluem de forma harmônica. Cada nível hierárquico codifica as funções de pertinência, as regras individuais, as bases de regras e os sistemas nebulosos, respectivamente. A co-evolução permite que sejam estabelecidas relações de hierarquia e cooperação entre indivíduos representando os diferentes parâmetros dos sistemas nebulosos. O modelo original apresenta bom desempenho para uma grande classe de problemas. Entretanto, devido ao uso de múltiplas populações, com informações significativamente diferentes, o ajuste dos parâmetros evolutivos do sistema se torna um problema complexo. Portanto, o objetivo do uso de uma abordagem auto-adaptativa, na qual o próprio AG se encarrega de selecionar um bom conjunto de parâmetros, é liberar o usuário do processo de definição manual dos parâmetros evolutivos mantendo o bom desempenho do sistema nebuloso. A utilização do algoritmo evolutivo, não apenas para encontrar a solução do problema, mas também para ajustar uma série de parâmetros do próprio algoritmo se constitui em uma das principais contribuições deste trabalho. O desempenho do mecanismo de auto-adaptação de parâmetros evolutivos que está sendo proposto é avaliado em duas fases: inicialmente, a auto-adaptação é testada, utilizando-se problemas de otimização contínua e combinatória; depois, a auto-adaptação é aplicada para resolver o problema do projeto automático de sistemas baseados em regras nebulosas, e para isto, o sistema genético-nebuloso resultante é usado na aproximação de funções.

Palavras-chave: Sistemas genético-nebulosos, auto-adaptação, projeto automático de sistemas nebulosos.

# *Abstract*

Hybrid approaches, based on fuzzy systems, and equipped with learning capabilities by genetic algorithms are named genetic fuzzy systems. In this case, genetic algorithms are used to solve the complex problem of the automatic design of fuzzy systems. However, the question of how to define the evolutionary parameters of the genetic algorithm still persists. This dissertation proposes a method for self-adapting the evolutionary parameters of a genetic fuzzy system. The genetic fuzzy system approach that is being used has been inspired by a previous model, in which different populations encode partial solutions for the problem of automatic fuzzy system design, and are organized into four hierarchical levels, co-evolving in a harmonic way. Each hierarchical level encodes membership functions, individual rules, rule-bases and fuzzy systems, respectively. The co-evolution supports hierarchical and collaborative relations among individuals representing different parameters of fuzzy models. The original model performs well for a large class of problems. However, due to the use of multiples populations where significantly different information are being considered, the process of tuning evolutionary parameters becomes a complex problem. Thus, the objective of using a self-adaptive approach, where the own GA is capable of finding out a good set of parameters, is to free the user of manually defining evolutionary parameters, while preserving the good performance of the fuzzy system. The use of an evolutionary algorithm, not only to find out the problem solution, but also to tune a set of evolutionary parameters, constitutes one of the main contributions of this work. The performance of the self-adaptation mechanism, is evaluated in two phases: first, self-adaptation is tested considering continuous and combinatorial optimization problems; second, self-adaptation is applied to solve the problem of automatic fuzzy system design, for this, the resulting genetic-fuzzy system is used to approximate functions.

Keywords: Genetic fuzzy systems, self-adaptation, automatic design of fuzzy systems.

# 1 Introdução

A Inteligência Computacional (IC), também conhecida como Computação Flexível (*soft computing*)(CF) (JANG; SUN; MIZUTANI, 1997), é a área de estudo que contempla diferentes metodologias inspiradas em mecanismos naturais. Esta característica permite, muitas vezes, a obtenção de sistemas com melhor desempenho, autonomia, tratabilidade e verossimilhança (CORDÓN *et al.*, 2001a). Os principais ramos da Inteligência Computacional são: Redes Neurais Artificiais, Sistemas Nebulosos e Computação Evolutiva (JANG; SUN; MIZUTANI, 1997), que compreende, entre outras metodologias, os algoritmos genéticos (EIBEN, 2002). Os sistemas baseados em IC, em especial os sistemas nebulosos, são capazes de modelar, com um grau significativo de simplificação, conhecimento semelhante ao de um operador humano em um domínio específico, podendo ainda adaptar-se, adquirir conhecimento em um ambiente dinâmico e tornar mais clara a forma como cada decisão é tomada (CORDÓN *et al.*, 2001a).

Os trabalhos em redes neurais artificiais, por exemplo, têm sido motivados desde o começo pelo reconhecimento de que o cérebro humano processa informações de uma forma inteiramente diferente do computador digital convencional (HAYKIN, 1998). O cérebro é um sistema de processamento de informação complexo, não-linear e paralelo. Ele tem a capacidade de organizar seus constituintes estruturais, os neurônios, de forma a realizar certos processamentos, por exemplo o reconhecimento de padrões, de maneira mais eficiente que o mais rápido computador digital existente (HAYKIN, 1998). Considere, por exemplo, a visão humana, que é uma tarefa complexa de processamento de informação (CHURCHLAND; SEJNOWSKI, 1992). A função do sistema visual é fornecer uma representação do ambiente externo à sua volta e, mais importante do que isso, fornecer a informação necessária para interagir com este ambiente. Portanto, a solução dessa classe de problemas através de sistemas computacionais baseados em mecanismos naturais parece ser uma alternativa interessante. Entretanto, a complexidade inerente a estes mecanismos tornaria sua simulação computacional excessivamente demorada. Devido a restrições de poder computacional, sistemas inspirados na natureza fazem uso de simplificações significativas no processamento da informação.

Os algoritmos evolutivos por outro lado constituem uma classe de métodos de busca e

otimização inspirados nos princípios da evolução natural (GOLDBERG, 1989; HOLLAND, 1992). A computação evolutiva (CE) engloba técnicas como os algoritmos genéticos, estratégias evolutivas, programação genética e programação evolutiva (CORDÓN *et al.*, 2001a).

Os algoritmos genéticos (AGs) são algoritmos baseados no princípio da sobrevivência do mais apto e troca de material genético. Eles foram propostos e inicialmente investigados por John Holland na Universidade de Michigan em 1975 (HOLLAND, 1992). A evolução é caracterizada por diferentes níveis de representação (gene, cromossomo, indivíduo, espécie, ecossistema), e 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 funções de desempenho ou aptidão (*fitness surface*). Os AGs caracterizam-se como ferramentas de otimização genéricas que vêm ganhando popularidade principalmente devido a um conjunto particular de características (JANG; SUN; MIZUTANI, 1997):

- não dependem de informações sobre o gradiente da função objetivo;
- podem ser distribuídos em máquinas com processamento paralelo;
- podem ser utilizados em problemas de otimização contínuos ou discretos;
- utilizam regras de transição probabilísticas e são menos suscetíveis à convergência para falsos ótimos;
- podem ser facilmente combinados com outras técnicas de busca e otimização.

Já os sistemas nebulosos, baseados na teoria dos conjuntos nebulosos desenvolvida por Zadeh (1965), são uma das metodologias fundamentais no processamento e representação de informações lingüísticas, com mecanismos específicos para lidar com incerteza e imprecisão. Esta classe de sistemas estende os sistemas clássicos de inferência, por lidar com regras nebulosas, ou seja, regras "se-então" baseadas em termos lingüísticos representados por conjuntos nebulosos (PEDRYCZ; GOMIDE, 1998). O cumprimento da condição do antecedente, mesmo que parcial, ativa a execução do conseqüente, ou seja, uma ação é realizada. Estes atributos tornam possível a aplicação de sistemas baseados em regras nebulosas ou *Fuzzy Rule-Based Systems*(FRBS) em diversos problemas de controle, classificação e modelagem (KLIR; YUAN, 1995; PEDRYCZ; GOMIDE, 1998). Em um sentido mais amplo, um FRBS é um sistema em que a lógica nebulosa é utilizada como ferramenta para representar diferentes formas de conhecimento sobre o problema estudado, assim como para modelar as interações e relações existentes entre suas variáveis.

Cada paradigma da inteligência computacional apresenta um desempenho destacado em um tipo de problema, conforme mostrado na tabela 1.1.

Tabela 1.1: Paradigmas da Inteligência Computacional e suas principais aplicações (JANG; SUN; MIZUTANI, 1997)

| Metodologia | Aplicação |
|---|---|
| Redes neurais | Aprendizado e adaptação |
| Sistemas nebulosos | Representação do conhecimento |
| Computação Evolutiva | Busca aleatória sistemática |

A facilidade de integração dessas metodologias é a principal força da Inteligência Computacional. O projeto de sistemas híbridos, muitas vezes, requer esta integração de forma colaborativa ao invés de um processo competitivo. Por exemplo, a eficácia dos sistemas nebulosos no tratamento de sistemas imprecisos e com informações incertas pode ser associada à capacidade de aprendizado de redes neurais artificiais (JANG, 1993), ou aos eficientes mecanismos de otimização dos algoritmos genéticos (CORDÓN *et al.*, 2001b). Os sistemas híbridos baseados em sistemas nebulosos e com aprendizado através de algoritmos genéticos são também denominados *Genetic Fuzzy Systems* (GFS) ou sistemas genético-nebulosos (CORDÓN *et al.*, 2001b; DELGADO; VON ZUBEN; GOMIDE, 2004; HERRERA; MAGDALENA, 1998) e deverão ser investigados na aplicação da auto-adaptação proposta nesta dissertação.

## 1.1 Caracterização do problema

As duas principais etapas durante o projeto de um FRBS (CORDÓN *et al.*, 2001a; GUILLAUME; MAGDALENA, 2006) são:

1. O projeto do mecanismo de inferência;
2. A geração da base de conhecimento (base de regras e dados).

O problema do projeto de um FRBS do ponto de vista de otimização envolve a busca de uma base de conhecimento ótima, ou seja, a identificação dos parâmetros pertinentes ao projeto da base de conhecimento, a definição de uma base de conhecimento parametrizada que será descrita no capítulo 4 na qual o conjunto de parâmetros obtido satisfaça um critério de otimização. Atualmente, um esforço considerável tem sido utilizado no desenvolvimento de metodologias capazes de extrair automaticamente uma base de conhecimento a partir de dados numéricos (JANG; SUN; MIZUTANI, 1997). Em particular, muitas metodologias têm sido propostas com o objetivo de realizar esta tarefa empregando algoritmos genéticos (CORDÓN

*et al.*, 2001a, 2004; KLIR; YUAN, 1995; PEDRYCZ; GOMIDE, 1998; YUAN; ZHUANG, 1996; CASTRO; CAMARGO, 2004; BONISSONE; KHEDKAR; CHEN, 1996; CARSE; FOGARTY; MUNRO, 1996; CHEONG; LAI, 2000; DE SOUSA; MADRID, 2000; GONZALEZ; HERRERA, 1997; GONZALEZ; PEREZ, 1996; GUROCAK, 1999; HERRERA; LOZANO; VERDEGAY, 1995; NAGAI; ARRUDA, 2002; MARUO; DELGADO, 2006)

Conforme discutido anteriormente, um sistema genético-nebuloso é um sistema nebuloso dotado de um processo de aprendizado baseado em algoritmos genéticos (CORDÓN *et al.*, 2001a). Uma das classes mais promissoras de GFS são os sistemas genético-nebulosos baseados em regras (GFRBS), em que um algoritmo genético é utilizado para evoluir ou ajustar diversos parâmetros de um FRBS (CORDÓN *et al.*, 2001a, 2004). Neste contexto, os parâmetros da base de conhecimento e operadores do processo de inferência constituem o espaço de otimização, ou espaço fenotípico, que deve ser transformado em uma representação genética, o espaço genotípico. Para o AG explorar o espaço genotípico, ele necessita de alguns mecanismos para gerar variantes das soluções-candidatas existentes. O objetivo do processo de busca é maximizar ou minimizar uma função de aptidão que descreve o comportamento do sistema. Neste trabalho, o processo de evolução é o resultado da interação entre a avaliação, seleção e criação de soluções candidatas que representam a base de conhecimento e mecanismo de inferência de um sistema nebuloso.

Delgado (2002) introduziu um método de projeto de FRBS cuja evolução ocorre de forma modular e hierárquica, com mecanismos co-evolutivos associados (DELGADO; VON ZUBEN; GOMIDE, 2000, 2001, 2004). Este método utiliza um AG trabalhando em 4 diferentes populações que codificam as informações dos diferentes níveis hierárquicos para fornecer um conjunto completo de parâmetros do FRBS. Essa configuração permite a implementação de um processo eficaz de cooperação e competição entre indivíduos que permite a obtenção automática de bases de regras com menor complexidade (CORDÓN *et al.*, 2001a). Entretanto, como será discutido a seguir, o modelo apresenta uma dificuldade adicional no processo de definição dos parâmetros do AG.

## 1.2 Motivações

A simulação de processos evolutivos biológicos permite a compreensão de como a evolução guia os seres vivos na direção de um maior nível de inteligência (FOGEL; ANGELINE; FOGEL, 1995). O processo de otimização, inerente à seleção dos elementos mais bem adaptados, melhora a qualidade média das soluções obtidas ao longo das gerações. Entretanto, um

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, denominado explotação. Esta integração resulta em uma capacidade de adaptação da CE a diferentes espaços de busca (GOLDBERG, 1989), o que permite a aplicação de CE em vários problemas práticos de busca e otimização (como por exemplo o problema do caixeiro viajante (ROSENKRANTZ; STEARNS; II, 1977), ajuste de parâmetros de controladores (GUROCAK, 1999; FABRO; NEVES JR.; ARRUDA, 2005; NAGAI; ARRUDA, 2005)) nos quais outras estratégias de solução podem ser menos adequadas.

O uso de sistemas nebulosos é adequado em muitas aplicações práticas nas quais há a necessidade de se modelar o conhecimento, ou pelo menos, tornar o conhecimento obtido interpretável para um operador humano, como suporte a decisão em aplicações médicas, finanças e comércio. Nestes casos, a representação do conhecimento via abordagens convencionais, baseadas na lógica bi-valorada, apresenta uma séria desvantagem: sua inabilidade de lidar com a incerteza e imprecisão. Conseqüentemente, essas abordagens não fornecem um modelo adequado para uma forma de raciocínio semelhante ao raciocínio humano (CORDÓN *et al.*, 2001a). Devido a essa propriedade, os FRBS têm sido utilizados com sucesso em uma grande variedade de problemas em diferentes áreas de estudo nas quais a incerteza e imprecisão se manifestam de diferentes maneiras (BARDOSSY L. DUCKSTEIN, 1995; CHI; YAN; PHAM, 1996; HIROTA, 1994; ALCALÁ *et al.*, 2000; PEDRYCZ, 1996; YAGER, 1992).

Assim, os sistemas híbridos baseados em sistemas nebulosos e algoritmos genéticos parecem candidatos promissores para solucionar o problema do projeto automático de FRBS. O projeto de um FRBS pode ser visto como um problema de busca ou otimização para o qual nenhuma informação adicional sobre o espaço de busca (como o gradiente da função objetivo, por exemplo) está disponível. Deste modo, os algoritmos genéticos podem ser utilizados como mecanismos de busca ampla uma vez que possuem habilidade de explorar espaços de busca grandes exigindo apenas uma medida escalar de desempenho. De forma adicional, a habilidade dos AGs de encontrar soluções próximas do ótimo em espaços de busca complexos e sua estrutura de armazenamento das variáveis do problema através de um genótipo fazem dos AGs uma das metodologias promissoras para buscar uma base de conhecimento em FRBS. Para o problema do projeto de FRBS, esse conhecimento pode estar nas variáveis lingüísticas, parâmetros das funções de pertinência, regras nebulosas, número de regras, entre outras. Essas propriedades motivaram o uso de AGs no desenvolvimento de muitas abordagens de projeto automático de FRBS (CORDÓN *et al.*, 2001a).

Entretanto, embora úteis para o projeto dos parâmetros de FRBS, a definição dos parâ-

metros evolutivos de AGs se torna um problema mal-definido (*ill-defined*), mal-estruturado e complexo (EIBEN; HINTERDING; MICHALEWICZ, 1999). Os valores desses parâmetros irão influir decisivamente se o algoritmo irá encontrar soluções próximas do ótimo e se ele encontrará essas soluções de forma eficiente. A estrutura hierárquica de Delgado, von Zuben e Gomide (2004) torna o problema ainda mais complexo pois utiliza quatro populações com informações diferentes. O projeto dos parâmetros evolutivos de sistemas desse tipo apresenta algumas características:

- Os níveis hierárquicos codificam informações de natureza diferente. Portanto não há evidência que parâmetros adequados para um módulo irão funcionar satisfatoriamente em outros níveis hierárquicos;
- Durante diferentes etapas do processo evolutivo, diferentes parâmetros podem apresentar características mais interessantes. Não existe evidência de que níveis hierárquicos distintos apresentem a mesma velocidade de evolução;
- Os parâmetros de populações diferentes podem interagir de maneira complexa. Portanto, a definição dos parâmetros evolutivos de um nível hierárquico isoladamente pode levar a escolhas sub-ótimas.

O projeto de heurísticas para o controle dos parâmetros evolutivos é um problema complexo (freqüentemente mais complexo que o próprio problema que está sendo resolvido (EIBEN; HINTERDING; MICHALEWICZ, 1999)), além de requerer conhecimento adicional. Não existe nenhuma garantia que heurísticas adequadas para um problema apresentem desempenho satisfatório em outros problemas similares (EIBEN; SCHUT; WILDE, 1998).

O controle de parâmetros torna-se mais complexo em abordagens que utilizam múltiplas populações com informações distintas. Neste caso parâmetros de populações diferentes podem interagir de forma complexa. Nestes casos, conforme apontado por Eiben, Schut e Wilde (1998), Eiben, Hinterding e Michalewicz (1999), um dos paradigmas mais promissores para o controle de parâmetros é o uso de auto-adaptação.

## 1.3 Objetivos

Este trabalho propõe a aplicação de técnicas de auto-adaptação em algoritmos genéticos com o intuito de obter um GFRBS inspirado na estrutura proposta por Delgado (2002), mas com um número reduzido de parâmetros evolutivos a serem definidos pelo usuário. O objetivo

é a obtenção de um GFRBS com bom desempenho e elevado grau de autonomia. Para isso, o sistema de auto-adaptação dos parâmetros evolutivos deverá ser desenvolvido em duas etapas, conforme detalhado a seguir.

### 1.3.1 Algoritmo genético auto-adaptativo

Na primeira etapa, um protótipo de um algoritmo genético com auto-adaptação de múltiplos parâmetros será construído para validar a metodologia proposta. Este protótipo será avaliado através de uma aplicação em problemas de otimização contínua e de otimização combinatória. A principal vantagem deste tipo de algoritmo em relação a um algoritmo genético convencional é a facilidade de uso devido à menor quantidade de parâmetros evolutivos definidos pelo usuário. O desempenho do algoritmo será comparado com um AG utilizando os valores definidos por de Jong (1975) que são reconhecidos como um conjunto de parâmetros "padrão" que apresenta desempenho razoável para um grande número de aplicações (EIBEN; HINTERDING; MICHALEWICZ, 1999).

### 1.3.2 GFRBS auto-adaptativo

Na segunda etapa, a abordagem auto-adaptativa será validada através do problema do projeto automático de sistemas nebulosos. O sistema resultante é baseado no coevol-GFS proposto por Delgado, von Zuben e Gomide (2004), o qual apresenta bom desempenho para uma ampla classe de problemas. Entretanto devido ao uso de múltiplas populações com informações significativamente diferentes, o ajuste dos parâmetros evolutivos do sistema é um problema complexo. O objetivo do uso de uma abordagem auto-adaptativa, na qual o próprio AG se encarrega de selecionar um bom conjunto de parâmetros, é liberar o usuário do processo de obtenção manual dos parâmetros evolutivos mantendo o bom desempenho do coevol-GFS.

## 1.4 Estrutura da Dissertação

Esta dissertação divide-se em seis capítulos.

**Capítulo 1** Discute o problema do projeto automático de sistemas nebulosos e apresenta a motivação para seu tratamento utilizando algoritmos genéticos. Os objetivos da dissertação são definidos e o conteúdo da dissertação é detalhado.

**Capítulo 2** Neste capítulo é apresentada uma discussão inicial sobre sistemas nebulosos, pro-

curando localizá-los dentro do contexto das aplicações a serem abordadas neste trabalho. A seguir, são especificados os principais componentes de um sistema nebuloso, de modo a permitir o entendimento do modelo que está sendo proposto.

**Capítulo 3** Apresenta os conceitos básicos relacionados à teoria dos algoritmos genéticos, a serem utilizados no desenvolvimento da dissertação. A seguir são apresentados os mecanismos de determinação de parâmetros evolutivos, enfatizando métodos de controle auto-adaptativo de parâmetros.

**Capítulo 4** Aqui, são descritos o sistema coevol-GFS que serviu de inspiração para o modelo proposto, as modificações realizadas e o mecanismo de auto-adaptação de parâmetros evolutivos proposto neste trabalho. Finalmente, essas duas técnicas são combinadas para a obtenção de um sistema genético-nebuloso co-evolutivo, com auto-adaptação dos parâmetros evolutivos.

**Capítulo 5** Descreve as abordagens que serão usadas na comparação com o modelo proposto, e apresenta os resultados das simulações realizadas. Este capítulo apresenta ainda uma análise geral dos resultados obtidos durante todo o desenvolvimento do trabalho.

**Capítulo 6** Traz as conclusões sobre os resultados obtidos e também propostas de pesquisas futuras utilizando a metodologia proposta nesta dissertação.

# 2 *Sistemas nebulosos*

Os sistemas nebulosos constituem uma das áreas mais importantes na tecnologia de processamento de informação contemporânea encontrando aplicações em sistemas de controle, reconhecimento de padrões e visão computacional (PEDRYCZ; GOMIDE, 1998).

O conceito de conjuntos nebulosos foi inicialmente desenvolvido por Zadeh (1965) com a idéia de capturar, representar e processar dados associados a noções lingüísticas ou objetos com fronteiras mal-definidas de maneira formal. O resultado foi a formalização de uma plataforma de processamento de informações imprecisas inspirada no conhecimento humano. Desta maneira os sistemas nebulosos representam uma possível ponte entre o processamento lingüístico e o processamento numérico (DUBOIS; PRADE, 1998).

A lógica nebulosa pode ser entendida como uma extensão da lógica binária e da lógica multi-valores. Na lógica binária uma proposição sempre assume valores verdadeiro ou falso. Em uma lógica multi-valores, além disso, o valor da proposição pode assumir alguns valores intermediários. A lógica nebulosa, em contrapartida, assume que os valores-verdade são conjuntos nebulosos permitindo a utilização de um raciocínio aproximado lingüisticamente interpretável.

Um sistema de inferência nebuloso é uma plataforma baseada nos conceitos da teoria dos conjuntos nebulosos, regras do tipo se-então e raciocínio nebuloso. A utilização dessa classe de sistemas representa um marco na tecnologia de processamento de informação pois os sistemas baseados em regras nebulosas apresentam grande habilidade para expressar a ambiguidade e subjetividade inerentes ao raciocínio humano (PEDRYCZ; GOMIDE, 1998). Em alguns trabalhos o termo sistema difuso também é utilizado para designar um sistema nebuloso.

De acordo com a discussão apresentada por Lin e Lee (1996), o uso de sistemas baseados em regras nebulosas é adequado para sistemas com as seguintes características:

- O modelo matemático do sistema não existe, ou se existe é de difícil tratamento;
- O modelo matemático é muito complexo para ser avaliado em tempo real ou requer uma quantidade de memória maior que a disponível;

- As variáveis de entrada do sistema são contínuas;
- O processo envolve a interação com um operador humano;
- As variáveis de entrada estão sujeitas a ruído ou são coletadas por sensores de baixa precisão.

Muitas metodologias de análise de dados pressupõem precisão dos dados e medidas exatas. Na prática isso ocorre somente em raríssimas situações. A interpretação mais correta das medidas deveria ser como intervalos cujo comprimento depende da precisão dos dispositivos de medição (BERTHOLD, 1999).

Na seção 2.1 será apresentado o conceito de conjuntos nebulosos. Serão abordados tópicos como funções de pertinência e operadores nebulosos. Na seção 2.2 será introduzido o conceito de relações e regras nebulosas além do mecanismo clássico de raciocínio nebuloso. O conceito de regras nebulosas será essencial para a seção 2.3 onde serão brevemente descritos os tipos de sistemas nebulosos. A seção 2.4 apresenta as considerações associadas ao projeto de sistemas nebulosos.

## 2.1 Conjuntos nebulosos

Os conjuntos clássicos encontram aplicações na matemática e na ciência da computação. Entretanto, este tipo de conjunto não é adequado para tratar a natureza do pensamento humano que tente a ser imprecisa e abstrata (JANG; SUN; MIZUTANI, 1997).

O conjunto $\mathbb{X}$ é denominado universo de discurso da variável $x$ ou simplesmente universo, e pode consistir de objetos discretos (ordenados ou não) ou um espaço contínuo.

Seja $x$ uma variável, um conjunto nebuloso é definido como:

$$A = (x, \mu_A(x) \mid x \in \mathbb{X}) \tag{2.1}$$

onde $\mu_A(x)$ é uma função de pertinência que indica o grau de compatibilidade de um objeto $x$ ao conjunto nebuloso $A$. Assim $\mu_A(x) = 1$ indica que o elemento pertence totalmente ao conjunto e $\mu_A(x) = 0$ indica que o elemento não pertence ao conjunto $A$. Define-se uma função de pertinência como um mapeamento da forma:

$$\mu_A : \mathbb{X} \mapsto [0, 1] \tag{2.2}$$

Um conjunto clássico (*crisp set*) é um tipo específico de conjunto nebuloso no qual um valor

ou objeto pertence completamente ao conjunto ou é totalmente incompatível a ele. A função de pertinência de um conjunto clássico só pode ser mapeada para os valores $\{0, 1\}$, ou seja, sua função característica (JANG; SUN; MIZUTANI, 1997; PEDRYCZ; GOMIDE, 1998).

Aos conjuntos nebulosos estão associados, tipicamente, termos lingüísticos como por exemplo: "baixo", "médio" e "alto". Esses conjuntos são caracterizados pelas funções de pertinência $\mu_{baixo}$, $\mu_{médio}$ e $\mu_{alto}$ respectivamente. Para um valor de entrada específico, se o valor de $\mu_{baixo} > 0$ então a expressão "$x$ é baixo" é verdadeira com um grau de pertinência $\mu_{baixo}$.

### 2.1.1 Definições

#### Suporte

O suporte de um conjunto nebuloso A é o conjunto de todos os pontos em $\mathbf{X}$ tais que $\mu_A(x) > 0$, onde:

$$suporte(A) = \{x \mid \mu_A(x) > 0\} \tag{2.3}$$

#### Núcleo

O núcleo de um conjunto nebuloso A é o conjunto de todos os pontos em $\mathbf{X}$ em que $\mu_A(x) = 1$:

$$núcleo(A) = \{x \mid \mu_A(x) = 1\} \tag{2.4}$$

#### Normalidade

Um conjunto nebuloso A é normal se seu núcleo for não-vazio. Ou seja, $\exists x \mid \mu_A(x) = 1$.

#### Pontos de corte

Um ponto de corte de um conjunto nebuloso A é um ponto $x \in \mathbf{X}$ definido como:

$$corte(A) = \{x \mid \mu_A(x) = 0.5\} \tag{2.5}$$

#### $\alpha$-cortes

Um $\alpha$-corte de um conjunto nebuloso A é um conjunto clássico definido por:

$$A_\alpha = \{x \mid \mu_A(x) \geq \alpha\} \tag{2.6}$$

De forma semelhante pode-se definir um $\alpha$-corte forte:

$$A'_\alpha = \{x \mid \mu_A(x) > \alpha\} \tag{2.7}$$

Utilizando esta notação, pode-se expressar o suporte e o núcleo de um conjunto nebuloso como:

$$suporte(A) = A'_0 \tag{2.8}$$

e

$$núcleo(A) = A_1 \tag{2.9}$$

respectivamente.

**Convexidade**

Um conjunto nebuloso $A$ é convexo se e somente se para quaisquer pontos $x_1, x_2 \in \mathbb{X}$ e $\varpi \in [0,1]$,

$$\mu_A(\varpi x_1 + (1-\varpi)x_2) \geq \min\{\mu_A(x_1), \mu_A(x_2)\}. \tag{2.10}$$

Conseqüentemente, $A$ é convexo se todos seus $\alpha$-cortes forem convexos.

### 2.1.2 Simetria

Um conjunto nebuloso é simétrico se sua função de pertinência for simétrica ao redor de um certo ponto $c \in \mathbb{X}$, ou seja:

$$\mu_A(c+x) = \mu_A(c-x), \quad \forall x \in \mathbb{X} \tag{2.11}$$

**Subconjunto**

Um conjunto nebuloso A está contido em um conjunto nebuloso B se e somente se $\mu_A(x) \leq \mu_B(x), \forall x \in \mathbb{X}$. Ou de forma equivalente pode-se dizer que A é um subconjunto de B. Formalmente:

$$A \subseteq B \Longleftrightarrow \mu_a(x) \leq \mu_B(x), \quad \forall x \in \mathbb{X} \tag{2.12}$$

### 2.1.3 Formatos de funções de pertinência

Em geral, o formato das funções de pertinência é restrito a uma certa classe de funções pseudo-trapezoidais, que podem ser caracterizadas por um conjunto de parâmetros específicos. Os formatos mais comuns são: triangular, trapezoidal e Gaussiana (DELGADO, 2002). Além desses formatos tradicionais existe uma forma amplamente utilizada em aplicações práticas: o conjunto unitário (*singleton*).

**Função triangular** : A função triangular apresenta três parâmetros $\{a,b,c\}$ e pode ser escrita como:

$$\mu_{triangular}(x;a,b,c) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a < x \leq b \\ \frac{c-x}{c-b}, & b < x \leq c \\ 0, & x > c \end{cases} \tag{2.13}$$

Alternativamente, pode-se reescrever a formulação 2.13 como:

$$\mu_{triangular}(x;a,b,c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \tag{2.14}$$

onde $b$ é o valor modal de $\mu$ e $a$ e $b$ representam os limites inferior e superior dos elementos compatíveis com o conjunto nebuloso respectivamente. Graficamente, pode-se interpretar o conjunto $\{a,b,c\}, a \leq b \leq c$ como as coordenadas dos três vértices do triângulo formado pela função de pertinência.

**Função trapezoidal** : A função trapezoidal é especificada por um conjunto de quatro parâmetros: $\{a,b,c,d\}$, conforme a formulação 2.15:

$$\mu_{trapezoidal}(x;a,b,c,d) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a < x \leq b \\ 1, & b < x \leq c \\ \frac{d-x}{d-c}, & c < x \leq d \\ 0, & x > d \end{cases} \tag{2.15}$$

alternativamente, pode-se reescrever a formulação 2.15 como:

$$\mu_{trapezoidal}(x;a,b,c,d) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right) \tag{2.16}$$

Graficamente, pode-se interpretar o conjunto $\{a,b,c,d\}, a \leq b \leq c \leq d$ como as coordenadas dos quatro vértices do trapézio formado pela função de pertinência. Para o caso especial em que $b = c$ a função trapezoidal se reduz a uma função triangular. Para o caso especial $a = b$ e $c = d$ tem-se um conjunto clássico.

**Função Gaussiana** : Uma função de pertinência Gaussiana é especificada por dois parâmetros, a média ($\omega$) e a abertura ($\sigma$), conforme a equação 2.17:

$$\mu_{Gaussiana}(x;\omega,\sigma) = e^{-\frac{1}{2}(\frac{x-\omega}{\sigma})^2} \tag{2.17}$$

A função Gaussiana é simétrica ao redor de $\omega$ e sua abertura é especificada pelo parâmetro $\sigma$. Podemos também afirmar que o suporte de $Gaussiana(x,\omega,\sigma)$ é infinito.

**Função *singleton*** Uma função singleton é uma função delta de Kronecker (PROAKIS; MANOLAKIS, 1996) com altura controlada pelo parâmetro $h \in \mathbb{R} \mid h < 1$. Ela pode ser especificada pelo conjunto de parâmetros $\{h,\iota\}$ conforme a formulação 2.18

$$\mu_{singleton}(x;h,\iota) = \begin{cases} h, & x = \iota \\ 0, & caso\ contrário \end{cases} \tag{2.18}$$

Em geral o valor de $h = 1$, nesse caso o suporte e o núcleo do conjunto nebuloso serão coincidentes.

**Função de irrelevância** Para algumas regras nebulosas, o valor de uma variável de entrada pode ser irrelevante para a ativação dessas regras. Nesse caso, pode-se considerar que a função de pertinência, para uma agregação do tipo "e", implementada por uma norma-t (KLEMENT; MESIAR; PAP, 2000), associada ao termo lingüístico irrelevante é uma função do tipo:

$$\mu(x) = 1, \quad \forall x \in \mathbb{X} \tag{2.19}$$

Se os operadores nebulosos respeitarem a propriedade de condição de contorno, a agregação de termos lingüísticos irrelevantes não afetará a ativação da regra.

Devido à sua formulação simples e eficiência computacional, as funções do tipo triangular e trapezoidal são amplamente utilizadas, especialmente em implementações com requisitos de funcionamento em tempo real (JANG; SUN; MIZUTANI, 1997). Contudo, o fato dessas funções de pertinência serem constituídas por segmentos de reta faz com que a derivada de primeira ordem da função não seja contínua. Isto é importante se for utilizado algum método de otimização baseado no gradiente. A ausência de continuidade nas transições entre segmentos de reta

pode acarretar na divergência do método. Mesmo em métodos que independem da diferenciabilidade das funções de pertinência, o uso de funções não lineares pode ser recomendável por permitir transições mais suaves no valor de pertinência. Com base nessa observação, funções do tipo Gaussiana, além das triangulares e trapezoidais, foram utilizadas neste trabalho.

A especificação das funções de pertinência também pode ser realizada utilizando a notação alternativa (JANG; SUN; MIZUTANI, 1997):

$$A = \begin{cases} \sum_{\forall x \in \mathbf{X}} \mu_A(x)/x, & \mathbf{X} \text{ é um universo discreto} \\ \int_{\forall x \in \mathbf{X}} \mu_A(x)/x, & \mathbf{X} \text{ é um universo contínuo} \end{cases} \tag{2.20}$$

Neste contexto, os símbolos $\int$ e $\sum$ não devem ser interpretados como os operador de soma e integração e sim como a união contínua e discreta dos pares $(x, \mu_A(x))$, respectivamente. De forma similar, o símbolo / não denota a divisão e é utilizado somente como um marcador. Seja $A$ um conjunto nebuloso indicando o desejo de se morar em uma determinada cidade, é possível expressar esse conjunto, para um determinado objetivo, utilizando a notação anterior:

$$A = 0.2/Paranaguá + 0.9/Atlanta + 1.0/Vancouver + 0/Governador\ Valadares$$

A especificação de um formato para uma função de pertinência nem sempre é óbvia, podendo inclusive não estar ao alcance do conhecimento de um especialista para a aplicação desejada. No entanto existem sistemas nebulosos cujos parâmetros das funções de pertinência são completamente definidos pelo especialista. Nestes casos, a escolha de funções triangulares e trapezoidais é mais comum porque a idéia de definição de regiões de pertinência total, média e nula é mais intuitiva que a especificação de valor modal e dispersão associados ao projeto de funções Gaussianas. Existem ainda algumas classes de métodos experimentais para a determinação de funções de pertinência como a abordagem horizontal, abordagem vertical, comparação em pares, inferência baseada na especificação do problema, estimação paramétrica e agrupamento nebuloso (PEDRYCZ; GOMIDE, 1998). A utilização da classe adequada depende da aplicação, em particular, da forma como a incerteza se manifesta e é observada durante o experimento (PEDRYCZ; GOMIDE, 1998). Entretanto em trabalhos mais recentes existe a tendência ao projeto automático de sistemas nebulosos nos quais os parâmetros das funções de pertinência são ajustados no sentido de otimizar alguma função objetivo. Em geral essa otimização é realizada a partir de um conjunto significativo de dados de treinamento, ou seja, um conjunto de dados de treinamento que descreva adequadamente o comportamento do sistema.

### 2.1.4 Operações com conjuntos nebulosos

As operações de combinação, comparação ou agregação de conjuntos clássicos são de fundamental importância para o processamento de informações. De forma semelhante, as operações de união, interseção e negação possuem operações similares quando os operandos são conjuntos nebulosos.

#### Negação nebulosa

Uma negação nebulosa é uma função contínua $C : [0,1] \mapsto [0,1]$ que satisfaz os seguintes requisitos axiomáticos:

1. Condições de contorno: $\begin{cases} C(0) = 1 \\ C(1) = 0 \end{cases}$
2. Monotonicidade: $C(a) \geq C(b)$, se $a \leq b$.

Todas as funções que satisfaçam esses requisitos formam a classe geral de negações nebulosas. O primeiro requisito garante que as negações nebulosas podem ser utilizadas para conjuntos clássicos. A segunda condição garante que um aumento do grau de pertinência de um conjunto indica uma diminuição da pertinência de sua negação. Um requisito opcional, que não costuma ser satisfeito pela maioria das negações nebulosas, é a condição de involução da negação nebulosa: $C(C(a)) = a$ que garante que a negação dupla do conjunto é o próprio conjunto.

Os tipos mais comuns de negação nebulosa encontrados na literatura são:

- Complemento: $C_1(a) = 1 - a$
- Sugeno: $C_2(a,s) = \frac{1-a}{1+s.a}$, $s > -1$
- Yeager: $C_3(a,w) = (1 - a^w)^{\frac{1}{w}}$, $w > 0$

Podemos observar que $C_2(a,0) = C_1(a)$ e também $C_3(a,1) = C_1(a)$.

#### Interseção

De forma geral, a operação de interseção entre conjuntos nebulosos é realizada através de uma norma triangular ou norma-t. Ao contrário da interseção clássica, uma norma-t é capaz de agregar conjuntos em universos de discurso diferentes. Uma norma-t é uma operação definida como $\mathbf{t} : [0,1] \times [0,1] \mapsto [0,1]$ que satisfaz as seguintes propriedades:

1. Condições de contorno: $\begin{cases} 0 \ \mathbf{t} \ x = 0 \\ 1 \ \mathbf{t} \ x = x \end{cases}$
2. Monotonicidade: Se $x \leq y$ e $w \leq z$, então $x \ \mathbf{t} \ w \leq y \ \mathbf{t} \ z$
3. Comutatividade: $x \ \mathbf{t} \ y = y \ \mathbf{t} \ x$
4. Associatividade: $x \ \mathbf{t} \ (y \ \mathbf{t} \ z) = (x \ \mathbf{t} \ y) \ \mathbf{t} \ z$

O primeiro critério indica que as t-normas podem ser generalizadas para conjuntos clássicos. O segundo requisito implica que um decréscimo nos valores de pertinência a A ou B não pode produzir um aumento no valor de pertinência $A \ \mathbf{t} \ B$. O terceiro critério indica que o operador independe da ordem dos conjuntos nebulosos combinados. O quarto critério generaliza o operador para qualquer número de conjuntos nebulosos.

Existem alguns tipos de normas triangulares mais utilizados na literatura:

- mínimo: $a \ \mathbf{t_1} \ b = \min\{a, b\}$
- produto algébrico: $a \ \mathbf{t_2} \ b = a.b$
- produto restrito: $a \ \mathbf{t_3} \ b = \max\{0, a + b - 1\}$
- produto drástico: $a \ \mathbf{t_4} \ b = \begin{cases} a, & se \ b = 1 \\ b, & se \ a = 1 \\ 0, & se \ a, b < 1 \end{cases}$
- Schweizer e Sklar: $a \ \mathbf{t_5} \ b = [max\{0, (a^{-p} + b^{-p} - 1)\}]^{-\frac{1}{p}}, \ p > 0$
- Yeager: $a \ \mathbf{t_6} \ b = 1 - \min\{1, [(1-a)^q + (1-b)^q]^{\frac{1}{q}}\}, \ q > 0$
- Dubois e Prade: $a \ \mathbf{t_7} \ b = \frac{a.b}{\max\{a,b,\alpha\}}, \ \alpha \in [0, 1]$
- Hamacher: $a \ \mathbf{t_8} \ b = \frac{a.b}{\gamma + (1-\gamma)(a+b-a.b)}, \ \gamma > 0$
- Frank: $a \ \mathbf{t_9} \ b = \log_s[1 + \frac{(s^a - 1)(s^b - 1)}{s-1}], \ s > 0$
- Sugeno: $a \ \mathbf{t_{10}} \ b = \max\{0, (\lambda + 1)(a + b - 1) - \lambda.a.b\}, \ \lambda \geq -1$
- Dombi: $a \ \mathbf{t_{11}} \ b = \frac{1}{1 + \left[(a^{-1} - 1)^\lambda + (b^{-1} - 1)^\lambda\right]^{\frac{1}{\lambda}}}, \ \lambda > 0$

Observa-se ainda que:

$$\lim_{p\to 0} a \ \mathbf{t_5}\ b = a \ \mathbf{t_2}\ b \tag{2.21}$$

$$\lim_{p\to \infty} a \ \mathbf{t_5}\ b = a \ \mathbf{t_1}\ b \tag{2.22}$$

ou seja, é possível obter uma norma triangular a partir de outra.

Em geral as normas-t não satisfazem as demais propriedades de interseção dos conjuntos clássicos. Por exemplo a lei da contradição pode ser aplicada somente com combinações específicas de operadores de interseção e complemento. Portanto, dados a norma $\mathbf{t_1}$ e o complemento $C_1$, para um valor de pertinência $\mu_A = 0.5$ tem-se:

$$A \cap \bar{A} = \mu_A \ \mathbf{t_1}\ C_1(\mu_A) = \min\{\mu_A, 1-\mu_A\} = 0.5 \neq 0$$

Além disso, o princípio da idempotência é aplicável somente para $\mathbf{t_1}$. Portanto, tem-se:

$$\bigcap_{i=1}^{n} A \leq \bigcap_{i=1}^{m-1} A \tag{2.23}$$

Propriedades adicionais como continuidade e diferenciabilidade de normas-t podem ser importantes na otimização de sistemas nebulosos através de métodos baseados no gradiente.

### União

Analogamente, existem operadores específicos para realizar a união de conjuntos nebulosos. A operação de união entre conjuntos nebulosos é realizada através de uma co-norma triangular (co-norma-t) ou norma-s. Uma norma-s é uma operação binária $\mathbf{s} : [0,1]\times[0,1] \mapsto [0,1]$ que satisfaz as seguintes propriedades:

1. Condições de contorno: $\begin{cases} 0 \ \mathbf{s}\ x = x \\ 1 \ \mathbf{s}\ x = 1 \end{cases}$
2. Monotonicidade: Se $x \leq y$ e $w \leq z$, então $x \ \mathbf{s}\ w \leq y \ \mathbf{s}\ z$
3. Comutatividade: $x \ \mathbf{s}\ y = y \ \mathbf{s}\ x$
4. Associatividade: $x \ \mathbf{s}\ (y \ \mathbf{s}\ z) = (x \ \mathbf{s}\ y) \ \mathbf{s}\ z$

O primeiro critério implica que as s-normas podem ser generalizadas para conjuntos clássicos. O segundo requisito indica que um decréscimo nos valores de pertinência a A ou B

não podem produzir um aumento no valor de pertinência de $A \bigcup B$. O terceiro critério indica que o operador independe da ordem dos conjuntos nebulosos combinados. O quarto critério generaliza o operador para qualquer número de conjuntos nebulosos.

