Álvaro Roberto Silvestre Fialho

Exploração de Relações Entre as Técnicas Nebulosas e Evolutivas da Inteligência Computacional

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo (EPUSP) como parte dos requisitos para a obtenção do Título de Mestre em Engenharia no Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

São Paulo
2007

Álvaro Roberto Silvestre Fialho

Exploração de Relações Entre as Técnicas Nebulosas e Evolutivas da Inteligência Computacional

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo (EPUSP) como parte dos requisitos para a obtenção do Título de Mestre em Engenharia no Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

Área de concentração: Inteligência Artificial

Orientador:
Prof. Livre-Docente
Marco Túlio Carvalho de Andrade

São Paulo
2007

Ficha Catalográfica

Fialho, Álvaro Roberto Silvestre
Exploração de Relações Entre as Técnicas Nebulosas e Evolutivas da Inteligência Computacional. São Paulo, 2007.

109 p.

Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Inteligência Artificial. 2. Fuzzy (Inteligência Artificial). 3. Computação Evolutiva. 4. Sistemas Híbridos. I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais. II. t

Dedicatória

A Deus, a meus pais Alvaro e Cristina e irmão Alesi, e à minha amada esposa Camila.

Agradecimentos

Agradeço a Deus em primeiro lugar, pois sem Ele nada disso teria se concretizado. Através dEle eu renovo as minhas forças a cada dia.

Aos meus pais Alvaro e Cristina e a meu irmão Alesi, sou eternamente grato pelo amor e carinho, por terem me dado todo o apoio necessário para alcançar mais este objetivo, e por formarem essa família especial que eu amo muito e que é um exemplo nos dias de hoje.

À minha eterna companheira Camila, minha esposa amada, agradeço pelo amor incondicional, por ser paciente comigo nos dias ruins, por me motivar sempre e, principalmente, agradeço muito por ter me esperado, por ter suportado junto comigo esse longo período à distância que agora chega ao fim.

Ao meu orientador, Professor Marco Túlio, um agradecimento especial pela amizade, parceria, camaradagem, enfim, a sua orientação e a sua conduta, tanto no sentido acadêmico como no pessoal, com certeza me ensinaram muito, só tenho a lhe agradecer por esse período.

A todos os meus amigos e colegas da Escola Politécnica da Universidade de São Paulo. Em especial ao amigo Fábio Lopes Caversan, cujas longas conversas contribuíram muito para o meu trabalho. E aos outros membros do Laboratório de Engenharia de Conhecimento (KNOMA). Esses dois anos de convivência quase diária não poderiam ter sido melhores, tenho certeza que levarei daqui muitas amizades verdadeiras.

Sim, eu amo Deus.
Ele é a fonte de minha existência, é o meu Senhor e o meu Salvador.
Sem Ele eu não sou nada, mas com Ele eu posso todas as coisas,
através de Jesus Cristo, que me fortalece.
Filipenses 4, 13.

Resumo

FIALHO, A. R. S. **Exploração de Relações Entre as Técnicas Nebulosas e Evolutivas da Inteligência Computacional**. 2007. 109 f. Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo, São Paulo, 2007.

Neste trabalho foi realizada uma busca por relações, regras e transformações entre duas metodologias constituintes da Inteligência Computacional - a Computação Nebulosa e a Computação Evolutiva. Com a organização e sistematização da existência de tais transformações, obtém-se uma mudança na modelagem de soluções que as utilizam de forma conjunta, possibilitando que teorias e modelos bem estabelecidos em uma das metodologias possam ser aproveitados pela outra de uma forma mais robusta, correta por construção, intrínseca e transparente.

Um modelo foi proposto para direcionar o trabalho de pesquisa. Através da análise desse modelo e de uma revisão bibliográfica realizada, transformações pontuais entre as metodologias foram elencadas, e posteriormente consolidadas por meio de experimentos práticos: uma Base de Conhecimento (BC) de um Controlador Lógico Nebuloso foi criada e modificada, conforme a necessidade, através de um Algoritmo Genético (AG).

Com a abordagem desenvolvida, além da criação de BCs a partir de pouquíssimo conhecimento sobre o domínio do problema, tornou-se possível a inserção de novos "comportamentos desejados" em BCs já existentes, automaticamente, através de AGs. Os resultados desses experimentos, realizados sobre uma plataforma computacional especificada e implementada para este fim, foram apresentados e analisados.

Palavras-chave: Inteligência Artificial. Inteligência Computacional. Computação Nebulosa. Computação Evolutiva. Sistemas Inteligentes Híbridos.

Abstract

FIALHO, A. R. S. **Exploration of Relations Between the Fuzzy and the Evolutionary Techniques of Computational Intelligence**. 2007. 109 f. Dissertation (Master´s Degree) - Escola Politécnica da Universidade de São Paulo, São Paulo, 2007.

This work addressed a search of relations, rules and transformations between two Computational Intelligence constituent methodologies - Fuzzy Computing and Evolutionary Computing. The existence of these relations changes the actual way of solutions modeling that uses these methodologies, allowing the utilization of well established theories and models of one technique by the other in a more robust, intrinsic and transparent way.

Besides the research and systematization of points that indicate the existence of relations between the two methodologies, a model to guide these exploration was proposed. By this model analysis and by the bibliographic revision made, punctual transformations were pointed out, and further consolidated through practical experiments: a Knowledge Base (KB) of a Fuzzy Logic Controller was created and modified automatically by a Genetic Algorithm.

With the developed approach, besides the creation of KBs, it became possible to automatically insert new "desired behaviors" to existent KBs. The results of such experiments, realized through a computational platform specified and implemented to this task, were presented and analyzed.

Keywords: Artificial Intelligence. Computational Intelligence. Fuzzy Computing. Evolutionary Computing. Hybrid Intelligent Systems.

Lista de Figuras

2.1 Diagrama de Hassi-Euler do conjunto nebuloso “horas de sono”. 27

2.2 Formatos de algumas funções de pertinência. 27

2.3 Diagrama de Hassi-Euler da variável linguística “nota” 29

2.4 Diagrama típico de um Sistema de Inferência Nebuloso. 30

2.5 Modelo de neurônio artificial. 43

2.6 Principais tipos de aplicações híbridas utilizando IC. 44

3.1 Diagrama GK, reproduzido de Wagner, Jansh-Pôrto e Weber (1988). . . . 47

3.2 Eixos do Diagrama GK e as técnicas de IC. 48

3.3 Transporte de elementos entre os eixos do Diagrama AC. 51

3.4 Estágio atual do diagrama Andrade-Caversan-Fialho. 52

4.1 Algumas intersecções entre as áreas de atuação das metodologias de IC. . . 57

4.2 Troca de informações entre um AG e um CLN. 59

4.3 Sistema integrando CE e CF através de transformações. 60

4.4 Transporte de elementos entre os eixos do Diagrama ACF. 66

5.1 Diagrama de estados de uma instância da UCIP, reproduzido de Caversan (2006). 68

5.2 Diagrama de casos de uso e componentes da plataforma UCIP, reproduzido de Caversan (2006). 69

5.3 Modelo lógico do pacote para realização de CE 71

5.4 Diagrama de classes do pacote para realização de CE 72

5.5 Diagrama de atividades representando a operação com CE 73

5.6 Exemplo de tabuleiro e da sua respectiva codificação evolutiva. 74

5.7 Diagrama de classes do pacote MT, adaptado de Caversan (2006). 75

6.1 Robô autoguiado com os seus sensores (F, L e R) e a saída (θ), reproduzido de Caversan (2006). 78

6.2 Interface principal do ambiente de simulação estendido. 79

6.3 Trajetórias obtidas a partir de BCs criadas de forma diferente: a) Especialistas b) Algoritmo Genético. 90

6.4 Mapa original, com robô se locomovendo em círculos. 91

6.5 Mapa estendido, com um robô atingindo todas as regiões-alvo. 93

6.6 Robô "centralizado". 94

6.7 Robô "suavizado". 95

A.1 Modelo lógico do pacote para realização de Computação Nebulosa (CF) . . 105

A.2 Diagrama de classes do pacote para realização de CF. 107

A.3 Diagrama de atividades representando a operação com CF. 108

Lista de Tabelas

4.1 Exemplo de isomorfismo entre sistemas lógicos formais 55

Lista de Algoritmos

1 Procedimento padrão de algoritmos evolutivos 36
2 Algoritmo para exemplo de representação Programação Genética (PG) . . 41
3 Conversão de Indivíduo para Base de Conhecimento Nebulosa 85
4 Construtor de Genes para Conjunto Nebuloso 86
5 Conversão de Base de Conhecimento Nebulosa para Indivíduo 88