Da mesma forma que as normas-t, existem alguns tipos de co-normas triangulares mais utilizadas na literatura:

- máximo: $a\ \mathbf{s_1}\ b = \min\{a, b\}$
- soma algébrica: $a\ \mathbf{s_2}\ b = a + b - a.b$
- soma restrita: $a\ \mathbf{s_3}\ b = \max\{1, a + b\}$
- soma drástica: $a\ \mathbf{s_4}\ b = \begin{cases} a, & se\ b = 0 \\ b, & se\ a = 0 \\ 1, & se\ a, b > 0 \end{cases}$
- Schweizer e Sklar: $a\ \mathbf{s_5}\ b = 1 - [max\{0, ((1-a)^{-p} + (1-b)^{-p} - 1)\}]^{-\frac{1}{p}},\ p > 0$
- Yeager: $a\ \mathbf{s_6}\ b = \min\{1, [a^q + b^q]^{\frac{1}{q}}\},\ q > 0$
- Dubois e Prade:$a\ \mathbf{s_7}\ b = [a + b - a.b - \frac{\min\{a,b,(1-\alpha)\}}{\max\{1-a,1-b,\alpha\}},\ \alpha \in [0, 1]$
- Hamacher: $a\ \mathbf{s_8}\ b = \frac{a+b-(\gamma-2)a.b}{1+(\gamma-1)a.b},\ \gamma > 0$
- Frank: $a\ \mathbf{s_9}\ b = 1 - \log_s[1 + \frac{(s^{1-a}-1)(s^{1-b}-1)}{s-1}],\ s > 0$
- Sugeno: $a\ \mathbf{s_{10}}\ b = \min\{1, a + b - \gamma.a.b\},\ \lambda \geq -1$
- Dombi: $a\ \mathbf{s_{11}}\ b = \frac{1}{1+\left[(a^{-1}-1)^{-\lambda}+(b^{-1}-1)^{-\lambda}\right]^{\frac{-1}{\lambda}}},\ \lambda > 0$

Em geral as normas-s não satisfazem às demais propriedades de união dos conjuntos clássicos. Por exemplo a lei da exclusão do meio pode ser aplicada somente com combinações específicas de operadores de união e complemento. Por exemplo, dados a norma $\mathbf{s_1}$ e o complemento $C_1$, para um valor de pertinência $\mu_A = 0.5$ tem-se:

$$A \cup \bar{A} = \mu_A\ \mathbf{s_1}\ C_1(\mu_A) = \max\{\mu_A, 1 - \mu_A\} = 0.5 \neq 1$$

Além disso, o princípio da idempotência é aplicável somente para $\mathbf{s_1}$. Portanto tem-se:

$$\bigcup_{i=1}^{n} A \geq \bigcup_{i=1}^{n-1} A \tag{2.24}$$

Propriedades adicionais como continuidade e diferenciabilidade de normas-s podem ser importantes na otimização de sistemas nebulosos através de métodos baseados no gradiente.

**Outros operadores**

Além dos operadores apresentados nesse capítulo, existem alguns operadores de agregação específicos como os operadores compensatórios propostos por Zimmermann e Zysno (1980), operadores de soma simétrica de Dubois e Prade (1980), operador de média generalizada de Dyckhoff e Pedrycz (1984) e o operador de média ponderada ordenada, proposto por Yager (1988).

Além da agregação, algumas operações com conjuntos nebulosos podem resultar em medidas de distância, igualdade, possibilidade, necessidade e compatibilidade (DUBOIS; PRADE, 1980; KLEMENT; MESIAR; PAP, 2000; PEDRYCZ; GOMIDE, 1998). A possibilidade e a necessidade apresentam algumas interpretações:

**possibilidade** indica o grau de sobreposição entre dois conjuntos nebulosos:

$$\mu_{Poss(A,B)} = \sup_{x \in \mathbf{X}} \min(\mu_A(x), \mu_B(x)) \tag{2.25}$$

**necessidade** indica o grau de inclusão de um conjunto no complemento de outro:

$$\mu_{Nec(A,B)} = \inf_{x \in \mathbf{X}} \max(\mu_A(x), 1 - \mu_B(x)) \tag{2.26}$$

## 2.2 Regras nebulosas e raciocínio nebuloso

As regras nebulosas e raciocínio nebuloso são os principais componentes dos sistemas de inferência nebulosos. Estes são uma das ferramentas importantes para a modelagem baseada na teoria dos conjuntos nebulosos (JANG; SUN; MIZUTANI, 1997), como será visto na próxima seção.

### 2.2.1 Relações nebulosas

As relações nebulosas binárias são conjuntos nebulosos definidos em $\mathbf{X}_1 \times \mathbf{X}_2$ que mapeiam cada elemento de $\mathbf{X}_1 \times \mathbf{X}_2$ em um grau de pertinência $\mu_{\mathcal{R}}(x_1, x_2)$ ao conjunto $\mathcal{R}$.

As relações com conjuntos clássicos são casos específicos de relações nebulosas. É possível capturar as dependências entre variáveis sem a fixação de uma caracterização direcional parti-

cular. Ou seja, não existe domínio e contra-domínio (PEDRYCZ; GOMIDE, 1998). Aplicações de relações nebulosas incluem áreas como controle nebuloso e tomada de decisões. O mesmo raciocínio pode ser utilizado para produzir relações nebulosas $n$-árias.

**Relação nebulosa binária**

Sejam $\mathbb{X}_1$ e $\mathbb{X}_2$ dois universos de discurso. Pode-se definir o conjunto nebuloso $\mathcal{R}$ como:

$$\mathcal{R} : \mathbb{X}_1 \times \mathbb{X}_2 \mapsto [0,1] \tag{2.27}$$

$$\mathcal{R} = \{((x,y), \mu_{\mathcal{R}}(x,y)) \mid (x,y) \in \mathbb{X}_1 \times \mathbb{X}_2\} \tag{2.28}$$

é uma relação nebulosa binária no universo $\mathbb{X}_1 \times \mathbb{X}_2$. Por exemplo, seja $\mathbb{X}_1 \equiv \mathbb{X}_2 \equiv \mathbb{R}^+$ e $\mathcal{R}$ = "$x_2$ é muito maior que $x_1$". Supondo que a função de pertinência de $\mathcal{R}$ possa ser aproximada por:

$$\mu_{\mathcal{R}}(x,y) = \begin{cases} \frac{x_2 - x_1}{x_1 + x_2 + 2}, & se\ x_2 > x_1 \\ 0, & se\ x_2 \leq x_1 \end{cases} \tag{2.29}$$

Para os universos de discurso $\mathbb{X}_1 = \{3,4,5\}$ e $\mathbb{X}_2 = \{3,4,5,6,7\}$, pode-se expressar a mesma relação nebulosa $\mathcal{R}$ através de uma matriz de relação:

$$\mathcal{R} = \begin{bmatrix} 0 & 0.111 & 0.200 & 0.273 & 0.333 \\ 0 & 0 & 0.091 & 0.167 & 0.231 \\ 0 & 0 & 0 & 0.077 & 0.143 \end{bmatrix} \tag{2.30}$$

onde o elemento na linha $i$ e coluna $j$ indica o grau de pertinência ao conjunto muito maior entre o elemento $i$ de $\mathbb{X}_1$ e o elemento $j$ de $\mathbb{X}_2$.

Outros exemplos de relações nebulosas binárias foram levantados por Jang, Sun e Mizutani (1997):

- $x_1$ está próximo de $x_2$, onde $x_1$ e $x_2$ são números;
- $x_1$ depende de $x_2$, onde $x_1$ e $x_2$ são eventos;
- $x_1$ e $x_2$ são semelhantes, onde $x_1$ e $x_2$ são pessoas ou objetos;
- se $x_1$ é grande então $x_2$ é pequeno, onde $x_1$ é uma observação e $x_2$ é a ação correspondente.

Expressões do tipo "se $x_1$ é $A$, então $y$ é $B$" são utilizadas com freqüência em sistemas de inferência nebulosos e são usualmente denominadas regras nebulosas.

**Relações nebulosas multidimensionais**

As relações nebulosas multidimensionais, também conhecidas como relações nebulosas $n$-árias, podem ser obtidas por generalização das relações nebulosas bidimensionais. Seja a relação nebulosa $\mathcal{R}$:

$$\mathcal{R} = \mathbb{X}_1 \times \ldots \times \mathbb{X}_n \mapsto [0,1] \tag{2.31}$$

analogamente à definição de relações nebulosas binárias o grau de pertinência de um vetor de entrada $\mathbf{x} = [x_1 \cdots x_n]$ a $\mathcal{R}$ será dado por $\mu_{\mathcal{R}}(\mathbf{x})$. Formalmente

$$\mathcal{R} = \{(\mathbf{x}, \mu_{\mathcal{R}}(\mathbf{x})) \mid \mathbf{x} \in \mathbb{X}_1 \times \ldots \times \mathbb{X}_n\} \tag{2.32}$$

### 2.2.2 Operações com relações nebulosas

As relações nebulosas definidas em intervalos de entrada idênticos podem ser combinadas através de operações de união, interseção e complemento. Conforme definido anteriormente, relações nebulosas são conjuntos nebulosos e portanto podem ser combinadas através dos operadores discutidos na seção 2.1. Sejam:

$$\begin{aligned} \mathcal{R}_1 &= \left\{ \left( (x_1, x_2), \mu_{\mathcal{R}_1}(x_1, x_2) \right) \middle| (x_1, x_2) \in \mathbb{X}_1 \times \mathbb{X}_2 \right\} \\ \mathcal{R}_2 &= \left\{ \left( (x_1, x_2), \mu_{\mathcal{R}_2}(x_1, x_2) \right) \middle| (x_1, x_2) \in \mathbb{X}_1 \times \mathbb{X}_2 \right\} \end{aligned} \tag{2.33}$$

As operações de união, interseção e complemento de $\mathcal{R}_1$ e $\mathcal{R}_2$ são dadas por:

**União** $\mu_{\mathcal{R}_1 \cup \mathcal{R}_2}(x_1, x_2) = \mu_{\mathcal{R}_1}(x_1, x_2)\ \mathbf{s}\ \mu_{\mathcal{R}_2}(x_1, x_2)$

**Interseção** $\mu_{\mathcal{R}_1 \cap \mathcal{R}_2}(x_1, x_2) = \mu_{\mathcal{R}_1}(x_1, x_2)\ \mathbf{t}\ \mu_{\mathcal{R}_2}(x_1, x_2)$

**Complemento** $\mu_{\overline{\mathcal{R}_1}}(x_1, x_2) = C\left(\mu_{\mathcal{R}_1}(x_1, x_2)\right)$

### 2.2.3 Composição de relações nebulosas

As relações nebulosas definidas em produtos cartesianos distintos podem ser combinadas utilizando operadores de composição. Existem diversos operadores de composição possíveis sendo que o mais conhecido é a composição max min proposta por Zadeh (1965).

**Composição max min**

Sejam $\mathcal{R}_1$ e $\mathcal{R}_2$ duas relações nebulosas definidas em $\mathbf{X}_1 \times \mathbf{X}_2$ e $\mathbf{X}_2 \times \mathbf{X}_3$ respectivamente. A composição max min de $\mathcal{R}_1$ e $\mathcal{R}_2$ é um conjunto nebuloso definido por:

$$\mathcal{R}_1 \circ \mathcal{R}_2 = \left\{ \left[ (x_1, x_3), \mu_{\mathcal{R}_1 \circ \mathcal{R}_2}(x_1, x_3) \right] \middle| \, x_1 \in \mathbf{X}_1, x_2 \in \mathbf{X}_2, x_3 \in \mathbf{X}_3 \right\} \tag{2.34}$$

onde:

$$\mu_{\mathcal{R}_1 \circ \mathcal{R}_2}(x_1, x_3) = \max_{x_2 \in \mathbf{X}_2} \min(\mu_{\mathcal{R}_1}(x_1, x_2), \mu_{\mathcal{R}_2}(x_2, x_3)) \tag{2.35}$$

Quando as relações $\mathcal{R}_1$ e $\mathcal{R}_2$ são expressas como matrizes de relação, o cálculo de $\mathcal{R}_1 \circ \mathcal{R}_2$ é semelhante a uma multiplicação matricial, exceto pelo fato que os operadores de multiplicação e soma são substituídos por min e max respectivamente. Apesar da composição max min ser amplamente utilizada, ela dificulta a associação de métodos da programação matemática na definição dos parâmetros ótimos. A composição max prod foi proposta como uma alternativa à composição max min (JANG; SUN; MIZUTANI, 1997), entre outras coisas, para alcançar maior tratabilidade matemática.

**Composição max prod**

Assumindo a mesma notação utilizada na definição da composição max min, pode-se definir a composição max prod como:

$$\mu_{\mathcal{R}_1 \circ \mathcal{R}_2}(x_1, x_3) = \max_{x_2 \in \mathbf{X}_2} \mu_{\mathcal{R}_1}(x_1, x_2).\mu_{\mathcal{R}_2}(x_2, x_3) \tag{2.36}$$

**Composição genérica**

Analogamente pode-se utilizar o mesmo raciocínio para definir uma composição através de quaisquer pares de normas-s e normas-t (NGUYEN; GANESH; GONG, 1998).

$$\mu_{\mathcal{R}_1 \odot \mathcal{R}_2}(x_1, x_3) = \mathbf{s}_{x_2 \in \mathbf{X}_2} \mu_{\mathcal{R}_1}(x_1, x_2) \mathbf{t} \mu_{\mathcal{R}_2}(x_2, x_3) \tag{2.37}$$

ou ainda:

$$\mu_{\mathcal{R}_1 \odot \mathcal{R}_2}(x_1, x_3) = \mathbf{t}_{x_2 \in \mathbf{X}_2} \mu_{\mathcal{R}_1}(x_1, x_2) \mathbf{s} \mu_{\mathcal{R}_2}(x_2, x_3) \tag{2.38}$$

### Variáveis lingüísticas

Conforme apontado por Zadeh, técnicas convencionais para análise de sistemas são essencialmente inadequadas para o tratamento de sistemas baseados no conhecimento humano, cujo comportamento é influenciado pela percepção, julgamento e emoções. Essa é uma manifestação do princípio da incompatibilidade enunciado por Zadeh: "Com o aumento da complexidade do sistema, nossa habilidade de realizar indicações precisas e significativas sobre seu funcionamento diminui até que é alcançado um nível em que precisão e significado se tornam características quase mutuamente exclusivas" (ZADEH, 1973). Isso motivou Zadeh a propor o conceito de variáveis lingüísticas como uma alternativa na modelagem do pensamento humano em que a informação é processada através de conjuntos nebulosos.

Formalmente uma variável lingüística é caracterizada por uma quíntupla:

$$\langle \mathcal{X}, \mathcal{T}(\mathcal{X}), \mathbf{X}, \mathcal{G}, \mathcal{M} \rangle \tag{2.39}$$

onde:

- $\mathcal{X}$ é o nome da variável lingüística;
- $\mathcal{T}(\mathcal{X})$ é o conjunto de termos lingüísticos. Cada elemento de $\mathcal{T}(\mathcal{X})$ representa um rótulo dos termos que a variável $\mathcal{X}$ pode assumir;
- $\mathbf{X}$ é o universo de discurso da variável lingüística $\mathcal{X}$;
- $\mathcal{G}$ é a gramática para geração dos rótulos;
- $\mathcal{M}$ é a regra que associa cada rótulo a um conjunto nebuloso no universo $\mathbf{X}$, representando o seu significado.

Por exemplo, seja a variável lingüística velocidade ($\mathcal{X}$ = *velocidade*) no universo de discurso $\mathbf{X} = \{x \in \mathbf{X} \mid 0 \leq x \leq 150\}$ e uma variável base $x \in \mathbf{X}$. Um conjunto de termos lingüísticos associados à variável velocidade poderia ser: $\mathcal{T}(\textit{velocidade})$ = {“muito-baixa”, “baixa”, “média”, “alta”, “muito-alta”}. Seja $\mathcal{P}(\mathbf{X})$ a coleção de todas as combinações possíveis de conjuntos nebulosos definidos no universo $\mathbf{X}$. Pode-se definir a relação:

$$\mathcal{M} : \mathcal{T}(\mathcal{X}) \mapsto \mathcal{P}(\mathbf{X}) \tag{2.40}$$

que estabelece uma relação biunívoca entre o conjunto de termos $\mathcal{T}(\mathcal{X})$ a uma partição nebulosa $\mathcal{P}(\mathbf{X})$.

A gramática $\mathcal{G}$ define como os termos lingüísticos primários {"baixa", "média", "alta"} serão associados aos modificadores {"muito", "pouco", "maior", "menor", "ou", "não"} para formar os nomes dos termos não-primários. Usualmente as funções de pertinência associadas às variáveis lingüísticas primárias têm formatos conhecidos. Os conjuntos nebulosos associados aos termos não primários, em contrapartida podem ser obtidos através de modificadores pré-especificados (PEDRYCZ; GOMIDE, 1998).

O conjunto $\mathcal{P}(\mathbf{X})$ define a partição nebulosa do universo de discurso $\mathbf{X}$. Uma partição nebulosa pode ser uniforme (os termos lingüísticos têm funções de pertinência de formato semelhante transladadas e eqüidistantes) ou não-uniforme. A granularidade da partição é definida pelo número de termos lingüísticos. Um número baixo de termos lingüísticos define uma partição esparsa ou grossa, enquanto um número alto resulta em uma partição fina (DELGADO, 2002).

A partição do universo pode ser vista também como uma forma de compressão nebulosa de dados. Utilizando o agrupamento nebuloso de informações de natureza similar (*fuzzy clusters*), pode-se desprezar parte da informação inútil, indesejada ou redundante. Assim, a granularização da partição pode ser usada para direcionar a análise nos aspectos de interesse permitindo maior ênfase em áreas específicas dos universos das variáveis de entrada (PEDRYCZ; GOMIDE, 1998).

Não existe uma metodologia consistente para a determinação da partição nebulosa ideal. Em geral essa tarefa é realizada manualmente através da intervenção de um especialista ou utilizando um método de particionamento automático a partir de dados de treinamento. Na ausência de um especialista ou de abordagens de ajuste automático, partições utilizando de 5 a 7 termos lingüísticos uniformemente distribuídos são utilizadas com freqüência (PEDRYCZ; GOMIDE, 1998).

### 2.2.4 Regras nebulosas

As regras nebulosas são também conhecidas como implicações nebulosas ou declarações condicionais nebulosas. Elas são apropriadas quando o conhecimento do problema é resultado de associações empíricas e experiências de um operador humano, ou quando se deseja uma representação lingüística do conhecimento adquirido.

Em geral, as regras nebulosas assumem a forma:

$$\text{Se } x \text{ é } A \text{ então } y \text{ é } B \tag{2.41}$$

onde $A$ e $B$ são rótulos lingüísticos definidos nos universos $\mathbf{X}$ e $\mathbf{Y}$ respectivamente. Freqüentemente *x é A* é denominado de antecedente ou premissa, enquanto *y é B* é denominado de conseqüente ou conclusão (JANG; SUN; MIZUTANI, 1997).

A regra nebulosa da equação 2.41 pode ser abreviada por $A \rightarrow B$ e pode ser definida como uma relação nebulosa $\mathcal{R}$ no produto cartesiano $\mathbf{X} \times \mathbf{Y}$. Seja $\mu_{\mathcal{R}}$ a função de pertinência associada à relação nebulosa $\mathcal{R}$, a semântica da regra nebulosa pode ser especificada através de uma função $f : [0,1] \times [0,1] \mapsto [0,1]$:

$$\mu_{\mathcal{R}}(x,y) = f(\mu_A(x), \mu_B(x)) \tag{2.42}$$

De acordo com Pedrycz e Gomide (1998), as relações nebulosas induzidas por regras nebulosas são derivadas de três classes principais de funções: conjunções nebulosas, disjunções nebulosas e implicações nebulosas sendo as conjunções e implicações as mais comuns. As conjunções e as disjunções podem ser vistas como generalizações duais do produto cartesiano nebuloso através de normas-t e normas-s, enquanto implicações nebulosas representam generalizações das implicações da lógica multi-valores de Lukasiewicz (1970).

**Conjunção nebulosa** : A conjunção nebulosa, representada pela relação $\mathcal{R} : A \mapsto B = A \ \mathbf{t} \ B$ é uma função $f_t : [0,1] \times [0,1] \mapsto [0,1]$ onde $f_t(\mu_A(x), \mu_B(y)) = \mu_A(x) \ \mathbf{t} \ \mu_B(y)$ e $\mathbf{t}$ pode ser qualquer norma t. As funções mais utilizadas são as propostas por Mamdani e Assilian (1999) e Larsen (1980) que utilizam o mínimo e o produto algébrico respectivamente:

**Mamdani** $f_m(\mu_A(x), \mu_B(y)) = \min\{\mu_A(x), \mu_B(y)\}$

**Larsen** $f_p(\mu_A(x), \mu_B(y)) = \mu_A(x).\mu_B(y)$

**Disjunção nebulosa** : A disjunção nebulosa, representada pela relação $\mathcal{R} : A \mapsto B = A \ \mathbf{s} \ B$ é uma função $f_s : [0,1] \times [0,1] \mapsto [0,1]$ onde $f_s(\mu_A(x), \mu_B(y)) = \mu_A(x) \ \mathbf{s} \ \mu_B(y)$ e $\mathbf{s}$ pode ser qualquer norma s.

**Implicação nebulosa** A implicação nebulosa, representada pela relação $R : A \mapsto B = A \Rightarrow B$ é uma função $f_i : [0,1] \times [0,1] \mapsto [0,1]$ que pode ser classificada em duas categorias básicas:

**Implicação-S** : $f_{iS}(\mu_A(x), \mu_B(x)) = \mu_{\overline{A}}(x) \ \mathbf{s} \ \mu_B(y)$. Essa categoria é uma generalização da implicação utilizando lógica de conjuntos convencionais $p \Rightarrow q = \overline{p} \cup q$

**Implicação-R** $f_{iR}(\mu_A(x), \mu_B(y)) = \sup_{c \in [0,1]}\{\mu_A(x) \ \mathbf{t} \ c \leq \mu_B(y)\}, \forall (x,y) \in \mathbf{X} \times \mathbf{Y}$

O exemplo mais conhecido dessa classe é a implicação de Lukasiewicz (PEDRYCZ; GOMIDE, 1998): $f_l(\mu_A(x), \mu_B(y)) = \min(1, 1 - \mu_A(x) + \mu_B(y))$

### 2.2.5 Computação utilizando regras nebulosas

As regras nebulosas são ferramentas eficientes na modelagem de sentenças em linguagem natural (JANG; SUN; MIZUTANI, 1997). Esta modelagem utiliza o conceito de variáveis lingüísticas para gerar regras nebulosas que são mapeadas através de relações nebulosas permitindo a investigação de diferentes esquemas de raciocínio aproximado. Nestes esquemas, procedimentos de inferência baseados no conceito de regra composicional de inferência são utilizados para derivar conclusões, a partir de uma base de dados contendo regras e fatos conhecidos.

#### Regra composicional de inferência

O procedimento conhecido como regra composicional de inferência é uma generalização do processo de avaliação de uma função em um dado ponto. Seja $f(x)$ uma função que associa um valor de entrada $x \in \mathbb{X}$ a um valor de saída $y \in \mathbb{Y}$. Dado $a \in \mathbb{X}$ pode-se facilmente inferir $b = f(a)$. A generalização desse procedimento permite que o valor de entrada da função seja um intervalo e o contradomínio de $f(x)$ seja um intervalo também. Para encontrar o intervalo $\mathbb{B}$ correspondente ao intervalo de entrada $\mathbb{A}$ utiliza-se a extensão cilíndrica de $\mathbb{A}$ e encontra-se sua interseção $K$ com a curva $f(x)$. A projeção de $K$ em $\mathbb{I}$ indica o intervalo $\mathbb{B}$.

**extensão cilíndrica** : A extensão cilíndrica em $\mathbb{X} \times \mathbb{Y}$ de qualquer conjunto nebuloso $A$ em $\mathbb{X}$ é uma relação nebulosa com uma função de pertinência:

$$\mu_{cil(A)}(x, y) \equiv \mu_A(x) \tag{2.43}$$

**projeção** : A projeção em $\mathbb{Y}$ de uma relação nebulosa $\mathcal{R}$ é dada por um conjunto nebuloso com função de pertinência

$$\mu_{proj_Y(\mathcal{R})}(y) \equiv \sup_{x \in X} \mu_{\mathcal{R}}(x, y) \tag{2.44}$$

A extensão cilíndrica e a projeção têm comportamentos complementares, pois afetam as dimensões das relações ou conjuntos nebulosos envolvidos de forma diferente. A extensão cilíndrica eleva a dimensão do argumento expandido, por exemplo de $\mathbb{R}^1$ para $\mathbb{R}^2$ enquanto a projeção realiza uma operação inversa diminuindo a dimensão do argumento (PEDRYCZ; GOMIDE, 1998).

O mecanismo composicional de inferência necessita de objetos compatíveis dimensionalmente em sua fase intermediária como por exemplo a obtenção da interseção $K$ descrita a seguir.

Seja $A$ um conjunto nebuloso com universo de discurso em $\mathbf{X}$ e a relação nebulosa com função de pertinência $\mu_{\mathcal{R}} : \mathbf{X} \times \mathbf{Y} \mapsto [0, 1]$. A operação de interseção entre $A$ e $\mathcal{R}$ não é possível devido a incompatibilidade entre suas dimensões. Para obter a compatibilidade necessária é utilizado o operador de extensão cilíndrica em $A$.

Uma vez definidos esses dois operadores nebulosos é possível obter um conjunto nebuloso $B'$ a partir de um conjunto nebuloso $A'$ através da relação nebulosa $\mathcal{R}$. Este procedimento composicional de inferência foi proposto por Zadeh (1975) e pode ser resumido nos seguintes passos:

1. Obter a extensão cilíndrica de $A'$: $cil(A')$;
2. Encontrar a interseção $K = cil(A') \, \mathbf{t} \, \mathcal{R}$;
3. Projetar $K$ em $Y$: $B' = proj_{y \in \mathbf{Y}}(K)$.

Assim, o conjunto nebuloso $B'$ é obtido da seguinte forma:

$$B' = proj_{x \in \mathbf{X}}(cil(A') \, \mathbf{t} \, \mathcal{R}) \tag{2.45}$$

Considerando-se que a interseção entre conjuntos nebulosos pode ser realizada por qualquer norma triangular e a projeção é definida pelo operador sup. A função de pertinência de $B'$ é calculada por:

$$\begin{aligned} \mu_{B'}(y) \quad = \quad & \sup_{x \in \mathbf{X}} [\mu_{cil(A')}(x, y) \, \mathbf{t} \, \mu_{\mathcal{R}}(x, y)] \\ & \sup_{x \in \mathbf{X}} [\mu_{A'}(x) \, \mathbf{t} \, \mu_{\mathcal{R}}(x, y)] \end{aligned} \tag{2.46}$$

o que resulta numa composição sup$\mathbf{t}$. Portanto,

$$B' = A' \circ \mathcal{R} \tag{2.47}$$

Utilizando a regra de inferência composicional, pode-se formalizar o processo de inferência nebuloso através de um conjunto de regras nebulosas se-então. Este processo de inferência é geralmente denominado raciocínio aproximado ou raciocínio nebuloso.

**Raciocínio nebuloso**

O raciocínio nebuloso, também conhecido como raciocínio aproximado, é um procedimento de inferência que deriva suas conclusões de um conjunto de regras nebulosas e fatos conhecidos.

A regra básica de inferência na lógica binária é o *modus ponens*, segundo a qual pode-se inferir a veracidade de uma proposição $B$ a partir da veracidade de uma outra proposição $A$ e da implicação $A \Rightarrow B$.

$$\begin{array}{lll} & (fato) & X \text{ é } A \\ Modus\ Ponens & (regra) & \text{se } X \text{ é } A \text{ então } Y \text{ é } B \\ \hline & (conclusão) & Y \text{ é } B \end{array} \tag{2.48}$$

O mecanismo de raciocínio aproximado permite a utilização do *Modus Ponens generalizado* (JANG; SUN; MIZUTANI, 1997).

$$\begin{array}{lll} & (fato) & X \text{ é } A' \\ Modus\ Ponens\ generalizado & (regra) & \text{se } X \text{ é } A \text{ então } Y \text{ é } B \\ \hline & (conclusão) & Y \text{ é } B' \end{array} \tag{2.49}$$

Embora o mecanismo de raciocínio nebuloso utilize o *modus ponens generalizado* na computação das regras nebulosas, a principal diferença para o mecanismo de raciocínio aproximado é que as proposições $A$, $A'$, $B$ e $B'$ são conjuntos nebulosos. Na definição de uma base de regras nebulosas alguns passos devem ser seguidos:

1. definição da semântica das regras nebulosas;
2. definição de como a conclusão será extraída a partir dos fatos e da regra nebulosa.

Sejam $A$, $A'$ e $B$ conjuntos nebulosos definidos nos universos de discurso $\mathbf{X}$, $\mathbf{X}$ e $\mathbf{Y}$, respectivamente. Considerando que a implicação nebulosa $A \Rightarrow B$ é expressa pela relação $\mathcal{R}$ definida em $\mathbf{X} \times \mathbf{Y}$. Então o conjunto nebuloso $B'$ induzido por "x é $A'$" e a regra nebulosa "se $X$ é $A$ então $Y$ é $B$" é definida por:

$$\begin{aligned} B' &= A' \circ \mathcal{R} \\ &= A' \circ (A \Rightarrow B) \end{aligned} \tag{2.50}$$

e a função de pertinência de $B'$ é definida por:

$$\mu_{B'}(y) = \sup_{x \in \mathbf{X}} [\mu_{A'}(x)\ \mathbf{t}\ \mu_R(x,y)] \tag{2.51}$$

onde $\mathbf{t}$ representa a interseção entre dois conjuntos nebulosos, implementada por uma norma-t.

**Raciocínio nebuloso com múltiplos antecedentes**

Uma regra nebulosa com dois antecedentes geralmente assume a forma "se $x$ é $A$ e $y$ é $B$ então $z$ é $C$". O raciocínio aproximado para essa regra pode ser visto na formulação 2.52

$$\begin{array}{lll} & (fato) & x \text{ é } A' \text{ e } y \text{ é } B' \\ \textit{Modus Ponens} & (regra) & \text{se } x \text{ é } A \text{ e } y \text{ é } B \text{ então } z \text{ é } C \\ \hline & (conclusão) & z \text{ é } C' \end{array} \tag{2.52}$$

A regra nebulosa pode ser colocada na forma simplificada $A \bigcap B \Rightarrow C$ que pode ser transformada na relação nebulosa ternária $\mathcal{R} = (A \bigcap B) \bigcap C$ supondo que a regra seja uma implicação nebulosa de Mamdani.

A conclusão $C$ pode ser expressa como:

$$C' = (A' \cap B') \circ (A \cap B \Rightarrow C) \tag{2.53}$$

Geralmente o grau de compatibilidade de cada premissa do antecedente pode ser analisado separadamente. Este procedimento é denominado de processo de inferência escalonado:

$$\begin{aligned} \mu_{C'}(z) &= \sup_{x,y} \{[\mu_{A'}(x) \,\mathbf{t}\, \mu_{B'}(y)] \;\mathbf{t}\; [\mu_A(x) \,\mathbf{t}\, \mu_B(y) \,\mathbf{t}\, \mu_C(z)]\} \\ &= \left\{ \sup_{x,y} [\mu_{A'}(x) \,\mathbf{t}\, \mu_{B'}(y) \mathbf{t}\, \mu_A(x) \,\mathbf{t}\, \mu_B(y)] \right\} \mathbf{t}\, \mu_C(z) \\ &= \underbrace{\left\{ \sup_{x} [\mu_{A'}(x) \,\mathbf{t}\, \mu_A(x)] \right\}}_{w_1} \mathbf{t} \underbrace{\left\{ \sup_{y} \left[ \mathbf{t}\, \mu'_B(y) \,\mathbf{t}\, \mu_B(y) \right] \right\}}_{w_2} \mathbf{t}\, \mu_C(z) \\ &= \underbrace{(w_1 \,\mathbf{t}\, w_2)}_{\eta} \,\mathbf{t}\, \mu_C(z) \end{aligned} \tag{2.54}$$

onde,

- $w_1$ e $w_2$ são as possibilidades $Poss(A, A')$ e $Poss(B, B')$, respectivamente conforme definido na equação 2.25. Normalmente, essas componentes indicam o *grau de compatibilidade* entre o fato e cada antecedente associado a uma variável lingüística;
- $\eta$ é o grau de ativação da regra nebulosa obtido da agregação dos graus de compatibilidade;
- $\mu_{C'}(z)$ é o conseqüente qualificado obtido da agregação do grau de ativação de uma regra com a função associada ao conseqüente.

**Raciocínio nebuloso com múltiplas regras**

Em geral os sistemas nebulosos utilizam mais de uma regra nebulosa para realizar o processo de inferência. Entretanto a saída do sistema deve ser um único conjunto nebuloso. Seja $m$ o número de regras nebulosas, define-se uma relação nebulosa $\mathcal{R}_{sys}$:

$$\mathcal{R}_{sys} : \mathbf{Y}_1 \times \cdots \times \mathbf{Y}_m \mapsto [0,1] \tag{2.55}$$

que é obtida da agregação de todas as regras qualificadas do sistema nebuloso. Em geral o operador de agregação de regras é uma norma-s mas operadores de média são relativamente comuns também.

Supondo que ambas as regras sejam conjunções nebulosas e o operador de agregação das regras seja o operador max tem-se:

$$\begin{aligned} C' &= (A' \times B') \circ (\mathcal{R}_1 \cup \mathcal{R}_2) \\ &= [(A' \times B') \circ \mathcal{R}_1] \cup [(A' \times B') \circ \mathcal{R}_2] \\ &= C'_1 \cup C'_2 \end{aligned} \tag{2.56}$$

onde $C'_1$ e $C'_2$ são os conjuntos nebulosos resultantes da inferência das regras 1 e 2 definidos na equação 2.53. A função de pertinência do conjunto nebuloso $C'$ também é conhecida como função de pertinência global de saída (JANG; SUN; MIZUTANI, 1997).

## 2.3 Sistemas nebulosos

Um sistema de inferência nebuloso é uma ferramenta de computação baseada nos conceitos de conjuntos nebulosos, regras nebulosas e raciocínio nebuloso. Eles encontram aplicações em diversas áreas como controle automático, classificação de dados, tomada de decisões, predição de séries temporais, robótica e reconhecimento de padrões (JANG; SUN; MIZUTANI, 1997). Outras nomenclaturas para os sistemas nebulosos encontradas na literatura incluem: sistemas nebulosos baseados em regras, sistemas especialistas nebulosos (KANDEL, 1992), modelos nebulosos (TAKAGI; SUGENO, 1985b; SUGENO; KANG, 1988), memórias associativas nebulosas (KOSKO, 1992) e controlador lógico nebuloso (KANDEL, 1992; LEE, 1990).

A estrutura básica de um sistema de inferência nebuloso é constituída de três componentes principais:

**base de regras** que contém um conjunto de regras nebulosas;

**base de dados** que define as funções de pertinência utilizadas nas regras nebulosas;

**mecanismo de inferência** que executa o procedimento de inferência sobre as regras nebulosas e retorna fatos para formar uma saída razoável ou conclusão.

Observa-se que o sistema básico de inferência nebuloso pode receber tanto conjuntos nebulosos quanto entradas não-nebulosas (que são interpretadas como conjuntos do tipo *singleton*), mas as saídas produzidas são geralmente conjuntos nebulosos. Em alguns casos é necessário obter como saída um valor escalar, especialmente em casos em que o sistema nebuloso será utilizado agindo em um sistema não-nebuloso. Portanto é necessário um método de transformação da saída nebulosa em não-nebulosa para extrair um valor escalar que melhor represente o conjunto nebuloso. O tipo de sistema nebuloso utilizado ao longo desse trabalho, entretanto, não necessita de tal procedimento que não será profundamente detalhado. Mais informações sobre este processo podem ser obtidas em outras obras da literatura e.g. (PEDRYCZ; GOMIDE, 1998; JANG; SUN; MIZUTANI, 1997; CORDÓN *et al.*, 2001a).

### 2.3.1 Modelos de sistemas nebulosos

Existem vários tipos de sistemas nebulosos. Na maioria dos casos, o antecedente é formado por proposições lingüísticas e a distinção entre os modelos se dá no conseqüente das regras nebulosas. Entre os modelos mais conhecidos pode-se destacar:

**Lingüístico ou Mamdani** utiliza conjuntos nebulosos nos conseqüentes das regras nebulosas (MAMDANI; ASSILIAN, 1975). A saída final é representada por um conjunto nebuloso resultante da agregação da saída inferida de cada regra. Para se obter uma saída final não nebulosa adota-se um dos métodos de transformação da saída nebulosa em não nebulosa (JANG; SUN; MIZUTANI, 1997; PEDRYCZ; GOMIDE, 1998; CORDÓN *et al.*, 2001a).

**Interpolativo ou Takagi-Sugeno** o conseqüente é representado por uma função das variáveis de entrada (TAKAGI; SUGENO, 1993). A saída final é obtida pela média ponderada das saídas inferidas de cada regra. Os coeficientes de ponderação são dados a partir do grau de ativação de cada regra.

**Tsukamoto** utiliza funções de pertinência monotônicas no conseqüente (TSUKAMOTO, 1993). A saída do sistema é um valor não-nebuloso obtido da média ponderada das saídas inferidas de cada regra.

### Modelo de Takagi-Sugeno

O modelo de Takagi-Sugeno-Kang (TAKAGI; SUGENO, 1985a, 1985b; SUGENO; KANG, 1988) (também conhecido como TSK ou simplesmente TS) é uma abordagem sistemática para a geração de regras nebulosas a partir de um conjunto de dados de entrada-saída. Uma regra típica de um sistema com duas variáveis de entrada utilizando um modelo TSK tem a forma:

$$\text{Se } x_1 \text{ é } A \text{ e } x_2 \text{ é } B \text{ então } z = g(\mathbf{w}, \mathbf{x}) \tag{2.57}$$

onde:

- $A$ e $B$ são conjuntos nebulosos no antecedente;
- $g(\mathbf{w}, \mathbf{x})$ é uma função das variáveis de entrada $\mathbf{x} = [x_1, x_2]$ com parâmetros $\mathbf{w}$, também conhecida como conseqüente de Takagi-Sugeno (TAKAGI; SUGENO, 1985a)

Geralmente $g(\mathbf{w}, \mathbf{x})$ é uma função polinomial de $\mathbf{x}$ mas pode ser qualquer função que descreva apropriadamente o comportamento do sistema em uma região determinada pelo antecedente da regra. Quando $g(\mathbf{w}, \mathbf{x})$ é um polinômio de primeira ordem o sistema de inferência nebuloso é denominado modelo nebuloso TSK de primeira ordem. Este modelo foi proposto inicialmente por Takagi e Sugeno (1985b), Sugeno e Kang (1988). Quando $g(\mathbf{w}, \mathbf{x})$ é uma função constante o sistema de inferência resultante é um modelo nebuloso TSK de ordem zero, que pode ser visto como um caso especial de um modelo de Mamdani no qual cada conseqüente é especificado por uma função singleton. Delgado, von Zuben e Gomide (2001) propuseram o uso de conseqüentes de Takagi-Sugeno de ordem 2 com uma componente não-linear para o problema de identificação de sistemas nebulosos. Neste trabalho foram utilizados os conseqüentes propostos por Delgado (2002).

A saída de um modelo nebuloso TSK de ordem zero é uma função suave de suas variáveis de entrada enquanto as funções de pertinência vizinhas apresentam um grau de sobreposição suficiente. Ou seja, a sobreposição entre as funções de pertinência no antecedente de um modelo de Takagi-Sugeno-Kang de ordem zero é condição necessária para a suavidade do mapeamento entrada-saída.

Como cada regra possui uma saída convencional, a saída global é obtida através da média ponderada. Na prática, o operador de média ponderada é algumas vezes substituído pela soma ponderada visando reduzir o custo computacional, especialmente durante o treinamento do sistema de inferência nebuloso. Entretanto, essa simplificação pode causar uma perda do significado lingüístico da partição nebulosa, a menos que a soma dos graus de ativação ($\sum_i \eta_i$)

seja próxima da unidade (JANG; SUN; MIZUTANI, 1997).

Ao contrário de sistemas nebulosos de Mamdani, os modelos nebulosos TSK não seguem estritamente a regra composicional de inferência. Essa característica é inconveniente quando as entradas do modelo são nebulosas (JANG; SUN; MIZUTANI, 1997). É possível realizar a combinação de conjuntos nebulosos no antecedente para encontrar o grau de ativação de cada regra. Entretanto a saída resultante da média ponderada (ou soma ponderada) não é um conjunto nebuloso. Isso vai contra a idéia de que um modelo nebuloso deveria ser capaz de propagar sua incerteza das entradas para as saídas de maneira apropriada.

Muitas vezes um modelo nebuloso TSK simples é capaz de fornecer um comportamento complexo. Essa característica é interessante pois, apesar de gerar regras sem um significado lingüístico óbvio, o número de regras necessárias para modelar um conjunto de dados entrada-saída pode ser menor que o necessário em um sistema com regras tipo Mamdani (DE CARVALHO E PAIVA, 1999).

O modelo  Takagi e Sugeno (1985a) foi proposto como resultado de um esforço para se desenvolver, de forma sistemática, uma abordagem para a geração de regras nebulosas a partir de dados entrada-saída.

Neste trabalho somente os modelos nebulosos TSK foram considerados.

## 2.4 Projeto de sistemas nebulosos

Sistemas nebulosos são ferramentas úteis de tratamento de informações imprecisas como aquelas retiradas do conhecimento e experiências humanas. Contudo seu projeto apresenta algumas dificuldades:

- O projeto manual de um sistema nebuloso pode se tornar um processo complexo requerendo algumas vezes conhecimento profundo da aplicação;
- O conhecimento sobre a aplicação pode não estar disponível;
- Em alguns casos, não há uma metodologia consistente para o particionamento dos intervalos de discurso em partições nebulosas interpretáveis;
- O projeto do processo de transformação da saída nebulosa em não-nebulosa é, em alguns casos, de difícil tratamento matemático;

- O projeto de sistemas visando a minimização do erro pode gerar sistemas não-interpretáveis por um operador humano;
- Independente do esforço despendido no projeto do sistema nebuloso não há garantia que os parâmetros deste sistema sejam ótimos.

Considerando essas dificuldades, um grande esforço tem sido despendido na busca de métodos eficazes de projeto automático de sistemas nebulosos. Alguns dos paradigmas mais conhecidos são aqueles baseados em redes neurais artificiais ou algoritmos genéticos. Neste sentido, o presente trabalho aborda uma alternativa ao projeto automático de sistemas nebulosos. Particular atenção foi dada ao fato de tentar evitar que o usuário deva fornecer ou alterar dados de entrada. Ou seja, busca-se uma alternativa o mais independente possível do usuário.

# 3 *Algoritmos genéticos*

Os algoritmos genéticos (AG) são ferramentas de busca e otimização baseadas em princípios da seleção natural e genética. Eles utilizam uma forma simplificada do princípio da sobrevivência do mais apto e uma troca de informação estruturada, porém aleatória entre soluções candidatas (GOLDBERG, 1989). Os algoritmos genéticos encontram aplicações nas mais diversas áreas tais como: otimização de funções (HERRERA; LOZANO, 2001), logística, bioinformática (SCAPIN; LOPES, 2005; TSUNODA; LOPES, 2006) e telecomunicações (MOGNON, 2004).

Os algoritmos genéticos foram desenvolvidos por Holland (1975), seus colegas e estudantes na Universidade de Michigan (GOLDBERG, 1989). Os objetivos principais da pesquisa foram: abstrair e explicar rigorosamente o mecanismo adaptativo de processos naturais e projetar sistemas computacionais que imitassem alguns mecanismos de sistemas biológicos.

A seção 3.1 apresenta uma introdução ao conceito de desempenho relativo de otimização e a motivação para o uso de algoritmos genéticos. As seções 3.2 à 3.7 apresentam a fundamentação dos algoritmos genéticos. A seção 3.8 discute as formas de determinação de parâmetros evolutivos de um algoritmo genético. A subseção 3.8.1 faz uma breve revisão bibliográfica da classificação das técnicas de controle dos parâmetros evolutivos. A subseção 3.8.2 discute a técnica de ajuste de parâmetros mais detalhadamente. A subseção 3.8.3 discute a técnica de controle de parâmetros. A subseção 3.8.4 discute a abordagem auto-adaptativa, que serviu de inspiração para a técnica adotada neste trabalho.

Este capítulo apresenta alguns princípios básicos de algoritmos genéticos. Entretanto, vale salientar que o texto apresentado não se propõe a ser uma documentação sobre o assunto. Para mais informações, outras obras da literatura podem ser consultadas (GOLDBERG, 1989; MICHALEWICZ, 1994; CORDÓN *et al.*, 2001a; EIBEN; SMITH, 2003)

## 3.1 Objetivos da otimização

Antes de examinar o funcionamento de um algoritmo genético simples, deve-se esclarecer os objetivos desta técnica. Quando deseja-se otimizar uma função ou processo, busca-se melhorar o desempenho com o intuito de alcançar um ponto ótimo. Existe uma distinção entre o processo de melhora das soluções e a obtenção do ótimo propriamente dito. Na avaliação de técnicas de otimização é comum analisar somente a convergência do método (seu sucesso na busca da solução ótima) e esquecer completamente seu desempenho relativo, ou seja, seu desempenho comparado com outras técnicas de solução do mesmo problema. A análise exclusiva da convergência entretanto nem sempre é natural. Por exemplo, considere uma pessoa encarregada de tomar decisões. Qual é a forma como se avaliam suas decisões? Qual o critério utilizado para inferir se ele realizou um bom ou mal trabalho? Normalmente, diz-se que ele realizou um bom trabalho se ele fez seleções adequadas do tempo e recursos necessários. A qualidade é avaliada em comparação com seus competidores. Dificilmente a qualidade do trabalho de uma pessoa é avaliada por um critério de "realização-da-perfeição". Analogamente, a não-convergência para o ótimo nem sempre é uma questão crítica para muitos problemas da vida real. Neste contexto, o principal objetivo passa a ser a melhora da solução atual, ou seja, é possível obter um nível de desempenho satisfatório com os recursos disponíveis? A importância de encontrar o ótimo decresce com o aumento da complexidade dos sistemas. Em muitos casos, a solução ótima pode ser desconhecida ou sua busca exceder os limites razoáveis de recursos despendidos (GOLDBERG, 1989).

Portanto, o resultado do processo evolutivo nem sempre é avaliado em termos de convergência. Em muitos casos soluções quase ótimas são aceitas, considerando-se o tempo computacional requerido para obte-las.

## 3.2 Inspiração natural

De forma geral, o processo evolutivo é um mecanismo de busca no qual uma população de possíveis soluções para o problema da sobrevivência do mais apto é constantemente avaliada. Entretanto, o processo evolutivo utilizado pelos algoritmos genéticos é razoavelmente simplificado.

Os algoritmos genéticos tentam imitar o processo evolutivo das espécies. Ao contrário de algumas técnicas tradicionais de otimização que trabalham com apenas uma solução candidata por ciclo iterativo, os algoritmos genéticos pressupõem o uso de uma população de indivíduos

(que codificam as soluções-candidatas). Essa abordagem é interessante por diminuir significativamente a probabilidade de convergência para ótimos locais. Além disso, as variáveis da solução candidata devem estar codificadas no código genético de um indivíduo através de um conjunto de genes. O código genético do indivíduo define, de forma indireta, as variáveis da solução candidata. A seqüência de genes de um indivíduo ou organismo é geralmente denominada de genótipo enquanto a representação destes genes em variáveis do problema é denominada de fenótipo. A relação associando genótipo e fenótipo não é necessariamente biunívoca e nestes casos não é possível associar um fenótipo a apenas um genótipo. A representação através de genes é interessante por necessitar de poucas informações adicionais para funcionar adequadamente. Técnicas baseadas no gradiente, ao contrário, necessitam que o espaço de busca seja diferenciável para qualquer ponto do espaço de busca. Algoritmos genéticos, em contrapartida, necessitam somente de informações da qualidade da solução associada ao genótipo que, em geral, pode ser inferida a partir de uma função objetivo. Entretanto, caso existam informações adicionais, os algoritmos genéticos podem ser combinados com outras formas de busca para incrementar seu desempenho (HART; KRASNOGOR; SMITH, 2004).

A cada geração do processo evolutivo estas soluções sofrem um mecanismo de seleção que visa mimetizar, de forma simplificada, o princípio da sobrevivência do mais apto. Através de uma regra de transição probabilística soluções mais aptas podem trocar material genético gerando novos indivíduos que contêm soluções parciais do problema herdadas de seus pais. Além disso, o código genético dos indivíduos está sujeito a mudanças aleatórias através de um operador análogo à mutação genética.

### 3.2.1 Algumas definições

Formalmente, um algoritmo genético é uma tupla (SMITH, 1998)

$$AG = \left(P^0, \delta^0, \lambda, \varphi, l, F, G, U\right) \tag{3.1}$$

onde:

- $P^0 = (h_1^0, \ldots, h_\varphi^0) \in I^{\varphi \times l}$ é a população inicial;
- $h_i^t \in I^l$ é o indivíduo na posição $i$ na geração $t$;
- $I$ é o alfabeto do gene (para codificação binária $I = \{0, 1\}$);

- $\delta^0$ é o conjunto inicial dos parâmetros estratégicos , ou seja, o conjunto de parâmetros do AG (taxa e tipo de mutação, taxa e tipo de recombinação, tamanho de população, etc);
- $\varphi \in \mathbb{N}$ é o tamanho da população;
- $\lambda \in \mathbb{N}$ é o número de descendentes;
- $l \in \mathbb{N}$ é o tamanho do cromossomo;
- $F : I^l \mapsto \mathbb{R}^+$ é a função de avaliação;
- $G : I^{\varphi \times l} \mapsto I^{\lambda \times l}$ é a função de reprodução;
- $U : I^{\varphi \times l} \times I^{\lambda \times l} \mapsto I^{\varphi \times l}$ é a função de atualização;
- $\mathbb{N}$ é o conjunto dos números naturais;
- $\mathbb{R}$ é o conjunto dos números reais;
- $\mathbb{R}^+$ é o conjunto dos números reais não-negativos.

Geralmente, assume-se que a função objetivo do algoritmo genético é uma maximização restrita somente a valores positivos (SMITH, 1998; GOLDBERG, 1989; EIBEN; SMITH, 2003). Embora nada possa ser afirmado sobre a função de avaliação neste momento, esta restrição pode ser satisfeita através de um processo de normalização.

## 3.3 Codificação da solução

A codificação de cada indivíduo da população indica a forma como as variáveis do problema são armazenadas e processadas pelo algoritmo genético através de entidades denominadas genes.

O AG clássico utiliza um alfabeto binário $I = \{0, 1\}$. Entretanto, segundo Michalewicz (1994), esta forma de codificação pode ser inadequada para resolver problemas de otimização numérica. É possível mapear uma seqüência de bits para um número inteiro, mas as trocas de material genético podem ocorrer em pontos de corte situados no meio de genes codificando números inteiros. Essa operação pode gerar um gene totalmente novo, o que vai contra o princípio do operador de troca de material genético de transferir características dos pais para os filhos. Nestes casos é possível evitar este comportamento utilizando genes inteiros ($I \subseteq \mathbb{N}$). Um número inteiro pode facilmente ser mapeado para um número real desde que seus limites inferior e superior sejam conhecidos (GOLDBERG, 1989):

$$v_k^{real} = \frac{v_k^{int} - l_k}{u_k - l_k} \tag{3.2}$$

onde $v_k^{real}$ é o valor fenotípico do gene e $v_k^{real} \in [l_k, u_k]$. Seja $\epsilon$ a diferença máxima tolerável no mapeamento inteiro-real, o número de bits necessário para representar adequadamente o gene, utilizando um mapeamento linear, é:

$$n_{bits} = log_2\left(1 + \frac{u_k - l_k}{\epsilon}\right) \tag{3.3}$$

Para precisões grandes, o número de bits necessário pode tornar-se alto. Em muitos problemas de otimização contínua isso pode ser impraticável. Para esta classe de problema a utilização de genes codificados com números reais ($I \subseteq \mathbb{R}$) pode ser uma alternativa interessante. É possível o uso de codificação real na solução de problemas de otimização inteira ou combinatorial. Entretanto, como será visto ainda neste capítulo, essa escolha apresenta a desvantagem de aumentar significativamente o espaço de busca do algoritmo.

Em problemas de otimização que envolvem restrições é possível melhorar o desempenho do algoritmo através de modificações na codificação. Entretanto, o problema de como lidar com indivíduos infactíveis não é trivial (MICHALEWICZ, 1995b):

- Dados dois indivíduos $h_i$ factível e $h_j$ infactível, a aptidão de $h_i$ deve ser maior que a de $h_j$ $\Big(F(h_i) > F(h_j)\Big)$ mesmo que $h_j$ seja mais próximo da solução?
- Indivíduos infactíveis devem ser considerados nocivos e eliminados da população?
- Soluções infactíveis devem ser movidas para o ponto mais próximo no espaço factível?
- Devem-se atribuir penalizações para soluções pertencentes ao espaço infactível? Qual a severidade dessas penalizações?

Analisando somente problemas de otimização numérica, particularmente problemas de otimização não-linear, as técnicas de controle de restrições em algoritmos genéticos podem ser classificadas da seguinte maneira (MICHALEWICZ, 1995b):

- uso de funções de penalização (DAVIS, 1987);
- rejeição de indivíduos infactíveis (MICHALEWICZ, 1995a; MICHALEWICZ; NAZHIYATH, 1995; MICHALEWICZ; SCHOENAUER, 1996);

- manutenção da factibilidade da população através de codificação e operadores genéticos (DAVIDOR, 1991, 1990);
- separação da função objetivo das restrições (COELLO, 2000);
- métodos híbridos (ADELI; CHENG, 1994);
- outros métodos (MICHALEWICZ, 1995b).

A abordagem mais comum é o uso de funções de penalização. A aptidão de um indivíduo é diminuída se este violar qualquer restrição. Idealmente, a penalidade deve ser a mais baixa possível, logo acima do limite em que soluções infactíveis tornam-se ótimas (este critério é denominado de regra da mínima penalidade (DAVIS, 1987; SMITH, 1998)). Entretanto, embora bastante simples, na prática é difícil implementar essa regra pois a localização exata das regiões de factibilidade e infactibilidade são usualmente desconhecidas. O uso de penalidades severas funciona bem para espaços de busca convexos. Entretanto essa abordagem apresenta sérias limitações em casos em que o espaço de busca infactível é muito maior que o espaço de busca factível. Além disso, para muitos problemas, o espaço de busca é não-convexo ou a solução ótima está localizada em regiões próximas ao limite entre factibilidade e infactibilidade. Nesses casos o sistema pode alcançar a solução ótima mais facilmente se ele puder "atravessar" a região não factível (MICHALEWICZ, 1995b). Em contrapartida a necessidade de estimar a aptidão de indivíduos infactíveis para aplicações com funções de avaliação complexas pode tornar-se problemática devido ao custo computacional envolvido.

A rejeição de indivíduos infactíveis pode ser vista como um caso limite em que as funções de penalização divergem. Michalewicz (MICHALEWICZ, 1995a; MICHALEWICZ; NAZHIYATH, 1995; MICHALEWICZ; SCHOENAUER, 1996) comparou a abordagem de rejeição de indivíduos infactíveis com a de penalização e seus resultados indicam um desempenho superior da técnica de penalização baseada na distância da região de factibilidade.

Algoritmos de reparação são particularmente úteis para aplicações em problemas de otimização combinatorial (como o caixeiro viajante, problema da mochila, cobertura de conjuntos, etc) em que é relativamente fácil reparar um indivíduo infactível (MICHALEWICZ, 1995b). Essa versão reparada pode ser utilizada somente na avaliação dos indivíduos (COELLO, 1999), isto é:

$$F(h_i) = F(h'_i) \tag{3.4}$$

onde $h'_i$ é a versão reparada de $h_i$ e $F$ é a função de avaliação. Alternativamente $h'_i$ pode substituir o indivíduo original $h_i$ na população.

O uso de codificação e operadores genéticos especiais foi estudado por Davis (1987), Davidor (1991, 1990) e Michalewicz *et al.* (1996). O uso de operadores especiais é útil para a aplicação para a qual eles foram projetados, mas sua generalização para outros problemas similares é questionável (MICHALEWICZ, 1995b). Seu uso requer conhecimento do espaço de busca da aplicação, que podem não estar disponíveis.

A técnica de separação da função objetivo das restrições consiste em redefinir a otimização de um objetivo em um problema multi-objetivo com $m+1$ objetivos onde $m$ indica o número de restrições (DEB, 2000). Então, uma técnica de otimização multi-objetivo é aplicada ao problema. Para problemas com muitas restrições essa técnica não é adequada pois a complexidade do modelo ultrapassa limites razoáveis.

O controle híbrido de restrições considera o uso de uma outra técnica (geralmente uma abordagem de otimização numérica) para tratar as restrições do algoritmo evolutivo. A principal desvantagem dessa abordagem é o aumento do número de parâmetros necessários que devem ser determinados empiricamente. Os resultados obtidos com essa técnica não são significativamente melhores que uma abordagem por penalização (ADELI; CHENG, 1994).

Outros métodos de destacado uso na literatura são sistemas imunológicos artificiais (DE CASTRO; TIMMIS, 2002) e algoritmos sociais (MICHALEWICZ, 1995b).

Neste trabalho, as abordagens de penalização, reparação e codificação foram utilizadas em contextos diferentes:

**penalização** na garantia das restrições de um problema combinatorial;

**reparação** na garantia da interpretabilidade das partições nebulosas. A reparação ocorre no nível I da estrutura hierárquica proposta por Delgado (2002);

**codificação** na garantia das restrições dos problemas de otimização contínua e nos demais níveis do sistema genético-nebuloso hierárquico

No problema de otimização combinatória considerado neste trabalho, para a garantia das restrições, utilizou-se a mesma forma de penalização do trabalho de Kimbrough *et al.* (2002). Isto porque o objetivo é avaliar o desempenho da abordagem auto-adaptativa considerando a mesma função de avaliação. Por outro lado, os problemas de otimização contínua abordados nesta dissertação devem estar restritos a uma região compacta. A satisfação dessa restrição foi realizada através de uma codificação adequada. A correção por reparação no problema do projeto automático de sistemas nebulosos foi escolhida pois uma abordagem por codificação

demandaria conhecimento específico do problema, que não existe. Uma abordagem por penalização seria possível, entretanto o custo computacional associado à avaliação de um sistema nebuloso no modelo adotada neste trabalho é elevado. Neste caso a manutenção de indivíduos infactíveis na população e sua avaliação demandaria um gasto computacional alto sem existir garantias de convergência para uma solução factível.

### Cromossomos

Um cromossomo é representado por um vetor de genes com comprimento $l$:

$$\mathbf{h}_i^t = [h_{i1}^t, \ldots h_{il}^t], \quad h_{ik}^t \in I, \forall k \in \{1, \ldots, l\} \tag{3.5}$$

onde $\mathbf{h}_i^t$ representa o cromossomo na posição $i$ dentro da população na geração $t$ e $h_{ik}^t$ representa o gene $k$ deste cromossomo. Cada gene ocupa um lugar definido no cromossomo. Esse lugar definido é denominado locus gênico.

Na representação da solução, é possível ainda utilizar duas abordagens (EIBEN; SMITH, 2003). A abordagem *Pittsburgh* utiliza um único cromossomo para codificar a solução completa para o problema. A abordagem *Michigan*, ao contrário, utiliza um conjunto de cromossomos $k \in 1, \ldots, l$ para compor a solução. Cada cromossomo passa a conter somente uma solução parcial para o problema e é necessário um mecanismo de agregação dessas soluções parciais para a obtenção da solução.

### Espaço de busca

O espaço de busca de um algoritmo genético é dado pelo número total de possíveis combinações de cromossomos. Seja $card(I)$ a cardinalidade do alfabeto do gene, o espaço de busca do AG é dado por:

$$ss(AG) = [card(I)]^l \tag{3.6}$$

onde a cardinalidade de qualquer alfabeto contínuo é teoricamente infinita (mas geralmente restrita pela arquitetura computacional utilizada). Portanto o espaço de busca de um algoritmo genético com codificação real é, desprezadas as restrições da arquitetura, infinito.

## 3.4 Teoria dos esquemas

Um esquema é um padrão de similaridade descrevendo um conjunto de vetores com certos elementos em comum (HOLLAND, 1992). Por questão de simplificação esta análise será focada somente no alfabeto binário $I = \{0, 1\}$. Inicialmente adiciona-se um símbolo especial a este alfabeto, o símbolo de *don't care* $*$. Utilizando este novo alfabeto é possível criar seqüências (vetores) e seu significado pode ser resumido da seguinte forma: um esquema combina com um vetor se, para cada posição, um símbolo 1 está presente na mesma posição tanto no esquema quanto no vetor. De maneira análoga, um símbolo 0 deve estar presente na mesma posição tanto no esquema quanto no vetor. Ou ainda, um símbolo $*$ indica que o esquema e o vetor combinam independente do símbolo nesta posição. Por exemplo, o esquema $[*000]$ combina tanto com o vetor [1000] quanto com [0000]. Portanto, os esquemas são uma ferramenta poderosa e compacta de definir similaridades entre vetores utilizando um alfabeto finito. É preciso ainda enfatizar que $*$ é apenas um meta-símbolo (um símbolo representando outros símbolos) e não é explicitamente processado pelo algoritmo.

Considerando uma população de cromossomos $P^t$ isoladamente, têm-se poucas informações para inferir o desempenho do algoritmo. Entretanto quando a análise é feita a partir das similaridades entre os indivíduos tem-se uma informação importante sobre as regiões mais promissoras do espaço de busca. Inicialmente é importante contabilizar o número de esquemas únicos presentes na população. O número total de esquemas possíveis depende do tamanho do cromossomo e da cardinalidade do alfabeto. Seja $u$ o número total de esquemas possíveis, portanto tem-se:

$$u = (card(I) + 1)^l \tag{3.7}$$

Entretanto, considerando um único vetor de genes, supondo $v_{ik}^t = 0, \forall k \in \{1, \ldots, l\}$, o número de esquemas possíveis para este cromossomo é $2^l$. Conseqüentemente, para uma população de $\phi$ cromossomos, o número total de esquemas varia entre $2^l$ e $\varphi \times 2^l$. Através do mecanismo de seleção, os padrões mais aptos têm uma probabilidade maior de sobrevivência. Como conseqüência tem-se, em média, um crescimento do número dos esquemas associados a indivíduos mais aptos. Holland demonstrou que esta população hipotética processa efetivamente $O(\varphi^3)$ padrões. Este resultado, conhecido como paralelismo implícito é considerado um dos principais fatores no sucesso dos AGs.

Através do teorema do esquema que será visto nesse capítulo é possível mostrar que um esquema associado a indivíduos com aptidão acima da média apresenta uma tendência a crescimento (GOLDBERG, 1989). Esse crescimento garante a sobrevivência de esquemas associ-

ados a indivíduos mais aptos ao mecanismo de seleção natural. Entretanto, é possível que um esquema esteja associado a indivíduos acima da média mas não contenha genes que afetem diretamente a aptidão do indivíduo. Essa observação será importante, pois garante o funcionamento do controle auto-adaptativo dos algoritmos genéticos.

## 3.5 Operadores de variação

Os principais operadores de variação são a Mutação e a Recombinação ou cruzamento. Uma distinção entre estes dois operadores pode ser feita de acordo com o número de indivíduos utilizados como entradas do operador. O operador de mutação utiliza como operando um único indivíduo e produz um único indivíduo em sua saída. O operador de recombinação, em contrapartida, atua em mais de um indivíduo e produz, geralmente, mais de um indivíduo em sua saída. Em geral, operadores de recombinação utilizam dois indivíduos como "pais" e o termo cruzamento (inspirado da biologia evolutiva) é freqüentemente utilizado como um sinônimo para recombinação com dois "pais".

### 3.5.1 Recombinação

Todos os operadores de combinação envolvem a interação de dois ou mais indivíduos. A forma geral para este operador é dada pela equação 3.8:

$$\rho : I^{\lambda \times l} \times \delta \mapsto I^{q \times l} \tag{3.8}$$

onde $\rho$ indica o operador de recombinação e $q$ indica o número de indivíduos gerados.

O operador de recombinação é executado em dois passos:

- Um conjunto de $r$ operandos é selecionado a partir de uma coleção de indivíduos;
- Um vetor de $k$ pontos de troca $\mathbf{v}$ é selecionado aleatoriamente entre 1 e $l-1$. Indivíduos são gerados a partir de trocas de material genético utilizando os pontos sorteados.

Em geral, nas aplicações que utilizam algoritmos genéticos as taxas de recombinação costumam ser maiores que as taxas de mutação (GOLDBERG, 1989). Baseados nessa premissa, muitos autores consideram que a recombinação é o principal operador de variação dos algoritmos genéticos (EIBEN; HINTERDING; MICHALEWICZ, 1999).

A recombinação de múltiplos pontos caracteriza-se pela troca de material genético entre as

posições

$$\begin{array}{ll} \{v_{2i+1}+1,\ldots,v_{2(i+1)}-1\}, & i \in \left\{0,\ldots,\left\lfloor \frac{r}{2} \right\rfloor\right\} \\ \{v_{2i+1}+1,\ldots,l\}, & i > \left\lfloor \frac{r}{2} \right\rfloor \end{array} \tag{3.9}$$

$\forall i \in \{1,\ldots,\left\lceil\left(\frac{r}{2}\right)\right\rceil\}$. Por exemplo, suponha uma recombinação com $r = 2$ e $k = 4$ entre os dois indivíduos $A_1$ e $A_2$:

$$\begin{aligned} A_1 &= 0\,1\,1\,0\,1\,1\,1\,0\,0\,1\,1\,0\,1\,0\,1\,0\,1 \\ A_2 &= 1\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,0\,0\,1\,0\,1\,1 \end{aligned}$$

Suponha que o pontos de troca $\mathbf{v} = [3,5,7,12]$. $A_1$ e $A_2$ deverão trocar material genético entre as posições $\{4,5\}$ e $\{8,\ldots,12\}$.

$$\begin{aligned} A_1 &= 0\,1\,1\,|\,\mathbf{0\,1}\,|\,1\,1\,|\,\mathbf{0\,0\,1\,1\,0}\,|\,1\,0\,1\,0\,1 \\ A_2 &= 1\,1\,0\,|\,\mathbf{1\,0}\,|\,1\,0\,|\,\mathbf{0\,1\,1\,0\,0}\,|\,0\,1\,0\,1\,1 \end{aligned}$$

O operador de recombinação deve gerar dois novos indivíduos $A'_1$ e $A'_2$.

$$\begin{array}{rl} & A_1 = 0\,1\,1\,|\,\mathbf{0\,1}\,|\,1\,1\,|\,\mathbf{0\,0\,1\,1\,0}\,|\,1\,0\,1\,0\,1 \\ \rho_1 & A_2 = 1\,1\,0\,|\,\mathbf{1\,0}\,|\,1\,0\,|\,\mathbf{0\,1\,1\,0\,0}\,|\,0\,1\,0\,1\,1 \\ \hline & A'_1 = 0\,1\,1\,|\,\mathbf{1\,0}\,|\,1\,1\,|\,\mathbf{0\,1\,1\,0\,0}\,|\,1\,0\,1\,0\,1 \\ & A'_2 = 1\,1\,0\,|\,\mathbf{0\,1}\,|\,1\,0\,|\,\mathbf{0\,0\,1\,1\,0}\,|\,0\,1\,0\,1\,1 \end{array}$$

Em muitos trabalhos, o número de pontos de corte do operador de recombinação de múltiplos pontos é reduzido a 1. Nestes casos o operador é denominado de recombinação de um-ponto.

O operador de recombinação de múltiplos pontos é capaz de aproveitar o fenômeno da epistasia na medida que, ao agrupar genes que codificam variáveis relacionadas ou mesmo acopladas, a probabilidade de uma recombinação separar estas variáveis será função do número alelos entre estes genes. Utilizando indivíduos $A_1$ e $A_2$ que combinam com dois esquemas $1***0$ e $**11*$ respectivamente, pode-se estimar a probabilidade de quebra do segundo esquema em $\frac{1}{4}$ (existem 4 pontos de corte possíveis mas apenas 1 deles é capaz de separar o esquema). O primeiro esquema, em contrapartida, será certamente quebrado se este cromossomo sofrer recombinação independente do sorteio (qualquer ponto de corte escolhido terá como consequência a quebra desse esquema). Neste caso um processo de codificação adequado (agrupando genes pertencentes ao mesmo esquema) pode incrementar significativamente o desempenho do AG. Seja $\mathbb{H}$ um esquema qualquer, a probabilidade da operação de recombinação de um ponto

acarretar em uma quebra de $\mathbb{H}$ pode ser dada por:

$$P_{Hr} = p_c \frac{d(\mathbb{H})}{l-1} \tag{3.10}$$

onde:

- $P_{Hr}$ é a probabilidade de quebra do esquema por uma operação de recombinação;
- $p_c$ é a probabilidade de recombinação;
- $d(\mathbb{H})$ é a distância máxima entre dois genes relevantes do esquema.

Além da recombinação de múltiplos pontos, é possível utilizar um operador de recombinação uniforme, em que a probabilidade de dois genes serem separados pelo operador independe de sua posição no genótipo. Suponha o cromossomo do exemplo anterior:

$$\begin{aligned} A_1 &= 0\ 1\ 1\ 0\ 1 \\ A_2 &= 1\ 1\ 0\ 1\ 0 \end{aligned}$$

Suponha que foram escolhidos os pontos de troca $\mathbf{v} = [1,3]$. Neste caso $A_1$ e $A_2$ deverão trocar material somente nas posições 1 e 3

$$\begin{aligned} A_1 &= (\mathbf{0})\ 1\ (\mathbf{1})\ 0\ 1 \\ A_2 &= (\mathbf{1})\ 1\ (\mathbf{0})\ 1\ 0 \end{aligned}$$

gerando os indivíduos $A''$ e $B''$

$$\begin{array}{rrcl} & A_1 &=& (\mathbf{0})\ 1\ (\mathbf{1})\ 0\ 1 \\ \rho_2 & A_2 &=& (\mathbf{1})\ 1\ (\mathbf{0})\ 1\ 0 \\ \hline & A_1'' &=& (\mathbf{1})\ 1\ (\mathbf{0})\ 1\ 0 \\ & A_2'' &=& (\mathbf{0})\ 1\ (\mathbf{1})\ 0\ 1 \end{array}$$

O desempenho da recombinação é influenciado pela codificação escolhida. Para uma combinação de esquemas com variáveis relacionadas próximas no genótipo, é provável que uma recombinação de múltiplos pontos apresente um desempenho superior. Entretanto, na ausência de conhecimento sobre a dependência entre as variáveis do problema, o uso de esquemas com grande distância entre os genes apresenta uma probabilidade maior de quebra devido à recombinação levando a um desempenho sub-ótimo. Uma recombinação uniforme costuma ser mais recomendável nesses casos.

O mecanismo de recombinação é relativamente simples envolvendo somente a geração de

números aleatórios, cópia do genótipo e a troca parcial de código genético. Apesar disso, a troca de informação estruturada, porém aleatória, realizada pelo operador de recombinação fornece grande parte do poder de busca do algoritmo genético.

### 3.5.2 Mutação

Ao contrário do operador de recombinação discutido anteriormente o operador de mutação recebe um indivíduo como operando, age apenas em um gene e retorna um novo indivíduo como resultado:

$$\xi : I^l \times \delta \mapsto I^l \tag{3.11}$$

onde $\xi$ indica o operador de mutação.

A mutação é um processo de busca aleatória e é necessária porque ocasionalmente a seleção natural e a recombinação podem eliminar material genético potencialmente promissor da população antes que ele seja avaliado adequadamente. Nos algoritmos genéticos a mutação evita este tipo de perda através de uma alteração aleatória do valor codificado em um determinado locus gênico (GOLDBERG, 1989).

Considerando um esquema, a probabilidade de uma mutação destruir este padrão em um vetor é dada pela relação entre o tamanho do padrão e o tamanho do cromossomo, independente da codificação utilizada e, ao contrário da recombinação, da posição dos genes do esquema. Formalmente:

$$P_{Hm} = p_m.o(\mathbb{H}) \tag{3.12}$$

onde:

- $P_{Hm}$ é a probabilidade de quebra do esquema por uma operação de mutação;
- $p_m$ é a probabilidade de mutação;
- $o(\mathbb{H})$ é o número de genes relevantes no esquema.

Em geral, o operador de mutação depende da codificação escolhida, em particular do alfabeto do gene. Para um alfabeto binário a mutação consiste somente na troca do valor do gene por seu contrário. Para um alfabeto inteiro, o operador de mutação pode atuar trocando somente um bit em seu gene ou substituindo o gene por outro sorteado aleatoriamente. As duas formas de mutação apresentam desempenho similar mas o custo computacional da troca de bits

é sensivelmente menor. Quando o algoritmo utiliza a codificação real, os dois operadores mais comuns são a mutação não uniforme e uniforme. Neste caso, a mutação uniforme substitui o valor do gene sem utilizar informação sobre seu valor atual, substituindo-o por um valor aleatório. Para a codificação real, Janikow e Michalewicz (1991) investigaram o efeito da mutação não uniforme, em que o passo da mutação depende de uma realização aleatória $0 < r < 1$ e do contador da geração $t$:

$$\epsilon(h_k^t) = \begin{cases} h_k^t + \Delta(t, \max(I) - h_k^t), r \geq 0.5 \\ h_k^t - \Delta(t, h_k^t - \min(I)), r < 0.5 \end{cases} \tag{3.13}$$

para $k = 1, \ldots, l$. A função $\Delta(t, y)$ retorna um valor no intervalo $[0, y]$ tal que a probabilidade de $\Delta(t, y)$ se aproximar de zero aumenta com o crescimento de $t$. O efeito deste operador causa uma busca uniforme no início da evolução (devido aos baixos valores de $t$) que se torna restrita nos estágios finais da evolução. A forma mais comum para a função $\Delta(t, y)$ é ilustrada na equação 3.14:

$$\Delta(t, y) = y.r.\left(1 - \frac{t}{T}\right)^b \tag{3.14}$$

onde $r$ é uma realização aleatória em $[0, 1]$, $T$ é o número máximo de gerações e $b$ é o parâmetro do grau de não-uniformidade do operador.

## 3.6 Mecanismo de seleção

O mecanismo de seleção consiste em escolher, dentre a população de soluções-candidatas, um conjunto de soluções de forma a privilegiar indivíduos com aptidão mais alta. A importância de uma especificação adequada do mecanismo de seleção é crucial para o desempenho de um algoritmo genético pois este componente afeta significativamente a pressão seletiva do algoritmo genético. Esse mecanismo é uma forma simplificada do princípio da sobrevivência do mais apto. Formalmente:

$$\zeta : I^{\varphi \times l} \times \delta \mapsto I^{\lambda \times l} \tag{3.15}$$

onde $\zeta$ indica o mecanismo de seleção, $\lambda$ indica o número de descendentes e $l$ é o comprimento do cromossomo.

### Seleção por roleta

Uma forma intuitiva de tratar o princípio da sobrevivência do mais apto é atribuir uma probabilidade de sobrevivência proporcional à aptidão de cada indivíduo. Essa forma de seleção também é conhecida como seleção proporcional ao desempenho ou seleção por roleta. Seja a

população do algoritmo em uma geração qualquer $j$:

$$P^j = \{\mathbf{h}_1^j, \ldots, \mathbf{h}_\varphi^j\} \tag{3.16}$$

Cada indivíduo possui uma medida de desempenho $F(h_i^j) \in \mathbb{R}^+$. A probabilidade do indivíduo $i$ ser selecionado é dada por:

$$P_{sel}(\mathbf{h}_i^j) = \frac{F(\mathbf{h}_i^j)}{\sum_{k=1}^{\varphi} F(\mathbf{h}_k^j)} \tag{3.17}$$

Este tipo de seleção apresenta problemas se os indivíduos da população apresentarem medidas de desempenho muito próximas. Além disso, se um indivíduo apresenta desempenho significativamente maior que o resto da população, existe uma grande chance de ocorrer a convergência prematura do algoritmo devido à diminuição abrupta da diversidade genética da população. Nestes casos, mecanismos de controle da pressão seletiva do algoritmo genético podem atenuar este efeito (GOLDBERG, 1989). Este tipo de seleção ainda exige a avaliação de todos os indivíduos da população aumentando significativamente o custo computacional do algoritmo.

#### Seleção por torneio

A seleção por torneio é uma opção atraente para substituir a seleção por roleta devido à sua menor pressão seletiva e custo computacional (DELGADO, 2002). Seja $k \in \{2, \ldots, \varphi\}$ o tamanho do torneio e $B(\mathbf{h}_i^j)$ uma medida de desempenho ou variedade genética. O processo de seleção consiste em obter um vetor de indivíduos $\mathbf{v} \in I^{k \times l}$ e escolher $h$ de forma que:

$$h = \arg\max_{\mathbf{h}_i^j \in \mathbf{v}} B(\mathbf{h}_i^j) \tag{3.18}$$

Uma outra variedade de torneio utiliza uma realização aleatória entre $[0, 1]$ e, caso essa realização seja menor que uma probabilidade $p$, um outro indivíduo é escolhido para sobreviver.

## 3.7 Atualização da população

Utilizando $O \in I^\lambda$ para indicar o conjunto de descendentes, uma geração do algoritmo pode ser escrita como (SMITH, 1998):

$$V^t = \zeta(P^t, \delta^t) \tag{3.19}$$

$$A_i^t = \rho(V^t, \delta^t), \forall i \in \{1 \cdots \lambda\} \tag{3.20}$$

$$O_i^t = \xi(A_i^t, \delta^t), \forall i \in \{1 \cdots \lambda\} \tag{3.21}$$

$$P^{t+1} = U(O^t \cup P^t) \tag{3.22}$$

onde $A_i^t$ e $V^t$ não representam entidades significativas e foram utilizados somente com fins de simplificação da notação.

O algoritmo genético proposto por Holland utiliza uma abordagem na qual toda a população de uma geração é substituída por seus descendentes na geração seguinte, ou seja $\lambda = \varphi$ (GOLDBERG, 1989). Holland (1992) e de Jong (1975), também consideraram o efeito da substituição de uma proporção menor da população em cada geração. Outra classe comum de estratégia de atualização de população são os algoritmos genéticos de estado estacionário (*steady state AG's*). Eles substituem apenas uma pequena porção da população a cada geração ($\lambda \ll \phi$) (SMITH, 1998). É possível demonstrar que, sob determinadas condições, as duas classes de algoritmos genéticos apresentam o mesmo comportamento.

Supondo $m(\mathbb{H}, t)$ o número de indivíduos pertencentes a um esquema durante a geração $t$, o número de indivíduos pertencentes a este esquema na próxima geração é dado pelo Teorema do esquema (GOLDBERG, 1989):

$$m(\mathbb{H}, t+1) \geq m(\mathbb{H}, t)\overline{P}_{sel}(\mathbb{H})\,(1 - P_{Hr} - P_{Hm}) \tag{3.23}$$

Para um algoritmo genético com seleção por roleta e recombinação de um-ponto, desconsiderando o efeito do $r$-Elitismo que será apresentado na seção 3.7.1 tem-se:

$$m(\mathbb{H}, t+1) \geq m(\mathbb{H}, t)\frac{\overline{F}(\mathbb{H}, t)}{\overline{F}(P^t)}\left[1 - p_c\frac{d(\mathbb{H})}{l-1} - p_m.o(\mathbb{H})\right] \tag{3.24}$$

onde:

- $\overline{F}(\mathbb{H}, t)$ é a aptidão média dos indivíduos de um determinado esquema na geração $t$;
- $\overline{F}(P^t)$ é a aptidão média dos indivíduos de toda a população na geração $t$.

Portanto, esquemas que apresentem uma aptidão maior que a média $\overline{F}(\mathbb{H}, t) > \overline{F}(P^t)$ apresentam uma tendência a aumentar seu número de indivíduos, ou seja, $m(\mathbb{H}, t+1) > m(\mathbb{H}, t)$. Este crescimento é restrito pela ação dos operadores genéticos ou operadores de variação. Nota-se

que não é necessário que os genes do esquema codifiquem variáveis do problema. Essa observação é importante para a compreensão do mecanismo de auto-adaptação pois os genes que codificam os parâmetros evolutivos não afetam diretamente a aptidão dos indivíduos associados a seus esquemas.

### 3.7.1 *r*-Elitismo

O termo elitismo, introduzido por de Jong (1975), está associado à adoção de uma operação de reprodução $G : I^{n\times l} \mapsto I^{r\times l}$ que reproduz os $r$ melhores indivíduos de uma geração para a geração seguinte. Estes indivíduos poderiam ser perdidos caso não fossem reproduzidos de forma determinística para a próxima geração, ou se sofressem a ação dos operadores de variação. Geralmente, estratégias elitistas associadas aos métodos de seleção melhoram o desempenho de um AG (MITCHELL, 1996).

Supondo $r = 1$, tem-se:

$$G(P^t) = \arg \max_{i\in\{1,\ldots,\phi\}} F(\mathbf{h}_\mathbf{i}^\mathbf{t}) \tag{3.25}$$

onde $\mathbf{h}_\mathbf{i}^\mathbf{t}$ é um indivíduo da população $P^t$ na geração $t$ e $F(\mathbf{h})$ é a função de avaliação para o indivíduo $\mathbf{h}$.

O funcionamento básico de um algoritmo genético clássico (sGA) (GOLDBERG, 1989) pode ser descrito pelo fluxograma da figura 3.1, adaptada de Cordón *et al.* (2001a)

## 3.8 Parâmetros evolutivos

As duas etapas mais importantes na aplicação de um algoritmo evolutivo de busca para um problema particular são a especificação da representação (codificação) e da função de avaliação (aptidão). Estas duas etapas realizam a correspondência entre o contexto original do problema e o algoritmo heurístico de solução do problema. Deve-se ressaltar, conforme visto anteriormente, que algoritmos genéticos, assim como outros algoritmos baseados em heurísticas, nem sempre armazenam as variáveis do problema de fato. A forma como estes parâmetros são armazenados e processados tem grande impacto no desempenho do algoritmo de solução.

A definição do algoritmo evolutivo envolve, também, a escolha de alguns componentes como os operadores de variação (geralmente mutação e recombinação) correspondentes à representação, um mecanismo de seleção para escolher os sobreviventes e uma população inicial (EIBEN; HINTERDING; MICHALEWICZ, 1999; MARUO; LOPES; DELGADO, 2004). Cada um destes componentes pode conter parâmetros, por exemplo: a probabilidade de muta-

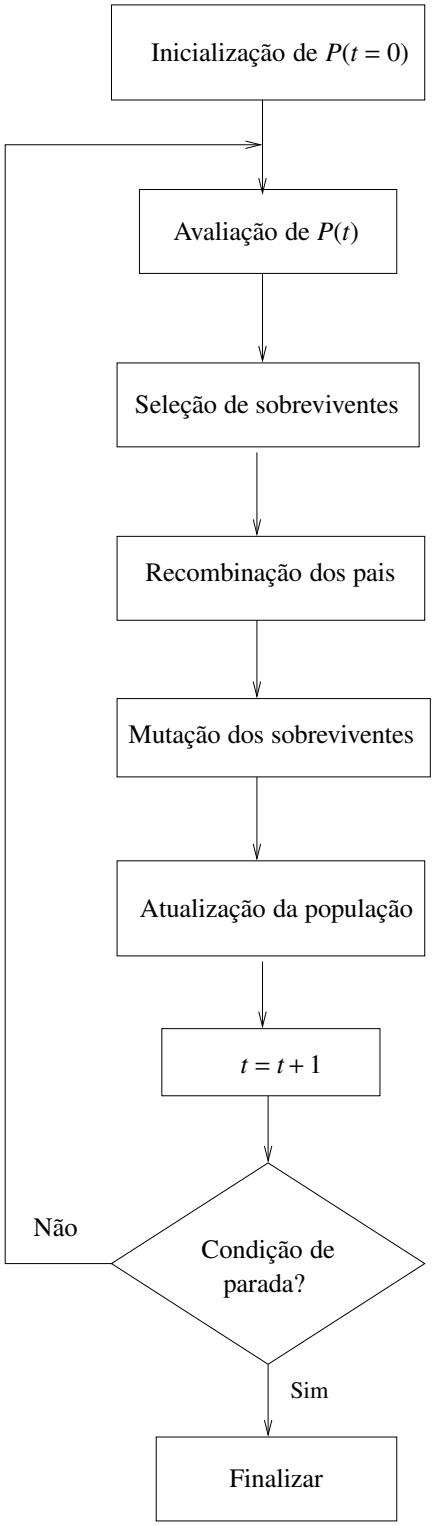



Figura 3.1: Estrutura principal de um AG (CORDÓN *et al.*, 2001a)

ção, o tamanho do torneio utilizado na seleção ou o tamanho da população. Os valores utilizados para estes parâmetros irão influir decisivamente se o algoritmo irá encontrar soluções próximas ao ótimo e se ele encontrará esta solução de forma eficiente.

Segundo Eiben, o ajuste dos parâmetros associados à rodada de um algoritmo evolutivo é um problema mal-estruturado, mal-definido e complexo (EIBEN; HINTERDING; MICHALEWICZ, 1999). E é exatamente nesta classe de problemas que algoritmos evolutivos apresentam um desempenho destacado. Apesar de sua importância, não existe uma metodologia consistente para a determinação dos parâmetros evolutivos que são, geralmente, arbitrados dentro de intervalos pré-definidos. A utilização do algoritmo evolutivo, não apenas para encontrar a solução do problema, mas também para ajustar uma série de parâmetros do próprio algoritmo parece ser uma alternativa razoável (MARUO; LOPES; DELGADO, 2004) e constitui uma das principais contribuições deste trabalho.

De forma geral, são observadas duas principais abordagens para definição dos valores destes parâmetros: o ajuste de parâmetros e o controle de parâmetros. A abordagem de ajuste de parâmetros corresponde à prática de tentar encontrar bons valores para estes parâmetros antes da rodada do algoritmo e a aplicação destes valores, que devem permanecer constantes durante toda a execução. Na seção 3.8.2 discute-se o fato de que que normalmente um conjunto de parâmetros que permanecem fixos durante a rodada do algoritmo parece ser inapropriado, independente do problema. O controle de parâmetros é uma abordagem alternativa na qual a rodada inicia com um conjunto de parâmetros que sofrem alterações durante o processo evolutivo.

### 3.8.1 Classificação

De forma geral, na classificação das técnicas de determinação de parâmetros evolutivos, alguns aspectos podem ser considerados:

1. Quais componentes do algoritmo sofrerão a ação do controle de parâmetros?
2. A forma como será feita esta mudança (isto é, será usada uma regra determinística, uma regra baseada no estado da evolução ou uma abordagem auto-adaptativa?).
3. O escopo/nível de mudança (o controle irá agir em nível populacional? Em nível individual? Em somente um gene?).
4. A evidência sobre a qual será baseada a mudança dos parâmetros (monitoração do desempenho dos operadores, diversidade da população, etc).

Alguns componentes são mais adequados para sofrer controle de parâmetros (EIBEN; HINTERDING; MICHALEWICZ, 1999):

- codificação;
- função de avaliação;
- operadores de variação e suas probabilidades;
- mecanismo de seleção;
- mecanismo de substituição;
- população (tamanho e topologia).

Em relação à forma como será realizada a adaptação, as técnicas de controle de parâmetros podem ser classificadas em três grande grupos (EIBEN; HINTERDING; MICHALEWICZ, 1999):

**(i) Controle determinístico** ocorre quando o valor de um parâmetro é alterado segundo alguma regra determinística. Esta regra independe do estado da evolução. Normalmente, uma regra baseada no número da geração atual é utilizada.

**(ii) Controle adaptativo** ocorre quando existe alguma forma de realimentação do estado da busca e esta informação é utilizada para determinar a direção e a magnitude da mudança efetuada.

**(iii) Controle auto-adaptativo** a idéia da "evolução da evolução" pode ser utilizada para implementar a auto-adaptação dos parâmetros. Nesta forma, os parâmetros controlados são codificados no cromossomo e sofrem a ação dos operadores genéticos da mesma forma que os genes que codificam a solução do problema. Os valores adequados para estes parâmetros estratégicos têm uma maior probabilidade de gerar bons indivíduos que terão uma maior chance de sobreviver, produzir descendentes e propagar estes parâmetros evolutivos.