Lista de Abreviaturas

AC Andrade-Caversan

ACF Andrade-Caversan-Fialho

AG Algoritmo Genético

BD Base de Dados

BR Base de Regras

BC Base de Conhecimento

CE Computação Evolutiva

CF Computação Nebulosa

CLN Controlador Lógico Nebuloso

CN Computação Neural

CNAO Controlador Nebuloso Auto-Organizável

GK Gajski-Kuhn

IA Inteligência Artificial

IC Inteligência Computacional

ICM Intelligent Computational Model

MT Model Transformation

PG Programação Genética

RNA Rede Neural Artificial

SIN Sistema de Inferência Nebuloso

UCIP Unified Computational Intelligence Plataform

Sumário

1 Introdução **18**

1.1 Motivação e Justificativa 20

1.2 Objetivos 21

1.3 Metodologia 22

1.4 Estrutura do Trabalho 22

2 Inteligência Computacional (IC) **24**

2.1 Computação Nebulosa (CF) 25

2.1.1 Conjuntos Nebulosos 26

2.1.2 Operações sobre Conjuntos Nebulosos 28

2.1.3 Variáveis Linguísticas - Base de Dados 28

2.1.4 Regras Linguísticas - Base de Regras 29

2.1.5 Sistemas de Inferência Nebulosos (SINs) 30

2.1.6 Vantagens e Desvantagens 32

2.2 Computação Evolutiva (CE) 33

2.2.1 Algoritmos Evolutivos 33

2.2.2 Estratégias Evolutivas (EEs) 37

2.2.3 Programação Evolutiva (PE) 38

2.2.4 Algoritmos Genéticos (AGs) 38

2.2.5 Programação Genética (PG) 40

2.2.6 Vantagens e Desvantagens 41

2.3 Computação Neural (CN) 42

2.3.1 Vantagens e Desvantagens 44
2.4 Aplicações Híbridas 44
2.5 Considerações Finais do Capítulo 45

3 Modelos para a Busca por Relações 46

3.1 Diagrama Gajski-Kuhn (GK) 46
3.2 Diagrama Andrade-Caversan (AC) 49
3.3 Diagrama Andrade-Caversan-Fialho (ACF) 52
3.4 Considerações Finais do Capítulo 53

4 Busca de Transformações entre CE e CF 54

4.1 Relações Isomórficas 54
4.2 Substituição de Técnica 55
4.2.1 CF Para Aprendizado 56
4.2.2 CE Para Classificação 56
4.2.3 Análise Crítica 57
4.3 Sistemas Híbridos 58
4.3.1 Híbridos Inter-Comunicativos - Troca de Informações 58
4.3.2 Híbridos Polimórficos - Transformação de Componentes 59
4.4 Transformações entre CE e CF 60
4.4.1 Base de Dados (BD) 60
4.4.2 Base de Regras (BR) 63
4.4.3 Base de Conhecimento (BC) 65
4.5 Considerações Finais do Capítulo 65

5 Plataforma Computacional para Exploração das Relações 67

5.1 Especificação dos Requisitos 67
5.2 Especificação do Projeto 70

5.2.1 Pacote *Evolutionary Computing* (EC) 70

5.2.2 Pacote *Model Transformation* (MT) 74

5.3 Considerações Finais do Capítulo 75

6 Experimentos Computacionais 76

6.1 Navegação de Robôs Autoguiados 76

6.2 Criação e Aprimoramento de Bases de Conhecimento (BC) 77

6.2.1 Ambiente de Simulação 79

6.2.2 Abordagem Baseada em Shi, Eberhart e Chen (1999) 81

6.2.3 Inicialização 82

6.2.4 Operadores Genéticos 83

6.2.5 Transformações Implementadas CE ⇒ CF 84

6.2.6 Transformações Implementadas CF ⇒ CE 86

6.3 Experimentos Realizados 89

6.3.1 Criação 1 - Comparação com Solução Existente 89

6.3.2 Criação 2 - Evitar Colisões (EV) 90

6.3.3 Análise Crítica - Experimentos de Criação 91

6.3.4 Aprimoramento 1 - EV + Regiões-Alvo (RA) 92

6.3.5 Aprimoramento 2 - EV + RA + Centralizador (CE) 93

6.3.6 Aprimoramento 3 - EV + RA + CE + Suavizador (SU) 94

6.4 Considerações Finais do Capítulo 94

7 Considerações Finais 97

7.1 Cumprimento dos Objetivos Propostos 97

7.2 Contribuições do Trabalho 98

7.3 Trabalhos Futuros 99

Referências 101

Anexo A – Pacote FC - Plataforma Unified Computational Intelligence Plataform (UCIP) 105

1 Introdução

Na solução de problemas *ideais*, os sistemas a serem modelados ou controlados são descritos por informações completas e precisas. Entretanto, quando trata-se de resolver problemas do *mundo real*, nota-se que tais sistemas são tipicamente mal-definidos, difíceis de serem modelados, e/ou possuem grandes espaços de soluções, tornando muito difícil a solução através da abordagem algorítmica ou a modelagem matemática tradicionais. Mas esses mesmos problemas são facilmente resolvidos pelo aparato que compõe a chamada inteligência humana.

É nesse contexto que entra a Inteligência Computacional (IC), uma associação de metodologias computacionais que explora o potencial para a criação de "Máquinas Inteligentes", através da modelagem de comportamentos e mecanismos, que tentam imitar organismos biologicamente constituídos (FOGEL; FUKUDA; GUAN, 1999), fazendo uso da tolerância de imprecisões, incertezas, e verdades parciais, para atingir tratabilidade, robustez, e melhor conformidade com a realidade (ZADEH, 1994). Entre as suas técnicas constituintes, podemos destacar como principais: a Computação Nebulosa (*Fuzzy* - CF), a Computação Evolutiva (CE) e a Computação Neural (CN).

A CF é a metodologia que tenta imitar o raciocínio humano, aproximado, sem se preocupar com a exatidão numérica, para tratar de problemas vagos e mal-definidos. Trata-se do ramo simbólico da IC, que têm como base a lógica nebulosa e a teoria de conjuntos nebulosos, uma extensão da teoria de conjuntos tradicional. Lançada a partir da publicação de Zadeh (1965), a CF têm como uma de suas principais características a robustez do seu mecanismo de raciocínio (TSAKONAS; DOUNIAS, 2002), e é muito utilizada em aplicações de controle, possibilitando a modelagem sistematizada do conhecimento especialista para a solução do problema.

A CE é a metodologia de busca por soluções através da modelagem computacional da teoria da evolução. Basicamente, um indivíduo da teoria da evolução das espécies representa um candidato à solução de um determinado problema. Assim, o algoritmo roda sobre uma população de soluções-candidatas, com uma função que avalia a aptidão

(*fitness*) de cada indivíduo como solução para o problema, executando assim o papel do meio-ambiente - "Só os mais aptos sobrevivem". Através de operadores genéticos de cruzamento, cópia (elitismo) e mutação, chega-se a uma solução ótima, ou muito perto disso. É muito utilizada em problemas de otimização, principalmente naqueles cujo espaço de busca é muito grande e não se têm um bom conhecimento sobre o domínio.

Conhecida como o ramo conexionista da IC, a CN é a metodologia que se inspira na estrutura física do cérebro para a modelagem de soluções. Camadas de neurônios artificiais interligadas por sinapses, com os seus respectivos pesos, recebem o nome de Redes Neurais Artificiais (RNAs). Bastante utilizadas em problemas de classificação de padrões, as RNAs necessitam ser treinadas com um grande conjunto de amostras (exemplos) de relações de entrada-saída para, assim, atingir uma boa generalização do problema.

Em Andrade (2002), propôs-se a idéia de que essas três metodologias da IC citadas possuem relações, regras e transformações entre si, o que muda a forma de modelagem de soluções que as utilizam. Exemplificando-se, um problema cuja solução tradicional é modelada em uma técnica, poderia ser convertido para outra técnica e resolvido utilizando-se a mesma. Com isso, têm-se mais uma alternativa de obtenção de resultados, sejam esses resultados da mesma natureza ou não.

Por tratar-se de uma linha de pesquisa muito abrangente, buscando relações entre as três metodologias citadas, ela foi dividida em 3 sub-pesquisas, sendo elas: 1) a busca pelas relações, regras e transformações entre CF e CN; 2) entre CF e CE; e 3) entre CE e CN; sendo que cada uma delas foi designada para um trabalho de mestrado.

As relações entre CF e CN foram as primeiras a serem estudadas (CAVERSAN, 2006). Um problema de navegação de veículo autoguiado foi modelado e resolvido utilizando-se um Sistema de Inferência Nebuloso (CF). Este modelo foi então transformado para a forma de Redes Neurais Artificiais (CN), resolvendo o problema de forma satisfatória também. Para isso, algumas transformações de CF para CN foram definidas, porém as transformações de volta (CN para CF) representam uma alternativa para trabalhos futuros.

Neste trabalho foi realizada a segunda das sub-pesquisas citadas, a busca de relações, regras e transformações entre técnicas de CF e CE. Para chegar até a transformação "total" de um domínio para outro, caso seja possível, é preciso definir relações e transformações ponto-a-ponto, partindo-se dos componentes mais básicos, subindo o nível de abstração gradativamente. A busca e validação dessas transformações pontuais foi o principal foco seguido por esta pesquisa.

Um modelo foi proposto para direcionar a busca, o qual recebeu o nome de diagrama Andrade-Caversan-Fialho (ACF), sendo baseado no diagrama Gajski-Kuhn (GK) (GAJSKI; KUHN, 1983). Dado que não existe apenas uma técnica representando cada um dos domínios em questão, diferentes caminhos foram abordadas como possíveis objetos de estudo para a busca por relações. Um histórico sobre as mudanças e a justificativa delas terem acontecido também faz parte do escopo deste trabalho. A alternativa analisada apresenta os Controladores Lógicos Nebulosos (CLNs) como representantes da CF, e os Algoritmos Genéticos (AGs) como representantes da CE.

Através da análise do diagrama e de um levantamento bibliográfico, transformações pontuais foram elencadas e validadas através da realização de experimentos: uma Base de Conhecimento (BC) nebulosa foi criada e posteriormente modificada, conforme a necessidade, através de um Algoritmo Genético (CE). A partir da abordagem desenvolvida, tornou-se possível também a inserção de novos "comportamentos desejados" em BCs já existentes. Os resultados desses experimentos, realizados sobre uma plataforma computacional especificada e implementada para este fim, foram analisados.

1.1 Motivação e Justificativa

Esta linha de pesquisa teve sua origem em Andrade (2002), quando vislumbrou-se a possibilidade das técnicas constituintes da IC possuírem relações isomórficas entre si. Uma vez constatada a existência de tais relações, teorias e modelos bem estabelecidos em uma das técnicas podem ser utilizados em outras. Por exemplo, os métodos de otimização de CE sendo utilizados pela CF, e a capacidade de utilização de conhecimento especialista de CF sendo utilizada pela CE. De acordo com Caversan e Andrade (2005), aprofundando-se mais, a existência de tais transformações pode levar à constatação de um Modelo Unificado da Inteligência Computacional, uma generalização da qual essas técnicas seriam especializações ou casos particulares.

Tal aproveitamento de vantagens de uma técnica em outra é comumente alcançado através do hibridismo. De acordo com Tsakonas e Dounias (2002), as duas metodologias principais deste trabalho têm sido utilizadas de forma conjunta em várias aplicações de controle e na soluções de problemas complexos de otimização. Tanto a CF sendo ajustada pela CE (CORDON et al., 2004), quanto a CE sendo controlada pela CF (ARNONE; DELL'ORTO; TETTAMANZI, 1994), ambas as abordagens já foram extensivamente testadas e utilizadas e se demonstraram muito eficientes, conforme apresentado em Cordon, Herrera e Lozano (1996). Essa grande "sinergia" entre as duas metodologias em questão é

mais uma grande motivação para esta pesquisa.

Em um hibridismo "fraco", as diferentes técnicas operam individualmente, trocando resultados e parâmetros entre si. Mas o que este trabalho busca são relações mais fortes, nas quais elementos de uma técnica são transformados em elementos da outra, o que resulta em uma hibridez mais intrínseca e transparente.

Um exemplo que motiva este objetivo: dado que têm-se um problema para o qual deseja-se uma solução nebulosa, como por exemplo, controlar a navegação de um robô móvel, porém não consegue-se abstrair uma modelagem para o mesmo na forma de um Controlador Lógico Nebuloso (CLN), dado que a sua Base de Conhecimento não está disponível ou está incompleta. Para isso, uma abordagem evolutiva, como os AGs, pode ser utilizada para a criação/otimização automática desse elemento essencial do CLN. Uma população genética é criada, com cada indivíduo representando uma solução nebulosa para o problema, e o processo de evolução é iniciado. Uma vez que um desses indivíduos atinge um *fitness* alto (de acordo com o comportamento desejado do controlador), ele pode ser transformado de volta para o domínio da CF, solucionando o problema.

Outro fator motivante é que já existem referências abrangendo transformações pontuais entre as duas metodologias em questão, que se mostraram muito úteis, como por exemplo em Hoffmann e Pfister (1997) e Shi, Eberhart e Chen (1999), nos quais são apresentados esquemas de codificação que permitem a transformação de objetos da CF para objetos da CE.

Além disso, o primeiro pesquisador a ingressar nesta linha, buscando relações entre as técnicas de CF e CN, obteve resultados na sua pesquisa, publicada em Caversan (2006). Antes disso, um artigo expondo a proposta como um todo foi publicado em uma revista nacional (CAVERSAN; ANDRADE, 2005), sendo aceito após ser julgado por pesquisadores da área, sendo isso mais um motivo para a continuidade deste trabalho.

1.2 Objetivos

1. Apresentar argumentos que suportam a idéia da existência de transformações entre as técnicas de CF e CE.

2. Propor abordagens e modelos para a busca dessas transformações.

3. Fundamentar, organizar e sistematizar as transformações encontradas.

4. Especificar a parte evolutiva da plataforma de apoio à pesquisa.

5. Realizar modificações na parte de CF da mesma plataforma, caso seja necessário.
6. Desenvolver um protótipo a partir destas especificações, com as transformações elencadas.
7. Realizar experimentos com o protótipo desenvolvido.
8. Analisar os resultados obtidos, mostrando a aplicabilidade e utilidade das transformações propostas.

1.3 Metodologia

Por utilizar uma abordagem empírica, o primeiro passo necessário para o cumprimento dos objetivos listados foi a proposta de um modelo, útil para direcionar a pesquisa, o diagrama Andrade-Caversan-Fialho.

Paralelamente, foi realizado um estudo das possíveis abordagens que a pesquisa poderia trilhar, e das técnicas pertinentes a cada uma das metodologias que seriam analisadas. O caminho dos Sistemas Inteligentes Híbridos foi tomado, tendo os Controladores Lógicos Nebulosos como representantes da CF e os Algoritmos Genéticos representando a CE.

Definidas a abordagem e as técnicas, foi realizada uma revisão das transformações pontuais espalhadas pela literatura, a fim de analisar a aplicabilidade de transporte de pontos de um domínio para o outro. Além da viabilidade de implementação, fez-se importante também verificar as vantagens que poderiam ser obtidas com tais transformações, para assim justificá-las.

Elencadas as possíveis transformações, uma plataforma computacional existente foi estendida para cobrir também experimentos no domínio da CE. A partir da validação dessa extensão, foram implementados os transportes previamente selecionados.

Com isso, experimentos puderam ser realizados, consolidando de fato a utilidade e aplicabilidade de tais transformações. Os resultados foram analisados, e as vantagens obtidas com a utilização de abordagem desenvolvida foram destacadas.

1.4 Estrutura do Trabalho

O próximo capítulo tem como objetivo a fundamentação teórica na área de IC, traçando as diferenças mais aparentes entre este campo de pesquisa e a Inteligência Artificial (IA),

e apresentando os principais conceitos das técnicas que fazem parte do objetivo deste trabalho.

No terceiro capítulo são expostas e ilustradas algumas propostas de modelos analisadas para serem utilizadas no direcionamento da pesquisa, com o diagrama atual sendo brevemente descrito. É realizada também uma contextualização da linha de pesquisa, desde as suas idéias originais apresentadas em Andrade (2002) até os últimos resultados já publicados em Caversan (2006), somados ao que foi idealizado por esta pesquisa.

O capítulo quatro apresenta uma explanação sobre as intersecções entre as metodologias de IC, tópico muito importante para a busca por relações. Alguns caminhos abandonados e o caminho seguido pela pesquisa são apresentados. Depois disso, são descritas algumas transformações pontuais já publicadas, as quais foram posteriormente analisadas para implementação.

O quinto capítulo apresenta a especificação da plataforma computacional, com um resumo do que já havia sido feito em trabalhos anteriores, e apresentando com um maior nível de detalhe o que foi realizado neste trabalho.

No sexto capítulo é apresentado o problema proposto para os experimentos, assim como os experimentos realizados, juntamente com uma análise dos resultados obtidos.

O sétimo e último capítulo traz as considerações finais do trabalho, relacionando as contribuições obtidas com o mesmo e apresentando possíveis idéias de continuidade nesta linha de pesquisa.

2 Inteligência Computacional (IC)

Este capítulo tem como objetivo a fundamentação teórica na área de IC, traçando as diferenças mais aparentes entre este campo de pesquisa e a Inteligência Artificial (IA), e apresentando os principais conceitos das técnicas que fazem parte do objetivo deste trabalho, introduzindo também o conceito de sistemas híbridos.

A palavra *inteligência* é considerada até hoje um termo vago e indefinido. Segundo Fogel, Fukuda e Guan (1999), pesquisadores se esforçaram na tentativa de escrever programas que simplesmente demonstravam um nível de competência igual ou maior que a dos humanos em alguma atividade, ao invés de se preocuparem com os processos fundamentais do comportamento inteligente, satisfazendo assim, em partes, o teste de Turing. Alguns exemplos simples, entretanto, podem demonstrar que apenas o fato de superar os humanos em certa atividade não é o suficiente para considerar um programa inteligente, como por exemplo:

- um relógio pode marcar o tempo com muito mais precisão do que uma pessoa;
- um sistema de radar pode determinar a velocidade de um carro com muito mais precisão do que uma pessoa;
- uma calculadora pode fazer contas com precisão e velocidade muito maiores que uma pessoa.

Como os exemplos acima, muitas outras tarefas são melhor executadas por máquinas do que por pessoas, e as máquinas ainda tem a vantagem de não possuírem as fragilidades dos humanos, como por exemplo, idade avançada, doenças, cansaço. Mas o fato dessas máquinas serem melhores que os humanos em determinadas atividades não faz delas inteligentes, mas sim apenas máquinas úteis para auxiliar os humanos.

O *Deep Blue*, por exemplo, derrotou em 1997 o campeão mundial de xadrez Garry Kasparov. Uma conquista louvável, mas que não traz nenhum avanço para o entendimento

da inteligência, e nem oferece uma teoria que se encaixe com o que ocorre com organismos biologicamente constituídos. Trata-se de uma aplicação que representa muito bem a IA, construída para vencer jogos de xadrez, sem se preocupar com a maneira com que os humanos aprendem, raciocinam e tomam decisões. Utilizando heurísticas para buscas em árvore e, acima de tudo, força bruta, o *Deep Blue* era capaz na época de avaliar 200 milhões possíveis posições de xadrez por segundo, milhões de vezes mais do que um jogador humano faz. Para atingir o seu objetivo, ele usava uma função de avaliação desenvolvida por humanos, testada com humanos e, o mais importante, gerada sem utilizar nenhum aprendizado próprio, de acordo com Clark (1997). É muito fácil perceber que as técnicas e recursos utilizados neste sistema não são os elementos-chave da inteligência biológica.