Utilizando esta terminologia, e adicionando o ajuste de parâmetros, pode-se construir a árvore da figura 3.2:

Em relação ao critério "escopo da mudança", uma atualização dos parâmetros estratégicos pode afetar somente um gene (mudança paramétrica), um indivíduo (mudança individual) ou a

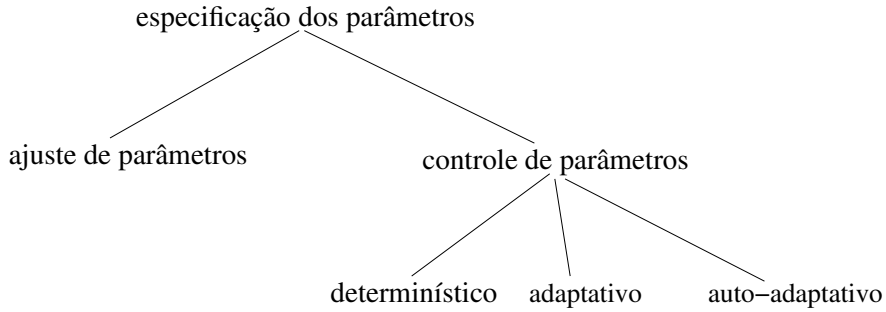



Figura 3.2: Taxonomia da especificação de parâmetros de um algoritmo evolutivo (EIBEN; HINTERDING; MICHALEWICZ, 1999)

população inteira (mudança populacional), outros componentes ou mesmo a função de avaliação. Por exemplo uma mudança no passo da mutação pode afetar um gene, um indivíduo ou toda a população, dependendo da implementação. Em contrapartida, uma mudança na função de avaliação sempre agirá no escopo populacional. Logo o critério do escopo normalmente depende do componente adaptado e de sua implementação. Por este motivo, o critério do escopo é considerado por muitos autores como secundário (EIBEN; HINTERDING; MICHALEWICZ, 1999).

Outro critério possível é a evidência sobre a qual será baseada a atualização do parâmetro. A forma mais usual é monitorar o progresso do processo evolutivo, isto é, o desempenho dos parâmetros estratégicos. Entretanto outras medidas podem ser utilizadas, sendo a mais comum a diversidade genética da população. Esta informação é utilizada como entrada de um processo que tem como saída os novos parâmetros dos componentes do algoritmo. Apesar de ser uma distinção significativa, este critério é válido somente para o controle de parâmetros adaptativo. Devido a esta restrição muitos autores consideram este critério como de importância secundária.

A classificação da forma de especificar parâmetros proposta por Eiben, Hinterding e Michalewicz (1999) é baseada em dois aspectos: como o mecanismo de adaptação funciona e a escolha dos componentes do algoritmo que sofrerão ação deste mecanismo. A classificação de Eiben foi feita com base na compilação de outras formas de classificação (ANGELINE, 1995; HINTERDING; MICHALEWICZ; EIBEN, 1997; SMITH; FOGARTY, 1997). Cada uma com diferentes critérios que resultam em diferentes esquemas de classificação.

A classificação proposta por Angeline (1995) é baseada no nível de adaptação e no tipo de regra de atualização. Em particular, três níveis de adaptação são considerados: nível populacional, nível individual e nível de componente. Ao mesmo tempo existem dois tipos de mecanismos de atualização: absoluto e regras empíricas. As regras absolutas são pré-determinadas e especificam como as mudanças devem ser efetuadas. Em contrapartida, regras empíricas modi-

ficam os valores dos parâmetros através da competição entre eles. A classificação de Angeline considera o algoritmo como um todo, sem analisar individualmente seus componentes (mutação, recombinação, seleção, etc).

A classificação proposta por Hinterding, Michalewicz e Eiben (1997) difere da classificação de Angeline ao considerar um nível adicional de adaptação (nível ambiental), e faz uma divisão mais detalhada do tipo e mecanismos de atualização, dividindo-os em categorias determinística, adaptativa e auto-adaptativa. Assim como a classificação de Angeline, o algoritmo é considerado como um todo, sem analisar seus componentes isoladamente.

A classificação de Smith e Fogarty (1997), Smith (1998) é a mais detalhada. Ela é baseada em três critérios de divisão: o que está sendo adaptado, o escopo da adaptação e a base para mudança. Este último critério é ainda dividido em duas categorias: a evidência na qual se baseia a mudança e a regra que executa esta mudança. Além disso existem dois tipos de regras: absolutas e empíricas.

Segundo Smith (1998) os primeiros estudos no controle de parâmetros podem ser classificados conforme ilustrado na figura 3.3. Utilizar-se-á ao longo deste capítulo o critério de classificação de Eiben baseado nos dois principais critérios de classificação: o que é modificado e como esta mudança é realizada. Este tipo de classificação é ortogonal (ou seja, seus critérios são independentes entre si) e consiste em três formas de mudança: determinística, adaptativa e auto-adaptativa e seis categorias de componentes: representação, função de avaliação, operadores de variação, seleção, substituição e população.

### 3.8.2 Ajuste de parâmetros

Inicialmente, os algoritmos genéticos utilizavam codificação binária, recombinação de um ponto, mutação por negação de bit e seleção por roleta (com ou sem elitismo). O projeto do algoritmo era limitado à escolha dos parâmetros evolutivos, ou parâmetros estratégicos, como taxa de mutação, taxa de recombinação e tamanho da população. O ajuste do conjunto de parâmetros era puramente empírico não havendo uma justificativa razoável para a escolha realizada.

Dois trabalhos foram especialmente importantes na tentativa de incrementar o ajuste de parâmetros. de Jong (1975) utilizou um extenso conjunto de problemas de otimização e encontrou um conjunto de valores para os parâmetros estratégicos que tiveram bom desempenho para grande parte destes problemas. Grefenstette (1986) utilizou o algoritmo genético como um meta-algoritmo para otimizar os mesmos parâmetros entretanto utilizando um conjunto diferente de problemas. Os valores encontrados para os parâmetros estratégicos estão na tabela

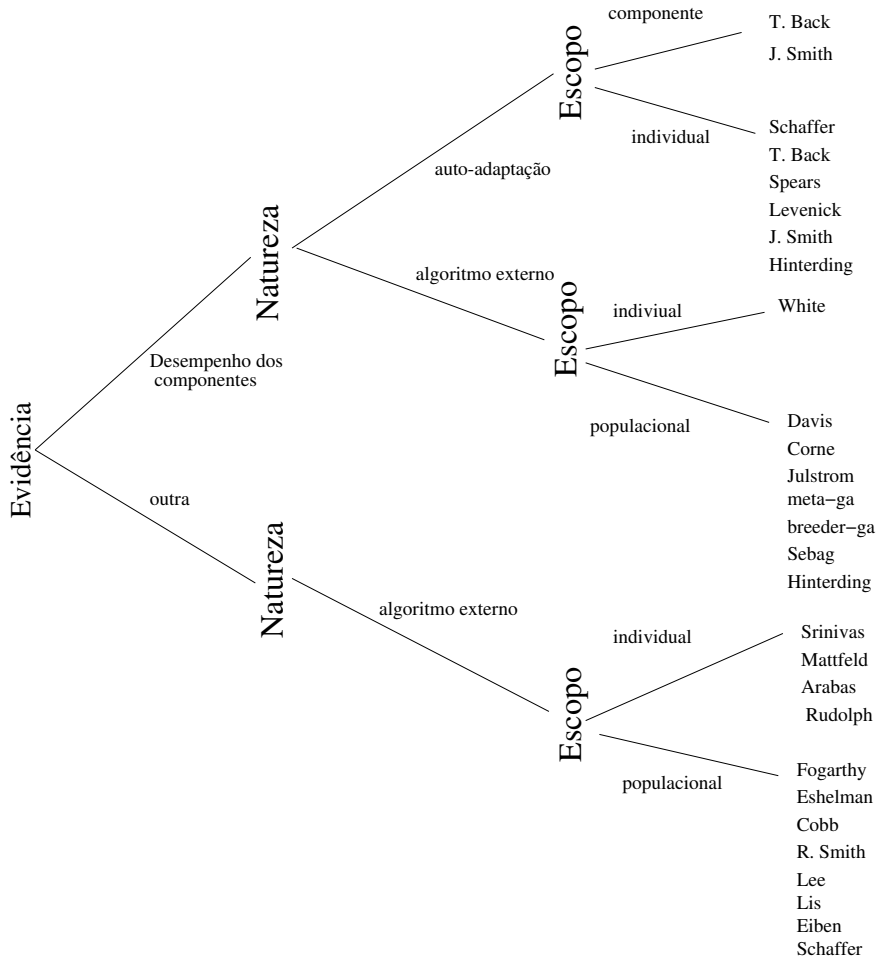



Figura 3.3: Classificação dos primeiros trabalhos em controle de parâmetros (SMITH, 1998)

3.1.

É importante salientar que em ambos os experimentos o objetivo foi encontrar um conjunto de parâmetros ótimo e geral. Neste contexto o termo geral se refere ao fato destes parâmetros apresentarem bom desempenho para um amplo espectro de problemas (GOLDBERG, 1989). Entretanto, a visão contemporânea sobre os algoritmos evolucionários em geral reconhece que tipos específicos de problemas demandam configurações específicas no algoritmo evolucionário para atingir um desempenho satisfatório (BÄCK; FOGEL; MICHALEWICZ, 1997). Assim assume-se que o escopo de um conjunto ótimo de parâmetros deve ser necessariamente restrito. Desta forma, fica reforçada a necessidade de encontrar técnicas eficientes para facilitar a busca de bons conjuntos de parâmetros que sejam específicos para uma aplicação. Os parâmetros de De Jong se tornaram populares devido à ausência de conhecimento adicional em muitas aplicações. Devido a sua maior popularidade, os parâmetros de De Jong foram escolhidos como valores de referência nos experimentos.

Tabela 3.1: Conjunto de parâmetros estratégicos de de Jong (1975) e Grefenstette (1986)

| Parâmetro | De Jong | Grefenstette |
|---|---|---|
| tamanho da população | 50 | 80 |
| probabilidade de recombinação | 0.6 | 0.45 |
| probabilidade de mutação | 0.001 | 0.01 |
| estratégia de seleção | elitista | elitista |

A forma tradicional de ajuste de parâmetros altera os valores de somente um parâmetro por vez, podendo acarretar em escolhas sub-ótimas. Isso ocorre devido à forma complexa com que os parâmetros estratégicos interagem (MARUO; LOPES; DELGADO, 2004). Em contrapartida, o ajuste de mais parâmetros ao mesmo tempo demanda uma grande quantidade de experimentos. De forma geral, as desvantagens da abordagem de ajuste de parâmetros baseada em experimentação podem ser resumidas em:

- Parâmetros não são independentes, mas tentar todas as diferentes combinações de valores é virtualmente impossível;
- O processo de ajuste de parâmetros é demorado, mesmo que a otimização ocorra somente com um parâmetro, desprezado suas interações;
- Para um dado problema os parâmetros encontrados não são necessariamente ótimos, mesmo se o processo de busca demandou um esforço computacional significativo (EIBEN; HINTERDING; MICHALEWICZ, 1999)

Uma desvantagem da abordagem de ajuste de parâmetros, independente da forma como eles são definidos, é baseada na observação de que a rodada de um algoritmo evolucionário é um processo inerentemente dinâmico e adaptativo e o uso de parâmetros estáticos parece ser contraditório. Reforçando esta premissa, alguns resultados empíricos indicam que, para diferentes etapas do processo evolutivo, valores distintos para os parâmetros estratégicos apresentam características mais interessantes (MARUO; LOPES; DELGADO, 2004; BÄCK, 1992a, 1992b, 1993; DAVIS, 1989; HESSER; MäNNER, 1991; SOULE; FOSTER, 1997; SYSWERDA, 1991). Por exemplo, mutações mais agressivas podem ser benéficas nas gerações iniciais, colaborando para uma exploração mais ampla do espaço de busca enquanto mutações menos abruptas podem ser necessárias nas gerações finais no intuito de realizar um ajuste fino em indivíduos sub-ótimos. Isto indica que o uso de parâmetros estáticos pode ser uma causa de perda de desempenho do algoritmo.

Outras alternativas para a abordagem de ajuste por experimentação incluem o ajuste de parâmetros por analogia e o uso de análise teórica (HESSER; MäNNER, 1991). O ajuste de parâ-

metros por analogia consiste em utilizar um conjunto de parâmetros que se mostraram eficazes para um problema similar ao proposto. Não é claro, entretanto, até que ponto a similaridade entre problemas percebida pelo usuário indica que o conjunto de parâmetros estratégicos também seja similar. Quanto à análise teórica, atualmente a complexidade do processo evolutivo e as características dos problemas interessantes limita o uso desta técnica (somente após simplificações significativas no algoritmo ou no modelo do problema). Portanto a eficiência destes resultados teóricos ainda é discutível (EIBEN; HINTERDING; MICHALEWICZ, 1999). Existem resultados interessantes na investigação do tamanho ótimo da população (GOLDBERG, 1989; GOLDBERG; DEB; CLARK, 1992; HARIK *et al.*, 1997; THIERENS, 1996) e taxas dos operadores genéticos (BÄCK, 1993; GOLDBERG; DEB; THIERENS, 1991; SCHAFFER; MORISHIMA, 1987; THIERENS; GOLDBERG, 1993), porém estes resultados foram obtidos somente para problemas de otimização com funções simples e sua aplicabilidade a outros tipos de problema permanece indefinida.

### 3.8.3 Formas de controle de parâmetros

Conforme mencionado na seção 3.8.1, utilizando a forma como os parâmetros são atualizados, segundo a classificação de Hinterding, Michalewicz e Eiben (1997) pode-se identificar três principais formas de controle de parâmetros: controle determinístico, adaptativo e auto-adaptativo.

#### Determinístico

A forma mais simples de controle de parâmetros é o controle determinístico. Uma maneira de realizar este tipo de controle é substituir um parâmetro qualquer $p_{estat}$ por uma função $p_{det}(t)$ onde $t$ indica o contador de gerações (EIBEN; HINTERDING; MICHALEWICZ, 1999). Conforme indicado na seção 3.8.2 o problema de encontrar parâmetros estáticos adequados para um problema específico é complexo e dependente de outros fatores. Desta forma, a busca por uma função $p_{det}(t)$ conveniente é um problema ainda mais complexo. Outra possível desvantagem desta abordagem é o fato de mudanças dos parâmetros estratégicos serem comandadas somente pelo contador de gerações $t$ sem utilizar nenhuma informação sobre o estado da busca, ou seja, o usuário tem total controle sobre o valor do parâmetro estratégico $p_{det}(t)$ e seu valor é totalmente previsível. Entretanto existem vários casos na literatura de pesquisadores que conseguiram incrementar o desempenho de seus algoritmos evolutivos utilizando regras determinísticas. Segundo Eiben, Hinterding e Michalewicz (1999) isso pode ser explicado pela simples mudança dos valores. Segundo ele, uma escolha sub-ótima de $p_{det}(t)$ freqüentemente

leva a melhores resultados que uma escolha sub-ótima de $p_{estat}$.

Um dos primeiros estudos sobre o controle da taxa ótima de mutação foi realizado por Fogarty (1989) que utilizou um mecanismo determinístico para diminuir a taxa de mutação $p_m(t)$ ao longo do processo evolutivo. Em outro trabalho, Hesser e Männer (1991) derivaram uma função para $p_m(t)$ segundo a equação 3.26:

$$p_m(t) = \sqrt{\frac{\alpha}{\beta}} \times \frac{e^{(\frac{-\gamma t}{2})}}{\varphi \sqrt{L}} \tag{3.26}$$

onde $\alpha, \beta, \gamma$ são constantes, $\varphi$ é o tamanho da população e $t$ é o contador de gerações.

Janikow e Michalewicz (1991) investigaram o efeito da mutação não uniforme, em que o passo da mutação depende de uma realização aleatória $0 < r < 1$ e do contador da geração $t$, conforme visto nesse capítulo. A diminuição do passo de mutação pode ser interessante em estágios avançados da evolução em que algumas soluções necessitam apenas de ajustes-finos em seus parâmetros.

### Adaptativo

Um dos primeiros trabalhos no controle de parâmetros adaptativo foi realizado por Bäck (1993) no estudo das taxas de mutação como uma função da distância da solução ao ótimo, sendo:

$$p_m(F(\mathbf{x})) \approx \frac{1}{2\,(F(\mathbf{x}) + 1) - l} \tag{3.27}$$

Uma desvantagem dessa abordagem é a restrição da função de aptidão das soluções ao intervalo $[0, 1]$. É possível realizar uma normalização de $F(\mathbf{x})$ para satisfazer essa restrição, entretanto essa operação exige o conhecimento, ou uma boa estimativa, da aptidão da solução ótima. Essa informação nem sempre está disponível.

Uma das formas mais simples de incorporar o controle de parâmetros adaptativo é utilizar a regra de 1/5 de sucessos de Rechenberg (1973). Esta regra afirma que a proporção de mutações bem-sucedidas em relação a todas as mutações deve ser 1/5. Assim, se a proporção for maior que 1/5 então o passo da mutação deve ser aumentado e se a proporção for menor que 1/5 o passo deverá ser diminuído. Uma mutação é considerada bem-sucedida se produz um descendente mais apto que seu progenitor. De forma opcional esta verificação pode ser realizada apenas a cada $q$ gerações. Seja $\sigma(t)$ uma métrica do passo da mutação durante a geração $t$, o parâmetro $\sigma(t)$ pode ser, por exemplo, a distância entre o limite superior e inferior da realização aleatória que define o passo da mutação, se a função densidade de probabilidade da distribuição

for uniforme. A métrica $\sigma(t)$ pode ser também o desvio padrão da função densidade de probabilidade para um passo de mutação com distribuição Gaussiana. Então a atualização dessa métrica pode ser alterada da seguinte maneira:

$$\sigma(t) = \begin{cases} \sigma(t-q), & t \text{ módulo } q \neq 0 \\ \text{caso contrário} & \begin{cases} \sigma(t-q)/c, & \text{se } p_s > 1.5 \\ \sigma(t-q).c, & \text{se } p_s < 1.5 \\ \sigma(t-q), & \text{se } p_s = 1.5 \end{cases} \end{cases} \tag{3.28}$$

onde $p_s$ é a proporção de mutações bem-sucedidas e $0.817 \leq c \leq 1$ modula a intensidade da mudança.

### 3.8.4 Auto-adaptativo

O princípio da auto-adaptação consiste em identificar os parâmetros mais interessantes dos componentes de um algoritmo evolutivo e adicionar um conjunto de genes a cada cromossomo, codificando estes parâmetros evolutivos. Esses genes não devem, teoricamente, influir diretamente no valor da aptidão dos indivíduos. Entretanto esses genes também constituem esquemas. Conforme a equação 3.23, esquemas que contenham indivíduos mais aptos tendem a dominar a população (GOLDBERG, 1989). Da mesma forma, parâmetros evolutivos capazes de gerar indivíduos mais aptos tendem a ter uma aptidão média maior em relação a esquemas menos eficazes. Dessa forma o uso de auto-adaptação não requer mudanças na função de avaliação. O próprio mecanismo de seleção do algoritmo genético permite a sobrevivência de bons indivíduos gerados por bons parâmetros evolutivos.

Os cromossomos, entretanto, devem ser modificados para conter os genes associados a seus parâmetros evolutivos. Considerando que o número de parâmetros associados aos componentes adaptados é $p$, cada indivíduo é codificado através de um cromossomo com $l = l_{sol} + p$ genes. Onde $l_{sol}$ é o número de genes codificando as variáveis do problema. Seja $\mathbf{h}_i$ um cromossomo qualquer, sua nova codificação será:

$$\mathbf{h}_i = [\mathbf{h}_i^{sol} \partial_i^t] \tag{3.29}$$

onde $\partial_i^t \subseteq \delta_i^t$ indica o conjunto genes correspondentes aos parâmetros evolutivos do indivíduo que sofrerão auto-adaptação.

Durante o processo evolutivo, os operadores tratam uniformemente todos os genes dos cromossomos ignorando sua função (codificação da solução ou dos parâmetros estratégicos). Portanto, o uso da auto-adaptação não demanda modificações drásticas nos operadores de um algoritmo genético convencional. Assim, o uso de auto-adaptação de parâmetros é uma alter-

nativa relativamente simples de controle de parâmetros não exigindo grandes mudanças em um algoritmo genético convencional.

Um dos primeiros trabalhos nessa área foi realizado por Bäck (1992a) em que um AG foi modificado adicionando 20 bits ao genótipo do problema, que foram utilizados para codificar a taxa de mutação. Hinterding (1995) aplicou a auto-adaptação para o passo da mutação em problemas numéricos de otimização para um AG com codificação real. A auto-adaptação também é aplicável a problemas não-numéricos. Fogel, Angeline e Fogel (1995) utilizaram a auto-adaptação para controlar as probabilidades relativas dos cinco operadores de mutação para os componentes de uma máquina de estados finita. Hinterding (1997) utilizou um AG com abordagem *Michigan* para resolver problemas de corte de guilhotina. Um cromossomo é utilizado para representar a solução utilizando codificação mista enquanto outro cromossomo codifica os parâmetros utilizando codificação real. Neste caso a auto-adaptação é utilizada para evoluir a probabilidade de utilizar um entre dois operadores de mutação. Tao e Michalewicz (1998) propuseram um operador adaptativo para problemas de permutação. O operador foi denominado *inver-over* e aplica um número variável de inversões em um cromossomo. Além disso, o segmento a ser invertido é definido através de outro indivíduo selecionado aleatoriamente.

Spears (1995) utilizou a auto-adaptação no operador de recombinação aplicando o mesmo princípio de Bäck para adicionar um bit ao genótipo de seu GA que codifica o tipo de recombinação (escolhida entre uniforme e de dois-pontos). White e Oppacher (1994) limitaram o universo de probabilidades de recombinação para $p_c \in \{0, \frac{1}{N}, \frac{2}{N}, \ldots, 1\}$. Eles adicionaram $l_{sol}$ genes ao genótipo de cada indivíduo indicando a probabilidade de ocorrer recombinação uniforme nesse alelo. Para o gene $j$ nos indivíduos $i$ e $k$ ocorrerá recombinação se $\sqrt{p_c(h_i^j) \times p_c(h_k^j)} \geq r$ onde $r$ é uma realização aleatória $r \in [0, 1]$. Além disso se ocorrer cruzamento no ponto $j$ os descendentes deverão herdar também $p_c(h^j)$ dos pais. É possível, ainda utilizar a auto-adaptação em recombinações com mais de dois pais. Nesse caso existe um parâmetro adicional: a aridade da recombinação, ou seja, o número de indvíduos envolvidos em cada recombinação específica. Eiben, Sprinkhuizen-Kuyper e Thijssen (1998) propuseram um mecanismo de auto-adaptação da aridade da recombinação utilizando sub-populações concorrentes. A população é dividida em sub-populações disjuntas, cada uma utilizando uma aridade. Essas sub-populações evoluem independentemente por um período de tempo e trocam informação ao permitirem a migração de indivíduos após certo período. Um mecanismo evita a extinção prematura de sub-populações com baixo desempenho.

A abordagem de utilizar a auto-adaptação em mais de um parâmetro simultaneamente, apesar da evidência sobre a forma complexa com que os parâmetros interagem, não é muito comum

na literatura. Um dos primeiros trabalhos a utilizar a auto-adaptação simultânea de $p_r$ e $p_m$ foi realizado por Srinivas e Patnaik (1994). Mais recentemente Maruo, Lopes e Delgado (2004) utilizaram a auto-adaptação no tipo de recombinação, probabilidade de recombinação, probabilidade de mutação e tamanho de torneio comparando o desempenho da auto-adaptação de múltiplos componentes simultâneos ou isoladamente. Segundo Eiben, Hinterding e Michalewicz (1999) a adaptação simultânea de múltiplos parâmetros pode acarretar problemas relacionados ao comportamento inicial de um algoritmo evolutivo. Supondo uma população constituída de populações disjuntas, cada uma utilizando uma recombinação diferente, se o tamanho dessa sub-população depende do desempenho de seus parâmetros, os operadores que têm baixo desempenho durante um estágio da evolução terão dificuldades em propagar seus esquemas. Essa característica reduz as possibilidades de utilização desses parâmetros em outros estágios da evolução. Nesse contexto, o controle da pressão seletiva torna-se ainda mais importante. Neste trabalho, a auto-adaptação foi utilizada visando a evolução do algoritmo com baixa pressão seletiva. Além dessa preocupação com o desempenho dos operadores, o AG proposto utiliza um mecanismo de seleção por variedade genética visando diminuir a probabilidade de convergência prematura.

# 4 *Projeto automático de sistemas nebulosos utilizando algoritmos genéticos auto-adaptativos*

A modelagem de sistemas com base em ferramentas matemáticas tradicionais (como por exemplo, equações diferenciais) não é adequada para tratar problemas com informações imprecisas ou incertas como aqueles envolvendo o conhecimento humano. Uma das abordagens mais utilizadas no tratamento dessa classe de problemas é o uso de sistemas nebulosos baseados em regras, também conhecidos simplesmente como sistemas nebulosos. A modelagem nebulosa ou identificação nebulosa foi estudada por Takagi e Sugeno (1993) encontrando aplicações nas áreas de controle, predição e inferência (PEDRYCZ; GOMIDE, 1998; SUGENO, 1985). Os principais componentes de um sistema nebuloso, que devem ser especificados durante a modelagem ou projeto do sistema nebuloso são (CORDÓN *et al.*, 2001a):

**Base de dados** contém os termos lingüísticos utilizados nas regras nebulosas e suas funções de pertinência. Define a semântica que associa rótulos aos conjuntos nebulosos. Cada variável lingüística do problema tem uma partição nebulosa que contém os conjuntos nebulosos associados a seu domínio.

**Base de regras** é formada por uma coleção de regras nebulosas que podem ser agregadas por um operador de inferência. Ou seja, múltiplas regras podem ser ativadas pelo mesmo vetor de entrada.

Contudo, vários trabalhos científicos apontam algumas limitações no projeto deste tipo de sistemas:

- o conhecimento sobre o comportamento do sistema nem sempre está disponível;
- o projeto manual de sistemas nebulosos pode ser um processo demorado;
- apesar de sua importância, nem sempre existem metodologias específicas para o projeto de sistemas nebulosos.

Atualmente, o projeto automático de sistemas nebulosos tem considerado um balanço entre os seguintes aspectos: acuidade, interpretabilidade do sistema e autonomia do projeto. Algumas aplicações demandam sistemas nebulosos com saídas acuradas. Neste caso, o critério de acuidade terá um peso maior do que os outros. Entretanto, nos últimos anos, a interpretabilidade do sistema tem sido mais valorizada. O objetivo é obter um sistema com boa acuidade e que, ao mesmo tempo, seja facilmente interpretável por um operador humano. Neste caso é razoável que parte da acuidade seja sacrificada em prol da obtenção de um sistema nebuloso mais simples e mais interpretável. Quanto ao critério de autonomia do projeto, é desejável que o desenvolvimento do sistema nebuloso exija a mínima intervenção do usuário (DELGADO, 2002).

Em projetos com elevado grau de autonomia, ou seja, o desenvolvimento de sistemas nebulosos para os quais a maioria dos parâmetros é definida automaticamente, o número de parâmetros codificados pode tornar-se alto. Nesses casos, a evolução de uma única população de indivíduos, que codifiquem a solução completa para o problema, pode tornar-se inadequada. A sobrevivência de uma boa solução parcial para o problema (como por exemplo uma boa partição nebulosa) pode ser prejudicada por outras soluções parciais codificadas no mesmo indivíduo (como por exemplo uma base de regras ruim).

A abordagem que será apresentada neste capítulo é baseada na co-evolução proposta por Delgado (2002) em que o sistema nebuloso é codificado através de indivíduos em quatro populações distintas. Estas populações codificam soluções parciais do problema através de espécies diferentes e uma função de avaliação especial é necessária para mensurar o desempenho de indivíduos das diferentes espécies. Para garantir os requisitos mínimos de interpretabilidade, é possível estabelecer restrições em diferentes níveis hierárquicos garantindo a exclusão de indivíduos inválidos sob os critérios de interpretabilidade. Por exemplo, no seguinte trabalho, restrições na consistência da base de regras e de visibilidade da partição nebulosa foram utilizadas.

A presença de quatro populações de indivíduos durante o processo evolutivo dificulta a definição dos parâmetros evolutivos. Neste capítulo será abordado o uso da técnica de auto-adaptação de parâmetros discutida no capítulo 3 para tentar diminuir o número de parâmetros evolutivos que devem ser especificados pelo usuário, com ganhos significativos na autonomia do sistema.

Na seção 4.1 serão discutidos os critérios de qualidade das abordagens para o projeto automático de sistemas nebulosos. Na seção 4.2 será discutida a abordagem co-evolutiva e suas deficiências no critério de automação. A seção 4.3.1 discutirá como a abordagem auto-

adaptativa apresentada no capítulo 3 pode ser utilizada para melhorar o desempenho do sistema co-evolutivo no critério de automação. Na seção 4.2.6 é realizada uma revisão do método de identificação dos conseqüentes ótimos de Takagi-Sugeno não-lineares (DELGADO; VON ZUBEN; GOMIDE, 2001).

## 4.1 Aspectos fundamentais no projeto de sistemas nebulosos

Segundo Delgado (2002), existem três aspectos principais a serem considerados durante o projeto automático de sistemas nebulosos: acuidade, interpretabilidade e autonomia. A importância de cada critério dependerá da natureza da aplicação do sistema.

**acuidade** O desempenho de um sistema nebuloso pode ser avaliado pela comparação de sua resposta com um modelo real através de um conjunto de dados de teste[1]. Este critério é particularmente importante quando o sistema nebuloso é utilizado, por exemplo, em aplicações na área de controle. A acuidade é geralmente medida através de uma métrica de erro. Para problemas de aproximação de funções, a métrica mais comum é dada pelo erro quadrático médio (EQM). O EQM é definido como:

$$EQM = \frac{1}{N}\sum_{p=1}^{N}(y_p - \hat{y}(\mathbf{x}_p))^2 \tag{4.1}$$

onde $y_p$ é a saída desejada, $\mathbf{x}_p$ é a entrada $p$ do sistema nebuloso e $\hat{y}(\mathbf{x}_p)$ é a saída estimada do sistema para a entrada $x_p$, onde $p = \{1,\ldots,N\}$.

**interpretabilidade** Recentemente a interpretabilidade do sistema passou a ser outro aspecto fundamental no projeto de sistemas nebulosos. Os primeiros trabalhos utilizando o paradigma de sistemas nebulosos se preocupavam somente com a acuidade do sistema projetado sem se preocupar com a dificuldade de entendimento de seu funcionamento por um operador humano (JIN, 2000; SETNES; BABUSKA; VERBRUGGEN, 1998). De certa forma, estes sistemas operavam de forma semelhante a uma "caixa-preta", nas quais o sistema possui um bom desempenho mas não se sabe exatamente como ele funciona. Contudo, apesar da importância da interpretabilidade dos sistemas nebulosos, não existe nenhuma definição consistente para esse conceito (JIN, 2000). No decorrer desse trabalho será utilizada a definição proposta por Delgado (2002) para o critério da interpretabilidade. Assim serão considerados quatro aspectos:

[1] Um subconjunto dos dados disponíveis que não é utilizado durante a fase de treinamento

**visibilidade** Este critério diz respeito à forma como as partições nebulosas são obtidas pelos modelos de projeto automático. A visibilidade está associada a dois parâmetros complementares ($\gamma$ e $\kappa$) que definem as propriedades de $\gamma$-completude e $\kappa$-sobreposição.

**$\gamma$-completude** Também conhecida como propriedade de cobertura do universo (ESPINOSA; VANDEWALLE, 2000), essa propriedade estabelece um grau mínimo $\gamma$ de sobreposição entre funções de pertinência de forma a garantir o particionamento sem falhas (GONZALEZ; PEREZ, 1998; JANG, 1993). Para partições formadas somente por conjuntos convexos pode-se escrever:

$$\inf_{x_i \in \mathbf{X}_i} \left[ \max_{j,k \in \mathcal{T}(\mathbf{X}_i), j \neq k} \mu_j(x_i), \mu_k(x_i) \right] > \gamma \tag{4.2}$$

onde $x_i$ é um valor qualquer no universo de discurso $\mathbf{X}_i$ da variável de entrada $i$ do sistema nebuloso.

**$\kappa$-sobreposição** Esta propriedade estabelece um grau máximo de sobreposição entre funções de pertinência na obtenção da partição nebulosa do universo (DELGADO; VON ZUBEN; GOMIDE, 2001) evitando a geração de partições com funções de pertinência completamente sobrepostas. Supondo partições formadas somente por conjuntos convexos, tem-se:

$$\sup_{x_i \in \mathbf{X}_i} \left[ \min_{j,k \in \mathcal{T}(\mathbf{X}_i), j \neq k} \mu_j(x_i), \mu_k(x_i) \right] < \kappa \tag{4.3}$$

**simplicidade** Uma forma para avaliar a complexidade de uma regra nebulosa é mensurar o número de antecedentes necessários para ativar essa regra. Para sistemas com muitas variáveis de entrada, o uso de regras lingüísticas com muitas condições no antecedente dificulta o entendimento por um operador humano (ISHIBUCHI; TAKEUCHI; NAKASHIMA, 2001). A inclusão de condições irrelevantes (*don't care conditions*) implementadas pela função de irrelevância descrita na seção 2.1.3 é uma alternativa interessante pois, além de simplificar significativamente a semântica das regras nebulosas, apresenta ganhos na generalização do sistema (ISHIBUCHI; NAKASHIMA, 1999; DELGADO, 2002)

**compactação** Sistemas com muitas regras nebulosas também são considerados de difícil compreensão por um operador humano. Em sistemas com muitas variáveis de entrada ou que apresentem partições com muitos termos lingüísticos, o número possível de regras pode ser exageradamente alto. Segundo Ishibuchi, Takeuchi e Nakashima (2001) pode-se mensurar o grau de compactação da base de regras pelo

número efetivo de regras que compõem o sistema nebuloso.

**consistência** A definição de inconsistência em sistemas nebulosos ocorre quando existem regras com mesmo antecedente mas conseqüentes diferentes. No sistema proposto, o método de obtenção dos conseqüentes de Takagi Sugeno realiza uma verificação de dependência linear entre os graus de ativação das regras através dos dados de treinamento. Para tanto, é calculada a condição de uma matriz contendo os graus de ativação de todas as regras para todos os dados de treinamento. Através da estimativa da condição dessa matriz, é possível verificar a existência de colunas linearmente dependentes. Um caso particular de dependência linear é: $\eta_i(\mathbf{x}_p) = \eta_j(\mathbf{x}_p)$, $\forall p \in \{1,\ldots,N\}$, ou seja, regras com mesmo antecedente. Devido à dependência linear, regras com mesmo antecedente são seqüencialmente eliminadas até restar somente uma. Conseqüentemente, os sistemas nebulosos resultantes são consistentes.

**autonomia do projeto** A ausência de uma metodologia específica para o projeto de sistemas nebulosos, além da complexidade inerente ao problema, motivou o estudo de formas de se adicionar algum mecanismo de aprendizado aos sistemas nebulosos. Entretanto o projeto desse tipo de sistemas é uma tarefa complexa e, mesmo depois de prontos, geralmente esses sistemas possuem parâmetros de entrada. A autonomia do projeto é um critério relativamente subjetivo. Neste trabalho, considera-se que o sistema com mais parâmetros a serem arbitrados possui uma menor autonomia.

O sistema proposto por Delgado (2002) é capaz de definir praticamente todos os parâmetros de um sistema nebuloso típico, entretanto possui o inconveniente de depender da intervenção do usuário para a definição dos parâmetros evolutivos de quatro populações diferentes. Considerando $\delta^i$ o conjunto de parâmetros evolutivos na geração $i$ e $\tau(\delta^i)$ o total de combinações possíveis para esse conjunto, o número de combinações possíveis de parâmetros para um sistema baseado na abordagem co-evolutiva para uma única geração é: $\left[\tau(\delta^i)\right]^4$ (isto desprezando-se os parâmetros associados à co-evolução do sistema, como será visto posteriormente). Existem ainda algumas evidências experimentais de que os valores de parâmetros de populações diferentes não estão correlacionados (MARUO; DELGADO, 2006). Logo, pode-se considerar que esse sistema, do ponto de vista de autonomia, apresenta algumas limitações e que a busca pelos parâmetros evolutivos desse tipo de sistema genético-nebuloso é realizada em um espaço de busca grande.

Durante o projeto de um GFS, dois fatores devem ser observados quando uma nova abordagem é proposta: a quantidade demandada de recursos computacionais e a necessidade de co-

nhecimento de um especialista para ajustar os valores dos parâmetros (MARUO; DELGADO, 2006). A figura 4.1 ilustra um GFS tradicional onde todos os parâmetros são codificados em um único indivíduo e é necessário o conhecimento de um especialista para especificar os parâmetros evolutivos.

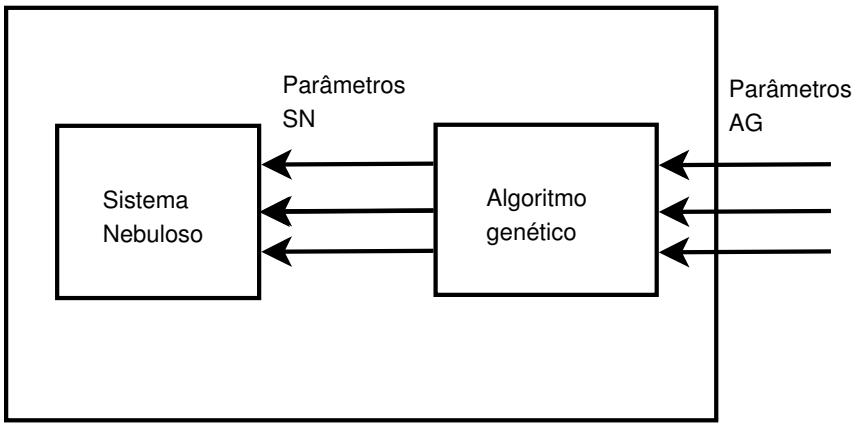



Figura 4.1: Abordagem tradicional de codificação de um GFS

Em geral existe um compromisso entre a flexibilidade do sistema e a eficiência do processo evolutivo. Apesar de apresentarem flexibilidade e alto-desempenho, muitos GFS apresentados na literatura necessitam do conhecimento de um especialista para definir um conjunto adequado de parâmetros evolutivos. (CORDÓN *et al.*, 2001a, 2004; KLIR; YUAN, 1995; PEDRYCZ; GOMIDE, 1998; YUAN; ZHUANG, 1996; CASTRO; CAMARGO, 2004; BONISSONE; KHEDKAR; CHEN, 1996; CARSE; FOGARTY; MUNRO, 1996; CHEONG; LAI, 2000; DE SOUSA; MADRID, 2000; GONZALEZ; HERRERA, 1997; GONZALEZ; PEREZ, 1996; GUROCAK, 1999; HERRERA; LOZANO; VERDEGAY, 1995). Isto diminui o grau de autonomia do sistema, pois a escolha de parâmetros adequados depende do problema e requer experiência prévia do projetista.

## 4.2 Sistema nebuloso co-evolutivo

O sistema proposto neste trabalho é baseado no coevol-GFS proposto por Delgado (2002) que apresenta as seguintes características:

- O sistema coevol-GFS é projetado automaticamente através de um algoritmo genético que é capaz de obter:

  **funções de pertinência** tipo, formato, localização e grau de sobreposição entre conjuntos nebulosos;

**regras nebulosas** número de regras e proposições nebulosas;

**operadores do mecanismo de inferência** Para modelos nebulosos do tipo TSK, os operadores de agregação nebulosa.

- O processo evolutivo se desenvolve em uma estrutura de co-evolução com quatro níveis (módulos): funções de pertinência ou conjunto de partições nebulosas (nível I), regras individuais (nível II), bases de regras (nível III) e modelos de inferência nebulosos (nível IV);

- Os módulos interagem durante o processo evolutivo trocando informações sobre a qualidade das soluções, permitindo uma busca mais eficaz;

- Os parâmetros dos conseqüentes são determinados utilizando o método dos mínimos quadrados.

Esta abordagem co-evolutiva foi proposta inicialmente por Delgado, von Zuben e Gomide (2001) como uma extensão da evolução hierárquica (DELGADO; VON ZUBEN; GOMIDE, 2001). Este processo foi inspirado pelo paradigma baseado em hierarquia de Moriarty e Miikkulainen (1998) e o modelo de co-evolução cooperativa de Potter e de Jong (2000). O sistema passa a ser representado por indivíduos de várias populações indicando níveis hierárquicos diferentes onde cada população está associada a um conjunto de soluções parciais para o problema. O particionamento da solução através de múltiplas populações é interessante por permitir a representação de entidades completamente diferentes em populações distintas, que podem utilizar uma codificação específica para cada nível hierárquico. Entretanto, essa abordagem exige a definição da função de aptidão das soluções parciais em níveis hierárquicos inferiores. Para populações muito grandes em níveis hierárquicos mais baixos, esse número adicional de avaliações pode ser inconveniente. No entanto, a abordagem proposta por Delgado, von Zuben e Gomide (2001) pode se beneficiar do uso de paralelismo. O cálculo da aptidão no nível IV, o processo computacional mais oneroso, independe do estado das outras soluções e pode ser calculado em múltiplos processadores simultaneamente. Nessa abordagem, as populações não precisam estar armazenadas no mesmo computador. Desta forma, é possível utilizar um esquema de memória distribuída (TANENBAUM; VAN RENESSE, 1985) diminuindo significativamente os requisitos de memória de um nodo do sistema. A co-evolução surge da interação entre as diferentes populações trocando informações sobre a qualidade dos indivíduos de outras populações para direcionar a busca para regiões mais promissoras.

### 4.2.1 Abordagem co-evolutiva

Conforme descrito por Shi, Eberhart e Chen (1999), a definição dos parâmetros de um sistema nebuloso é um problema equivalente a encontrar o mínimo de uma hiper-superfície associada com uma função objetivo. Esta hiper-superfície possui as seguintes características:

- é infinitamente grande, complexa e ruidosa;
- é não-diferenciável pois mudanças no número de regras nebulosas ocorrem de forma discreta e têm efeito descontínuo no desempenho do sistema nebuloso;
- é multimodal, ou seja, conjuntos nebulosos diferentes podem apresentar desempenho similar;
- é um problema mal-condicionado, já que pequenas mudanças podem causar grandes efeitos no desempenho de cada sistema.

Em problemas desse tipo, abordagens tradicionais, nas quais a solução completa para o problema é codificada em um único indivíduo (abordagem *Pittsburgh* ilustrada na figura 4.1), podem ser inadequadas (CORDÓN *et al.*, 2001a). O desempenho efetivo de um sistema nebuloso pode não ser corretamente medido sob a presença de um elemento destrutivo (por exemplo, a sobrevivência de uma boa partição nebulosa pode ser comprometida por sua associação a uma base de regras inadequada). Este fenômeno é razoavelmente comum nos estágios iniciais da evolução e pode evitar que regiões promissoras sejam exploradas pelo algoritmo. Além disso, os operadores genéticos tratam da mesma maneira informações de natureza completamente diferente codificadas no mesmo cromossomo.

### 4.2.2 Estrutura hierárquica

A estrutura hierárquica proposta por Delgado (2002) estabelece a divisão da solução em populações distintas, organizadas de forma hierárquica em que quatro populações codificam os diferentes parâmetros de sistemas nebulosos.