De acordo com Fogel, Fukuda e Guan (1999), a inteligência biológica não opera através de símbolos bem definidos, mas sim através de conceitos linguísticos que são sujeitos a interpretações diferentes e são, de alguma maneira, *nebulosos*, o que contrasta com a programação precisa, com as regras de comportamento precisamente definidas (*crisp*), processadas sequencialmente. O nosso cérebro, assim como de outros organismos vivos, não é divididos em fatos ordenados. Ao invés disso, são sistemas de processamento altamente paralelo, nos quais o conhecimento é distribuído em um "mar de conexões". Os humanos, chimpanzés, e até mesmo grupos sociais, como por exemplo, as formigas e as abelhas, todos constituem bons exemplos de sistemas inteligentes.

Ao invés de procurar melhorar as soluções, desenvolvendo heurísticas mais rápidas para busca em árvores, é interessante examinar os fatores para a geração de máquinas inteligentes, através da modelagem dos processos da natureza, um paradigma que atualmente é chamado como **Inteligência Computacional (IC)**. Entre as suas metodologias constituintes, podemos destacar como principais: **Computação Nebulosa (CF)**, **Computação Evolutiva (CE)** e **Computação Neural (CN)**. Este trabalho se concentra em duas delas, a CF e a CE, que serão apresentadas a seguir. A linha de pesquisa, como um todo, propõe a busca por relações entre as 3 metodologias. Por esse motivo, apesar de não estar no escopo deste trabalho, a CN também será brevemente descrita.

2.1 Computação Nebulosa (CF)

Para solucionar computacionalmente problemas complexos do mundo real, faz-se necessário com frequência capturar a experiência humana. De acordo com Andrade (2002), geralmente, as pessoas não baseiam seu raciocínio na dicotomia "Verdadeiro x Falso", e nem na exatidão numérica. A maneira como as pessoas encaram um problema muitas

vezes não permite a definição precisa de uma solução, em termos de números exatos, por exemplo, mas conduz a uma classificação ou agregação qualitativa em categorias gerais ou conjuntos de possíveis soluções.

É nesse contexto que entra a Computação Nebulosa (CF), proposta em Zadeh (1965), que permite a tomada de decisões baseada em valores qualitativos e não quantitativos, utilizando variáveis linguísticas e não numéricas para representar o problema e as regras utilizadas para resolvê-lo. Mesmo utilizando etiquetas de linguagem, esta técnica também permite tratar de maneira categórica questões imprecisas ou indefinidas.

2.1.1 Conjuntos Nebulosos

A CF é baseada na Teoria de Conjuntos Nebulosos, uma extensão da Teoria Clássica de Conjuntos, que não classifica elementos como pertencentes ou não-pertencentes a um determinado conjunto, ou seja, não trabalha com a dicotomia $\{falso, verdadeiro\}$ *ou* $\{0, 1\}$. Ao invés disso, ela mapeia aos elementos valores de pertinência intermediários entre a pertinência total e a não-pertinência.

A notação mais utilizada para representar a função de pertinência de um conjunto nebuloso A em um universo de discurso U se dá da seguinte forma: $\mu_A : U \rightarrow [0, 1]$. Essa função associa a cada elemento x pertencente ao universo U um número real $\mu_A(x)$ no intervalo $[0, 1]$, que representa o grau de pertinência do elemento x no conjunto A.

Existem várias formas de representar um conjunto nebuloso, dependendo basicamente da natureza e da dimensão do universo de discurso definido. A representação analítica é a mais utilizada quando o universo é discreto e composto de um número pequeno de elementos. Nesse tipo de representação, cada elemento do universo é apresentado na forma "$\mu_A(x_i)/x_i$", no qual o primeiro termo representa o grau de pertinência de x_i no conjunto A e o segundo termo representa o próprio elemento x_i. Tomando como exemplo o conjunto nebuloso simétrico "horas suficientes de sono por dia", em um universo de discurso de 4 a 12 horas, no qual 8 horas é considerado como totalmente pertinente ao conjunto, e sendo 8 também a largura do conjunto, a sua representação analítica se daria da seguinte forma:
$\mu_A(x) = \{0.0/4, 0.5/6, 1.0/8, 0.5/10, 0.0/12\}$

No caso do universo de discurso ser contínuo ou possuir uma grande quantidade de elementos discretos, a forma mais fácil de representar a sua função de pertinência é através do Diagrama de Hassi-Euler. Utilizando o mesmo conjunto citado como exemplo no

parágrafo anterior, o seu diagrama de Hassi-Euler ficaria na forma da figura 2.1.

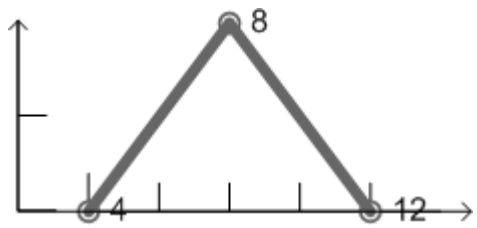


Figura 2.1: Diagrama de Hassi-Euler do conjunto nebuloso "horas de sono".

A função de pertinência de tal conjunto tem o formato triangular (*Triangular Fuzzy Number* - TFN). Alem deste, de acordo com Pedrycz e Gomide (1998), outros formatos de função de pertinência merecem ser destacados:

- Função "G" (exponencial);
- Função "S" (quadrática);
- Função Trapezoidal;
- Função Gaussiana;
- Função "exponential-like".

A função de pertinência trapezoidal, por ter sido utilizadas nos experimentos práticos deste trabalho, necessita de uma melhor apresentação. Tal função é dividida em 3 tipos, o *Right Fuzzy Number* (RFN), o *Flat Fuzzy Number* (FFN) e o *Left Fuzzy Number* (LFN), conforme pode ser visto na figura 2.2.

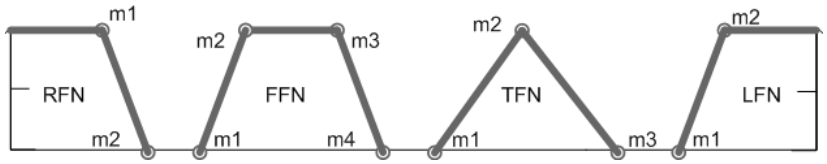


Figura 2.2: Formatos de algumas funções de pertinência.

Outras definições a respeito de conjuntos nebulosos, extraídas de Zadeh (1973), merecem ser citadas:

- Suporte: o conjunto de pontos do universo de discurso U em que $\mu_A(x)$ é positivo;
- *Crossover Point*: ponto de U no qual $\mu_A(x) = 0,5$;
- *Fuzzy Singleton*: um conjunto nebuloso cujo suporte é um único elemento.

2.1.2 Operações sobre Conjuntos Nebulosos

Os operadores básicos que se aplicam sobre os conjuntos nebulosos são extensões dos operadores utilizados em conjuntos clássicos: complemento, união e intersecção.

O operador de complemento é o correspondente do conectivo "NOT", sendo o complemento $\bar{A}$ de um conjunto nebuloso A definido por: $\mu_{\bar{A}}(x_i) = 1 - \mu_A(x_i)$.

A operação de união entre dois conjuntos nebulosos A e B representa o conectivo "OR". Podendo ser apresentada na forma de $A \cup B$ ou $A + B$, ela tem como definição básica a seguinte: $\mu_{A \cup B} = max\,[\mu_A(x_i), \mu_B(x_i)]$.

Além desse operador básico, de acordo com Rezende (2003), muitas funções, denominadas por "Normas S" ou "T-Conormas" ou "Co-Normas Triangulares", podem ser utilizadas como um operador generalizado da união entre conjuntos nebulosos, desde que tenham como requisitos as seguintes propriedades:

- Comutatividade: $S(A, B) = S(B, A)$;
- Associatividade: $S(A, S(B, C)) = S(S(A, B), C)$;
- Monotonicidade: se $A \leq B$ e $C \leq D$, então $S(A, C) \leq S(B, D)$;
- Coerência nos contornos: $S(A, 1) = 1$ e $S(A, 0) = A$.

A operação de intersecção entre dois conjuntos nebulosos A e B, representante do conectivo "AND", pode ser apresentada na forma $A \cap B$ ou $A * B$. A sua definição básica é a seguinte: $\mu_{A \cap B} = min\,[\mu_A(x_i), \mu_B(x_i)]$. Da mesma forma que o operador de união, de acordo com Rezende (2003), um operador generalizado de intersecção pode ser alcançado através de uma gama de funções, que recebem o nome de "T-Normas" ou "Normas Triangulares", necessitando das mesmas propriedades já citadas.

Outras operações também podem ser utilizadas, sendo elas: dilatação, concentração e intensificação. Definidas em Zadeh (1973), elas são agrupadas sobre o nome de modificadores linguísticos (*hedges*).

2.1.3 Variáveis Linguísticas - Base de Dados

De acordo com Rezende (2003), a capacidade de classificar de modo impreciso as variáveis de um problema, em termos de conceitos qualitativos em vez de quantitativos, traduz a idéia de uma variável linguística. Essas variáveis são formadas por vários termos

linguísticos que as caracterizam, com cada termo sendo representado por um conjunto nebuloso.

A variável linguística admite como valores apenas expressões linguísticas, como "muito quente", "pequeno", "aproximadamente médio", etc. De acordo com Lee (1990a), uma variável linguística pode ser definida pela quíntupla $(x, T(x), U, G, M)$, sendo x o nome da variável; $T(x)$ são os valores linguísticos que a variável pode assumir; U é o universo de discurso; G é a regra sintática para gerar nomes dos valores de x; e M é a regra semântica para relacionar cada valor com o seu significado.

Considerando-se $X = (nota, \{mal, medio, bem\}, [0, 10], G, M)$, a formação dos termos linguísticos com os seus respectivos conjuntos nebulosos seria, por exemplo, conforme a figura 2.3.

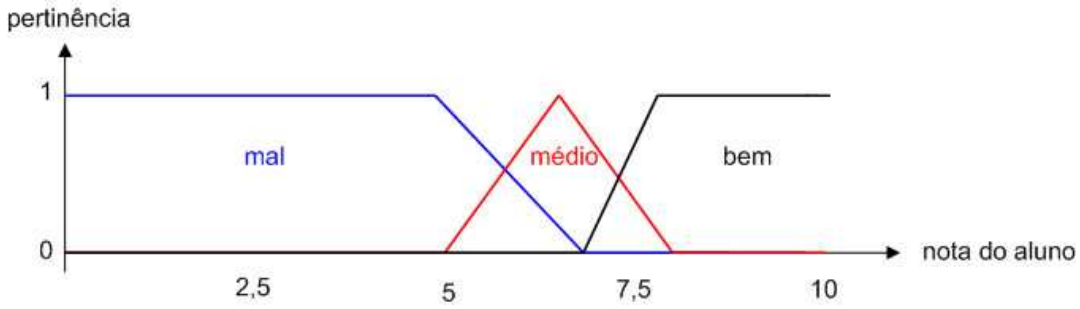


Figura 2.3: Diagrama de Hassi-Euler da variável linguística "nota"

O conjunto de variáveis linguísticas, com os seus respectivos conjuntos nebulosos, constitui a *Base de Dados* de um sistema nebuloso.

2.1.4 Regras Linguísticas - Base de Regras

As soluções que implementam CF, normalmente, são modeladas através de conhecimento especialista sobre o problema, e este conhecimento é representado e passado através de regras linguísticas, como por exemplo:

- Se o aluno estuda bastante, então a sua nota deve ser alta.

Generalizando essas regras, é possível descrevê-las da seguinte forma:

- If $\langle antecedente \rangle$ Then $\langle consequente \rangle$.

Para que a regra seja classificada como linguística, o antecedente ou o consequente precisa conter uma ou mais cláusulas linguísticas, como por exemplo, "Nota é Alta". O

antecedente representa um conjunto de condições que, quando satisfeitas (mesmo parcialmente), determinam o processamento do consequente da regra, através de um mecanismo de inferência. O consequente representa um conjunto de ações ou consequências (REZENDE, 2003).

O conjunto de regras linguísticas de um sistema nebuloso é comumente chamado de *Base de Regras*.

2.1.5 Sistemas de Inferência Nebulosos (SINs)

A partir das bases de dados e de regras, é possível a construção de sistemas para soluções de problemas de forma linguística ou simbólica, ou seja, sistemas que trabalham de forma mais semelhante ao raciocínio humano do que os sistemas que trabalham em termos numéricos, os quais recebem o nome de Sistemas de Inferência Nebulosos (SINs). Tais sistemas mapeiam entradas em saídas, podendo ser definidos como aproximadores de funções.

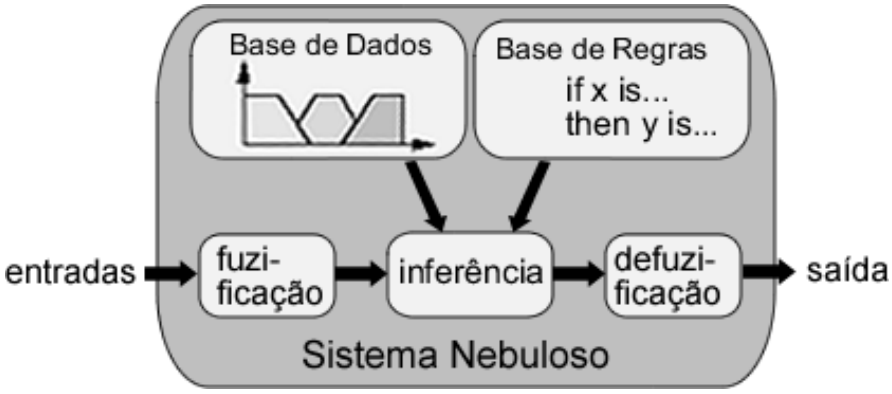


Figura 2.4: Diagrama típico de um Sistema de Inferência Nebuloso.

Conforme apresentado na figura 2.4, os SINs são constituídos pelos seguintes pontos, de acordo com Lee (1990a):

- Fuzificação: trata-se do processo de transformar os valores numéricos de entrada em valores linguísticos. Um dos métodos mais utilizados é a conversão direta das medidas em *singletons*. Caso os valores de entrada já sejam linguísticos, a fuzificação é descartada.

- Base de Dados: trata-se do conjunto de variáveis linguísticas, com os seus respectivos conjuntos nebulosos.

- Base de Regras: trata-se do conjunto de regras do sistema.

- Mecanismo de Inferência: trata-se da etapa onde são realizados os cálculos de inferência. De acordo com (JANG, 1992), os passos para a realização da inferência aproximada são:

  1. Compara-se os valores de entrada com as variáveis linguísticas dos antecedentes das regras contidas na base de regras do sistema, para assim obter um "grau de casamento parcial" entre cada dado da entrada e os antecedentes de cada regra;
  2. Combina-se as funções de pertinência não-nulas dos antecedentes, através de uma T-Norma, para assim obter o peso de disparo (*firing strength*) de cada regra;
  3. Gera-se o consequente de cada regra, de acordo com o seu respectivo peso calculado;
  4. Caso seja necessário uma saída numérica, os consequentes disparados são agregados para a defuzificação.

- Defuzificação: é onde os valores linguísticos são transformados em numéricos. Trata-se de uma etapa importante, dado que grande parte das aplicações envolvendo CF requerem resultados numéricos. De acordo com Lee (1990b), a saída deste processo pode ser definida como a melhor distribuição possibilística resultante de uma inferência nebulosa. Existem diferentes métodos de defuzificação, dentre os quais podem ser destacados (TSOUKALAS; UHRIG, 1997):

  – Centro de Gravidade: a saída numérica é calculada pelo centro geométrico da saída nebulosa, a qual é obtida através da união do resultado de todas as regras que envolvem uma determinada variável linguística de saída.

  – Centro das Somas: similar ao Centro de Gravidade, com a diferença de que as regiões sobrepostas da saída nebulosa são contadas mais de uma vez.

  – Média dos Máximos: é calculada uma média dos valores dos elementos da saída nebulosa que atingiram os seus valores máximos.

De acordo com Nguyen e Walker (1996), diferentes SINs podem ser modelados, de acordo com a variação dos seguintes parâmetros: os conjuntos nebulosos de entrada e saída; os operadores de T-Norma e T-Conorma utilizados; e o método de defuzificação escolhido. Em Cobo (2000), tais variações são agrupadas em três grandes categorias, apresentadas a seguir:

1. Mamdani: a saída nebulosa é obtida através de um operador de união entre os consequentes "ativos". Como operador de T-Norma, é utilizada apenas a função de mínimo (AND). Não importa o método de defuzificação empregado.

2. Tsukamoto: também similar ao Mamdani, mas com a restrição das funções de pertinência dos consequentes serem sempre monotônicas e não-decrescentes.

3. Sugeno: a saída de cada regra é uma combinação das variáveis de entradas. A saída final do sistema é calculada através da soma ponderada do peso de cada regra.

O Sistema de Inferência Nebuloso (SIN) representa o núcleo de diferentes aplicações de CF, como por exemplo quando aplicada em problemas de controle, recebendo o nome de *Controladores Lógicos Nebulosos (CLNs)*.

2.1.6 Vantagens e Desvantagens

A CF possibilita, através do conhecimento de um especialista sobre o comportamento de um sistema, definir conjuntos e regras nebulosas, realizando assim uma *Computação com Palavras* (ZADEH, 1996), que pode resultar em respostas nebulosas (qualitativas) ou categóricas (quantitativas).

De acordo com Rezende (2003), a CF pode contribuir com: uma teoria de lógica nebulosa bem-definida, mecanismos de raciocínio similar ao do ser humano (por meio do uso de termos linguísticos), acomodação de conhecimento de senso comum, conhecimento ambíguo e impreciso (mas racional), técnicas de aproximação universal, robustez e tolerância a falhas, e facilidade de desenvolvimento e manutenção.

Como desvantagem, a geração das regras nebulosas a partir do conhecimento do especialista é difícil; a definição das funções de pertinência é, em geral, baseada em uma avaliação subjetiva do especialista; e não existem técnicas de aprendizado dentro da égide da CF.

Trata-se de uma metodologia ideal para o tratamento de conceitos aproximados e de domínios cujos limites não são precisamente definidos (TSAKONAS; DOUNIAS, 2002). Como principais áreas de aplicação da CF, podemos destacar os sistemas de apoio à decisão e o controle de processos.

2.2 Computação Evolutiva (CE)

A Computação Evolutiva (CE) é uma metodologia na qual a teoria da evolução das espécies é adaptada para modelos computacionais. A idéia básica é manter uma população de *estruturas de conhecimento*, que vai evoluindo no decorrer do tempo, através de um processo de competição e variação controlada. Cada estrutura na população representa uma solução-candidata para o problema. Trata-se de uma metodologia de otimização, cujo processo produz soluções tão próximas da ótima quanto possível, dadas condições iniciais, restrições ambientais e parâmetros evolutivos corretos.

Várias abordagens para sistemas baseados na evolução foram propostas, tendo como principais diferenças os operadores genéticos utilizados e o tipo de codificação dos indivíduos. Dentre elas, podemos destacar:

- Estratégias Evolutivas: Rechenberg (1965);
- Programação Evolutiva: Fogel, Owens e Walsh (1966);
- Algoritmos Genéticos (AGs): Holland (1975) e
- Programação Genética (PG): Koza (1992).