A divisão das populações é realizada da seguinte forma:

**nível I** população de partições nebulosas;

**nível II** população de regras individuais;

**nível III** população de bases de regras;

**nível IV** população de sistemas nebulosos.

O esquema hierárquico demanda um algoritmo genético específico para cada nível conforme ilustrado na figura 4.2

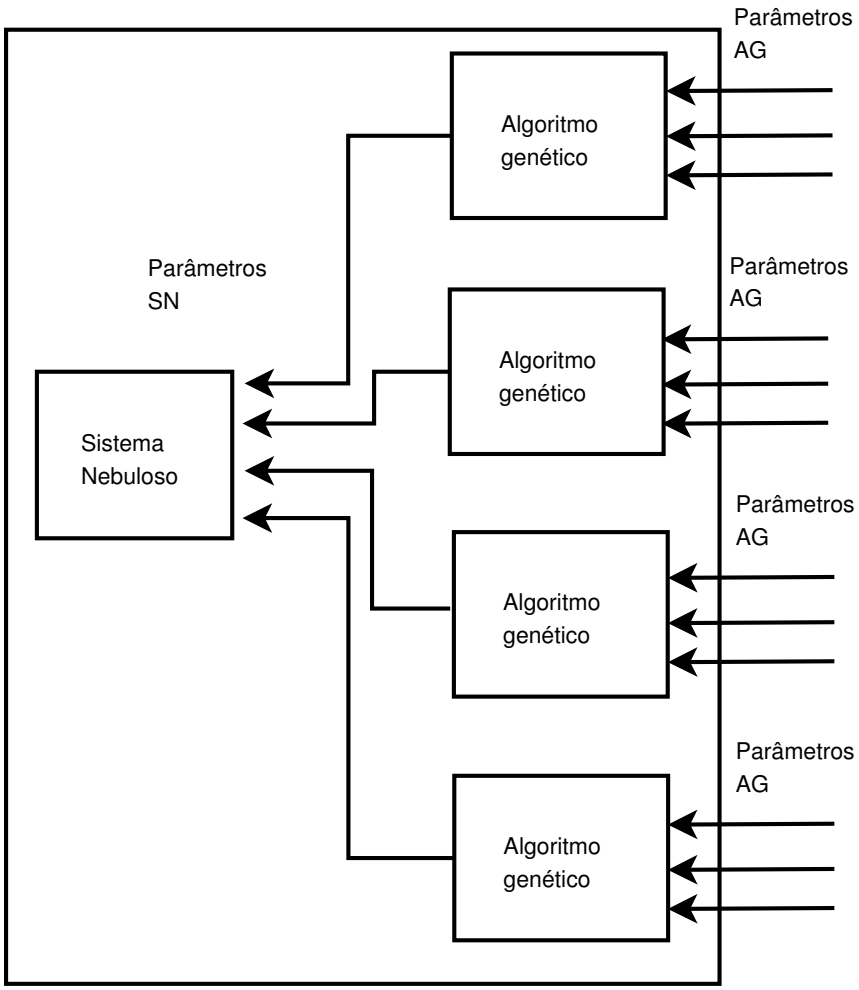



Figura 4.2: Abordagem hierárquica de codificação de um GFS

Delgado (2002) comparou as duas abordagens (ilustradas nas figuras 4.1 e 4.2) e, para todos os testes realizados, a abordagem co-evolutiva obteve resultados melhores que a abordagem tradicional de codificar a solução completa em um único indivíduo codificado em uma única população. Conforme será discutido na seção 4.3.1, essa abordagem hierárquica evita os problemas da codificação em um único cromossomo mas, em contrapartida, adiciona complexidade ao projeto dos parâmetros evolutivos.

Os indivíduos de níveis superiores são constituídos a partir de indivíduos de níveis inferiores. Dessa forma o desempenho de indivíduos de níveis superiores depende diretamente do

desempenho de indivíduos localizados em níveis inferiores. Um diagrama de classes simplificado do sistema nebuloso está na figura 4.3.

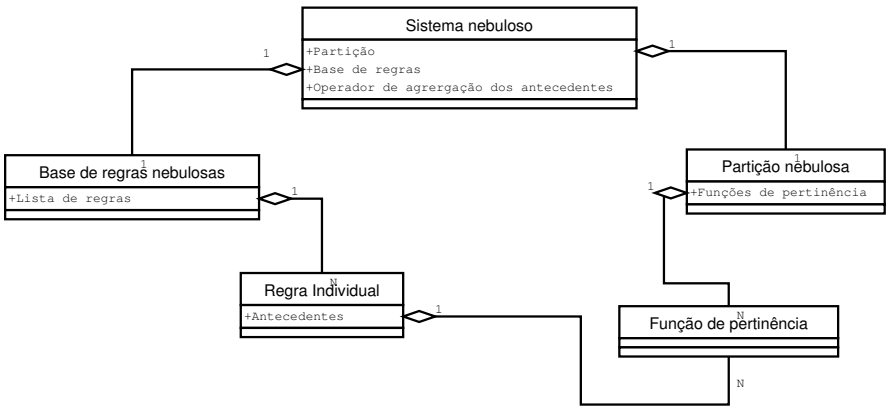



Figura 4.3: Hierarquia simplificada do sistema nebuloso proposto

Existe uma correspondência entre as classes desse diagrama e os níveis hierárquicos exceto o nível I em que estão codificados simultaneamente as funções de pertinência e as partições nebulosas. Devido aos requisitos de interpretabilidade do sistema nebuloso, a criação de uma população de funções de pertinência não é interessante (se uma função de pertinência pertence a mais de uma partição nebulosa, o processo de correção das partições sob o critério de interpretabilidade pode alterar uma função de pertinência tornando-a interpretável para uma partição e não-interpretável para outra).

### 4.2.3 Codificação

Um indivíduo da população de partições nebulosas (nível I) contém as funções de pertinência que discretizam o universo de discurso das variáveis de entrada. Essa população utiliza indivíduos com codificação real, a qual que será detalhada neste capítulo. As regras individuais do segundo nível hierárquico representam proposições nebulosas (MARUO; DELGADO, 2006). Esta população aceita combinações das funções de pertinência contidas no nível I indicadas por índices, ou seja, sua posição na partição nebulosa. A figura 4.4 ilustra o esquema geral de codificação adotado.

No nível II são codificadas as regras individuais, ou seja, diferentes combinações de termos lingüísticos no antecedente. A codificação é inteira e representa um apontador para o nível I onde estão codificadas as respectivas funções de pertinência.

Neste trabalho, foi feita uma modificação na codificação proposta por Delgado (2002) para a população de bases de regras (nível III). Ao contrário da proposta original que utiliza a co-

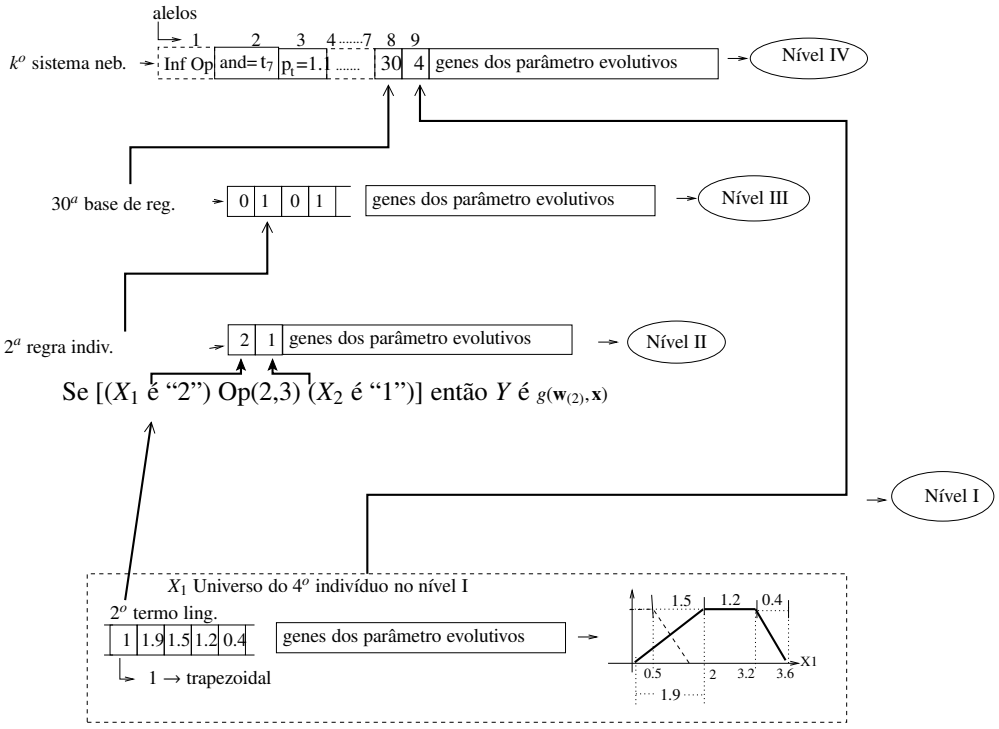



Figura 4.4: Colaboração hierárquica entre espécies

dificação inteira, aqui adota-se a codificação binária em que um valor 1 ou 0 no bit $i$ indica a presença ou ausência, respectivamente, da $i$-ésima regra na população de regras individuais. A codificação de Delgado utiliza um vetor de inteiros em que o valor de cada inteiro representa uma proposição (regra) da base de regras. Esta codificação inteira permite que a mesma base de regras contenha múltiplas cópias da mesma regra codificadas no mesmo cromossomo. Logo, indivíduos do nível II apresentam uma tendência a apresentar múltiplas cópias em regras com alta aptidão levando a um desempenho sub-ótimo do sistema. A codificação binária diminui significativamente esse efeito. Se houver garantia de que $\mathbf{h}^t_{i,II} \neq \mathbf{h}^t_{j,II}$, $\forall i, j \in \{1, \ldots, \phi_{II}\} \mid i \neq j$, onde $\mathbf{h}^t_{k,II}$ indica o indivíduo na posição $k$ do nível hierárquico II na geração $t$, pode-se afirmar que todas as regras de qualquer base de regras codificada com a codificação proposta são diferentes.

Já no nível IV, cada indivíduo representa um sistema de inferência nebuloso. Cada indivíduo possui, diferente da proposta de Delgado (2002), codificação inteira e seus genes, além de conter informações sobre o operador de agregação dos antecedentes das regras (genes 2 e 3), associam uma base de regras específica (gene 8) a uma única partição nebulosa (gene 9). Os genes 4 a 7 são específicos do modelo de Mamdani e não serão considerados neste trabalho. De forma semelhante ao nível III, o oitavo gene contém um apontador para a posição da base de regras (nível III) associada ao sistema de inferência nebuloso e o nono gene contém o índice

relativo ao nível I da partição nebulosa deste sistema.

No exemplo da figura 4.4 o sistema de inferência nebuloso $k$ utiliza a trigésima base de regras no nível III e a quarta partição nebulosa do nível I, codificadas nos alelos 8 e 9 respectivamente. Para cada indivíduo do nível IV, os antecedentes da base de regras associada são agregados utilizando a proposição codificada no nível IV (alelo 2). Neste caso a agregação do antecedente é dada pela norma:

$$a\,\mathbf{t}_7\,b = \frac{ab}{p_t + (1-p_t)(a+b-ab)}$$

onde o alelo 2 indica a norma $\mathbf{t}_7$ e $p = 1.1$ conforme a informação codificada no nível IV (alelo 3). Na trigésima base de regras observa-se que o segundo bit possui um valor "verdadeiro", logo a segunda regra individual faz parte da base de regras do sistema $k$.

No nível II observa-se que a segunda regra individual possui a forma:

$$Se\ [X_1\ é\ \text{“2“}\ \mathbf{t}_5\ X_2\ é\ \text{“1“}\ então\ Y\ é\ g(\mathbf{w_2}, \mathbf{x})]$$

onde $X_1$ e $X_2$ estão codificados na quarta partição nebulosa (no nível I) e $g(\mathbf{w_2}, \mathbf{x})$ é o conseqüente TSK da segunda regra nebulosa.

A interação entre os níveis hierárquicos também envolve os operadores co-evolutivos, que serão abordados neste capítulo, que alteram as probabilidades de mutação baseados na aptidão de indivíduos de níveis inferiores. Neste caso, genes indicando indivíduos com menor aptidão têm uma maior probabilidade de sofrer mutação.

### 4.2.4 Avaliação da aptidão

Na abordagem proposta, a avaliação da aptidão de um indivíduo depende diretamente de indivíduos de níveis hierárquicos diferentes. Por exemplo, é difícil avaliar o desempenho de uma base de regras nebulosas sem que ela esteja vinculada a um sistema de inferência nebuloso. Da mesma forma o desempenho de um sistema de inferência nebuloso depende diretamente da partição nebulosa que ele utiliza. Assim é necessário um esquema de cálculo de aptidão que compartilhe informações da aptidão dos indivíduos entre as diferentes populações. Utilizando a avaliação proposta por Delgado, von Zuben e Gomide (2001) tem-se:

**Sistema nebuloso** Para o nível IV, o desempenho do sistema nebuloso será dado por

$$F_{IV}(SN_i^t) = \frac{1}{\sqrt{EQM(SN_i^t)}} \tag{4.4}$$

**Base de regras** Para o nível III, a aptidão é dada pelo sistema nebuloso mais apto que utilize a base de regras $k$:

$$F_{III}(BR_k^t) = \max(F(SN_b^t) \cdots F(SN_d^t)) \tag{4.5}$$

onde $b, \ldots, d$ são os sistemas nebulosos que utilizam a base de regras $k$.

**Regra individual** Para o nível II, a aptidão do sistema é dada pela média das aptidões dos sistemas que utilizem a regra $j$. Como o sistema pode apresentar um bom desempenho da base de regras mas não necessariamente devido à presença específica da regra $j$ essa forma de avaliação parece mais apropriada

$$F_{II}(RI_j^t) = média(F(SN_m^t) \cdots F(SN_n^t)) \tag{4.6}$$

onde $m, \ldots, n$ são os sistemas nebulosos que utilizam a regras $j$.

**Partições nebulosas** Para o nível I a aptidão do indivíduo é novamente mensurada a partir do melhor sistema que contenha a partição $q$:

$$F_I(PN_q^t) = \max(F(SN_x^t) \cdots F(SN_z^t)) \tag{4.7}$$

onde $x, \ldots, z$ são os sistemas nebulosos que utilizam a partição nebulosa $q$.

### 4.2.5 Codificação das partições nebulosas

A codificação das partições nebulosas utiliza o esquema proposto por Delgado, von Zuben e Gomide (2001) que por sua vez é baseado no esquema de codificação de funções de pertinência triangulares utilizado por Lee e Takagi (1993). O processo de decodificação do cromossomo exige um passo adicional de conversão para gerar o fenótipo da função de pertinência pois existe a possibilidade da presença de múltiplos tipos de função.

Este tipo de representação apresenta as seguintes características:

**uniformidade** nessa implementação o tipo de funções de pertinência foi restrito a três: (trapezoidal, triangular e Gaussiano). Estes tipos de função possuem parâmetros diferentes (JANG, 1993) e é desejável que uma mutação em um gene específico acarrete o menor impacto possível em outros genes. Para isso foi utilizado um esquema de codificação com 5 genes por função de pertinência.

**relatividade** o uso indiscriminado de operadores evolutivos em partições codificadas com posições absolutas dos parâmetros das funções de pertinência possui uma alta probabilidade

de gerar indivíduos inválidos segundo o critério de interpretabilidade do sistema. Para evitar esse problema foi utilizado um esquema de codificação baseado em posições relativas, descrito a seguir, aliado a um processo de reparação de partições não-interpretáveis.

No esquema utilizado, uma função de pertinência é representada por uma tupla:

$$\mu_k =< S_k, L_k, C_{1k}, C_{2k}, R_k > \tag{4.8}$$

onde $k$ indica a posição da função de pertinência no universo de discurso. O parâmetro $S_k$ possui valores no conjunto $\{0,1,2,3\}$ onde o valor 0 indica uma função de irrelevância, ou seja, um termo lingüístico do tipo *don't care*, o valor 1 indica uma função de pertinência do tipo trapezoidal, o valor 2 indica uma função do tipo triangular e o valor 3 representa uma função do tipo Gaussiana. O parâmetro $C_{1k}$ indica a distância entre a função de pertinência $\mu_{k-1}$ e $\mu_k$ (se $k = 1$, $C_{1k}$ deve representar a distância entre o limite inferior do universo de discurso e $\mu_1$). Dessa forma pode-se concluir que a posição absoluta de um termo lingüístico depende dos termos lingüísticos anteriores. Como $C_{1k} \geq 0$, pode-se afirmar que $\sup_{x \in \mathbf{X}} núcleo(\mu_{k-1}) < \inf_{x \in \mathbf{X}} núcleo(\mu_k)$, ou seja, o limite superior do núcleo da função associada a um termo lingüístico "à esquerda" deve estar à esquerda do limite inferior do núcleo de uma função de pertinência associada a um termo lingüístico "à direita". O parâmetro $C_{2k}$ está diretamente associado à generalidade de $\mu_k$. Se o tipo da função for trapezoidal esse parâmetro codifica diretamente o tamanho do núcleo de $\mu_k$, do contrário $C_{2k}$ irá indicar indiretamente a posição do vértice central da função triangular ou a média da função Gaussiana. Assim como $C_{1k}$, seus valores devem ser restritos a valores positivos. Os parâmetros $L_k$ e $R_k$ estão associados ao suporte de $\mu_k$ se a função for dos tipos trapezoidal ou triangular. Se a função for da forma Gaussiana, este parâmetro, em conjunto com $C_{2k}$, indica indiretamente a dispersão da função. $L_k$ e $R_k$ devem ser restritos a valores positivos. A posição absoluta desses parâmetros é dada por $(c_{1k}, c_{2k}, c_{3k}, c_{4k})$:

- $c_{1k} = c_{2k-1} + C_{1k}$
- $c_{2k} = c_{1k} + C_{2k}$
- $l_k = c_{1k} - L_k$
- $r_k = c_{2k} + R_k$

O processo de decodificação de $\mu_k$ para funções trapezoidais é relativamente simples, os valores de $\{l_k, c_{1k}, c_{2k}, r_k\}$ equivalem aos parâmetros $\{a, m, n, b\}$. Para funções triangulares os parâmetros $a, m, b$ são equivalentes a $\{l_k, \frac{c_{1k}+c_{2k}}{2}, r_k\}$. Finalmente, para funções Gaussianas $\{\vartheta_k, \sigma_k\}$

equivalem a $\{\frac{c_{1k}+c_{2k}}{2}, \frac{L_k+C_{2k}+R_k}{6}\}$. Desta forma os limites práticos da função de pertinência Gaussiana não excederão $3\sigma$ (DELGADO, 2002). Os três tipos de funções de pertinência para $< l_k, c_{1k}, c_{2k}, r_k >=< -2, -1, 1, 0 >$ podem ser observados na figura 4.5:

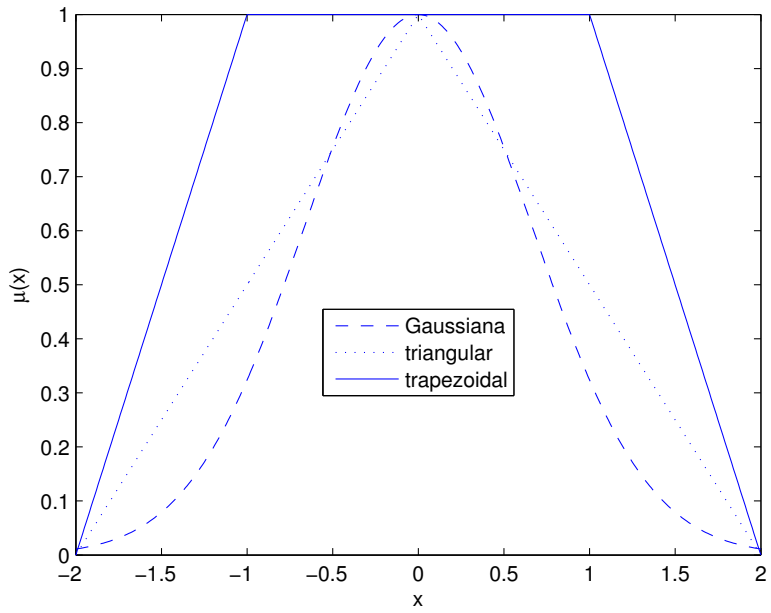



Figura 4.5: Gráfico dos tipos de funções de pertinência utilizados

Durante o processo de decodificação o algoritmo realiza a verificação da interpretabilidade da partição nebulosa. Considerando que o sistema utiliza somente funções convexas, ou seja, dados dois pontos $x_1$ e $x_2$ tal que $x_1 < x < x_2$, $\mu(x) \geq min\{\mu(x_1), \mu(x_2)\}$, pode-se garantir a interpretabilidade da partição nebulosa verificando somente as interseções entre funções de pertinência consecutivas e as interseções de funções de pertinência extremas e os limites superior e inferior do universo de discurso.

Segundo o critério adotado, a partição do universo de uma variável $x_k$ será considerada interpretável se:

$$\gamma \leq Poss(i, j) \leq \kappa, \quad \forall i, j \in \mathcal{T}(\S_{\|}) \tag{4.9}$$

Caso a interseção entre funções de pertinência não respeite a equação 4.9, utiliza-se um processo de correção das funções de pertinência (exceto as funções do tipo Gaussiana):

1. Inicialmente encontra-se o valor de pertinência da nova interseção:

$$\mu'_{j,k} = \max\left\{\min\left\{\mu_{j,k}(x_{j,k}), \kappa\right\}, \gamma\right\} \tag{4.10}$$

2. Encontra-se o valor da nova interseção $x_i$ tal que $\mu_j(x_i) = \mu'_{j,k}$

3. Corrige-se $\mu_j(x)$ tal que $\mu_j(x_i) = \mu'_{j,k}$

O processo de correção para as funções trapezoidal e triangular é trivial. A correção da função trapezoidal consiste em alterar $L_k$ para a função trapezoidal e $L_k$ e $C_{1k}$ para a função triangular. A função Gaussiana é simétrica em relação a sua média $\sigma$. Portanto não é possível realizar a correção de uma transição sem afetar a outra. Para este tipo de função foi utilizada uma correção específica: sejam $y_1$ e $y_2$ os valores de pertinência nos pontos de interseção da função Gaussiana com as funções imediatamente anterior e posterior respectivamente. Os valores para as novas interseções são calculados através de $y' = \max\{\min\{y,\kappa\},\gamma\}$. O valor em que a função anterior intercepta $y'_1$ e a função posterior intercepta $y'_2$ são armazenados em $x_1$ e $x_2$ respectivamente. Os parâmetros da função Gaussiana corrigida $(\vartheta,\sigma)$ serão dados pela solução do sistema de equações:

$$\begin{cases} x_1 = \vartheta - \sigma\sqrt{-2\ln y_1} \\ x_2 = \vartheta + \sigma\sqrt{-2\ln y_2} \end{cases} \tag{4.11}$$

### 4.2.6 Modelo de sistema nebuloso com conseqüentes Takagi-Sugeno não lineares

Conforme mencionado no capítulo 2, o uso de regras do tipo Takagi-Sugeno, apesar da menor interpretabilidade lingüística, é uma alternativa interessante no projeto automático de sistemas nebulosos. Isso ocorre porque o conseqüente da regra, ao contrário do Modelo Mamdani, é uma função paramétrica dos dados de entrada como na equação 4.12. Portanto, é possível a representação de sistemas com um número reduzido de regras diminuindo a complexidade do modelo. Dados os parâmetros associados aos antecedentes obtidos pelo processo evolutivo, o objetivo é encontrar os parâmetros associados aos conseqüentes das regras de forma a minimizar o erro.

$$R_j: \text{ Se } X_1 \text{ é } A_1^j \cdots \text{ e } X_n \text{ é } A_n^j \text{ então } Y \text{ é } g_j\left(\mathbf{a}_j, \mathbf{x}\right) \tag{4.12}$$

onde $g_j\left(\mathbf{a}_j, \mathbf{x}\right)$ é o conseqüente de Takagi Sugeno da regra $j$. Os tipos mais comuns de conseqüentes de Takagi-Sugeno são o constante ($Q = 0$) ou linear ($Q = 1$) (DELGADO; VON ZUBEN; GOMIDE, 2001; JANG; SUN; MIZUTANI, 1997). Neste trabalho, assim como em Delgado (2002), foram utilizados conseqüentes de ordem 2 com componentes não-lineares.

Dado um um vetor de entrada $\mathbf{x} = [x_1 \ldots x_n]^T$ e uma base de regras nebulosas obtidas da

população de nível III, o conseqüente $g_j$ da regra $j$ pode ser expresso como:

$$g_j\left(\mathbf{a}_j, \mathbf{x}\right) = a_{j0} + a_{j1}x_1 + \cdots + a_{jn}x_n + a_{j(n+1)}x_1x_1 + \cdots + a_{j(2n)}x_1x_n + \\ + a_{j(2n+1)}x_2x_2 + \cdots + a_{j\left(\frac{n(n-1)}{2}+2n\right)}x_nx_n \tag{4.13}$$

onde $\mathbf{a}_j = \left[a_{j0}\cdots a_{j\left(\frac{n(n-1)}{2}+2n\right)}\right]^T$, $j = 1,\ldots,m$ são os vetores de parâmetros associados às $m$ regras nebulosas. Assim, o problema passa a ser a obtenção dos vetores $\hat{\mathbf{a}}_\mathbf{j} = \left[\hat{a}_{j0},\ldots,\hat{a}_{j\left(\frac{n(n-1)}{2}+2n\right)}\right]$, $j \in \{1,\ldots,m\}$ que minimizem o erro do sistema. Uma forma de obter esses parâmetros seria codificá-los diretamente no indivíduo e deixar que o próprio processo evolutivo identifique o conjunto ótimo. Entretanto, uma abordagem alternativa que leva a resultados mais precisos é utilizar o método dos mínimos quadrados para encontrar $\hat{\mathbf{a}}_\mathbf{j}$. Contudo, conforme será discutido na seqüência, algumas condições devem ser satisfeitas para determinar $\hat{\mathbf{a}}_\mathbf{j}$ de forma determinística e fechada pelo método dos mínimos quadrados. Assim, o processo evolutivo se encarregará de obter somente os parâmetros do antecedente das regras. Após obter a informação sobre os antecedentes das $m$ regras, os conseqüentes podem ser calculados da seguinte forma:

- Seja $\eta_i(p)$ a agregação dos antecedentes da regra $i$ para o vetor de entrada $x_p = \left[x_{p1}\cdots x_{pn}\right]^T$. Então, a saída do sistema será dada por:

$$\hat{y}(x_p) = \frac{\sum_{i=1}^{m}\eta_i(p)g_i(\mathbf{a}_i, \mathbf{x}_p)}{\sum_{i=1}^{m}\eta_i(p)} \tag{4.14}$$

- Seja $\beta_i^p$ a ativação normalizada da regra $i$ para a entrada $x_p$ definida pela equação 4.15:

$$\beta_i^p = \frac{\eta_i(p)}{\sum_{i=1}^{m}\eta_i(p)} \tag{4.15}$$

  pode-se reescrever a equação 4.14 como:

$$y(p) = \mathbf{m}(p)\mathbf{a} \tag{4.16}$$

  onde $\mathbf{a} = \left[\mathbf{a}_1^T\cdots\mathbf{a}_m^T\right]^T$ é o vetor de parâmetros de todas as regras e:

$$\mathbf{m}(p) = \left[\beta_1^p\beta_1^p x_1 \cdots \beta_1^p x_n x_n \beta_2^p\beta_2^p x_1 \cdots \beta_m^p x_n x_n\right]^T \tag{4.17}$$

  ou seja, um sistema de equações lineares.

- Considerando $N$ o número de pares de treinamento, as saídas desejadas podem ser agrupadas em um vetor $\mathbf{y}$ da forma:

$$\mathbf{y} = \left[y_d(1)\cdots y_d(N)\right]^T \tag{4.18}$$

e definindo:

$$\Lambda = [\mathbf{m}(1) \cdots \mathbf{m}(N)]^T \tag{4.19}$$

pode-se encontrar **a** a partir do método dos mínimos quadrados (JANG; SUN; MIZUTANI, 1997):

$$\begin{aligned} \mathbf{y} &= \Lambda \mathbf{a} \\ \Lambda^T \mathbf{y} &= \Lambda^T \Lambda \mathbf{a} \\ \mathbf{a} &= \left(\Lambda^T \Lambda\right)^{-1} \Lambda^T \mathbf{y} \end{aligned} \tag{4.20}$$

Para calcular **a** através desse método deve-se tomar os seguintes cuidados:

- $\Lambda$ deve ter posto cheio. Essa condição pode sempre ser satisfeita se existir um processo de poda de colunas linearmente dependentes até que $\Lambda$ se torne uma matriz de posto cheio. Entretanto não há deterioração da qualidade da solução em comparação com a situação em que $\Lambda$ não possui posto cheio. Nestes casos a flexibilidade do modelo é excessiva quando comparada com a quantidade de informação disponível para o treinamento e deve ser reduzida de forma sistemática (DELGADO; VON ZUBEN; GOMIDE, 2000);
- $N \geq \frac{n(n-1)}{2} + 2n + 1$, ou seja o número de pares de treinamento deve ser maior ou igual ao número de parâmetros indeterminados. Essa condição normalmente é satisfeita, resultando em um sistema sobredeterminado;
- a dimensão da matriz $\Lambda^T \Lambda$ é limitada por $\frac{n(n-1)}{2} + 2n + 1$, ou seja a dimensão de $\Lambda$ cresce de forma quadrática com o número de variáveis de entrada. Além disso a complexidade temporal da fatoração LU de uma matriz pode ser aproximada por $O(k^3)$. Em sistemas com muitas variáveis de entrada pode ser necessário limitar a dimensão de $\Lambda^T \Lambda$ devido à limitação dos recursos computacionais disponíveis.

Uma alternativa ao uso do método descrito é o uso de um método dos mínimos quadrados recursivo (JANG; SUN; MIZUTANI, 1997). Entretanto, ambos os métodos apresentam baixo desempenho em situações em que a matriz de co-variância é mal condicionada (GOODWIN; NORTON; VISWANATHAN, 1985). Para a versão tradicional, um processo de eliminação de regras com antecedentes linearmente dependentes pode incrementar o desempenho do método.

### 4.2.7 Algoritmo de poda

Nesta seção será apresentado o algoritmo de eliminação de colunas linearmente dependentes (LD) da matriz $\Lambda$ sempre que ela for mal-condicionada. Este algoritmo se baseia na estimativa da condição da matriz pseudo-inversa de $\Lambda$ dada por $\Lambda^T\Lambda$. A condição de uma matriz é dada por:

$$cond(\mathbf{A}) = \| \mathbf{A} \| . \| \mathbf{A}^{-1} \| \tag{4.21}$$

onde:

- $rcond(\mathbf{A}) = cond^{-1}(\mathbf{A})$;
- A condição descreve a robustez da solução obtida pelo método dos mínimos quadrados a pequenas alterações na medição da saída dos dados de treinamento.
- se $rcond(\mathbf{A}) \rightarrow 1$ o sistema é bem-condicionado, ou seja, a solução é pouco suscetível a eventuais erros de medição ou ruídos.
- se $rcond(\mathbf{A}) \rightarrow 0$ o sistema é mal-condicionado, ou seja, pequenas mudanças nos dados de treinamento podem acarretar em grandes diferenças na resposta do sistema.

O cálculo da condição utilizando a equação 4.21 implica na inversão da matriz $\mathbf{A}$. A inversão da matriz é computacionalmente onerosa e pode ser evitada utilizando um estimador (MEYER, 2000). Nesse trabalho foi utilizado o estimador de Cheng e Higham (2001) que apresenta desempenho similar ao LAPACK (ANDERSON *et al.*, 1992) para a maioria dos casos, mas com menor complexidade de implementação.

O processo de poda aumenta o número de condição recíproca da matriz $\Lambda^T\Lambda$ eliminando de $\Lambda$ as colunas que apresentam maior contribuição para a diminuição de $rcond(\Lambda^T\Lambda)$.

O fluxograma do processo de poda pode ser visto na figura 4.6.

O algoritmo inicia com todas as colunas de $\Lambda$ presentes. Se o limiar de condição não for satisfeito, o sistema deve normalizar as colunas de $\Lambda$ fazendo $\sup_j \Lambda_{i,j} = 1$ e ordenar as colunas por norma euclidiana. O sistema elimina as 5 colunas com menor norma formando a matriz $\Lambda'$ e verifica se a condição de $\Lambda'^T\Lambda'$ satisfaz o limiar. Se não satisfaz, eliminam-se mais 5 colunas. Caso o limiar seja satisfeito, é realizado um processo de adição, uma a uma, de cada uma das 5 colunas eliminadas formando $\Lambda$". Caso $rcond(\Lambda"^T\Lambda")$ satisfaça o limiar, tenta-se adicionar outras colunas eliminadas. Ao final do processo, o método dos mínimos

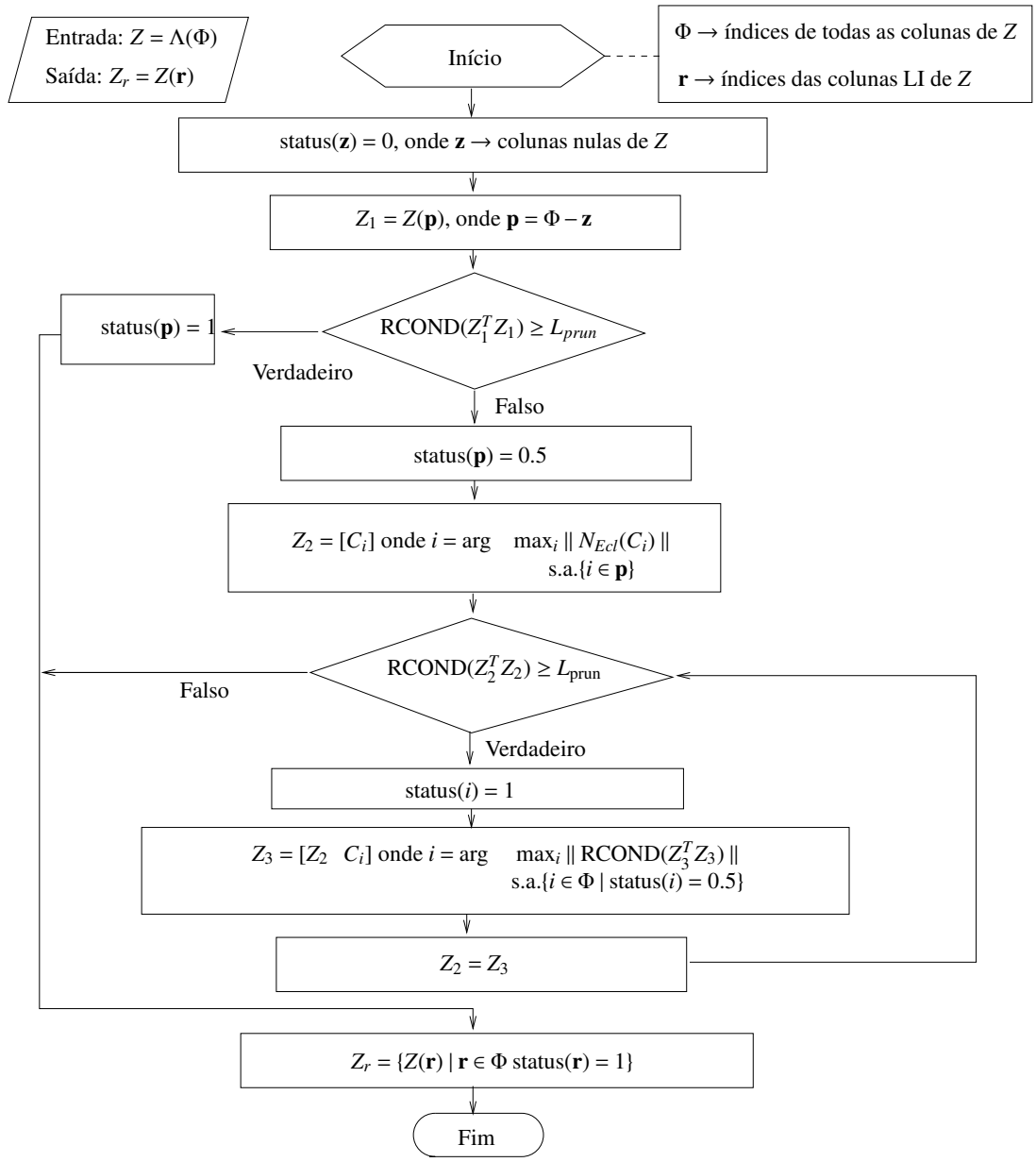



Figura 4.6: Algoritmo para o processo de poda.

quadrados deve trabalhar com Λ". Para uma descrição mais detalhada do algoritmo de poda, ver (DELGADO, 2002).

## 4.3 Sistema genético-nebuloso com auto-adaptação de parâmetros

Este trabalho propõe uma modificação no algoritmo co-evolutivo de Delgado (2002) o qual, embora seja capaz de realizar o projeto automático completo de um sistema nebuloso, apresenta uma complexidade adicional na determinação dos parâmetros evolutivos devido ao uso de múltiplas populações (vide figura 4.2).

Esta modificação, que envolve a auto-adaptação dos parâmetros evolutivos, representa a principal contribuição deste trabalho. Conforme visto no início do capítulo, a abordagem co-evolutiva possui bom desempenho nos critérios de acuidade e interpretabilidade com alguns problemas no critério de automação. Entretanto, foi demonstrado que uma abordagem por ajuste de parâmetros não é apropriada para a definição do conjunto de parâmetros evolutivos. Este problema se torna mais pronunciado quando emprega-se uma abordagem com múltiplas populações, como a abordagem co-evolutiva utilizada neste trabalho. Nesse contexto o uso de auto-adaptação de parâmetros parece uma alternativa interessante.

Para avaliar a qualidade da abordagem auto-adaptativa, um protótipo inicial de algoritmo genético foi proposto (AG-autoadap) e testado em problemas de otimização contínua e otimização combinatória (MARUO; LOPES; DELGADO, 2004). Em seguida esse algoritmo foi adaptado para a aplicação na identificação dos parâmetros de sistemas nebulosos utilizando a abordagem co-evolutiva (MARUO; DELGADO, 2006).

A figura 4.7 mostra o sistema proposto com um bom grau de autonomia sendo capaz tanto de encontrar a solução do problema quanto praticamente todos parâmetros do sistema.

### 4.3.1 Parâmetros auto-adaptativos do algoritmo genético

O sGA (Standard Genetic Algorithm) proposto por Holland apresenta dois operadores de variação básicos: a recombinação e a mutação. A recombinação é capaz de gerar descendentes que são semelhantes a seus pais. Isto acontece em dois casos:

**(i)** Quando um dos descendentes herda a maior parte de seu código genético de um único indivíduo;

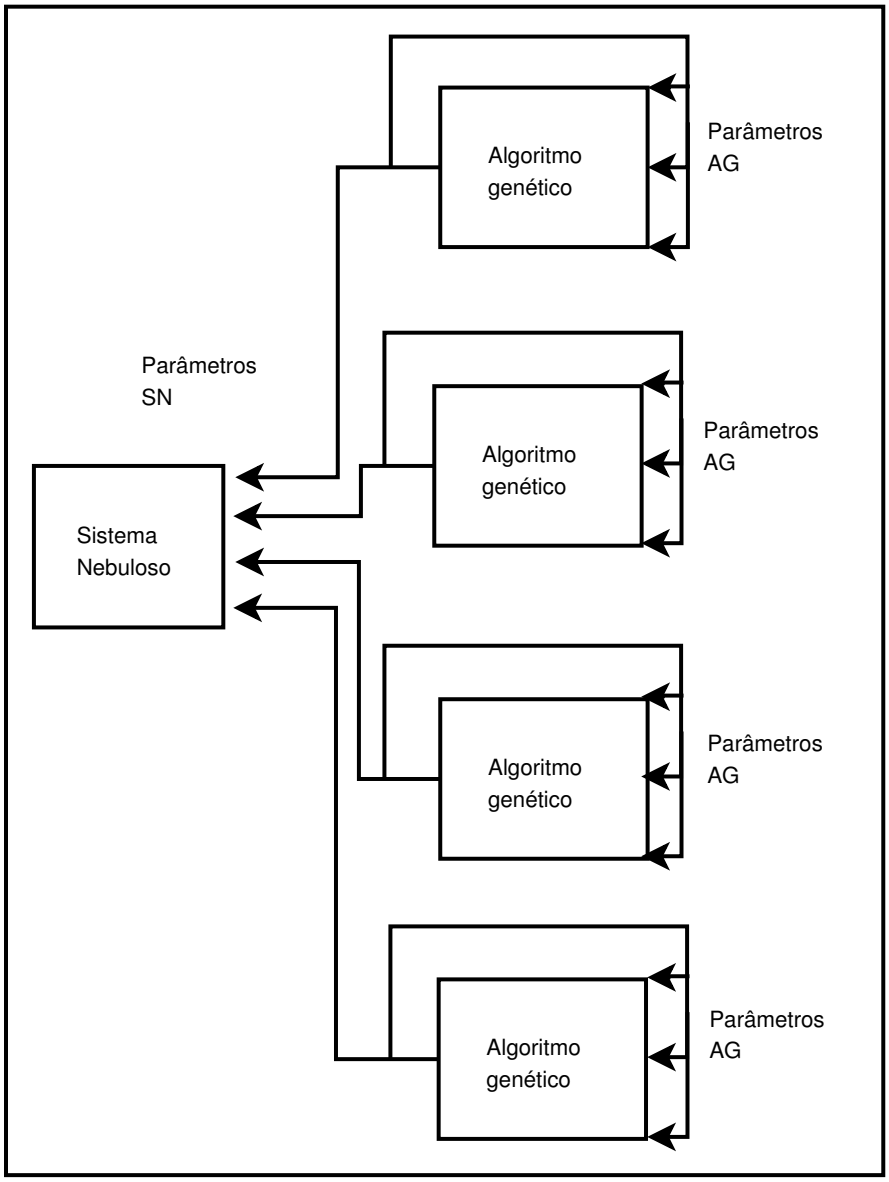



Figura 4.7: Abordagem auto-adaptativa

**(ii)** Quando os pais são relativamente próximos no espaço genotípico.

O segundo caso (ii) é relativamente comum nos estágios finais da evolução. Em (i) e (ii) a recombinação atua como um operador de busca local (POLI; LANGDON, 1998). A mutação, em contrapartida, é um operador que acarreta em mudanças pseudo-aleatórias nos indivíduos evitando a convergência prematura do algoritmo (GOLDBERG, 1989). Além dos operadores de variação é necessário também um mecanismo de seleção para a garantia da sobrevivência do mais apto. Cada um desses parâmetros possui valores distintos e no caso co-evolutivo cada população pode possuir um conjunto diferente desses parâmetros.

Como visto no capítulo 3, a auto-adaptação é realizada a partir da modificação do genótipo do indivíduo adicionando genes que codificam seus parâmetros evolutivos. Em geral, as abordagens encontradas na literatura buscam realizar a adaptação de somente um parâmetro evolutivo de cada vez (SPEARS, 1995; SCHAFFER; MORISHIMA, 1987; FOGARTY, 1989; SMITH; FOGARTY, 1996; THIERENS; GOLDBERG, 1993; SCHAFFER; MORISHIMA, 1987; DE JONG; SPEARS, 1992, 1991; SPEARS; ANAND, 1991; MUMFORD, 2003; THIERENS, 2002; KIMBROUGH *et al.*, 2002). Entretanto, conforme visto no capítulo 3, a definição de um parâmetro por vez tende a obter conjuntos sub-ótimos de parâmetros devido à forma complexa como eles interagem. Com base nessa premissa, a auto-adaptação proposta neste trabalho busca obter valores adequados para os seguintes parâmetros:

- recombinação (taxa e tipo);
- mutação (taxa para todas as codificações, tipo e não-uniformidade se a codificação for real);
- mecanismo de seleção (tamanho do torneio).

Considerando uma população com codificação real, o genótipo modificado, que é proposto nesse trabalho para conter os genes necessários para a auto-adaptação, pode ser visto na figura 4.8:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | Solução | $p_m$ | $p_r$ | tipo rec | tipo mut | não unif | tam. torn |
| 2 | Solução | $p_m$ | $p_r$ | tipo rec | tipo mut | não unif | tam. torn |
| | ⋮ | ⋮ | | | | ⋮ | |
| N | Solução | $p_m$ | $p_r$ | tipo rec | tipo mut | não unif | tam. torn |

Figura 4.8: Genótipo modificado para a auto-adaptação

Onde "Solução" indica os alelos codificando a solução do problema, $p_m$ indica a probabilidade de mutação, $p_r$ indica a probabilidade de recombinação, $tipo\ rec$ é o tipo de recombinação, $não\ unif$ codifica a não uniformidade da mutação e $tam.\ torn$ é o tamanho do torneio.

Para avaliar o desempenho de uma abordagem auto-adaptativa, foi construído um protótipo de um algoritmo genético auto-adaptativo (AG-autoadap). Este AG foi posteriormente modificado para a solução do problema do projeto automático de sistemas nebulosos (Capablanca).