Apesar de se tratarem de técnicas criadas por autores diferentes, em momentos diferentes e com focos diferentes, todas têm uma estrutura básica em comum: realizam reprodução, impõem variações aleatórias, promovem competição e executam seleção de indivíduos de uma dada população. De acordo com Zuben (2000), sempre que estes quatro processos estiverem presentes, seja na natureza ou em simulação computacional, a evolução é o produto resultante.

A seguir, serão feitas breves descrições da implementação genérica de um algoritmo evolutivo e de cada uma das técnicas citadas. Por fazerem parte do escopo deste trabalho, os Algoritmos Genéticos receberão uma melhor atenção do que as outras técnicas.

2.2.1 Algoritmos Evolutivos

Para descrever o processo evolutivo padrão, usa-se o termo *algoritmo evolutivo* como uma denominação comum a todas as técnicas. Os algoritmos evolutivos não são soluções "prontas para uso", mas sim uma lista de passos genéricos e adaptáveis para a solução de problemas, utilizados principalmente em problemas de otimização.

Os pontos principais de uma implementação evolutiva estão brevemente definidos, de forma genérica, a seguir:

- Indivíduo: candidato à solução do problema em questão, a sua codificação depende da técnica evolutiva escolhida e do problema a ser resolvido.
- Genes: cada atributo do indivíduo. Por exemplo, quando o indivíduo é codificado na forma de um vetor, cada posição do vetor é um gene; quando codificado na forma de árvore, cada nó da árvore é um gene.
- População - coleção de indivíduos que competem entre si pela sobrevivência.
- Aptidão do Indivíduo (*fitness*) - totalmente dependente do problema, define o quanto este candidato à solução está apto para solucionar o problema. Pode ser feita uma analogia com o papel do meio-ambiente na teoria da evolução das espécies: "só os indivíduos mais adequados/aptos/fortes sobrevivem".
- Seleção - independente de técnica e de domínio, a seleção é comumente realizada em duas etapas. Na primeira delas, indivíduos são selecionados a partir da geração atual para a produção de filhos, o que pode ser feito de diversas maneiras, conforme apresentado a seguir (OBITKO, 2007):
  - Roleta: Uma roleta imaginária é dividida em N parcelas, sendo N a soma da aptidão de todos os indivíduos. Assim, cada indivíduo ocupará o seu conjunto de parcelas da roleta, proporcionalmente ao seu valor de aptidão. Quanto maior for a aptidão, maior será o conjunto de parcelas ocupadas pelo indivíduo na roleta, e assim, maior será a probabilidade dele ser selecionado.
  - Posicionamento (Ranking): O indivíduo com pior aptidão recebe 1 parcela da roleta, o segundo pior recebe 2 parcelas, e assim por diante. Esse tipo de seleção é mais indicado que o método anterior em casos onde a distribuição de *fitness* seja muito desigual (ex: o melhor indivíduo tem 90% da soma de todas as aptidões). Nesse caso, utilizando o método da roleta, os demais indivíduos dificilmente seriam selecionados, o que geraria uma diversidade muito pequena da população. Como vantagem, todos os indivíduos tem uma chance maior de serem selecionados, entretanto a convergência se torna mais lenta, dado que os melhores indivíduos não diferem muito dos outros.
  - Torneio: Nesse método, "N" indivíduos são escolhidos aleatoriamente com a mesma probabilidade. Dentre os escolhidos, o que tiver maior aptidão é selecionado para ser "pai".

Ao final do algoritmo, uma nova seleção é feita, na qual escolhem-se quais indivíduos dentre os pais e os filhos sobreviverão para formar a nova população. Uma alternativa frequentemente utilizada para tal é chamada de elitismo, na qual os N melhores pais são sempre copiados para a próxima população, sendo o espaço restante preenchido pelos N melhores filhos. Outras alternativas podem ser definidas empiricamente, analisando-se o problema. Por exemplo: sobrevivem sempre os 50% melhores pais e os 50% melhores filhos são introduzidos como novos indivíduos na população.

- Operadores genéticos - dependentes da técnica evolutiva escolhida, tratam-se de operadores que atuam sobre os indivíduos para gerar a prole. Dentre os mais comuns, podemos destacar:
  - Cruzamento: cria novos indivíduos através da combinação dos atributos de um ou mais indivíduos "pais" previamente selecionados.
  - Cópia: como o próprio nome já diz, realiza uma cópia fiel do indivíduo da geração atual para a próxima. Muito utilizada na aplicação de elitismo, ou seja, os indivíduos mais aptos são copiados para a próxima geração para não haver perda do progresso obtido até então.
  - Mutação: cria novos indivíduos através de pequenas modificações de atributos em determinado indivíduo. Trata-se de um operador probabilístico que tenta introduzir características diferentes em populações, evitando que as soluções fiquem presas em mínimos locais.

O Algoritmo 1 representa uma típica implementação de algoritmo evolutivo. Normalmente a **população** de indivíduos $P(t) = \{x_1^t, ..., x_n^t\}$ é inicializada de forma aleatória, mas conhecimento específico do domínio do problema pode ser usado para facilitar a busca. Cada um dos indivíduos x_i^t é avaliado e produz alguma medida de **aptidão**. Para a formação de uma nova população (geração $t + 1$), indivíduos "pais" são **selecionados**. Alguns dos pais selecionados são simplesmente **copiados** (elitismo) para a próxima geração, enquanto outros sofrem alterações através dos operadores de **cruzamento** e **mutação**. Ao fim do ciclo existe outra seleção, a da sobrevivência (**substituição**), que define quais dentre os indivíduos da população t e da nova população criada sobreviverão para a próxima geração. Como **condição de parada**, as três opções mais utilizadas são: ou existe na população algum indivíduo que represente uma solução aceitável para o problema; ou o número máximo de gerações, previamente estipulado, foi atingido; ou o tempo máximo de execução, também previamente estipulado, foi atingido.

Algoritmo 1 Procedimento padrão de algoritmos evolutivos

```
início
   //inicializa o tempo/geração
   t := 0;
   //inicializa uma população de indivíduos aleatórios de acordo com o problema
   inicializa população P(t);
   //avalia a aptidão (fitness) de todos os indivíduos na população
   avalia P(t);
   //critério de parada pode ser tempo, número de gerações, fitness, etc.
   enquanto não ‘‘condição de parada’’ faça
      //aumenta o contador de tempo/gerações
      t := t + 1;
      //seleciona uma subpopulação para a produção de novos indivíduos
      P’ := seleciona pais P(t);
      //copia alguns pais selecionados para a prole (elitismo)
      cópia P’(t)
      //faz o cruzamento dos genes de alguns pais selecionados
      cruzamento P’(t);
      //adiciona perturbações à alguns da nova prole através da mutação
      mutação P’(t);
      //avalia a aptidão da nova prole
      avaliação P’(t);
      //seleciona os sobreviventes dentre a pop. pai e a pop. prole
      P := substituição (P(t),P’(t));
   fim enquanto
fim.
```

O desempenho e a eficiência de um algoritmo evolutivo está fortemente relacionado à definição de alguns parâmetros a serem utilizados, cujos valores devem ser estudados de acordo com as necessidades do problema e dos recursos disponíveis. Na maioria das vezes, esses parâmetros são definidos empiricamente e com base na análise do problema. Dentre eles, de acordo com Rezende (2003), é importante apresentar:

- Tamanho da População: afeta o desempenho global e a eficiência dos algoritmos evolutivos. Se a população é pequena, consequentemente a cobertura do espaço de busca por geração também é pequena, o que influencia muito no desempenho do algoritmo. Se a população é grande, fornecendo uma cobertura representativa sobre o domínio do problema, a possibilidade de acontecer uma convergência prematura para uma solução local em vez de global se torna muito pequena. Entretanto, quanto maior a população, mais recursos computacionais e temporais serão necessários. Tipicamente, a população é muito menor do que o universo a ser explorado.

- Taxa de Cruzamento: Quanto maior a probabilidade de realização de cruzamentos, mais diversidade será inserida na população. Mas se essa taxa for muito alta, indivíduos com boas características para a solução do problema poderão ser perdidos; e se for muito baixa, a busca pode estagnar. Dependendo do estágio em que se

encontra a busca, ou seja, dependendo da maturidade da população, tal taxa pode sofrer variações.

- Taxa de Mutação: Uma pequena probabilidade de realização de mutação já garante que a busca não se prenda em sub-regiões do espaço, possibilitando que qualquer ponto do espaço de busca seja atingido. Quanto maior a taxa de mutação, maior a aleatoriedade da busca. Assim como para o cruzamento, a taxa de mutação também pode variar de acordo com a maturidade da população.

É importante citar que, apesar das técnicas evolutivas possuírem essa estrutura comum, elas diferem em muitos detalhes. Por exemplo, existe uma grande variedade de mecanismos de seleção. A representação dos indivíduos pode ser desde vetores binários, até vetores com valores reais, árvores, máquinas de estado, etc. E os operadores genéticos de mutação e cruzamento, assim como as suas implementações particulares, se apresentam de muitas formas diferentes em cada uma das técnicas.

Dado que a seleção é um método genérico e um fator primordial de todo algoritmo evolutivo, torna-se necessária uma descrição mais abrangente das suas variantes mais utilizadas, o que será feito a seguir.

2.2.2 Estratégias Evolutivas (EEs)

As EEs surgiram a partir da publicação de Rechenberg (1965), tendo inicialmente como objetivo a solução de problemas hidrodinâmicos e de controle, sendo hoje em dia muito utilizadas por pesquisadores da Engenharia Civil, conforme Rezende (2003). Nesta técnica são empregados apenas operadores de mutação, o que acabou contribuindo muito em relação a análise e síntese destes operadores (ZUBEN, 2000).

As primeiras variantes de EEs não utilizavam o princípio de população, memorizando sempre apenas um ponto no espaço de busca (o "pai"), que poderia gerar um ou mais filhos, memorizando sempre apenas o melhor dos filhos para a próxima geração. Versões mais atuais empregam a população, sendo assim menos suscetíveis a convergências para mínimos locais, dado a maior diversidade (WIKIPEDIA, 2006a).

Trata-se da primeira técnica a explorar a teoria da evolução das espécies na solução de problemas. A partir dela foram derivadas a Programação Evolutiva, os Algoritmos Genéticos (AGs) e a Programação Genética (PG), que serão brevemente descritos a seguir.

2.2.3 Programação Evolutiva (PE)

Introduzida em Fogel, Owens e Walsh (1966), a PE foi originalmente desenvolvida com o objetivo de evoluir máquinas de estado finito. Apesar de ter evoluído esse tipo de estruturas originalmente, hoje em dia ela é empregada vastamente, não tendo uma estrutura de representação fixa.

De acordo com Fogel (1994), a PE hoje pode ser considerada como virtualmente equivalente às Estratégias Evolutivas em problemas de otimização, apresentando apenas algumas pequenas diferenças no que se diz respeito aos procedimentos de codificação e seleção dos indivíduos.

O seu principal operador é a mutação. Os indivíduos de uma população são vistos como indivíduos de diferentes espécies, ao contrário das outras técnicas evolutivas. Assim, cada pai, através da mutação, gera apenas um filho (WIKIPEDIA, 2006b).

2.2.4 Algoritmos Genéticos (AGs)

Os Algoritmos Genéticos (AGs) foram introduzidos em Holland (1975), tendo como objetivo inicial a formalização matemática e a explicação rigorosa dos processos de adaptação em sistemas naturais, e o desenvolvimento de sistemas artificiais que apresentassem os mecanismos originais encontrados em sistemas naturais (ZUBEN, 2000). Nesta técnica são empregados os operadores de cruzamento e mutação, que serão melhor apresentados na sequência.

É interessante notar que pesquisadores e usuários normalmente referem-se a "Algoritmos Genéticos" ou a "um Algoritmo Genético" e não "ao Algoritmo Genético". Isso se deve ao fato de que os AGs são uma classe de procedimentos com um conjunto de passos distintos e bem especificados, com cada um destes passos tendo inúmeras possíveis variações (REZENDE, 2003).

Apesar de existir uma grande quantidade de variantes de AGs, o seu mecanismo fundamental opera sobre a população de indivíduos (representando as possíveis soluções para o problema) e consiste de três operações básicas: avaliação da aptidão de cada indivíduo, aplicação de operadores genéticos, e formação de uma nova população. Estas três operações são repetidas até que o objetivo seja atingido, o que pode ser determinado como a solução ótima, ou a melhor solução encontrada após N gerações, ou N segundos de execução.

Descrever a execução de um AG aqui seria o mesmo que copiar a execução padrão de um algoritmo evolutivo. O que muda são os pontos que diferem de técnica para técnica, como por exemplo, o que de fato representam os indivíduos, a codificação e a representação dos mesmos e os operadores empregados.

2.2.4.1 Codificação

A codificação binária foi utilizada nos trabalhos pioneiros de John Holland, e até hoje é a mais comum, por ser de fácil utilização, manipulação e análise. Teoricamente, é uma codificação independente de problema, pois uma vez encontrada a representação em vetores binários, as operações-padrão podem ser utilizadas, facilitando o seu emprego em diferentes classes de problemas (REZENDE, 2003). Entretanto, ela não é uma representação "natural" para muitos problemas, nos quais as transformações de fenótipo para genótipo e vice-versa são muito difíceis, como por exemplo problemas que possuam valores contínuos. Como o próprio nome já diz, na codificação binária cada indivíduo é uma lista de bits 0 ou 1.

A codificação por permutação é muito utilizada em problemas de ordenamento. Nela, cada indivíduo é uma lista de números que representam posições. Como exemplo, o tradicional problema do caixeiro-viajante. Nesse caso, cada número refere-se a uma cidade, e a ordem dos números refere-se à ordem com que o caixeiro-viajante visitará as cidades.

A codificação por valores pode ser utilizada em problemas nos quais é necessário representar, por exemplo, números reais, caracteres, ou quaisquer outros objetos. Em problemas desse tipo, a codificação binária é muito difícil. A desvantagem é que, ao utilizar este tipo de codificação, na maioria das vezes torna-se necessário o desenvolvimento de operadores de mutação e cruzamento específicos para o problema.

2.2.4.2 Operadores de Cruzamento (*Crossover*)

Totalmente dependentes da codificação empregada, existem vários tipos diferentes de operadores de cruzamento, sendo os mais comuns apresentados a seguir (OBITKO, 2007):

- Ponto de Cruzamento Único: uma posição dos indivíduos é aleatoriamente selecionada, e o filho é formado copiando os genes do primeiro pai, do início até o ponto de cruzamento; e do outro pai, do ponto de cruzamento até o fim. Ex: **1100.10** $\times$ *0101.01* = **1100***01*.

- Dois Pontos de Cruzamento: duas posições dos indivíduos são aleatoriamente selecionadas, e o filho é formado copiando os genes do primeiro pai, do início até o primeiro ponto de cruzamento; do outro pai, do primeiro até o segundo ponto de cruzamento; e do primeiro pai novamente, do segundo ponto de cruzamento até o fim. Ex: **11.00.10** $\times$ *01.01.01* = **11***01***10**.

- Uniforme: os genes são copiados aleatoriamente do primeiro e do segundo pai. Ex: **110010** $\times$ *010101* = **1***1***0***1***10**.

- Aritmético: é realizada uma operação aritmética qualquer entre os pais para gerar a prole. Ex: **110010** $\cap$ *010101* = 010000.

2.2.4.3 Operadores de Mutação

Também dependentes da codificação empregada, existem diferentes maneiras de se realizar a mutação, sendo alguns brevemente descritos a seguir:

- Troca de Valores: uma ou mais posições do indivíduo são escolhidas aleatoriamente, e os seus valores são trocados por outros valores pertencentes ao universo de discurso do problema.
  Ex: $U = \{x \in N\}$ 1 (2) 3 4 5 6 $\Rightarrow$ 1 27 3 4 5 6.

- Aritmética: uma ou mais posições do indivíduo são escolhidas aleatoriamente, e é realizada uma operação aritmética qualquer sobre os seus respectivos.
  Ex: 1 2 3 4 (5) 6 $\stackrel{+4}{\rightarrow}$ 1 2 3 4 9 6.

- Troca de Posições: duas posições do indivíduo são escolhidas aleatoriamente, e os seus valores são trocados.
  Ex: 1 (2) 3 4 (5) 6 $\Rightarrow$ 1 5 3 4 2 6.

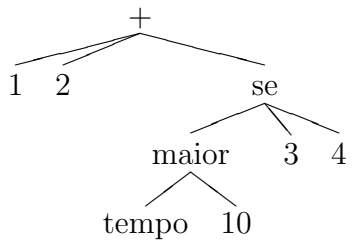
2.2.5 Programação Genética (PG)

A Programação Genética (PG) é a técnica mais recente sob a égide da CE, introduzida em Koza (1992). Trata-se de uma extensão dos AGs que tem por objetivo básico evoluir programas de computador ao invés de soluções codificadas, usando os mesmos princípios da evolução natural. Ou seja, os objetos que constituem a população não são vetores que codificam soluções para o problema em questão. Tratam-se de programas que, quando executados, "são" as soluções-candidatas para o problema.

Na PG, os indivíduos (programas) são codificados em árvores de tamanho variável. Por exemplo, um programa simples apresentado no Algoritmo 2 seria representado na forma da árvore apresentada logo abaixo.

Algoritmo 2 Algoritmo para exemplo de representação PG

```
se tempo maior 10
    retorna 1+2+3;
senão
    retorna 1+2+4;
fim se.
```



Com esse tipo de representação, têm-se como vantagem uma estrutura hierárquica, ao invés de vetores unidimensionais (AGs). Além disso, a estrutura é composta de simples funções, que podem ser facilmente codificadas usando uma linguagem de alto-nível.

2.2.6 Vantagens e Desvantagens

De acordo com Cordon, Herrera e Lozano (1995), os algoritmos evolutivos têm alcançado uma alta taxa de sucesso em problemas de busca e otimização. A razão de grande parte desse sucesso é a sua habilidade em explorar a informação acumulada sobre um espaço de busca inicialmente desconhecido, refinando a busca cada vez mais nos sub-espaços que se aproximam mais da solução ideal, sendo assim caracterizada a sua robustez. Essa é a sua principal característica, cuja vantagem pode ser vista principalmente em espaços de busca pouco conhecidos, grandes e complexos, para os quais as ferramentas clássicas de busca são inapropriadas, oferecendo assim uma alternativa válida para problemas que necessitam de uma busca eficiente.

Para muitos problemas de elevada complexidade, a obtenção da solução ótima por meio de técnicas convencionais se torna intratável, dado a quantidade inatingível de recursos computacionais e tempo necessários. Os algoritmos evolutivos nem sempre são capazes de atingir a solução ótima, mas na grande maioria das vezes conseguem obter soluções suficientemente boas em um tempo suficientemente pequeno, requerendo uma quantidade aceitável de recursos computacionais.