Estas duas abordagens serão testadas em diferentes contextos e os resultados da simulação mostrados no capítulo 5.

**Mecanismo de seleção**

Na abordagem proposta, a cada seleção de um indivíduo é sorteada uma realização aleatória com distribuição uniforme entre $[0,1]$. Caso esta realização seja maior que 0.8 é utilizado o mecanismo tradicional de seleção por torneio determinístico em que o vencedor será o indivíduo com o maior aptidão entre os selecionados. Caso contrário, um mecanismo de seleção por torneio em que o indivíduo com a maior distância genética em relação ao melhor indivíduo é selecionado para sobrevivência. Dessa maneira, espera-se que o algoritmo seja capaz de manter sua diversidade genética caracterizada por uma menor pressão seletiva. Para cromossomos com codificação real a medida é avaliada pela distância Euclidiana, enquanto para cromossomos com codificação inteira ou binária a seleção utiliza a distância de Hamming (GOLDBERG, 1989). O mecanismo de seleção é combinado, neste trabalho, com uma estratégia elitista em que o cromossomo com maior aptidão de uma geração tem sua sobrevivência garantida na geração seguinte.

No sistema AG-autoadap o tamanho do torneio é definido a partir de uma estatística do valor mais freqüente presente na população durante uma geração. O sistema inicializa um vetor de números inteiros $\mathbf{v}_{torn} = [00\cdots0]_{1\times\max\{0.05\phi,2\}}$ com valores nulos. Para cada indivíduo da população, o sistema lê o valor do tamanho do torneio $tam_{torn}$ codificado e incrementa o elemento $v_{torn}[tam_{torn}]$. Ao final desse processo o algoritmo encontra $t = \arg\max tam_{torn} \mid i \in \{1,\ldots 0.05\phi\}$ e utiliza esse valor como o tamanho do torneio. Essa abordagem apresenta a desvantagem de permitir que indivíduos com baixa aptidão comandem a escolha do tipo de torneio. Este mecanismo foi aperfeiçoado para o sistema Capablanca (auto-adaptação em um FIS) em que o tamanho do torneio é definido utilizando uma seleção proporcional à aptidão. O valor codificado no indivíduo selecionado com base na aptidão será utilizado por toda a população durante uma geração. Este método garante que um indivíduo com um valor mais alto de aptidão terá uma maior chance de definir o tamanho de torneio da população.

**Recombinação**

Para a operação de recombinação, é gerado um vetor de realizações aleatórias no intervalo $[0,1]$ de tamanho igual ao tamanho da população ($\varphi$). Cada valor desse vetor é comparado com a taxa de recombinação codificada no genótipo de cada indivíduo no alelo $p_r$. Se a realização for menor que o valor codificado este indivíduo será selecionado para sofrer recombinação.

Conforme visto no capítulo 3, os dois tipos mais comuns de recombinação são a recombinação uniforme e a recombinação de múltiplos-pontos. No AG-autoadap, ambos os indivíduos

têm uma probabilidade de comandar o tipo de recombinação de 0.5. Entretanto essa forma de seleção do tipo não é capaz de privilegiar indivíduos mais aptos e que, a princípio, apresentam melhores valores para seus parâmetros evolutivos. No sistema Capablanca, essa abordagem foi modificada da seguinte maneira: durante o processo evolutivo se dois indivíduos forem selecionados para sofrerem recombinação, o algoritmo deve verificar a aptidão dos dois indivíduos. Realiza-se então uma seleção por roleta entre os dois indivíduos e o tipo de recombinação será o codificado no indivíduo selecionado.

### Mutação

A operação de mutação gera uma matriz de dimensão $\varphi \times (l+p)$ com realizações aleatórias no intervalo $[0,1]$. Essas realizações são comparadas com a taxa de mutação efetiva. [2]. Se a realização aleatória for menor que essa probabilidade o indivíduo deverá sofrer mutação no gene correspondente.

As taxas de mutação são definidas para cada indivíduo baseadas no valor codificado em seu genótipo. Portanto, geralmente, todos os genes do indivíduo apresentam a mesma taxa de mutação $p_m$. Entretanto, o GFS proposto utiliza conhecimento da qualidade individual de alguns genes para alterar a taxa efetiva de mutação. Em genes de populações em níveis hierárquicos superiores, alguns genes codificam apontadores para indivíduos em níveis hierárquicos inferiores. Para esses genes específicos, as taxas de mutação podem ser alteradas conforme a aptidão do indivíduo apontado pelo gene, reafirmando o esquema de cooperação entre os diferentes níveis hierárquicos do sistema. Nestes casos, a taxa de mutação efetiva do indivíduo deverá ser multiplicada por uma constante $H$ contida no intervalo $[1,2]$. Essa constante assume um valor 1 se o gene apontar para o melhor indivíduo da população hierarquicamente inferior e 2 se apontar para o indivíduo com a menor aptidão. Evidentemente, valores intermediários de aptidão correspondem a valores intermediários de $H$.

O tipo de mutação depende do tipo de codificação escolhida. Caso a codificação seja binária, o único tipo de mutação possível é a mutação por mudança-de-bit. Para codificação inteira, é possível utilizar o mesmo mecanismo de mudança de bit da codificação binária. Se a codificação for do tipo real dois tipos de mutação podem ser utilizados: a mutação uniforme e a mutação não uniforme. Se o tipo de mutação codificado for não uniforme o parâmetro de não-uniformidade da mutação será utilizado, caso contrário esse parâmetro é desconsiderado.

[2]Conforme será visto neste capítulo, a taxa de mutação efetiva não corresponde a $p_m$ para genes que codificam níveis hierárquicos inferiores na estrutura co-evolutiva. Neste caso, $p_m$ deverá ser modificado através de uma constante $H$, onde $H$ assume valores entre 1 e 2

A figura 4.9 mostra um exemplo de atuação dos parâmetros evolutivos para a população no nível hierárquico IV. Neste exemplo, o indivíduo 5 apresenta a maior aptidão durante esta geração, o tamanho do torneio foi definido como 3 pelo quinto indivíduo (conforme visto, a probabilidade de um indivíduo comandar o tamanho do torneio é proporcional à sua aptidão). Os indivíduos 1 e 5 são selecionados para sofrer recombinação após a comparação de suas probabilidades de recombinação com a realização aleatória $v$. O tipo de recombinação é escolhido através de uma seleção proporcional à aptidão entre os dois indivíduos 1 e 5. Supondo uma vitória do indivíduo 5 (que apresenta a maior aptidão), o tipo de recombinação selecionado é uniforme (tipo 2 codificado no indivíduo 5). O operador de mutação age nos indivíduos 1, 2, 3 e 5. Os genes dos indivíduos 1 e 2 sofrem mutação uniforme enquanto o indivíduo 3 sofre ação de um operador de mutação não-uniforme. A taxa de mutação efetiva do gene no *locus* 5 do indivíduo 5 é alterada devido ao efeito da aptidão de indivíduos em nível hierárquico inferior. Para o indivíduo 5, a taxa de mutação, para o $5^o$ *locus* é multiplicada pelo fator de co-evolução $H = 2$ por estar associado ao indivíduo de menor aptidão da população hierarquicamente inferior($30^a$ base de regras).

## 4.4 Considerações finais

O sistema de inferência nebuloso proposto por Delgado (2002) apresenta bom desempenho nos critérios de acuidade e interpretabilidade. Entretanto, o uso de múltiplas populações aumenta significativamente a quantidade de parâmetros de entrada. A obtenção desses parâmetros é um problema mal-definido, mal-estruturado e complexo. Conseqüentemente o projeto de um GFS utilizando a abordagem co-evolutiva apresenta dificuldades. Nesse sentido, o uso de auto-adaptação de parâmetros parece ser uma alternativa interessante. Ela permite que cada indivíduo guie seu próprio processo evolutivo permitindo velocidades diferentes para pontos diferentes no espaço de busca.

Além disso, conforme será visto no próximo capítulo, o uso de indivíduos codificando bases de regras com codificação inteira apresentou um desempenho muito abaixo do esperado. Isso ocorreu, provavelmente, porque a codificação inteira permite que uma base de regras codifique várias vezes a mesma regra nebulosa em genes diferentes. Nestas condições, a codificação inteira dificulta a presença de outras regras em indivíduos mais aptos. Para resolver esse problema foi utilizada uma codificação binária que evita esse problema mas torna mais complexo o controle da simplicidade da base de regras nebulosas.

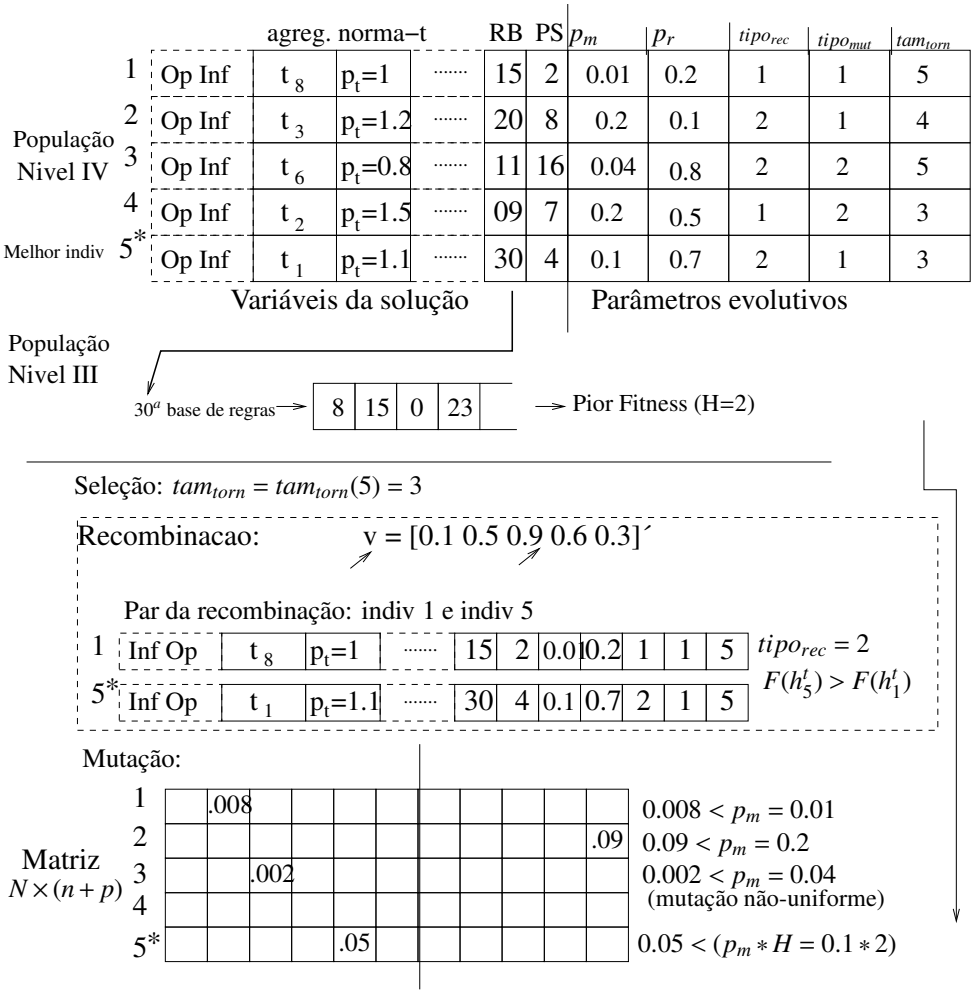



Figura 4.9: Atuação dos operadores genéticos

# 5 *Simulações e resultados*

O objetivo deste capítulo é avaliar o desempenho da auto-adaptação dos parâmetros evolutivos em diferentes contextos, em especial no problema de ajuste de parâmetros de sistemas genético-nebulosos. A abordagem auto-adaptativa será comparada a um algoritmo genético com parâmetros fixos baseados nos valores recomendados por de Jong (1975) e também com outras abordagens tradicionais descritas na literatura.

Este capítulo traz as simulações realizadas para diversos problemas: otimização de funções contínuas, um problema de otimização combinatória: o problema da mochila, aproximação de funções com e sem ruído. A seção 5.1 apresenta as abordagens utilizadas na comparação do desempenho da metodologia proposta. Nesta seção serão feitas considerações gerais sobre essas abordagens. A seção 5.2 avalia o desempenho do algoritmo genético auto-adaptativo através de problemas combinatoriais e de otimização contínua. A seção 5.3, a mais importante deste capítulo, avalia a abordagem proposta no capítulo 4 denominada Capablanca. Esta abordagem considera a auto-adaptação dos parâmetros estratégicos sob o problema de ajuste de parâmetros de um sistema nebuloso o qual será testado através do problema de aproximação de funções. A seção 5.3.1 apresenta o treinamento do sistema Capablanca com dados na ausência de ruído ao contrário da seção 5.3.2 na qual os dados foram perturbados por um ruído de distribuição normal para avaliar o efeito do sobre-treinamento.

Na avaliação do desempenho em problemas de otimização contínua e combinatorial o critério de qualidade da abordagem será a proximidade com o resultado ótimo conhecido. Na avaliação do problema de ajuste de funções, o critério de qualidade do desempenho será o erro médio quadrático obtido através de um conjunto de dados de teste.

A escolha de problemas combinatoriais se baseia na dificuldade de resolver esta classe de problemas com técnicas convencionais. A complexidade temporal de um problema, ou seja, o tempo necessário para resolver uma instância desse problema utilizando o algoritmo mais eficiente (KNUTH, 1973), está diretamente relacionada ao número de passos necessários para resolver uma instância desse problema. A classe de complexidade $P$ (polinomial) contém problemas que podem ser resolvidos por um dispositivo determinístico em tempo polinomial. A classe

de problemas de complexidade *NP* (Não-Polinomial) contempla problemas que podem ser solucionados por uma máquina não-determinística em tempo polinomial (GAREY; JOHNSON, 1979). Entretanto as soluções para esse tipo de problemas podem ser verificadas em tempo polinomial por um dispositivo determinístico. Alguns dos problemas mais famosos dessa classe são o problema do caixeiro viajante (TSP) (ROSENKRANTZ; STEARNS; II, 1977) e o problema da Mochila (KIMBROUGH *et al.*, 2002). A necessidade de diminuir o tamanho do espaço de busca motivou o uso de heurísticas na resolução destes problemas.

O uso de problemas de otimização contínua para avaliar o desempenho de algoritmos genéticos foi bastante discutido por de Jong (1975). Essas funções possuem características específicas que dificultam significativamente o processo de busca da solução ótima (presença de multimodos, não-separabilidade, com regiões aproximadamente planas, etc). Segundo Roubos e Babuska (2003), entretanto, o desempenho de um algoritmo em um conjunto de funções de teste não é um indicador confiável de sua qualidade em problemas reais.

A aproximação de funções através de sistemas nebulosos, desde que projetados com requisitos mínimos de interpretabilidade, apresenta a possibilidade de compreensão por um operador humano além de um grau razoável de imunidade ao ruído. O principal critério de qualidade das soluções adotado no problema de aproximação de funções adotado nesta seção será a acuidade do sistema. Um critério secundário a ser considerado é a interpretabilidade do sistema, mensurada através de critérios de visibilidade, simplicidade, compactação e consistência, conforme descrito no capítulo 4. A autonomia, medida pelo número de parâmetros a serem definidos pelo usuário também será analisada em todos os testes realizados.

Os resultados para o problema de otimização contínua serão comparados com os obtidos por Herrera e Lozano (2001) que utilizaram uma abordagem adaptativa através de um controlador nebuloso. Os resultados dos problemas combinatoriais com o algoritmo AG-autoadap serão confrontados com abordagens encontradas na literatura (KIMBROUGH *et al.*, 2002) e com o algoritmo genético proposto sem o uso da auto-adaptação (AG-fixo). Os resultados obtidos pela abordagem proposta (Capablanca) no problema de ajuste de funções serão confrontados com os obtidos pelas outras técnicas de inteligência computacional, particularmente abordagens baseadas em redes neurais artificiais, como o ANFIS (JANG; SUN; MIZUTANI, 1997), RBF e MLP por exemplo, e o próprio algoritmo genético proposto utilizando parâmetros fixos (fix-Capablanca). Quando aplicáveis, testes estatísticos serão realizados para avaliar a confiabilidade do resultado obtido.

A seção a seguir descreve mais detalhadamente cada uma das abordagens utilizadas.

## 5.1 Abordagens de comparação

Essa seção descreve os modelos utilizados ao longo desse capítulo. Os sistemas propostos (AG-autoadap, Capablanca e fix-Capablanca) tiveram seu desempenho comparado com outros paradigmas baseados em inteligência computacional.

**Abordagens propostas** Para validação da abordagem proposta, alguns protótipos foram implementados:

**AG-autoadap** consiste em um sGA com parâmetros auto-adaptativos que permite o uso de codificações binária, inteira e real. O sistema utiliza a auto-adaptação para obter automaticamente os parâmetros de seus operadores genéticos (taxa e tipo de recombinação e mutação) e de seu mecanismo de seleção. Contudo não é capaz de determinar automaticamente o tamanho de sua população nem o número máximo de gerações. Esses parâmetros devem ser obrigatoriamente passados pelo usuário. É possível definir limites superiores e inferiores para limitar a variação das taxas de mutação e recombinação. Na ausência desses limites o sistema utiliza um conjunto de valores "padrão" para os limites.

**fix-Capablanca** é uma forma simplificada do sistema coevol-GFS. A principal diferença aparece na codificação do nível hierárquico III. O coevol-GFS utiliza uma codificação inteira e o fix-Capablanca, em contrapartida, utiliza uma codificação binária.

**Capablanca** é a versão com parâmetros auto-adaptativos do sistema fix-Capablanca. Assim como o AG-autoadap, o tamanho das populações e o número máximo de gerações devem ser determinados pelo usuário.

**AG fixo** Visando avaliar os benefícios resultantes da auto-adaptação, o algoritmo genético auto-adaptativo foi comparado com uma versão similar mas que utiliza parâmetros evolutivos com valores fixos. Esses parâmetros foram estabelecidos de acordo com os valores sugeridos por de Jong (1975). Esta abordagem é essencialmente idêntica ao sGA descrito por Goldberg (GOLDBERG, 1989).

**Herrera e Lozano (2001)** Com o intuito de comparar a auto-adaptação com outras abordagens de adaptação de parâmetros, o algoritmo proposto teve seu desempenho comparado com a abordagem proposta por Herrera e Lozano (2001) em problemas de otimização de funções. Nesse trabalho Herrera construiu um sistema nebuloso para controlar automaticamente os valores dos parâmetros evolutivos. Apesar do excelente desempenho dessa abordagem, ela apresenta algumas desvantagens. O sistema nebuloso deve ser projetado

pelo usuário, demandando um conhecimento prévio nesta área. Outro problema encontrado é a variação do sistema nebuloso com melhor desempenho de acordo com a função otimizada. Dessa forma além do conhecimento prévio do usuário é necessário projetar um sistema nebuloso para cada problema a ser resolvido.

**Kimbrough *et al.* (2002)** No problema da mochila 0/1 a abordagem proposta é comparada com a abordagem proposta por Kimbrough *et al.* (2002) na qual utilizou-se um algoritmo genético com auto-adaptação exclusivamente nas taxas de mutação. A codificação binária utilizada por Kimbrough impede que a presença/ausência de um item apareça repetida em outro gene de um mesmo indivíduo.

**ANFIS** Para o problema do projeto automático de sistemas nebulosos uma abordagem de comparação baseada no paradigma neuro-nebuloso foi utilizada. O sistema ANFIS (JANG; SUN; MIZUTANI, 1997) proposto por Jang e Sun é reconhecidamente uma das referências mais importantes na comparação de técnicas de projeto de sistemas nebulosos baseados em aprendizado e adaptação (NAUCK; KRUSE, 1999; RIZZI; MASCIOLI; MARTINELLI, 1999). Este paradigma pode ser resumido da seguinte forma:

- Os operadores de inferência e formatos das funções de pertinência são fornecidos pelo usuário;
- Baseado na granularidade das partições nebulosas resultantes, uma base de regras fixa é gerada utilizando todas as combinações possíveis de funções de pertinência em universos de discursos diferentes sem utilizar regras com antecedentes do tipo *don't-care*;
- O treinamento do sistema ANFIS consiste em duas partes:
  - Durante o início da fase de treinamento os parâmetros dos conseqüentes de Takagi-Sugeno são obtidos através do método dos mínimos quadrados;
  - Os parâmetros dos antecedentes (partições nebulosas) são ajustados a partir de uma rede neural não-recorrente treinada pelo algoritmo da retropropagação utilizando os conseqüentes obtidos pelo método dos mínimos quadrados na primeira parte da fase de treinamento.

Essa abordagem possui um bom desempenho em problemas de aproximações de funções. Entretanto, algumas desvantagens podem ser observadas:

- O usuário precisa fornecer parâmetros de entrada como o formato das funções de pertinência e o mecanismo de inferência;

- O sistema não é capaz de realizar simplificações no número de funções de pertinência;
- O sistema não é capaz de realizar simplificações no número de regras na base de regras;
- O sistema não é capaz de utilizar regras com antecedentes do tipo *don't care*;
- As funções de pertinência escolhidas devem ser necessariamente contínuas e diferenciáveis;
- Não é possível realizar a avaliação da qualidade do sistema na ausência de dados de treinamento;
- Não há garantias de que o resultado obtido seja interpretável lingüisticamente.

**Abordagens baseadas em redes neurais artificiais** Na seção 5.3 os resultados do problema de aproximação de funções foram confrontados também com abordagens baseadas em redes neurais artificiais:

- Redes neurais com funções de base radial (RBF) (PARK; SANDBERG, 1991): Neste caso foi utilizada a função *newrbe* do MATLAB (THE MATHWORKS, INC., 2004) a qual usa uma matriz $P$ de dados de treinamento de entrada, um vetor $T$ contendo a saída desejada e uma constante de espalhamento *spread* para a camada contendo as funções de base radial. Esta função utiliza, um número de neurônios igual ao número de amostras de entrada em $P$. Cada *bias* na primeira camada é arbitrado com o valor $\frac{0.8326}{spread}$ garantindo a cobertura do espaço (THE MATHWORKS, INC., 2004). O valor de *spread* deve ser grande o bastante para que os neurônios apresentem uma resposta adequada a entradas que não pertencem aos dados de treinamento. Os pesos e *biases* da segunda camada são obtidos através da simulação da primeira camada e a solução da expressão linear:

$$\begin{bmatrix} w_2 & b_2 \end{bmatrix} \begin{bmatrix} o_1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = T \tag{5.1}$$

  onde $w_2$ indica o vetor de pesos da segunda camada, $b_2$ os *biases* da segunda camada e $o_1$ a saída da primeira camada.

  Caso a saída da segunda camada $o_1$ e a saída desejada sejam conhecidas e a segunda camada seja linear, o vetor de pesos pode ser calculado, de forma a minimizar o erro

médio quadrático, da seguinte forma:

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = T / \begin{bmatrix} & P & \\ 1 & \cdots & 1 \end{bmatrix} \tag{5.2}$$

Este problema apresenta $P$ restrições (pares de treinamento) e $P+1$ variáveis ($P$ pesos e 1 bias). Um problema linear com mais variáveis que restrições apresenta um número infinito de soluções com erro zero (MEYER, 2000).

O principal problema das redes neurais artificiais geradas pela função *newrbe* é a produção de uma camada intermediária com um número de neurônios igual ao número de amostras de treinamento. Portanto, a solução pode não ser aceitável em casos em que o número de vetores de entrada torna-se alto.

Os problemas de aproximação de funções foram executados com 225 amostras de treinamento, logo o número de neurônios com função de base radial é 225. O valor de *spread* foi arbitrado como 0.02.

- Perceptron de múltiplas camadas (MLP) (HAYKIN, 1998): Uma rede MLP treinada com o algoritmo da retropropagação pode ser visto como um método de realizar um mapeamento não-linear de entrada-saída de natureza geral. De acordo com o teorema da aproximação universal (HAYKIN, 1998) uma única camada oculta é suficiente para uma rede MLP computar uma aproximação uniforme para um dado conjunto de treinamento. Nos problemas tratados neste capítulo a rede MLP foi projetada com uma única camada intermediária e 25 neurônios na camada oculta.

  As redes RBF e MLP são exemplos de redes em camadas não-recorrentes, não lineares. Ambos são aproximadores universais porém essas duas arquiteturas diferem entre si em vários aspectos importantes (HAYKIN, 1998):

  – Uma rede RBF (em sua forma mais básica) tem uma única camada oculta, enquanto uma MLP pode ter uma ou mais camadas ocultas;

  – Tipicamente, os nós computacionais de um MLP, localizados em uma camada oculta ou em uma camada de saída, compartilham um modelo neural comum. Por outro lado, os nós computacionais na camada oculta de uma rede RBF são bastante diferentes e têm um propósito diferente daqueles da camada de saída da rede;

  – A camada oculta de uma rede RBF é não linear, enquanto que a camada de saída é linear. Em contrapartida, as camadas ocultas e de saída de um MLP usado como classificador de padrões são normalmente todas não-lineares. Quando o MLP é usado para resolver problemas de regressão não-linear, uma camada

linear para a saída é normalmente a escolha mais comum;

- O argumento da função de ativação de cada unidade oculta em uma rede RBF calcula a norma euclidiana entre o vetor de entrada e o centro daquela unidade, Enquanto isso, a função de ativação de cada unidade oculta em um MLP calcula o produto interno do vetor de entrada pelo vetor de peso sináptico daquela unidade;
- Os MLPs constroem aproximações globais de um mapeamento de entrada-saída não-linear. Por outro lado, as redes RBF utilizando não-linearidades localizadas com decaimento exponencial usam aproximações locais para mapeamentos de entrada-saída não lineares

### 5.1.1 Considerações sobre as abordagens de comparação

Algumas considerações devem ser realizadas na análise dos resultados obtidos:

**Número de rodadas** As abordagens baseadas em algoritmos genéticos podem apresentar um comportamento distinto para rodadas do mesmo algoritmo. Nesse contexto é necessário realizar múltiplas observações com o intuito de fazer análises estatísticas, que são mais significativas que resultados de uma única rodada.

- O AG-autoadap e o AG-fix foram executados 100 vezes para o problema de otimização de funções contínuas e 500 vezes para o problema da mochila;
- Herrera e Lozano (2001) utilizaram 30 rodadas para cada função contínua aproximada;
- Kimbrough *et al.* (2002) utilizaram 5 rodadas, porém com número de gerações relativamente elevado;
- O ANFIS (JANG; SUN; MIZUTANI, 1997) foi executado somente uma vez por que o mecanismo de inicialização dos pesos é determinístico;
- O RBF (PARK; SANDBERG, 1991) foi executado somente uma vez também porque o mecanismo de inicialização de pesos da implementação utilizada é determinístico;
- O MLP (HAYKIN, 1998) foi executado 30 vezes, cada uma com um conjunto de pesos diferentes gerados aleatoriamente;
- Capablanca/fix-Capablanca: o algoritmo Capablanca e sua versão fixa foram executados 30 vezes para cada problema proposto.

**Tamanho fixo das populações** Para as abordagens baseadas em algoritmos genéticos os tamanhos das populações ($\varphi$) foram estabelecidos da seguinte forma (o tempo de processamento aumenta linearmente com $\phi$):

- AG-autoadap e AG-fix utilizaram populações de $\varphi = 100$ indivíduos para os problemas de otimização de funções contínuas e da mochila;
- Herrera e Lozano (2001) utilizaram populações fixas de $\varphi = 5000$ indivíduos no problema de minimização de funções com 25 dimensões;
- Kimbrough *et al.* (2002) utilizaram populações com tamanho fixo de $\varphi = 50$ indivíduos para resolver o problema da mochila 0/1;
- Capablanca/fix-Capablanca: Analogamente ao coevol-GFS, o sistema proposto apresenta quatro populações de tamanho fixo co-evoluindo simultaneamente. Da mesma forma os tamanhos das populações são identificados como $\varphi_I$, $\varphi_{II}$, $\varphi_{III}$ e $\varphi_{IV}$.

**Operadores evolutivos** Os tipos de operadores genéticos são escolhidos com base no tipo de codificação do cromossomo.

**Recombinação** O processo evolutivo permite que o cromossomo sofra ação de dois tipos de recombinação ($\rho$): um-ponto e uniforme. O processo de troca de material genético da recombinação para codificação real pode manter parte da informação do gene original (como por exemplo uma média ponderada) (EIBEN; HINTERDING; MICHALEWICZ, 1999) ou simplesmente transferir o material genético entre os indivíduos. Na abordagem fix-Capablanca/Capablanca foi utilizada somente a abordagem de troca de material genético;

**Mutação** O operador de mutação ($\xi$) depende da codificação do cromossomo. Para a codificação binária o único tipo de mutação possível é a mutação pela inversão de bit. O tipo mais comum de mutação em cromossomos inteiros é a simples troca um dos bits da representação binária do número inteiro por seu complemento. Essa abordagem apresenta grande simplicidade computacional e generalidade por funcionar para qualquer alfabeto inteiro. A codificação real permite a modulação do passo da mutação de forma simples. Para essa forma de codificação os operadores de mutação mais comuns são a mutação uniforme e a mutação não-uniforme (MICHALEWICZ, 1994).

- AG-autoadap utilizou limites inferiores para a taxa de mutação maiores que os normais no problema de otimização de funções contínuas. Para esse problema, a taxa

de mutação mínima foi arbitrada como 0.04. No problema da mochila, os resultados utilizaram uma lógica diferente para determinação do tipo de recombinação e do torneio, baseado na escolha do valor com maior incidência em toda a população (MARUO; LOPES; DELGADO, 2004). Para outros problemas é possível arbitrar limites para os parâmetros evolutivos ou utilizar os valores padrão.

- Herrera e Lozano (2001) utilizaram codificação inteira com recombinação de um ponto e mutação com taxa fixa mas passo ajustado por um controlador nebuloso;
- Kimbrough *et al.* (2002) utilizaram codificação binária e mutação por troca de bit e recombinação de múltiplos pontos
- fix-Capablanca utiliza recombinação de um ponto com escolha aleatória do ponto de troca, mutação uniforme nos genes com codificação real e mutação por mudança de bit nos genes com codificação inteira e binária.
- Capablanca utiliza recombinação de um ponto com escolha aleatória do ponto de troca ou uniforme com número de pontos de troca aleatório, dependendo do genótipo do indivíduo. O algoritmo permite o uso de mutação uniforme ou não-uniforme nos genes com codificação real, de acordo com o genótipo do indivíduo, e mutação por mudança de bit nos genes com codificação inteira e binária.

**Iterações** Para as abordagens baseadas em algoritmos genéticos uma iteração do algoritmo corresponde a uma geração do processo evolutivo ($t$). No caso das abordagens baseadas em redes neurais, cada iteração corresponde a um ciclo de treinamento. É importante notar que o custo computacional de um ciclo de uma rede neural é muito menor que uma geração de um algoritmo evolutivo devido à necessidade de avaliar múltiplos indivíduos de uma população (JANG; SUN; MIZUTANI, 1997).

## 5.2 Resultados do AG-autoadap: Algoritmo genético auto-adaptativo

Nesta seção o algoritmo AG-autoadap será confrontado com outras abordagens para problemas de otimização contínua e combinatória.

### 5.2.1 Otimização de funções contínuas

Seja $\mathbb{R}^n$ o espaço euclidiano n-dimensional. O problema de minimização irrestrita de uma função contínua $f(\mathbf{x})$ consiste em encontrar $\hat{\mathbf{x}} = [x_1, x_2, \ldots x_n] \mid f(\hat{\mathbf{x}}) < f(\mathbf{x}) \forall \mathbf{x} \in \mathbb{R}^n$. A seguir

serão apresentadas algumas instâncias desse problema. As funções utilizadas são baseadas no trabalho de de Jong (1975). Entretanto para incrementar a dificuldade da busca foram utilizadas 25 variáveis de entrada. Tanto os processos de codificação e decodificação dos cromossomos como os operadores genéticos assumem que as variáveis do problema possuem limites superior e inferior finitos (GOLDBERG, 1989). Para permitir o uso de de algoritmos genéticos, a busca foi restrita à região compacta $\Omega \subset \mathbb{R}^n \mid \hat{\mathbf{x}} \in \Omega$.

Para todos os problemas de otimização contínua foram comparados algoritmos genéticos com codificação real (contínua) e inteira (discreta). Para os algoritmos com codificação inteira, foi realizada uma análise do tamanho do espaço de busca. Para os algoritmos com codificação real, o espaço de busca é sempre infinito.

Os algoritmos genéticos, por convenção, resolvem problemas de maximização sendo necessário transformar o problema de minimização em um problema de maximização. Isso foi realizado fazendo:

$$F(\mathbf{x}) = -f(\mathbf{x}) \tag{5.3}$$

Evidentemente, o uso de aptidões negativas dificulta significativamente o uso de seleção por roleta. As duas estratégias mais comuns são utilizar um escalonamento da aptidão em que o menor indivíduo tenha aptidão zero ou o uso de escalonamento sigma (HANCOCK, 1994).

A seguir serão descritas as funções contínuas utilizadas na etapa de simulação e obtenção de resultados:

**Função Esférica**

Considere o seguinte problema:

$$\min f_{sph}(\mathbf{x}) = \sum_{i=1}^{25} x[i]^2 \tag{5.4}$$

onde:

- $x[i]$ representa o elemento de índice $i$ no vetor de entrada $\mathbf{x}$ tal que $x[i] \in \Omega_i \mid -5.12 < x[i] < 5.12$, para $1 \leq i \leq 25$;
- $f_{S\,ph}(\mathbf{x})$ é uma função contínua, estritamente convexa e unimodal (HERRERA; LOZANO, 2001);

- $\hat{\mathbf{x}} = [0,0,\ldots,0]$;
- $f(\hat{\mathbf{x}}) = 0$.

A busca do mínimo global dessa função é um problema relativamente simples facilmente solucionável com técnicas de programação linear devido a sua convexidade. Utilizando uma codificação inteira, para uma precisão de duas casas decimais [1], o número mínimo de bits necessários para representar cada variável é 10. Considerando o problema para 25 variáveis tem-se um total de 250 bits por solução-candidata. Logo, o espaço de busca para esse problema é $2^{250} = 1.8093 \times 10^{75}$.

A função esférica para duas dimensões pode ser observada na figura 5.1. Os resultados da minimização da função esférica para 25 dimensões obtidos pelas diferentes abordagens estão na tabela 5.1

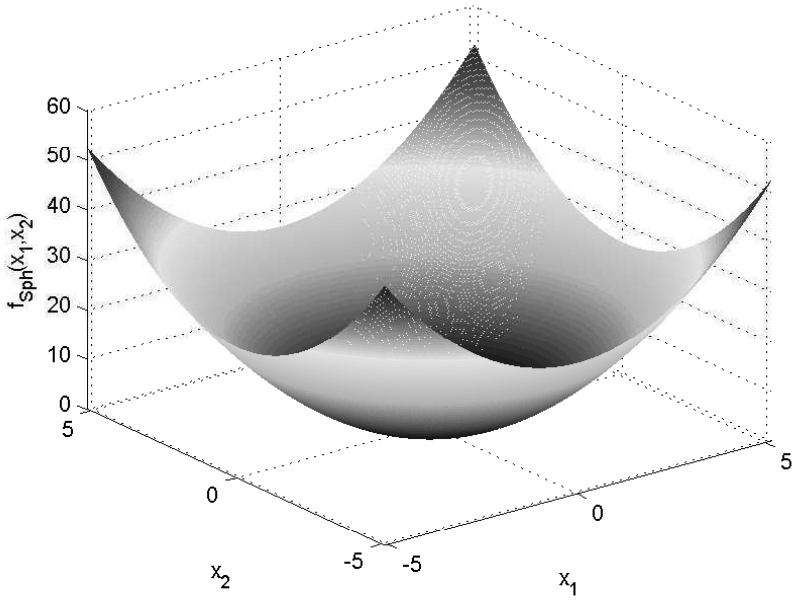



Figura 5.1: Gráfico da função esférica para duas dimensões

Onde Iteração(Melhor) indica a menor iteração na qual foi obtido o melhor resultado.

[1] A restrição ao número de casas decimais na codificação não está ligada ã precisão da função de aptidão do indivíduo na tabela 5.1. A função de aptidão sempre apresenta um contra-domínio real.

Tabela 5.1: Resultados da otimização da função esférica

| Abordagem | Média±Desv. pad. | Melhor Resultado |
|---|---|---|
| AG-autoadap(inteiro) | $0.00000 \pm 0.00000$ | 0.00000 |
| AG-fix(inteiro) | $0.00000 \pm 0.00000$ | 0.00000 |
| AG-autoadap(real) | $0.00000 \pm 0.00000$ | 0.00000 |
| AG-fix(real) | $0.00000 \pm 0.00000$ | 0.00000 |
| Herrera e Lozano (2001) | $0.00000 \pm 0.00000$ | 0.00000 |

**Função de Rosenbrock**

Considere o seguinte problema:

$$\min f_{Ros}(\mathbf{x}) = \sum_{i=1}^{24} \left[ 100(x[i+1] - x[i]^2)^2 + (x[i] - 1)^2 \right] \tag{5.5}$$

onde:

- $x[i]$ representa o elemento de índice $i$ no vetor de entrada $\mathbf{x}$ tal que $x[i] \in \Omega_i \mid -5.12 < x[i] < 5.12$, para $1 \leq i \leq 25$;
- $f_{Ros}(\mathbf{x})$ é uma função contínua, unimodal, com o ponto ótimo localizado em um vale parabólico íngreme com um fundo aproximadamente plano. Além disso $f_{Ros}$ é uma função não-separável, ou seja, suas variáveis interagem de forma não-linear (HERRERA; LOZANO, 2001);
- $\hat{\mathbf{x}} = [1, 1, \ldots, 1]$;
- $f(\hat{\mathbf{x}}) = 0$.

A busca do ótimo da função de Rosenbrock é um problema difícil pois o fundo do vale é aproximadamente plano, o que causa um progresso lento em vários algoritmos de busca. Essa característica faz com que esses algoritmos devam modificar a direção da busca continuamente para alcançar o ótimo. Além disso essa função possui interações não lineares (ou seja, é uma função não separável). Devido a essa característica, alguns autores a consideram um desafio para qualquer algoritmo de otimização contínua (ROJAS *et al.*, 1999). Assim como a otimização da função esférica o tamanho do espaço de busca da função de Rosenbrock, utilizando codificação inteira, é $2^{250} = 1.8093 \times 10^{75}$.

A função Rosenbrock para duas dimensões pode ser observada na figura 5.2. Os resultados da minimização da função Rosenbrock para 25 dimensões obtidos pelas abordagens estão na tabela 5.2

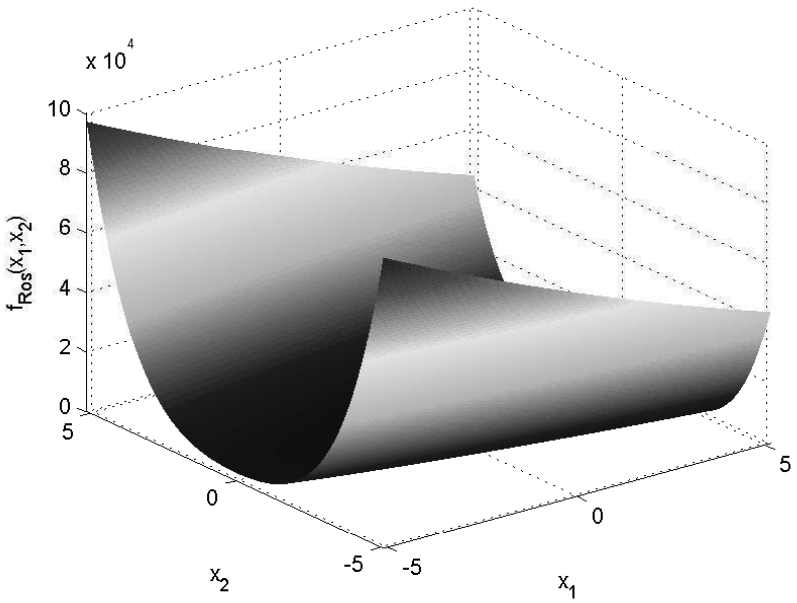



Figura 5.2: Gráfico da função Rosenbrock para duas dimensões

Tabela 5.2: Resultados da otimização da função Rosenbrock

| Abordagem | Média±Desv. pad. | Melhor Resultado |
|---|---|---|
| AG-autodap(inteiro) | $10.3525 \pm 3.50602$ | 0.56637 |
| AG-fix(inteiro) | $26.6064 \pm 5.02435$ | 0.99255 |
| AG-autoadap(real) | $10.9534 \pm 3.45678$ | 0.81352 |
| AG-fix(real) | $29.2324 \pm 5.23434$ | 1.02345 |
| Herrera e Lozano (2001) | $\mathbf{10.2 \pm 4.91}$ | 0.00000 |

Para as abordagens propostas, o melhor desempenho foi obtido utilizando codificação inteira com auto-adaptação. O AG-autoadap teve desempenho inferior em comparação com o trabalho de Herrera e Lozano. Entretanto este algoritmo foi projetado utilizando conhecimento específico da função de otimização (diminuição das taxas de mutação próximas do valor ótimo). O AG-autoadap, em contrapartida, é mais genérico pois não utilizou nenhuma otimização específica para o problema. Outra desvantagem da abordagem de Herrera e Lozano é a necessidade do projeto de um controlador nebuloso para cada problema tratado.

**Função de Schwefel**

Considere o seguinte problema:

$$\min f_{S\,ch}(\mathbf{x}) = \sum_{i=1}^{25}\left(\sum_{j=1}^{i} x[j]\right)^2 \tag{5.6}$$

onde:

- $x[i]$ representa o elemento de índice $i$ no vetor de entrada $\mathbf{x}$ tal que $x[i] \in \Omega_i \mid -65.536 < x[i] < 65.536$, para $1 \leq i \leq 25$;
- $f_{S\,ch}(\mathbf{x})$ é uma função contínua, unimodal, com o ponto ótimo localizado em um vale ainda mais íngreme que a equação 5.5 (HERRERA; LOZANO, 2001);
- $\hat{\mathbf{x}} = [0, 0, \ldots, 0]$;
- $f(\hat{\mathbf{x}}) = 0$.

A busca do ótimo da função de Schwefel é um problema difícil pois a busca através dos eixos coordenados apresenta baixa taxa de convergência, já que o gradiente de $f_{S\,ch}(\mathbf{x})$ não é orientado ao longo dos eixos. Além disso apresenta dificuldades similares às da função 5.5 mas com um vale ainda mais estreito (HERRERA; LOZANO, 2001). Utilizando uma codificação inteira, para uma precisão de três casas decimais, o número mínimo de bits necessários para representar cada variável é 17. Considerando o problema para 25 variáveis tem-se um total de 425 bits por solução-candidata. Logo, o espaço de busca para esse problema é $2^{425} = 8.6646 \times 10^{127}$ (o número estimado de átomos no universo é estimado em, no máximo, $10^{90}$ (HORÄK; KADLEC; SMRZ, 2002)).

A função Schwefel para duas dimensões pode ser observada na figura 5.3. Os resultados da minimização da função Schwefel para 25 dimensões obtidos pelas abordagens utilizadas estão na tabela 5.3

Tabela 5.3: Resultados da otimização da função Schwefel

| Abordagem | Média±Desv. pad. | Melhor Resultado |
|---|---|---|
| AG-autoadap(inteiro) | $0.12713 \pm 0.10372$ | 0.02314 |
| AG-fix(inteiro) | $0.35435 \pm 0.23264$ | 0.11744 |
| AG-autoadap(real) | $0.99112 \pm 0.82341$ | 0.18214 |
| AG-fix(real) | $1.10122 \pm 1.01282$ | 0.11346 |
| Herrera e Lozano (2001) | $\mathbf{9.25 \times 10^{-9} \pm 1.46 \times 10^{-8}}$ | 0.000000 |

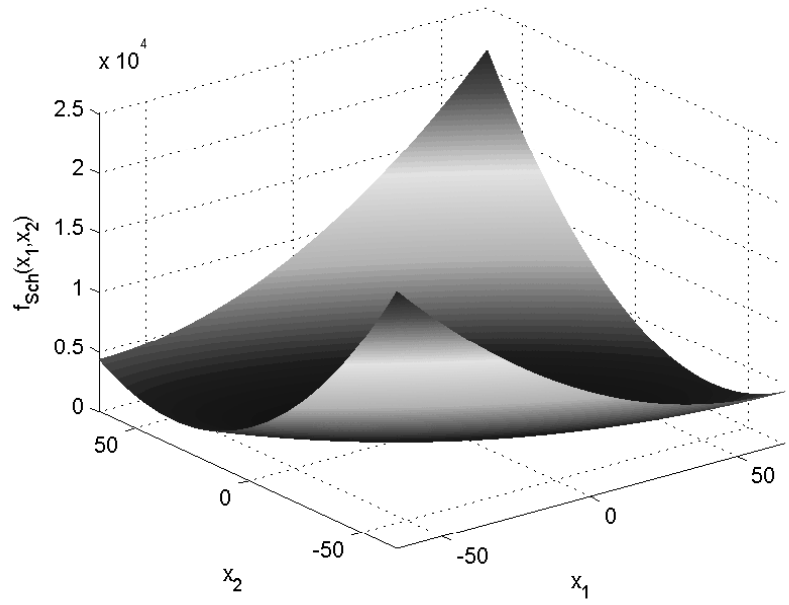



Figura 5.3: Gráfico da função Schwefel para duas dimensões

As abordagens propostas tiveram desempenho pior que o trabalho de Herrera e Lozano. Entretanto deve-se ressaltar, novamente, que o trabalho de Herrera utilizou conhecimento específico das funções otimizadas diminuindo a autonomia do sistema enquanto o AG-autoadap foi proposto com o intuito de servir como um algoritmo genético genérico, ou seja, não-dedicado a nenhuma aplicação. O algoritmo com auto-adaptação e codificação inteira apresentou o melhor desempenho dentre as abordagens propostas.

**Função de Griewangk**

Considere o seguinte problema:

$$\min f_{Gri}(\mathbf{x}) = \frac{1}{4000}\left(\sum_{i=1}^{25} x[i]^2\right) - \prod_{i=1}^{25} cos\left(\frac{x[i]}{\sqrt{i}}\right) + 1 \tag{5.7}$$

onde:

- $x[i]$ representa o elemento de índice $i$ no vetor de entrada $\mathbf{x}$ tal que $x[i] \in \Omega_i \mid -600.0 < x[i] < 600.0$, para $1 \leq i \leq 25$;
- $f_{Gri}(\mathbf{x})$ é uma função contínua, multimodal, não-separável;
- $\hat{\mathbf{x}} = [0,0,\ldots,0]$;

- $f(\hat{\mathbf{x}}) = 0$.

A busca do ótimo da função de Griewangk é um problema difícil, pois trata-se de uma função multimodal em que o algoritmo de busca deve subir um "morro" para chegar no próximo vale. Em contrapartida, uma propriedade indesejável é que ela torna-se mais fácil com o aumento da dimensão (HERRERA; LOZANO, 2001). Para uma precisão de uma casa decimal o número mínimo de bits necessários para representar cada variável é 11. Considerando o problema para 25 variáveis teremos um total de 275 bits por solução-candidata. Logo, o espaço de busca para esse problema é $2^{275} = 6.0708 \times 10^{82}$.

A função Griewangk para duas dimensões pode ser observada nas figuras 5.4(a) e 5.4(b). Os resultados da minimização a função Griewangk para 25 dimensões das abordagens utilizadas podem ser observados na tabela 5.4

O gráfico 5.4(a) mostra uma função semelhante à função esférica, entretanto observando o intervalo $[-10, 10]^2$ é possível notar que existem muitos mínimos locais na superfície da função de Griewangk dificultando a sua otimização.

Tabela 5.4: Resultados da otimização da função Griewangk

| Abordagem | Média±Desv. pad. | Melhor Resultado |
|---|---|---|
| AG-autoadap(inteiro) | $0.42341 \pm 0.34126$ | 0.00000 |
| AG-fix(inteiro) | $1.26349 \pm 0.56393$ | 0.23386 |
| AG-autoadap(real) | $7.51153 \pm 5.95152$ | 1.12415 |
| AG-fix(real) | $50.1351 \pm 23.9311$ | 3.66846 |
| Herrera e Lozano (2001) | $\mathbf{0.00000 \pm 0.00000}$ | 0.00000 |

Novamente, o algoritmo proposto por Herrera e Lozano apresenta um desempenho claramente superior às abordagens propostas. Dentre as abordagens propostas, o algoritmo genético com codificação inteira e auto-adaptação apresentou um desempenho relativo superior.

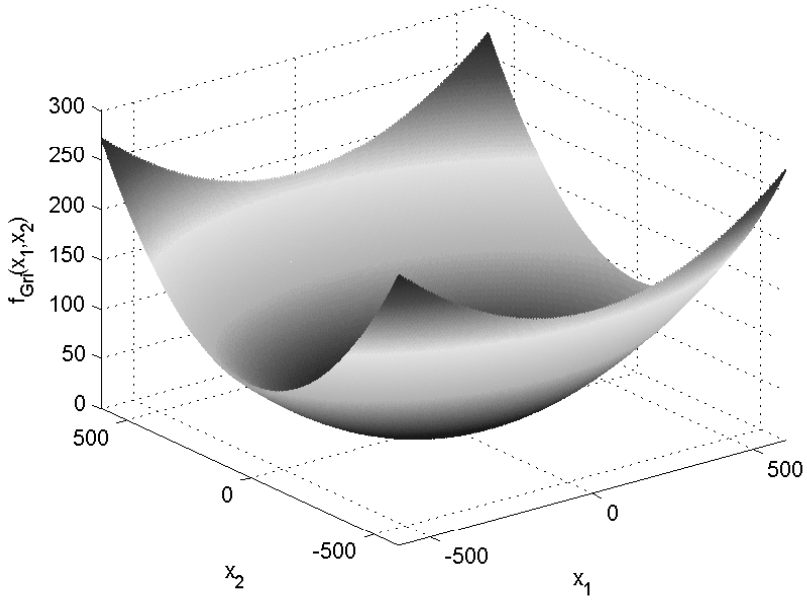
**Função de Rastrigin**

Considere o seguinte problema:

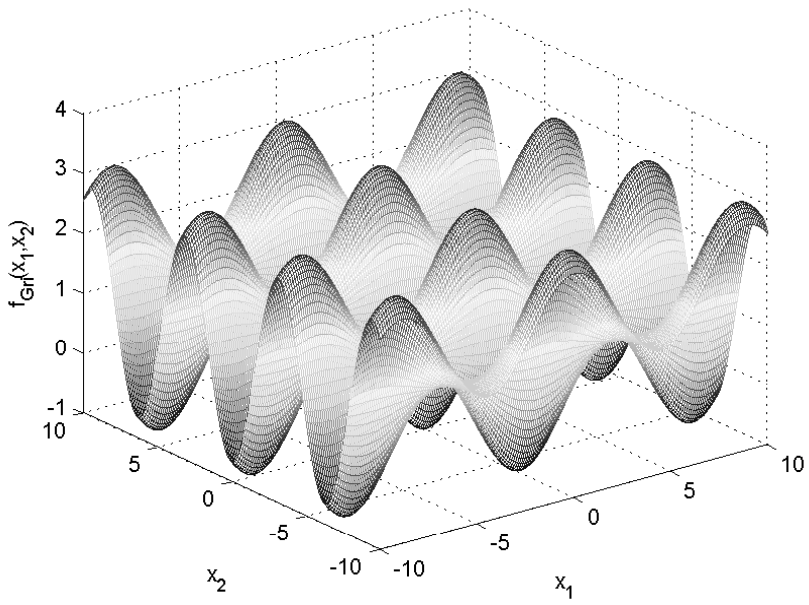
$$\min f_{Ras}(\mathbf{x}) = 250 + \sum_{i=1}^{25} \Big( x[i]^2 - 10\cos(2\pi x[i]) \Big) \tag{5.8}$$

onde:

- $x[i]$ representa o elemento de índice $i$ no vetor de entrada $\mathbf{x}$ tal que $x[i] \in \Omega_i \mid -5.12 < x[i] < 5.12$, para $1 \leq i \leq 25$;




(a) Gráfico da função Griewangk




(b) Gráfico da função Griewangk(detalhado)

Figura 5.4: Gráfico da função Griewangk para duas dimensões

- $f_{Ras}(\mathbf{x})$ é uma função contínua, multimodal e separável;
- $\hat{\mathbf{x}} = [0,0,\ldots,0]$;

- $f(\hat{\mathbf{x}}) = 0$.

A busca do ótimo da função de Rastrigin é um problema difícil pois trata-se de uma função multimodal em que há uma grande tendência de convergência para mínimos locais. Essa função é produzida a partir da equação 5.4 modulando-a com $a.\cos(\omega x[i])$ (HERRERA; LOZANO, 2001). Assim como a otimização das funções esférica e Rosenbrock, o tamanho do espaço de busca da função de Rastrigin é $2^{250} = 1.8093 \times 10^{75}$.

A função Rastrigin para duas dimensões pode ser observada na figura 5.5. Os resultados da minimização da função Rastrigin para 25 dimensões estão na tabela 5.5

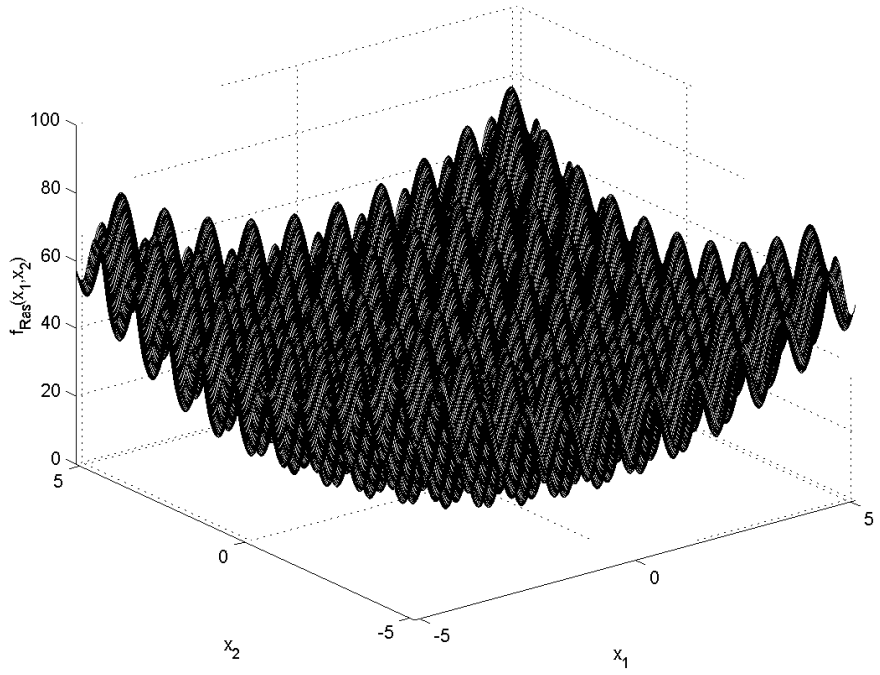



Figura 5.5: Gráfico da função Rastrigin para duas dimensões

Tabela 5.5: Resultados da otimização da função Rastrigin

| Abordagem | Média±Desv. pad. | Melhor Resultado |
|---|---|---|
| AG-autoadap(inteiro) | $\mathbf{0.00006 \pm 0.00004}$ | 0.000016 |
| AG-fix(inteiro) | $0.56234 \pm 0.03546$ | 0.002344 |
| AG-autoadap(real) | $45.5922 \pm 5.12333$ | 10.12343 |
| AG-fix(real) | $104.324 \pm 10.2345$ | 25.01445 |
| Herrera e Lozano (2001) | $0.0332 \pm 1.79 \times 10^{-1}$ | 0.000000 |

Ao contrário das outras funções de otimização, na função de Rastrigin o AG-autoadap obteve um desempenho superior ao do trabalho de Herrera e Lozano.

**Comentários gerais**

Em todas as funções de otimização utilizadas o melhor resultado entre as abordagens propostas foi obtido com uma combinação de codificação inteira e auto-adaptação. O desempenho da codificação inteira é justificável pelo tamanho do espaço de busca, que é infinito para codificação real.

### 5.2.2 Problema da Mochila

O problema da mochila é um problema de otimização combinatorial. Seu nome é derivado do problema de escolher o máximo de itens valiosos sem exceder os limites geométricos de uma mochila (ou o peso máximo permitido) que serão carregados em uma viagem. Um problema similar aparece freqüentemente na área administrativa, teoria da complexidade, criptografia e matemática aplicada (HOROWITZ; SAHNI, 1974; PISINGER, 2005).

Uma variação razoavelmente conhecida desse problema é o problema da mochila $0/1$ em que a quantidade de itens em cada mochila deve ser zero ou um. Outra restrição usualmente utilizada é que um item presente em uma mochila deverá estar em todas as mochilas. É importante ressaltar que mesmo utilizando essas restrições, a complexidade computacional do problema ainda é não-polinomial (GAREY; JOHNSON, 1979).

Este problema pode ser formulado do seguinte modo:

$$\max \sum_{i=1}^{\Upsilon} \sum_{j=1}^{M} K[i][j] \times Q[i][j] \tag{5.9}$$

onde $Q[i][j] \in \mathbb{N}$ sujeito a:

$$\sum_{i=1}^{\Upsilon} Q[i][j] \times W[i][j][d] \leq c[j][d], \quad j = 1 \ldots M, d = 1` \ldots D \tag{5.10}$$

onde,

- $M$ é o número de mochilas do problema;
- $D$ é o número de dimensões consideradas nas restrições do problema;
- $\Upsilon$ é o número de itens;
- $K[i][j]$ indica o lucro obtido ao carregar o objeto $i$ na mochila $j$. Em geral $K[i][j] = K[i][k]$, $\forall j,k \in \{1,\ldots,M\}$, ou seja, o lucro independe da mochila alocada;

- $Q[i][j]$ indica o número de itens $i$ alocados na mochila $j$. Para o problema da mochila 0/1 $Q[i][j] \in 0,1$;
- $W[i][j][d]$ indica a $d$-ésima dimensão (peso, volume, etc) do objeto $i$ na mochila $j$. Em geral $W[i][j][d] = W[i][k][d]$, $\forall j,k \in \{1,\ldots,M\}$, ou seja, a restrição considerada não depende da mochila utilizada;
- $c[j][d]$ indica a capacidade associada à dimensão $d$ da mochila $j$.
- $R = M \times D$ indica o número total de restrições do problema.

É importante notar que $D \neq M$, ou seja, as restrições podem aplicar-se a várias dimensões no mesmo problema (podem existir restrições de volume, comprimento, largura, altura, peso, etc) e, principalmente, que a solução deve conter somente quantidades inteiras de objetos.

O problema da mochila é usualmente resolvido utilizando técnicas de programação matemática (KOZANIDIS; MELACHRINOUDIS, 2004), apesar de não existir nenhum algoritmo com tempo polinomial para solucionar a versão genérica deste problema. O fato da solução conter somente números inteiros exige que algoritmos de programação inteira, como o *branch-and-bound*, sejam utilizados para obter-se uma solução factível.

O Instituto Zuze Berlim mantém um repositório de instâncias do problema da mochila (SKOROBOHATYJ, 2002). As instâncias utilizadas nesse trabalho foram flei, pb6, sent01, weish18, weish26, hp2, pet6, weing7 e weish22. A tabela 5.6 mostra as características dos problemas escolhidos.

Tabela 5.6: Instâncias do problema da mochila e suas características (SKOROBOHATYJ, 2002)

| Instância | itens | mochilas | solução ótima |
|---|---|---|---|
| weing7 | 105 | 2 | 1 095 445 |
| pb6 | 40 | 30 | 776 |
| pet6 | 39 | 5 | 10 618 |
| weish18 | 70 | 5 | 9 580 |
| weish22 | 80 | 5 | 8 947 |
| weish26 | 90 | 5 | 9 584 |
| hp2 | 35 | 4 | 3 186 |
| sent01 | 60 | 30 | 7 772 |

Para a garantia das restrições, utilizou-se a mesma forma de penalização do trabalho de Kimbrough *et al.* (2002). Desta maneira é possível comparar o desempenho da abordagem auto-adaptativa utilizando a mesma função de avaliação, ou seja, a mesma medida de qualidade da solução.

A tabela 5.7 compara o desempenho da abordagem auto-adaptativa com a abordagem baseada em parâmetros fixos para as instâncias da tabela 5.6. Essas instâncias foram avaliadas considerando 500 rodadas (diferentes sementes aleatórias foram geradas utilizando um gerador de números aleatórios (MATSUMOTO; NISHIMURA, 1998)) e foi realizada uma normalização onde o valor 1 representa a solução ótima para cada problema.

Tabela 5.7: Resultados do problema de otimização da mochila 0/1

| Instância | Abordagem | Média±desv. pad. | Melhor | Teste t |
|---|---|---|---|---|
| weing7 | AG-autoadap | $\mathbf{0.9997 \pm 0.0004}$ | 1.0000 | $2.4 \times 10^{-226}$ |
| | AG-fix | $0.9722 \pm 0.0104$ | 0.9926 | - |
| pb6 | AG-autoadap | $\mathbf{0.9858 \pm 0.0256}$ | 1.0000 | $1.5 \times 10^{-232}$ |
| | AG-fix | $0.9087 \pm 0.0749$ | 1.0000 | - |
| pet6 | AG-autoadap | $\mathbf{0.9941 \pm 0.0063}$ | 1.0000 | $1.1 \times 10^{-166}$ |
| | AG-fix | $0.9452 \pm 0.0264$ | 0.9932 | - |
| weish18 | AG-autoadap | $\mathbf{0.9994 \pm 0.0010}$ | 1.0000 | $1.2 \times 10^{-205}$ |
| | AG-fix | $0.9096 \pm 0.0382$ | 0.9889 | - |
| weish22 | AG-autoadap | $\mathbf{0.9984 \pm 0.0018}$ | 1.0000 | $1.1 \times 10^{-242}$ |
| | AG-fix | $0.8815 \pm 0.0409$ | 0.9818 | - |
| weish26 | AG-autoadap | $\mathbf{0.9978 \pm 0.0018}$ | 1.0000 | $7.7 \times 10^{-263}$ |
| | AG-fix | $0.8745 \pm 0.0039$ | 0.9851 | - |
| flei | AG-autoadap | $\mathbf{0.9828 \pm 0.0113}$ | 1.0000 | $1.6 \times 10^{-176}$ |
| | AG-fix | $0.9396 \pm 0.0226$ | 0.9920 | - |
| hp2 | AG-autoadap | $\mathbf{0.9854 \pm 0.0120}$ | 1.0000 | $4.8 \times 10^{-216}$ |
| | AG-fix | $0.9350 \pm 0.0221$ | 1.0000 | - |
| sent01 | AG-autoadap | $\mathbf{0.9985 \pm 0.0021}$ | 1.0000 | $3.7 \times 10^{-237}$ |
| | AG-fix | $0.6881 \pm 0.1118$ | 0.9634 | - |

Em todos os casos o AG-autoadap teve desempenho melhor que o de um sGA com parâmetros fixos com confiabilidade maior que 99% no teste t-student (p-value representado na última coluna da tabela 5.7). Para todos os casos o AG-autoadap foi capaz de encontrar a solução ótima em pelo menos uma rodada.

Os gráficos de evolução do algoritmo genético proposto e do algoritmo genético com parâmetros fixos do problema weing7 podem ser vistos na figura 5.6. O número de gerações foi limitado para 500 pois não houve evolução significativa após essas gerações.

O AG-autoadap apresenta uma maior diversidade genética ao longo das gerações permitindo uma busca eficaz.

Para avaliar a contribuição de cada componente que sofre a ação da auto-adaptação, uma série de rodadas do algoritmo com apenas um componente sofrendo auto-adaptação foi realizada. Os resultados dessa simulação estão na tabela 5.8.

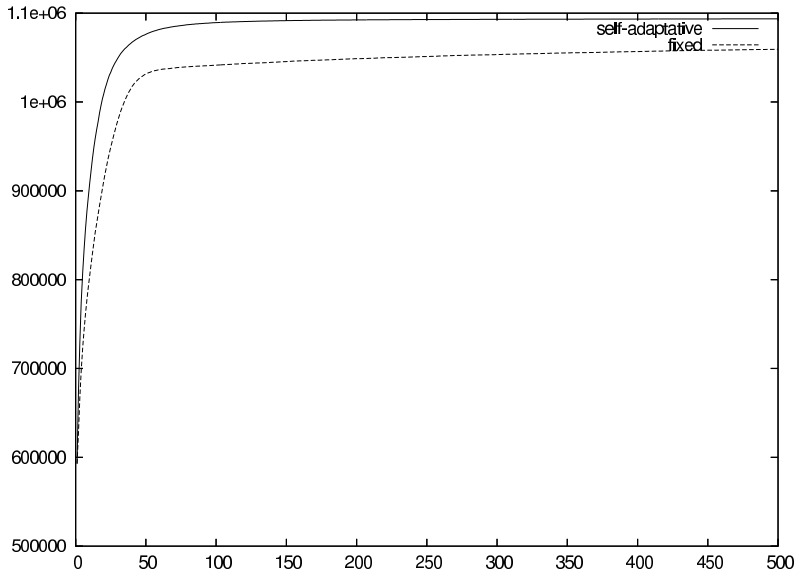



Figura 5.6: Evolução para a Instância WEING7 do problema da mochila para uma rodada aleatória do algoritmo (aptidão do melhor indivíduo média e média das médias da população)

Tabela 5.8: Resultados detalhados para diferentes níveis de autonomia

| Instância | Adaptação | Méd± Desv. pad. | Melhor | t-Student |
|---|---|---|---|---|
| | AG-autoadap | $0.9997 \pm 0.0004$ | 1.0000 | $2.4 \times 10^{-226}$ |
| | mutação | $0.9989 \pm 0.0036$ | 1.0000 | $6.6 \times 10^{-236}$ |
| | recombinação | $0.9921 \pm 0.0036$ | 0.9997 | $4.7 \times 10^{-174}$ |
| Weing7 | torneio | $0.9859 \pm 0.0036$ | 1.0000 | $2.0 \times 10^{-99}$ |
| | param fixos | $0.9722 \pm 0.0105$ | 0.9926 | – |
| | Kimbrough | $0.9993 \pm 0.0004$ | – | – |

Estes resultados evidenciam os benefícios da auto-adaptação, em especial da mutação, quando comparados a outros parâmetros evolutivos. Os resultados obtidos pela auto-adaptação da mutação foram semelhantes aos descritos no trabalho de Kimbrough *et al.* (2002). Neste caso a acuidade foi acompanhada pela autonomia, uma vez que o sistema mais acurado foi também aquele com menor número de parâmetros determinados pelo usuário.

## 5.3 Capablanca: sistema genético-nebuloso co-evolutivo com auto-adaptação de parâmetros

Diferente das seções anteriores, nesta seção a auto-adaptação foi aplicada no ajuste dos parâmetros de um sistema nebuloso. O sistema proposto Capablanca será confrontado com abordagens baseadas em GFS com parâmetros evolutivos fixos (fix-Capablanca) , modelos ba-

seados em redes neurais artificiais (RBF e PERCEPTRON) e modelos baseados em redes neuro-nebulosas (ANFIS). Em todos os casos o problema considerado será a aproximação de funções (com e sem ruído).

Para avaliar a qualidade da aproximação de funções através de um sistema nebuloso auto-adaptativo algumas considerações devem ser realizadas:

**Critérios de visibilidade** Na abordagem proposta, visando melhorar a interpretabilidade das partições nebulosas, os valores para os critérios de visibilidade ($\gamma$-completude e $\kappa$-sobre-posição) foram estabelecidos como $\gamma = 0.02$ e $\kappa = 0.95$.

**Granularidade máxima** A granularidade máxima do sistema (GranulMax) é determinada pelo número de termos lingüísticos do universo de discurso de cada variável de entrada ($x_1$,$x_2$, $\ldots$,$x_d$) definida como:

$$GranulMax = [G_1, G_2, \ldots, G_d],$$

onde $G_k$, $k = 1,\ldots,d$ representa o número máximo de funções de pertinência para cada universo de discurso. Este parâmetro deve ser especificado pelo usuário e indica um compromisso entre a acuidade e a interpretabilidade do sistema. O sistema é capaz de eliminar funções de pertinência gerando regras mais simples e partições mais compactas.

**Fixação da população de bases de regras** No problema de otimização de sistemas nebulosos o número de antecedentes de regras nebulosas possíveis (contendo tanto as regras específicas quanto as genéricas) é definido pela equação 5.11.

$$n_{regras} = \left( \prod_{i=1}^{d} G_i + 1 \right) - 1 \tag{5.11}$$

Este número cresce exponencialmente com o número de dimensões da entrada do sistema. Para um número alto de dimensões, o tamanho da população de regras individuais capaz de conter todas as possíveis combinações de antecedentes das regras torna-se inviável. Nesses casos é necessário estabelecer um tamanho máximo para essa população e seus cromossomos devem passar por um processo evolutivo convencional. Nos problemas tratados nessa seção o número de variáveis de entrada dos sistemas nebulosos testados é bastante reduzido. Para esse tipo de problemas é possível gerar uma população que contenha todos os valores possíveis de combinações de antecedentes. Essa população passa por um processo evolutivo particular: seus indivíduos possuem taxas de recombinação e mutação zero e o mecanismo de seleção garante que os cromossomos dessa população serão sempre preservados e reproduzidos nas próximas gerações em posições fixas;

**Inicialização das partições nebulosas** A geração inicial das funções de pertinência deve fornecer somente soluções válidas. Para garantir essa premissa, na abordagem proposta, a população de nível I é inicializada com partições uniformes.

- No ANFIS o formato das funções de pertinência devem ser especificados pelo usuário. É possível definir o formato de cada função de pertinência individualmente.
- No Capablanca e fix-Capablanca os tipos de funções de pertinência podem ser triangular, trapezoidal ou Gaussiana e são selecionados automaticamente através do processo evolutivo.

**Base de regras** Ao contrário do ANFIS que mantém uma base de regras nebulosas estática, a base de regras nebulosas das abordagens propostas sofre um processo de evolução ao longo da rodada do AG.

- No sistema ANFIS a base de regras é construída utilizando todas as combinações possíveis de antecedentes que contenham termos lingüísticos (base de regras específica). Essa abordagem possui o inconveniente de gerar uma base de regras composta por muitas regras quando o número de variáveis de entrada é elevado. Outra desvantagem é a impossibilidade de reduzir o tamanho da base de regras, que é a mais complexa possível utilizando somente as regras específicas, isto é, sem o uso de termos do tipo *don't care*.
- A abordagem proposta não utiliza nenhum artifício para limitar o tamanho da base de regras. Entretanto, permite o uso de regras genéricas (regras com *don't care*).

**Classes de funções no conseqüente de Takagi-Sugeno** Embora seja, teoricamente, possível o uso de conseqüentes não-lineares no ANFIS, esta funcionalidade não está presente na versão disponível.

- ANFIS: utiliza somente funções lineares no conseqüente das regras;
- Capablanca/fix-Capablanca: utiliza funções quadráticas e não lineares no conseqüente;

**Conjunto de treinamento e teste** Os sistemas avaliados utilizaram 225 amostras de treinamento e 2500 amostras de teste geradas aleatoriamente e distribuídas uniformemente. As amostras de treinamento foram utilizadas durante a fase de treinamento dos algoritmos e as amostras de teste durante a avaliação das soluções.

### 5.3.1 Aproximação de funções sem ruído

Esta subseção apresentará dois problemas de aproximação de funções para avaliar a abordagem proposta em termos de acuidade e interpretabilidade. Para isso utiliza-se um conjunto conhecido de funções em 2 dimensões que são combinações de funções unidimensionais descritas em (MITAIM; KOSKO, 2001).

**Função $g_1$:**

A função $g_1$ pode ser descrita como:

$$g_1 : \mathbb{R}^2 \mapsto \mathbb{R} \tag{5.12}$$

onde:

$$g_1(x_1, x_2) = 3x_1(x_1 - 1)(x_1 - 1.9)(x_1 + 0.7)(x_1 + 1.8).\sin(x_2) \tag{5.13}$$

Este mapeamento é produzido a partir de um produto de uma função polinomial com uma função trigonométrica (MITAIM; KOSKO, 2001). Para avaliar o desempenho da proposta neste mapeamento, os dados de treinamento e teste foram gerados aleatoriamente distribuídos uniformemente na região compacta $[-2, 2]^2$.

O gráfico de $g_1$ pode ser observado na figura 5.7 e os resultados da simulação podem ser vistos na tabela 5.9.

Tabela 5.9: Resultados para $g_1$:

| Abordagem | $\overline{MSE}$ | Desvio Padrão | MSE (melhor) | T-test |
|---|---|---|---|---|
| Capablanca | **0.0698** | 0.0202 | **0.0457** | $4 \times 10^{-7}$ |
| fix-Capablanca | 0.5328 | 0.6732 | 0.3201 | |
| MLP | 0.2346 | 0.1182 | 0.1215 | - |
| ANFIS | - | - | 0.0477 | - |
| RBF | - | - | 0.0721 | - |

O melhor resultado foi encontrado utilizando o sistema Capablanca. Este resultado apresenta uma boa confiabilidade de acordo com o teste estatístico t de student.

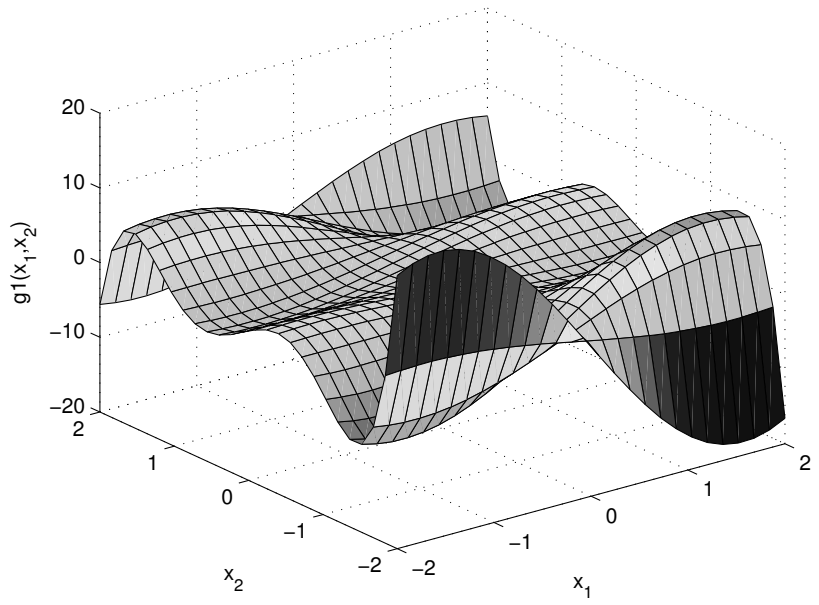



Figura 5.7: Função $g_1$

**Função $g_2$:**

A função $g_2$ pode ser descrita como:

$$g_2 : \mathbb{R}^2 \mapsto \mathbb{R} \tag{5.14}$$

onde:

$$g2(x_1, x_2) = 10\frac{\sin\left(10x_1^2 + 5x_2^2 - 6x_2\right)}{10x_1^2 + 5x_2^2 - 6x_2} \tag{5.15}$$

Este mapeamento é produzido a partir de um polinômio simples e um com base em funções trigonométricas (MITAIM; KOSKO, 2001).

O gráfico de $g_2$ pode ser observado na figura 5.8 e os resultados da simulação podem ser vistos na tabela 5.10.

Novamente, o melhor resultado foi encontrado utilizando o sistema Capablanca. Este resultado apresenta uma boa confiabilidade de acordo com o teste estatístico t de student.

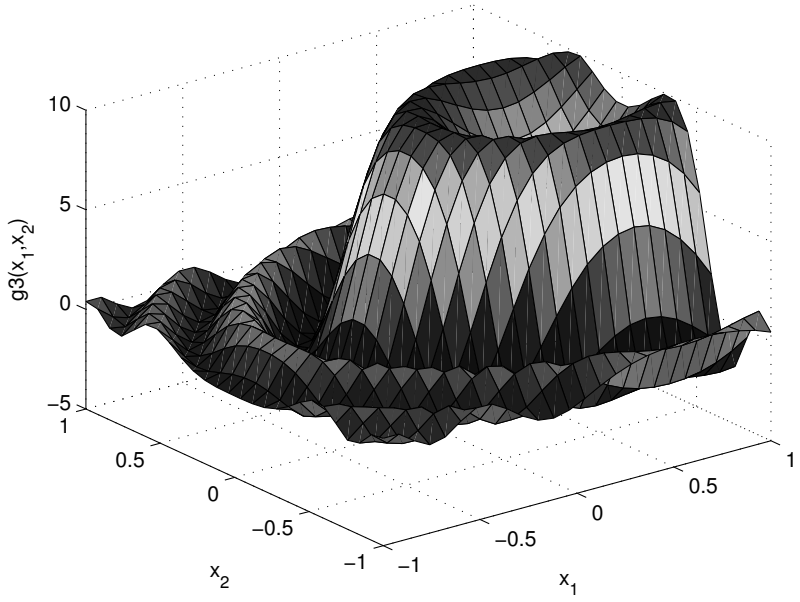



Figura 5.8: Função $g_2$

Tabela 5.10: Resultados para $g_2$:

| Abordagem | $\overline{MSE}$ | Desvio Padrão | MSE (melhor) | T-test |
|---|---|---|---|---|
| Capablanca | **0.3417** | 0.0832 | **0.3253** | 1E-5 |
| fix-Capablanca | 0.7605 | 0.0799 | 0.6865 | - |
| MLP | 0.6374 | 0.4428 | 0.5233 | - |
| ANFIS | - | - | 0.5234 | - |
| RBF | - | - | 0.3914 | - |

### 5.3.2 Aproximação de funções com ruído

Conforme Berthold (1999), o uso de modelos muito flexíveis, como o modelo proposto, estão mais sujeitos ao sobre-ajuste (*overfitting*) podendo incorporar as peculiaridades causadas por perturbações aleatórias. Portanto, nessa subseção utilizam-se funções com dados de treinamento perturbados por um ruído com distribuição normal $\mathcal{N}(\omega,\sigma)$, para mensurar o grau de sobre-ajuste.

**Função $g_3$:**

A função $g_3$ pode ser descrita como:

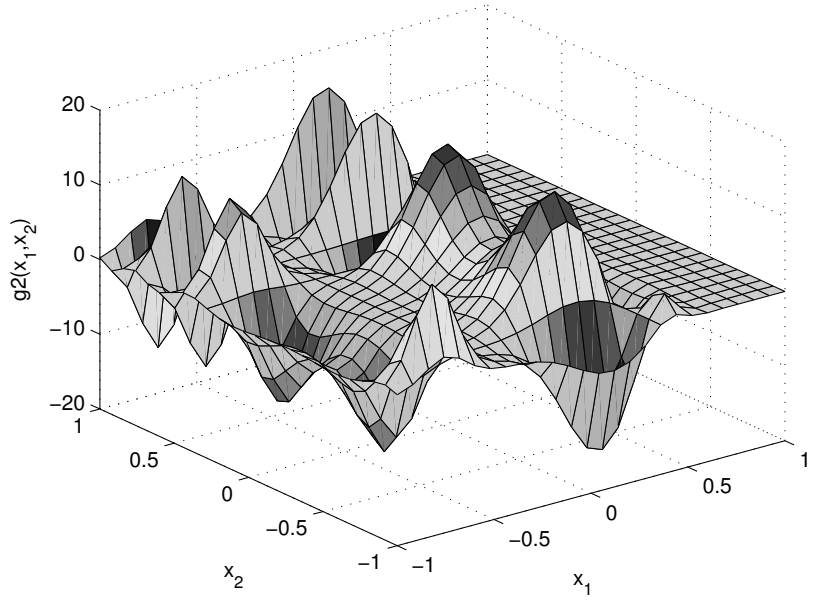



Figura 5.9: Função $g_3$

$$g_3 : \mathbb{R}^2 \mapsto \mathbb{R} \tag{5.16}$$

onde:

$$g3(x_1, x_2) = \frac{8}{5}\left(10e^{-\left(\frac{x_2-0.1}{0.25}\right)^2} - 8e^{-\left(\frac{x+0.75}{0.15}\right)^2} - 4e^{-\left(\frac{x-0.4}{0.1}\right)^2}\right).\sin(10x_1^2 + 5x_1 + 1) \tag{5.17}$$

Este mapeamento é produzido a partir de um produto de funções exponenciais e um polinômio baseado em funções trigonométricas (MITAIM; KOSKO, 2001). Os dados de treinamento e teste foram gerados na região compacta $[-1,1]^2$. Para avaliar o desempenho da proposta nesse mapeamento, os dados de treinamento e teste foram gerados aleatoriamente distribuídos uniformemente na região compacta $[-1,1]^2$.

O gráfico de $g_3$ pode ser observado na figura 5.9 e os resultados da simulação podem ser vistos na tabela 5.11.

O melhor resultado foi encontrado utilizando o sistema Capablanca. Este resultado apresenta uma boa confiabilidade de acordo com o teste estatístico t de student.

A partição nebulosa resultante da melhor aproximação de $g_2$ pelo sistema Capablanca pode

Tabela 5.11: Resultados para $g_3$:

| Abordagem | $\overline{MSE}$ | Desvio Padrão | MSE (melhor) | T-test |
|---|---|---|---|---|
| Capablanca | **2.6523** | 0.5872 | **1.4493** | 0.04 |
| fix-Capablanca | 2.7513 | 0.4119 | 1.9858 | |
| MLP | 5.0231 | 1.6601 | 4.0155 | - |
| ANFIS | - | - | 7.4030 | - |
| RBF | - | - | 7.9018 | - |

ser vista na figura 5.10 e a base de regras correspondente pode ser observada na tabela 5.12.

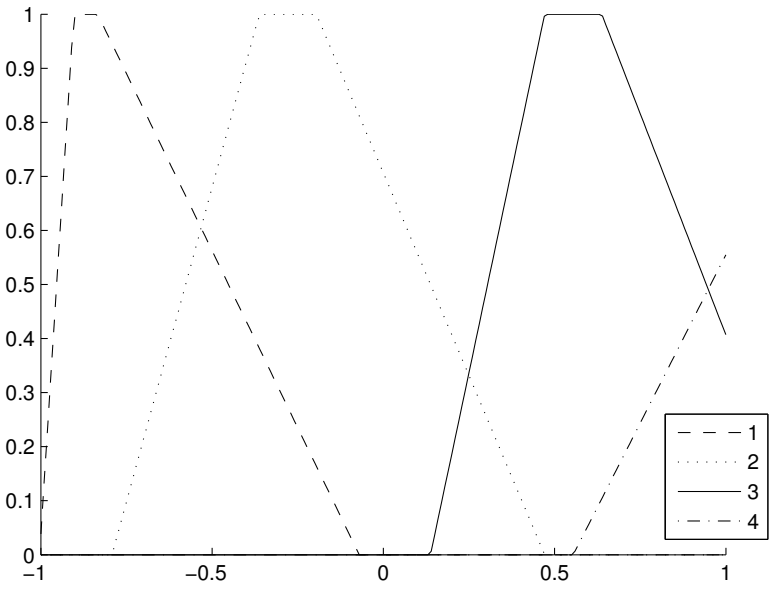
Tabela 5.12: Base de regras para $g_3$

| Antecedente | | |
|---|---|---|
| se $x_1$ é "1" | | |
| se $x_1$ é "2" | | |
| se $x_1$ é "3" | | |
| se $x_1$ é "4" | | |
| se $x_2$ é "1" | | |
| se $x_1$ é "1" | $t_4(2.71)$ | $x_2$ é "1" |
| se $x_1$ é "2" | $t_4(2.71)$ | $x_2$ é "1" |
| se $x_1$ é "3" | $t_4(2.71)$ | $x_2$ é "1" |
| se $x_2$ é "2" | | |
| se $x_1$ é "1" | $t_4(2.71)$ | $x_2$ é "2" |
| se $x_1$ é "2" | $t_4(2.71)$ | $x_2$ é "2" |
| se $x_1$ é "3" | $t_4(2.71)$ | $x_2$ é "2" |
| se $x_1$ é "4" | $t_4(2.71)$ | $x_2$ é "2" |
| se $x_1$ é "2" | $t_4(2.71)$ | $x_2$ é "3" |
| se $x_2$ é "4" | | |
| se $x_1$ é "2" | $t_4(2.71)$ | $x_2$ é "4" |
| se $x_1$ é "4" | $t_4(2.71)$ | $x_2$ é "4" |
| se $x_1$ é "1" | $t_4(2.71)$ | $x_2$ é "4" |

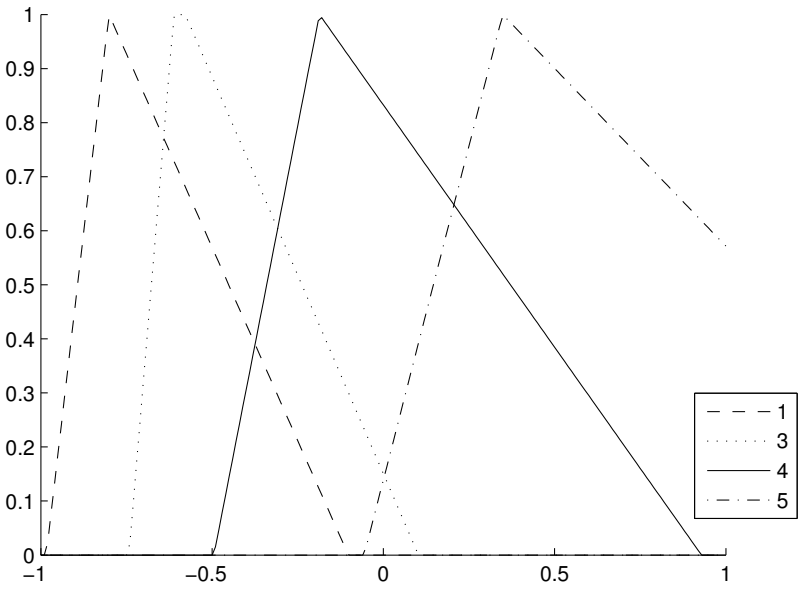
Neste caso, observa-se que a interpretabilidade do sistema é preservada. A tabela 5.12 mostra uma base de regras consistente e compacta formada por 19 regras nebulosas dentre as 35 possíveis (entre elas 7 regras genéricas, com antecedentes do tipo *don't care*)

Os gráficos dos parâmetros evolutivos ao longo da evolução podem ser observados nas figuras 5.11 . Observa-se que a evolução dos parâmetros é diferente para cada população que sofre co-evolução. Em todos os casos entretanto, há uma oscilação nas gerações iniciais e depois os parâmetros se estabilizam rapidamente. No caso da população do nível I, esta estabilização ocorre mais cedo.

Pode-se observar que os parâmetros evolutivos dos melhores indivíduos em níveis hierárquicos distintos são significativamente diferentes. Portanto, há uma evidência de que cada nível



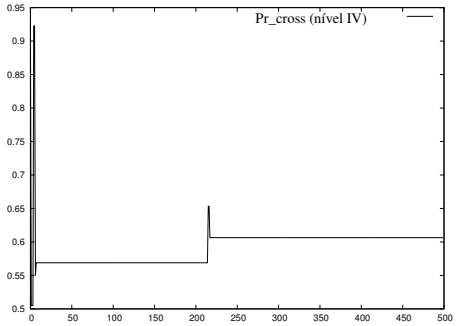

(a) partição nebulosa do universo $\mathbf{X}_1$



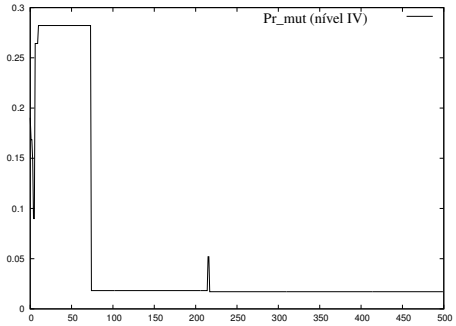

(b) partição nebulosa do universo $\mathbf{X}_2$

Figura 5.10: Partições nebulosas para $g_3$

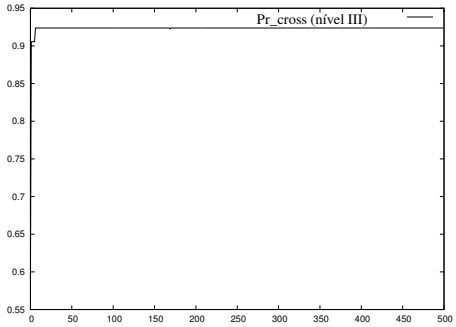
da estrutura hierárquica apresenta um conjunto de bons parâmetros evolutivos diferentes. Devido à complexidade da busca desses parâmetros uma abordagem baseada em auto-adaptação, além de apresentar melhores resultados para as funções de teste utilizadas, apresenta a vantagem



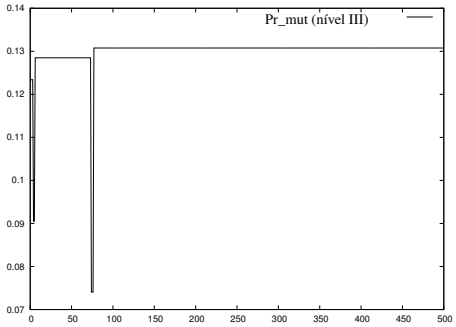

(a) Taxa de recombinação (sistemas nebulosos)



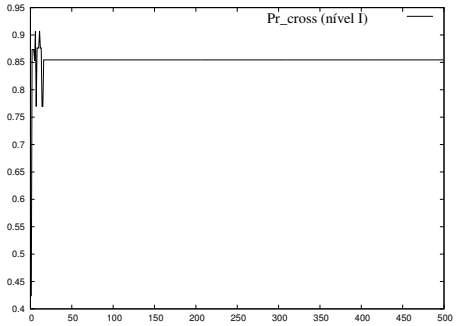

(b) Taxa de mutação (sistemas nebulosos)



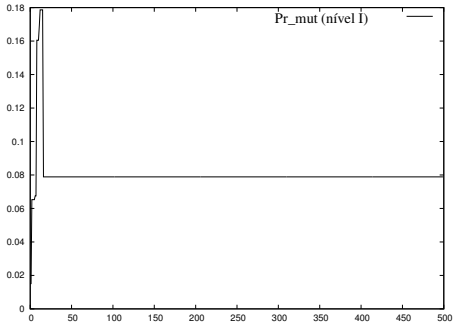

(c) Taxa de recombinação (base de regras)




(d) Taxa de mutação(base de regras)




(e) Taxa de recombinação (partições nebulosas)




(f) Taxa de mutação(partições nebulosas)

Figura 5.11: Gráficos dos parâmetros estratégicos de uma rodada da aproximação da função $g_3$

de liberar o usuário do projeto desses parâmetros melhorando significativamente a autonomia do sistema.

# 6 *Conclusões e trabalhos futuros*

## 6.1 Conclusões

Este trabalho apresentou a fundamentação teórica necessária para o uso de técnicas de auto-adaptação de parâmetros evolutivos em sistemas genético-nebulosos. Aqui a auto-adaptação foi inicialmente aplicada em um algoritmo genético padrão (sGA) utilizado para solucionar problemas de otimização. Na seqüência, o mecanismo de auto-adaptação foi utilizado no algoritmo genético-nebuloso co-evolutivo inspirado no modelo proposto por Delgado, von Zuben e Gomide (2004), com ganhos significativos em autonomia de projeto e desempenho do sistema, sem perdas na interpretabilidade do modelo. Os resultados permitem concluir que o uso da auto-adaptação em sistemas genético-nebulosos representa não só uma alternativa poderosa para liberar o usuário da tarefa de definição da grande maioria dos parâmetros associados, como traz ganhos em termos de desempenho, quando os sistemas auto-adaptativos são comparados às abordagens com parâmetros evolutivos fixos. Entretanto, cabe salientar que muitos outros problemas podem se beneficiar da metodologia de auto-adaptação proposta, uma vez que esta não requer grandes modificações nos algoritmos genéticos já existentes.

As principais contribuições desta dissertação foram:

**Fundamentação teórica do funcionamento da auto-adaptação** Apesar do uso de auto-adaptação de parâmetros em um número razoável de trabalhos, poucos autores se preocupam em explicar o funcionamento desse mecanismo de uma maneira formal. A principal dificuldade vem do fato dos parâmetros evolutivos não afetarem diretamente a aptidão dos indivíduos. Desta maneira, o mecanismo de seleção não seria, teoricamente, capaz de avaliar a qualidade dos parâmetros evolutivos. Entretanto o teorema dos esquemas mostra que o número de indivíduos que combinam com padrões de similaridade com desempenho melhor que a média apresentam tendência a aumentar. Este teorema não implica que os padrões de similaridade devam afetar diretamente o cálculo da aptidão do indivíduo. Supondo que um conjunto de parâmetros evolutivos apresentem um bom desempenho, eles irão gerar indivíduos com aptidão mais alta que outros parâmetros. Logo, esses in-

divíduos têm aptidão acima da média e os padrões de similaridade associados a esses indivíduos tendem a apresentar crescimento nas gerações seguintes. E um dos padrões associados a esses indivíduos contém justamente os parâmetros evolutivos que geraram indivíduos mais aptos. Desta maneira o mecanismo de seleção, ainda que os parâmetros evolutivos não contribuam diretamente para o cálculo da aptidão, se encarrega da seleção dos melhores parâmetros. As principais conseqüências dessa observação são:

- Bons conjuntos de parâmetros evolutivos tendem a gerar mais descendentes, logo têm uma maior chance de sobrevivência;
- O mecanismo de seleção se encarrega de eliminar parâmetros com baixo desempenho da população;
- Artigos recentes (EIBEN; SCHUT; WILDE, 1998) apontam para uma diminuição do custo computacional do algoritmo genético utilizando auto-adaptação em seu mecanismo de seleção.

E algumas outras características interessantes podem ser destacadas:

- O uso de auto-adaptação não exige conhecimento adicional do problema;
- A abordagem auto-adaptativa é flexível e pode, teoricamente, ser aplicada a qualquer problema formulado como um algoritmo genético;
- Os parâmetros são definidos automaticamente, sem intervenção do usuário;
- Outras heurísticas podem se beneficiar do mecanismo de auto-adaptação.

**Implementação do algoritmo genético auto-adaptativo** Um protótipo do algoritmo genético com auto-adaptação de múltiplos parâmetros foi construído para teste. Este algoritmo, denominado AG-autoadap, trabalhou com três tipos de codificação: real, inteira e binária. Seu desempenho foi comparado com uma abordagem baseada em controle adaptativo nebuloso proposta por Herrera e Lozano (2001) e um algoritmo genético utilizando o conjunto de parâmetros fixos proposto por de Jong (1975). Como esperado, o controlador nebuloso proposto por Herrera, projetado especificamente para a aproximação das funções de teste, obteve melhor desempenho para a maioria das funções contínuas. Para um problema combinatorial, entretanto, o algoritmo proposto apresentou um desempenho mais interessante que o AG auto-adaptativo proposto por Kimbrough *et al.* (2002).

**Implementação do sistema Capablanca** O sistema co-evolutivo proposto por Delgado, von Zuben e Gomide (2004) tem apresentado bom desempenho em problemas de aproximação de funções. Entretanto, a determinação de seus parâmetros evolutivos se torna

um problema complexo. Um protótipo do coevol-GFS foi implementado, com algumas pequenas modificações, e associado ao algoritmo genético auto-adaptativo descrito anteriormente, sendo denominado de sistema Capablanca. O desempenho do sistema Capablanca foi comparado com abordagens tradicionais de aproximações de funções baseadas em Inteligência Computacional (MLP, RBF, ANFIS) e um GFS co-evolutivo utilizando os parâmetros recomendados por De Jong (fix-Capablanca), sendo que o sistema proposto apresentou desempenho superior em todos os casos testados. O sistema Capablanca apresentou ainda a vantagem de não depender de software proprietário (o coevol-GFS foi implementado para ser executado no ambiente MATLAB© (THE MATHWORKS, INC., 2004)).

## 6.2 Trabalhos futuros

**AG autônomo** Durante esse trabalho alguns parâmetros evolutivos ainda dependem da definição do usuário e permanecem fixos durante a evolução. Os principais parâmetros, que merecem um estudo mais detalhado, e que deverão ser incluídos no esquema de auto-adaptação são:

- Tamanho da população;
- Número de gerações.

O objetivo é obter um sistema genético-nebuloso completamente autônomo, no qual todos os parâmetros (tanto do sistema nebuloso quanto do algoritmo genético) sejam definidos de forma automática.

**AG auto-adaptativo multi-objetivo** Com algumas modificações, o protótipo construído pode ser adaptado para solução de problemas multi-objetivo. Por exemplo problemas de tomada de decisão empresarial onde é desejável a maximização do lucro e a minimização do risco;

**Mecanismos de busca local no Capablanca** Em situações em que exista conhecimento adicional do problema, o desempenho do sistema Capablanca pode ser melhorado pela adição de um mecanismo de busca local;

**Uso de agrupamento nebuloso** A implementação de um algoritmo de agrupamento para encontrar uma granularidade máxima e indicar a localização inicial das funções de pertinência;

**Uso de conseqüentes de Mamdani** que são considerados mais interpretáveis que os modelos TSK;

**Uso de outros métodos de condicionamento de saída** que diminuam o custo computacional da avaliação de soluções;

**Simulação para espaços de busca dinâmicos** não abordados neste trabalho;

**Paralelização através de hardware programável** visando a diminuição do tempo de processamento;

**Evolução on-line** permitindo a aplicação do AG proposto em aplicações com função de aptidão dinâmicas.

# *Referências Bibliográficas*


ADELI, H.; CHENG, N.-T. Augmented Lagrangian Genetic Algorithm for Structural Optimization. *Journal of Aerospace Engineering*, v. 7, n. 1, p. 104–118, January 1994.

ALCALÁ, R.; CASILLAS, J.; CORDÓN, O.; HERRERA, F.; ZWIR, S. J. I. Learning and tuning fuzzy rule-based systems for linguistic modeling and their applications. In: ____. *KNOWLEDGE-BASED SYSTEMS. Techniques and Applications*. : Academic Press, 2000. III, p. 889–941.

ANDERSON, E.; BAI, Z.; BISCHOF, C.; DEMMEL, J.; DONGARRA, J.; CROZ, J. D.; GREENBAUM, A.; HAMMARLING, S.; MCKENNEY, A.; OSTROUCHOV, S.; SORENSEN, D. *LAPACK's user's guide*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1992. ISBN 0-89871-294-7.

ANGELINE, P. J. Adaptive and self-adaptive evolutionary computation. In: M., P.; ATTIKIOUZEL, Y.; MARKS, R.; D., F.; FUKUDA, T. (Ed.). *Computational Intelligence, A Dynamic System Perspective*. : IEEE Press, 1995. p. 152–161.

BÄCK, T. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In: MäNNER, R.; MANDERICK, B. (Ed.). *2nd Parallel Problem Solving from Nature 2*. : Elsevier, 1992. p. 87–96.

BÄCK, T. Self-adaptation in genetic algorithms. In: *1st European Conference on Artifical Life: Toward a practice of autonomous systems*. 1992. p. 263–271.

BÄCK, T. Optimal mutation rates in genetic search. In: FORREST, S. (Ed.). *Proceedings of the 5th International Conference on Genetic Algorithms*. San Mateo, CA, USA: Morgan Kaufmann, 1993. p. 2–8. ISBN 1-55860-299-2. Disponível em: <citeseer.ist.psu.edu/back93optimal.html>.

BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. (Ed.). *Handbook of Evolutionary Computation*. Bristol, UK, UK: IOP Publishing Ltd., 1997. ISBN 0750303921.

BARDOSSY L. DUCKSTEIN, I. B. A. Fuzzy rule-based classification of atmospheric circulation patterns. *International Journal of Climatolology*, v. 15, p. 1087–1097, 1995.

BERTHOLD, M. *Intelligent Data Analysis: An Introduction*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1999. ISBN 3540658084.

BONISSONE, P.; KHEDKAR, P.; CHEN, Y. Genetic algorithms for automated tuning of fuzzy controllers: A a train handling application. In: *Fifth IEEE International conference on Fuzzy Systems*. 1996. p. 674–680.

CARSE, B.; FOGARTY, T. C.; MUNRO, A. Evolving fuzzy rule based controllers using genetic algorithms. *Fuzzy Sets and Systems*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 80, n. 3, p. 273–293, 1996. ISSN 0165-0114.

CASTRO, P. A. D.; CAMARGO, H. A. Learning and optimization of fuzzy rule base by means of self-adaptive genetic algorithm. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*. 2004. v. 2, p. 1037–1042. ISSN 1098-7584.

CHENG, S. H.; HIGHAM, N. J. Parallel implementation of a block algorithm for matrix 1-norm estimation. In: *Euro-Par '01: Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing*. London, UK: Springer-Verlag, 2001. p. 568–577. ISBN 3-540-42495-4.

CHEONG, F.; LAI, R. Constraining the optimization of a fuzzy logic controller using an enhanced genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, v. 30, n. 1, p. 31–46, 2000.

CHI, Z.; YAN, H.; PHAM, T. *Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition.* : World Scientific, 1996.

CHURCHLAND, P. S.; SEJNOWSKI, T. J. *The computational brain*. Cambridge, MA, USA: MIT Press, 1992. ISBN 0-262-03188-4.

COELLO, C. A. C. Self-adaptive penalties for GA-based optimization. In: ANGELINE, P. J.; MICHALEWICZ, Z.; SCHOENAUER, M.; YAO, X.; ZALZALA, A. (Ed.). *Proceedings of the Congress on Evolutionary Computation*. Mayflower Hotel, Washington D.C., USA: IEEE Press, 1999. v. 1, p. 573–580. ISBN 0-7803-5537-7 (Microfiche). Disponível em: <citeseer.ist.psu.edu/75427.html>.

COELLO, C. A. C. Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering and Environmental Systems*, v. 17, p. 319–346, 2000.

CORDÓN, O.; GOMIDE, F. A. C.; HERRERA, F.; HOFFMANN, F.; MAGDALENA, L. Genetic fuzzy systems. new developments. *Fuzzy Sets and Systems*, v. 141, n. 1, p. 1–3, 2004.

CORDÓN, O.; HERRERA, F.; HOFFMANN, F.; MAGDALENA, L. *Genetic Fuzzy Systems. Evolutionary tuning and learning of fuzzy knowledge bases*. : World Scientific, 2001.

CORDÓN, O.; HERRERA, F.; HOFFMANN, F.; MAGDALENA, L. *Industrial Applications of Fuzzy Control*. : World Scientific Publishing Co. Pte. Ltd, 2001. ISBN 981-02-4016-3.

DAVIDOR, Y. *Genetic Algorithms and Robotics : A Heuristic Strategy for Optimization*. : World Scientific Publishing Co., 1990.

DAVIDOR, Y. A genetic algorithm applied to robot trajectory generation. In: ____. *Handbook of Genetic Algorithms*. : Van Nostrand Reinhold, 1991. cap. 12, p. 144–165.

DAVIS, L. *Genetic Algorithms and Simulated Annealing*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987. ISBN 0934613443.

DAVIS, L. Adapting operator probabilities in genetic algorithms. In: *Proceedings of the 3rd International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989. p. 61–69. ISBN 1-55860-066-3.

DE CARVALHO E PAIVA, R. P. P. *Identificação neuro-nebulosa Aspectos de interpretabilidade*. Dissertação (Mestrado) — Universidade de Coimbra, 1999.

DE CASTRO, L. R.; TIMMIS, J. *Artificial Immune Systems: A New Computational Intelligence Paradigm*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002. ISBN 1852335947.

DE JONG, K. *The Analysis of the behaviour of a Class of Genetic Adaptive systems*. Tese (Doutorado) — Department of Computer Science, University of Michigan, 1975.

DE JONG, K.; SPEARS, W. M. An analysis of the interacting roles of population size and crossover in genetic algorithms. In: SCHWEFEL, H. P.; MÄNNER, R. (Ed.). *Parallel Problem Solving from Nature - Proceedings of 1st Workshop, PPSN 1*. Dortmund, Germany: Springer-Verlag, Berlin, Germany, 1991. v. 496, p. 38–47. Disponível em: <citeseer.ist.psu.edu/dejong91analysis.html>.

DE JONG, K.; SPEARS, W. M. A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, v. 5, n. 1, p. 1–26, April 1992.

DE SOUSA, M. A. T.; MADRID, M. K. Optimization of takagi-sugeno fuzzy controllers using a genetic algorithm. In: *The Ninth IEEE International Conference on Fuzzy Systems*. 2000. v. 1, p. 30–35.

DEB, K. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, v. 186, n. 2, p. 311–338, 2000.

DELGADO, M.; VON ZUBEN, F.; GOMIDE, F. Evolutionary design of takagisugeno fuzzy systems: a modular and hierarchical approach. In: *Proceedings of the 9th IEEE International Conference on Fuzzy Systems*. 2000.

DELGADO, M. R. *Projeto Automático de Sistemas Nebulosos: Uma Abordagem Co-Evolutiva*. Tese (Doutorado) — Universidade Estadual de Campinas, 2 2002.

DELGADO, M. R.; VON ZUBEN, F.; GOMIDE, F. Hierarchical genetic fuzzy systems. *Inf. Sci.*, Elsevier Science Inc., New York, NY, USA, v. 136, n. 1-4, p. 29–52, 2001. ISSN 0020-0255.

DELGADO, M. R.; VON ZUBEN, F.; GOMIDE, F. Coevolutionary genetic fuzzy systems: a hierarchical collaborative approach. *Fuzzy Sets and Systems*, v. 141, p. 89–106, 2004.

DUBOIS, D.; PRADE, H. *Fuzzy Sets and Systems: Theory and applications*. : Academic Press, 1980.

DUBOIS, D.; PRADE, H. Soft computing, fuzzy logic and artificial intelligence. *Soft Computing*, v. 2, n. 1, p. 7–11, 1998.

DYCKHOFF, H.; PEDRYCZ, W. Generalized mean as a model of compensative connectives. v. 14, p. 143–154, 1984.

EIBEN, A. E. Evolutionary computing: the most powerful problem solver in the universe? *Dutch Mathematical Archive*, Nederlands Archief voor Wiskunde, v. 5/3, n. 2, p. 126–131, 2002.

EIBEN, A. E.; HINTERDING, R.; MICHALEWICZ, Z. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, v. 3, n. 2, 7 1999.

EIBEN, A. E.; SCHUT, M.; WILDE, A. R. Boosting genetic algorithms with self-adaptive selection. In: *Proceedings of the 5th IEEE Conference on Evolutionary Computation*. 1998. p. 787–792.

EIBEN, A. E.; SMITH, J. E. *Introduction to Evolutionary Computing*. Springer, 2003. ISBN 3-540-40184-9. Disponível em: <http://www.cs.vu.nl/ gusz/ecbook/ecbook.html>.

EIBEN, A. E.; SPRINKHUIZEN-KUYPER, I. G.; THIJSSEN, B. A. Competing crossovers in an adaptive ga framework. In: *Proceedings of the 5th IEEE Conference on Evolutionary Computation*. 1998. p. 787–792.

ESPINOSA, J.; VANDEWALLE, J. Constructing fuzzy models with linguistic integrity from numerical data-afreli algorithm. *IEEE Transactions on Fuzzy Systems*, v. 8, n. 5, p. 591–600, 2000.

FABRO, J. A.; NEVES JR., F.; ARRUDA, L. V. R. D. A proposal of a fuzzy-neuro predictive control, tuned by genetic algorithms, with application to the start-up control of a distillation column. In: NEDJAH, N.; DE MACEDO MOURELLE, L.; ABRAHAM, A.; KÖPPEN, M. (Ed.). *5th International Conference on Hybrid Intelligent Systems (HIS 2005), 6.9 November 2005, Rio de Janeiro, Brazil*. IEEE Computer Society, 2005. p. 539–541. ISBN 0-7695-2457-5. Disponível em: <http://doi.ieeecomputersociety.org/10.1109/ICHIS.2005.15>.

FOGARTY, T. C. Varying the probability of mutation in the genetic algorithm. In: SCHAFFER, J. D. (Ed.). *Proceedings of the 3rd International Conference on Genetic USA, June 1989*. 1989. p. 104–109.

FOGEL, L. J.; ANGELINE, P. J.; FOGEL, D. B. An evolutionary programming approach to self-adaptation finite state machines. In: *Evolutionary Programming*. 1995. p. 355–365.

GAREY, M. R.; JOHNSON, D. S. *Computer and Intractability: A Guide to the Theory of NP-Completeness.* : W. H. Freeman, 1979. ISBN 0-7167-1044-7.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. : Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201157675.

GOLDBERG, D. E.; DEB, K.; CLARK, J. H. Genetic algorithms, noise, and the sizing of populations. *Complex systems*, v. 6, p. 333–362, 1992.

GOLDBERG, D. E.; DEB, K.; THIERENS, D. Toward a better understanding of mixing in genetic algorithms. In: BELEW, R. K.; BOOKER, L. (Ed.). *Proceedings of the 4th International conference on Genetic Algorithms*. : Morgan Kauffman publishers Inc., 1991. p. 190–195.

GONZALEZ, A.; HERRERA, F. Multi-stage genetic fuzzy systems based on the iterative rule learning approach. *Mathware & Soft Computing*, v. 4, p. 233–249, 1997.

GONZALEZ, A.; PEREZ, R. A learning system of fuzzy control rules based on genetic algorithms. In: . 1996. p. 202–225.

GONZALEZ, A.; PEREZ, R. Completeness and consistency conditions for learning fuzzy rules. *Fuzzy Sets Syst.*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 96, n. 1, p. 37–51, 1998. ISSN 0165-0114.

GOODWIN, G.; NORTON, J.; VISWANATHAN, M. Persistency of excitation for nonminimal models of systems having purely deterministic disturbances. *IEEE Transactions on Automatic Control*, v. 30, n. 6, p. 589–592, 1985. ISSN 0018-9286.

GREFENSTETTE, J. J. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 16, n. 1, p. 122–128, 1986.

GUILLAUME, S.; MAGDALENA, L. Expert guided integration of induced knowledge into a fuzzy knowledge base. *Soft Computing*, Springer-Verlag, Berlin, Heidelberg, v. 10, n. 9, p. 773–784, 2006. ISSN 1432-7643.

GUROCAK, H. B. A genetic-algorithm-based method for tuning fuzzy logic controllers. *Fuzzy Sets Syst.*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 108, n. 1, p. 39–47, 1999. ISSN 0165-0114.

HANCOCK, P. J. B. An empirical comparison of selection methods in evolutionary algorithms. In: *Selected Papers from AISB Workshop on Evolutionary Computing*. London, UK: Springer-Verlag, 1994. p. 80–94. ISBN 3-540-58483-8.

HARIK; CANTU-PAZ; GOLDBERG; MILLER. The gambler's ruin problem, genetic algorithms, and the sizing of populations. In: *IEEECEP: Proceedings of The IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*. 1997. Disponível em: <citeseer.ist.psu.edu/article/harik97gamblers.html>.

HART, W. E.; KRASNOGOR, N.; SMITH, J. E. (Ed.). *Recent Advances in Memetic Algorithms*. : Springer-Verlag New York, Inc., 2004.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998. ISBN 0132733501.

HERRERA, F.; LOZANO, M. Adaptive genetic operators based on coevolution with fuzzy behaviors. *IEEE Transactions on Evolutionary Computation*, v. 5, n. 2, p. 149–165, 2001.

HERRERA, F.; LOZANO, M.; VERDEGAY, J. L. Tuning fuzzy controllers by genetic algorithms. *International Journal of Approximate Reasoning*, v. 12, p. 299–315, 1995.

HERRERA, F.; MAGDALENA, L. Introduction: Genetic fuzzy systems. *International Journal of Intelligent Systems*, v. 13, n. 10, p. 887–890, 1998.

HESSER, J.; MäNNER, R. Towards an optimal mutation probability for genetic algorithms. In: *PPSN I: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1991. p. 23–32. ISBN 3-540-54148-9.

HINTERDING, R. Gaussian mutation and self-adaptation for numeric genetic algorithms. In: *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*. 1995. p. 384–389.

HINTERDING, R. Self-adaptation using multi-chromosomes. In: *IEEE International Conference on Evolutionary Computation*. 1997. p. 87–91.

HINTERDING, R.; MICHALEWICZ, Z.; EIBEN, A. E. Adaptation in evolutionary computation: A survey. In: *Proceedings of the 4th IEEE International Conference on Evolutionary Computation*. 1997. p. 65–69.

HIROTA, K. (Ed.). *Industrial Applications of Fuzzy Technology*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1994. Translator-H. Solomon. ISBN 0387701095.

HOLLAND, J. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press, 1975.

HOLLAND, J. H. *Adaptation in natural and artificial systems*. : MIT Press, 1992. ISBN 0-262-58111-6.

HORÄK, A.; KADLEC, V.; SMRZ, P. Enhancing best analysis selection and parser comparison. In: *TSD '02: Proceedings of the 5th International Conference on Text, Speech and Dialogue*. London, UK: Springer-Verlag, 2002. p. 461–466. ISBN 3-540-44129-8.

HOROWITZ, E.; SAHNI, S. Computing partitions with applications to the knapsack problem. *J. ACM*, v. 21, n. 2, p. 277–292, 1974.

ISHIBUCHI, H.; NAKASHIMA, T. Genetic-algorithm-based approach to linguistic approximation of nonlinear functions with many input variables. *Fuzzy Systems Conference Proceedings*, v. 2, p. 779–784, 1999.

ISHIBUCHI, H.; TAKEUCHI, D.; NAKASHIMA, T. Ga-based approaches to linguistic modeling of nonlinear functions. In: *Joint 9th IFSA World Congress and 20th NAFIPS International Conference*. 2001. v. 2, p. 1229–1234.

JANG, J.-S. R. ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 23, p. 665–684, 1993. Disponível em: <citeseer.ist.psu.edu/article/jang93anfis.html>.

JANG, J.-S. R.; SUN, C.-T.; MIZUTANI, E. *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997. ISBN 0-13-261066-3.

JANIKOW, C.; MICHALEWICZ, Z. An experimental comparison of binary and floating point representations in genetic algorithms. In: *4th International Conference on Genetic Algorithms*. 1991. p. 31–38.

JIN, Y. Fuzzy modeling of high-dimensional systems: Complexity reduction and interpretability improvement. *IEEE Transactions on Fuzzy Systems*, v. 8, n. 2, p. 212–220, 2000.

KANDEL, A. *Fuzzy expert systems*. Boca Raton, FL, USA: CRC Press, Inc., 1992. ISBN 0-8493-4297-X.

KIMBROUGH, S. O.; LU, M.; WOOD, D. H.; WU, D.-J. Exploring a two-market genetic algorithm. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. : Morgan Kaufmann Publishers Inc., 2002. p. 415–422. ISBN 1-55860-878-8.

KLEMENT, E. P.; MESIAR, R.; PAP, E. *Triangular Norms*. : Kluwer, Dordrecht, 2000. (Trends in Logic, v. 8).

KLIR, G. J.; YUAN, B. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. : Prentice Hall PTR, 1995. ISBN 0131011715.

KNUTH, D. E. *Fundamental Algorithms*. Second. Reading, Massachusetts: Addison-Wesley, 1973. (The Art of Computer Programming, v. 1).

KOSKO, B. *Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992. ISBN 0-13-612334-1.

KOZANIDIS, G.; MELACHRINOUDIS, E. A branch & bound algorithm for the 0-1 mixed integer knapsack problem with linear multiple choice constraints. *Comput. Oper. Res.*, Elsevier Science Ltd., Oxford, UK, UK, v. 31, n. 5, p. 695–711, 2004. ISSN 0305-0548.

LARSEN, P. M. Industrial applications of fuzzy logic control. *Int. J. Man-Machine Stud.*, v. 12, p. 3–10, 1980.

LEE, C. C. Fuzzy logic in control systems: Fuzzy logic controller: Parts 1 and 2. *IEEE Transactions on Systems, Man and Cybernetics*, v. 20, n. 2, p. 404–435, 1990.

LEE, M.; TAKAGI, H. Dynamic control of genetic algorithms using fuzzy logic techniques. In: FORREST, S. (Ed.). *Proceedings of the Fifth International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1993.

LIN, C.-T.; LEE, C. S. G. *Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996. ISBN 0-13-235169-2.

LUKASIEWICZ, J. Philosophical remarks on many-valued systems of propositional logic. North Holland, 1970.

MAMDANI, E. H.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, v. 7, n. 1, p. 1–13, 1975.

MAMDANI, E. H.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. Hum.-Comput. Stud.*, Academic Press, Inc., Duluth, MN, USA, v. 51, n. 2, p. 135–147, 1999. ISSN 1071-5819.

MARUO, M.; DELGADO, M. R. Co-evolutionary genetic fuzzy system: a self-adapting approach. In: *Proceedings of the Ninth IEEE International Conference on Fuzzy Systems*. 2006.

MARUO, M. H.; LOPES, H. S.; DELGADO, M. R. Self-adapting evolutionary parameters: Encoding aspects for combinatorial. In: *Proceeding of 5th European Conference on Evolutionary Computation in Combinatorial Optimization*. : Springer Verlag, 2004. (Lecture Notes in Computer Science), p. 154–165.

MATSUMOTO, M.; NISHIMURA, T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, ACM Press, New York, NY, USA, v. 8, n. 1, p. 3–30, 1998. ISSN 1049-3301.

MEYER, C. D. (Ed.). *Matrix analysis and applied linear algebra*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000. ISBN 0-89871-454-0.

MICHALEWICZ, Z. *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. : Springer-Verlag New York, Inc., 1994. ISBN 3-540-58090-5.

MICHALEWICZ, Z. Genetic algorithms, numerical optimization, and constraints. In: ESHELMAN, L. (Ed.). *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann, 1995. p. 151–158.

MICHALEWICZ, Z. A survey of constraint handling techniques in evolutionary computation methods. In: MCDONNELL, J. R.; REYNOLDS, R. G.; FOGEL, D. B. (Ed.). *Proc. of the 4th Annual Conf. on Evolutionary Programming*. Cambridge, MA: MIT Press, 1995. p. 135–155. Disponível em: <citeseer.ist.psu.edu/michalewicz95survey.html>.

MICHALEWICZ, Z.; DASGUPTA, D.; RICHE, R. G. L.; SCHOENAUER, M. Evolutionary algorithms for constrained engineering problems. *Comput. Ind. Eng.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 30, n. 4, p. 851–870, 1996. ISSN 0360-8352.

MICHALEWICZ, Z.; NAZHIYATH, G. Genocop iii: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In: FOGEL, D. B. (Ed.). *Proceedings of the Second IEEE International Conference on Evolutionary Computation*. : IEEE Press, 1995. p. 647–651.

MICHALEWICZ, Z.; SCHOENAUER, M. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, v. 4, n. 1, p. 1–32, 1996. Disponível em: <citeseer.ist.psu.edu/article/michalewicz96evolutionary.html>.

MITAIM, S.; KOSKO, B. The shape of fuzzy sets in adaptive function approximation. *IEEE-FS*, v. 9, p. 637–656, Aug. 2001. Disponível em: <citeseer.ist.psu.edu/mitaim01shape.html>.

MITCHELL, M. *An introduction to genetic algorithms*. Cambridge, MA, USA: MIT Press, 1996. ISBN 0-262-13316-4.

MOGNON, V. R. *Algoritmos genéticos para otimização de arranjos de antenas*. Dissertação (Mestrado) — Universidade Federal do Paraná, 2004.

MORIARTY, D. E.; MIIKKULAINEN, R. Hierarchical evolution of neural networks. In: *Proceedings of the 1998 IEEE World Congress on Computational Intelligence*. Anchorage, Alaska, USA: IEEE Press, 1998. p. 428–433.

MUMFORD, C. L. Comparing representations and recombination operators for the multi-objective 0/1 knapsack problem. In: *CEC2003: Proceedings of The IEEE Conference on Evolutionary Computation, IEEE*. 2003. p. 854–861.

NAGAI, E. Y.; ARRUDA, L. V. R. D. Automatic generation of fuzzy models by genetic algorithms. In: *Proceedings of the 15th Triennial World Congress of the International Federation of Automatic Control*. 2002.

NAGAI, E. Y.; ARRUDA, L. V. R. D. Soft sensor based on fuzzy model identification. In: *Proceedings of the 16th World Congress of the International Federation of Automatic Control*. 2005.

NAUCK, D.; KRUSE, R. Neuro-fuzzy systems for function approximation. *Fuzzy Sets Syst.*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 101, n. 2, p. 261–271, 1999. ISSN 0165-0114.

NGUYEN, C. T.; GANESH, C.; GONG, K. F. A fuzzy logic-based intelligent controller for contact management data integration. In: *Proceedings of the IEEE World Congress on Computational Intelligence*. 1998. v. 2, p. 879–884.

PARK, J.; SANDBERG, I. W. Universal approximation using radial-basis-function networks. *Neural Computation*, v. 3, n. 2, p. 246–257, 1991.

PEDRYCZ, W. *Fuzzy Modelling: Paradigms and Practices*. Norwell, MA, USA: Kluwer Academic Publishers, 1996. ISBN 0792397037.

PEDRYCZ, W.; GOMIDE, F. *An Introduction to Fuzzy Sets: Analysis and Design*. : MIT Press, 1998.

PISINGER, D. Where are the hard knapsack problems? *Computers & Operations Research*, v. 32, n. 9, p. 2271–2284, September 2005. Disponível em: <http://dx.doi.org/10.1016/j.cor.2004.03.002>.

POLI, R.; LANGDON, W. B. On the search properties of different crossover operators in genetic programming. In: KOZA, J. R.; BANZHAF, W.; CHELLAPILLA, K.; DEB, K.; DORIGO, M.; FOGEL, D. B.; GARZON, M. H.; GOLDBERG, D. E.; IBA, H.; RIOLO, R. (Ed.). *Genetic Programming 1998: Proceedings of the Third Annual Conference*. University of Wisconsin, Madison, Wisconsin, USA: Morgan Kaufmann, 1998. p. 293–301. ISBN 1-55860-548-7.

POTTER, M. A.; DE JONG, K. A. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, v. 8, n. 1, p. 1–29, 2000.

PROAKIS, J. G.; MANOLAKIS, D. G. *Digital signal processing (3rd ed.): principles, algorithms, and applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996. ISBN 0-13-373762-4.

RECHENBERG, I. *Evolution Strategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. : Frommann-Holzboog, Stuttgart, 1973.

RIZZI, A.; MASCIOLI, F. M. F.; MARTINELLI, G. Automatic training of anfis networks. In: *Fuzzy Systems Conference Proceedings*. 1999. v. 3, p. 1655–1660.

ROJAS, I.; ORTEGA, J.; PELAYO, F. J.; PRIETO, A. Statistical analysis of the main parameters: in the fuzzy inference process. *Fuzzy Sets Syst.*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 102, n. 2, p. 157–173, 1999. ISSN 0165-0114.

ROSENKRANTZ, D. J.; STEARNS, R. E.; II, P. M. L. An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comput.*, v. 6, n. 3, p. 563–581, 1977.

ROUBOS, J.; BABUSKA, R. Comments on the benchmarks in "a proposal for improving the accuracy of linguistic modeling"and related articles. *IEEE Transactions on Fuzzy Systems*, v. 11, n. 6, p. 861– 865, 2003.

SCAPIN, M. P.; LOPES, H. S. An enhanced genetic algorithm for protein structure prediction using the 2d hydrophobic-polar model. In: *Lecture Notes in Computer Science - Proc. of 7th International Conference on Artificial Evolution*. 2005. v. 3871, p. 238–246.

SCHAFFER, J. D.; MORISHIMA, A. An adaptive crossover distribution mechanism for genetic algorithms. In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*. : Lawrence Erlbaum Associates, Inc., 1987. p. 36–40. ISBN 0-8058-0158-8.

SETNES, M.; BABUSKA, R.; VERBRUGGEN, H. Complexity reduction in fuzzy modeling. *Mathematics and Computers in Simulation*, v. 46, n. 5, p. 507–516, 1998.

SHI, Y.; EBERHART, R.; CHEN, Y. Implementation of evolutionary fuzzy systems. *IEEE Transactions on Fuzzy Systems*, v. 7, n. 2, p. 109–119, 1999.

SKOROBOHATYJ, G. *Integer/mixed-integer programming problems*. 2002. Http://elib.zib.de/pub/Packages/mp-testdata/ip/index.html.

SMITH, J.; FOGARTY, T. C. Self-adaptation of mutation rates in a steady-state genetic algorithm. *Proceedings of IEEE International Conference on Evolutionary Computation*, p. 318–323, 5 1996.

SMITH, J.; FOGARTY, T. C. Operator and parameter adaptation in genetic algorithms. *Soft Computing*, v. 1, n. 2, p. 81–87, 1997.

SMITH, J. E. *Self-adaptation in genetic algorithms*. Tese (Doutorado) — Faculty of computer studies and Mathematics, University of the West of England, Bristol, 1998.

SOULE, T.; FOSTER, J. A. Code size and depth flows in genetic programming. In: KOZA, J. R.; DEB, K.; DORIGO, M.; FOGEL, D. B.; GARZON, M.; IBA, H.; RIOLO, R. L. (Ed.). *Genetic Programming 1997: Proceedings of the Second Annual Conference*. Stanford University, CA, USA: Morgan Kaufmann, 1997. p. 313–320. Disponível em: <citeseer.ist.psu.edu/soule97code.html>.

SPEARS, W. M. Adapting crossover in evolutionary algorithms. In: *Proceedings of the Fourth Annual Conference on Evolutionary Programming*. San Diego, CA: , 1995.

SPEARS, W. M.; ANAND, V. A study of crossover operators in genetic programming. In: RAS, Z. W.; ZEMANKOVA, M. (Ed.). *Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems ISMIS 91*. Springer-Verlag, 1991. p. 409–418. Disponível em: <citeseer.ist.psu.edu/spears91study.html>.

SRINIVAS, M.; PATNAIK, L. M. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 24, n. 4, p. 656–667, 1994.

SUGENO, M. *Industrial Applications of Fuzzy Control*. : Elsevier Science Inc., 1985. ISBN 0444878297.

SUGENO, M.; KANG, G. T. Structure identification of fuzzy model. *Fuzzy Sets Syst.*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 28, n. 1, p. 15–33, 1988. ISSN 0165-0114.

SYSWERDA, G. Schedule optimization using genetic algorithms. In: ____. : Van Nostrand Reinhold, 1991. p. 332–349.

TAKAGI, T.; SUGENO, M. Derivation of fuzzy control rules from human operator's control actions. In: *Proceedings of the IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis*. 1985. p. 55–60.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, v. 15, n. 1, p. 116–132, 1985.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. In: DUBOIS, D.; PRADE, H.; YAGER, R. R. (Ed.). *Readings in Fuzzy Sets for Intelligent Systems*. San Mateo, CA: Kaufmann, 1993. p. 387–403.

TANENBAUM, A. S.; VAN RENESSE, R. Distributed operating systems. *ACM Comput. Surv.*, v. 17, n. 4, p. 419–470, 1985.

TAO, G.; MICHALEWICZ, Z. Inver-over operator for the tsp. In: *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1998. p. 803–812. ISBN 3-540-65078-4.

THE MATHWORKS, INC. *MATLAB The Language of Technical Computing*. version 7. 2004.

THIERENS, D. Dimensional analysis of allele-wise mixing revisited. In: *PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1996. p. 255–265. ISBN 3-540-61723-X.

THIERENS, D. Adaptive mutation rate control schemes in genetic algorithms. In: *CEC2002: Proceedings of The IEEE Conference on Evolutionary Computation, IEEE*. 2002. p. 980–985.

THIERENS, D.; GOLDBERG, D. E. Mixing in genetic algorithms. In: *Proceedings of the 5th International Conference on Genetic Algorithms*. : Morgan Kaufmann Publishers Inc., 1993. p. 38–47. ISBN 1-55860-299-2.

TSUKAMOTO, Y. An approach to fuzzy reasoning method. In: DUBOIS, D.; PRADE, H.; YAGER, R. R. (Ed.). *Readings in Fuzzy Sets for Intelligent Systems*. San Mateo, CA: Kaufmann, 1993. p. 523–529.

TSUNODA, D. F.; LOPES, H. S. Automatic motif discovery in an enzyme database using a genetic algorithm-based approach. *Soft Comput.*, v. 10, n. 4, p. 325–330, 2006.

WHITE, T.; OPPACHER, F. Adaptive crossover using automata. In: *PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1994. p. 229–238. ISBN 3-540-58484-6.

YAGER, R. R. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Trans. Syst. Man Cybern.*, IEEE Press, Piscataway, NJ, USA, v. 18, n. 1, p. 183–190, 1988. ISSN 0018-9472.

YAGER, R. R. Expert systems using fuzzy logic. In: YAGER, R. R.; ZADEH, L. A. (Ed.). *An Introduction to Fuzzy Logic Applications in Intelligent Systems*. Boston: Kluwer, 1992. p. 27–44.

YUAN, Y.; ZHUANG, H. A genetic algorithm for generating fuzzy classification rules,. *Fuzzy Sets and Systems*, v. 84, n. 1, 1996.

ZADEH, L. Fuzzy Sets. *Information and Control*, v. 3, n. 8, p. 338–353, 1965.

ZADEH, L. The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences*, v. 1, p. 119–249, 1975.

ZADEH, L. A. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. Systems, Man and Cybernetics*, SMC-3, p. 28–44, 1973.

ZIMMERMANN, H.; ZYSNO, P. Latent connectives in human decision making. *Fuzzy Sets and Systems*, v. 4, p. 37–51, 1980.

RESUMO:

Abordagens híbridas baseadas em sistemas nebulosos com aprendizado através de algoritmos genéticos são também denominadas de sistemas genético-nebulosos. Neste caso, os algoritmos genéticos são usados para resolver o problema do projeto automático de sistemas nebulosos. Entretanto, a questão da definição dos parâmetros evolutivos do algoritmo genético persiste. Esta dissertação propõe uma metodologia para o controle dos parâmetros evolutivos para um sistema genético-nebuloso. A abordagem do sistema genético-nebuloso que está sendo utilizada é inspirada em um modelo anterior, no qual diferentes populações codificam soluções parciais do problema do projeto automático de sistemas nebulosos, populações estas organizadas em quatro níveis hierárquicos que co-evoluem de forma harmônica. Cada nível hierárquico codifica as funções de pertinência, as regras individuais, as bases de regras e os sistemas nebulosos, respectivamente. A co-evolução permite que sejam estabelecidas relações de hierarquia e cooperação entre indivíduos representando os diferentes parâmetros dos sistemas nebulosos. O modelo original apresenta bom desempenho para uma grande classe de problemas. Entretanto, devido ao uso de múltiplas populações, com informações significativamente diferentes, o ajuste dos parâmetros evolutivos do sistema se torna um problema complexo. Portanto, o objetivo do uso de uma abordagem auto-adaptativa, na qual o próprio AG se encarrega de selecionar um bom conjunto de parâmetros, é liberar o usuário do processo de definição manual dos parâmetros evolutivos mantendo o bom desempenho do sistema nebuloso. A utilizaçao do algoritmo evolutivo, nao apenas para encontrar a soluçao do problema, mas também para ajustar uma série de parâmetros do próprio algoritmo se constitui em uma das principais contribuições deste trabalho. O desempenho do mecanismo de auto-adaptação de parâmetros evolutivos que está sendo proposto é avaliado em duas fases: inicialmente, a auto-adaptação é testada, utilizando-se problemas de otimização contínua e combinatória; depois, a auto-adaptação é aplicada para resolver o problema do projeto automático de sistemas baseados em regras nebulosas, e para isto, o sistema genético-nebuloso resultante é usado na aproximação de funções.

PALAVRAS-CHAVE

Sistemas genético-nebulosos, auto-adaptação, projeto automático de sistemas nebulosos.

2006

ÁREA/SUB-ÁREA DE CONHECIMENTO

1.00.00.00-3 – Ciências Exatas e da Terra

1.03.03.04-9 – Sistemas de Informação

N°: 415

# Livros Grátis

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

Milhares de Livros para Download:

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