Além disso, implementações evolutivas são soluções adaptativas, ou seja, soluções capazes de manter o desempenho mesmo quando o ambiente é sujeito a pequenas alterações em suas especificações no decorrer do tempo. Para isso, não é necessário reiniciar todo o processo de busca de uma solução frente a pequenas mudanças nas especificações do problema, já que refinamentos podem ser obtidos a partir das soluções atuais.

Outrossim, a CE torna possível a incorporação de conhecimento especialista sem a necessidade de recorrer ao conhecimento humano expresso, por exemplo, através de uma base de regras. O sistema evolutivo pode adquirir proficiência em domínios especialistas simplesmente realizando ações e recebendo a realimentação acerca das consequências das ações tomadas, podendo aqui ser feita uma analogia com o aprendizado supervisionado da CN.

É importante citar também que a CE possui um paralelismo intrínseco, boa tolerância a ruídos e dados incompletos, flexibilidade para otimizar múltiplas funções (que podem possuir objetivos conflitantes) e, por fim, são facilmente hibridizadas com outras técnicas e heurísticas (REZENDE, 2003).

Trata-se de uma metodologia ideal para tarefas de otimização em espaços de busca muito grandes, identificando soluções sub-ótimas de alta qualidade, sendo muito utilizada em domínios que sofrem de explosão combinatorial (TSAKONAS; DOUNIAS, 2002).

Em contra-partida, dependendo da aplicação, a CE pode apresentar limitações, tais como: o alto custo computacional na avaliação de cada indivíduo, a codificação complexa do fenótipo para o genótipo (e vice-versa), e a convergência prematura da população para ótimos locais (REZENDE, 2003).

2.3 Computação Neural (CN)

A Computação Neural (CN) pode ser descrita como um conjunto de estruturas computacionais que podem ser treinadas para aprender padrões a partir de exemplos, possuindo como principais características a adaptação e o aprendizado.

A sua técnica mais comum, batizada de Rede Neural Artificial (RNA), pode ser vista como um modelo matemático que imita as estruturas neurais biológicas, apresentando capacidade computacional adquirida por meio de aprendizado e generalização. O processamento da informação é feito de maneira paralela e distribuída, por meio de elementos processadores relativamente simples. Cada elemento processador corresponde a um neurônio artificial, ilustrado na figura 2.5.

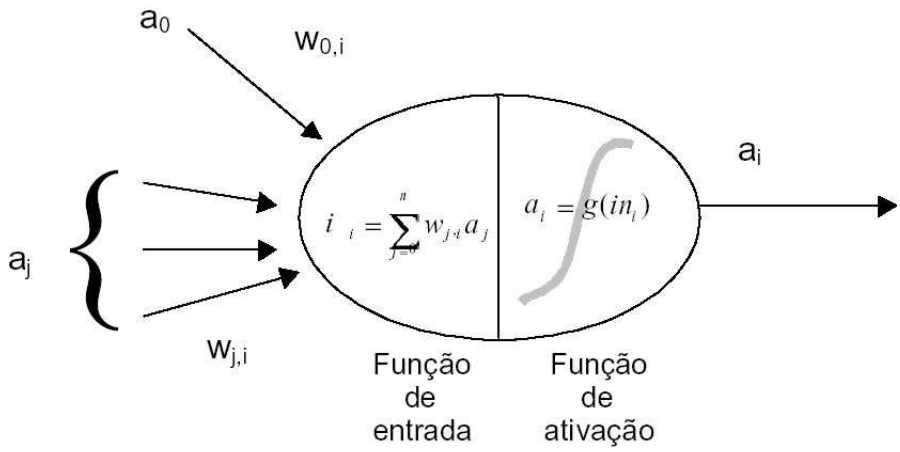


Figura 2.5: Modelo de neurônio artificial.

O aprendizado em RNAs está normalmente associado à capacidade das mesmas adaptarem os seus parâmetros como consequência da sua interação com o meio externo. O processo de aprendizado é interativo, e por meio dele a RNA deve melhorar o seu desempenho à medida que interage com o meio externo (REZENDE, 2003). Esse processo pode ser facilitado pela disponibilidade de um retorno parcial ou total. No caso de um retorno total (um supervisor disponibilizando uma avaliação para cada iteração ou um conjunto de treinamento descrevendo a saída correta para cada dado vetor de entrada), têm-se um *aprendizado supervisionado*. Quando é disponibilizado um retorno parcial (a cada intervalo x obtém-se um retorno de acerto ou erro), têm-se um *aprendizado por reforço*. Quando nenhum tipo de retorno é disponibilizado, têm-se o *aprendizado não-supervisionado* (BONISSONE, 1997).

A capacidade de generalização de uma RNA corresponde à sua capacidade de dar respostas coerentes para dados não apresentados a ela previamente durante o treinamento. O aprendizado e a generalização são etapas que andam juntas, ao invés da generalização surgir naturalmente como consequência do aprendizado.

O critério de desempenho que determina a qualidade do modelo neural e o ponto de parada de treinamento são pré-estabelecidos pelos parâmetros de treinamento e podem estar associados, por exemplo, ao valor do erro quadrático médio das respostas da RNA para um determinado conjunto de dados.

Como áreas de aplicação da CN, podemos destacar reconhecimento de caracteres, reconhecimento de voz, predição de séries temporais, modelagem de processos, controle, entre outras.

2.3.1 Vantagens e Desvantagens

De acordo com Rezende (2003), a CN oferece a possibilidade de aprendizado a partir de dados, modelagem empírica do comportamento humano, técnicas de aproximação universal, métodos de extração de conhecimento a partir dos dados, memórias associativas, técnicas de casamento de padrões, paralelismo massivo e robustez.

Apesar de todas essas vantagens, as RNAs exigem um longo tempo de treinamento, não possuem um mecanismo explicativo, nem um mecanismo automático e eficiente para auxiliar o desenvolvedor no projeto da rede.

De acordo com Tsakonas e Dounias (2002), trata-se de uma metodologia ideal para domínios cuja natureza seja puramente numérica.

2.4 Aplicações Híbridas

A hibridização consiste em misturar métodos inteligentes da Inteligência Computacional, fazendo com que os mesmos "trabalhem juntos". Como resultado, normalmente, se têm uma melhor solução para o problema, quando comparado com a utilização de um único método para o mesmo problema. A idéia principal é aproveitar as vantagens e as particularidades individuais das técnicas, aumentando assim a performance e a robustez dos sistemas que os empregam. A figura 2.6, reproduzida de Caversan e Andrade (2005), apresenta os principais tipos de aplicações híbridas envolvendo as metodologias de IC.

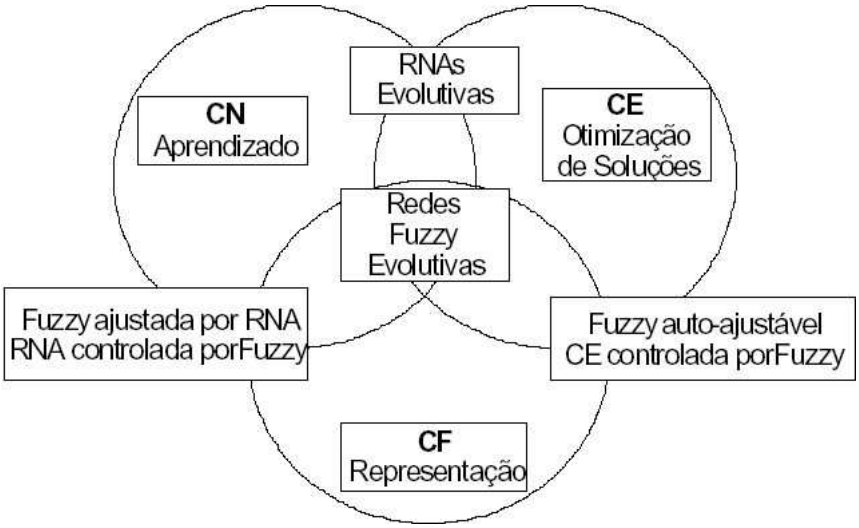


Figura 2.6: Principais tipos de aplicações híbridas utilizando IC.

Nos últimos anos têm-se visto uma utilização cada vez maior de sistemas híbridos, devido ao sucesso apresentado em muitos problemas complexos do mundo real. Com o

objetivo de melhor entender a razão pela qual as técnicas de IC apresentam bons resultados quando combinadas, é necessário observar suas vantagens e limitações. Cada uma destas técnicas tem uma contribuição a fazer na resolução de problemas, já brevemente citadas anteriormente neste texto.

2.5 Considerações Finais do Capítulo

As metodologias que constituem a Inteligência Computacional têm sido cada vez mais utilizadas em aplicações do mundo real, devido às suas principais características: o tratamento de dados imprecisos, incertos e a resolução de problemas em domínios nos quais o conhecimento ainda é incompleto.

Outra razão para a popularidade da sua utilização é a capacidade intrínseca que elas apresentam para trabalharem de forma conjunta, ou seja, para criar aplicações híbridas - sistemas que são baseados na integração ou trabalho conjunto de diferentes técnicas (BONISSONE et al., 1999).

Particularmente, aplicações híbridas envolvendo CF e CE têm sido utilizadas na solução de muitos problemas. A CE é aplicada na busca, aprendizado e otimização de componentes da CF; enquanto a CF é comumente empregada na utilização da experiência de especialistas para o gerenciamento dos parâmetros de controle da CE. Extensa revisão a respeito dessa utilização conjunta de técnicas podem ser vistas em Tsakonas e Dounias (2002) e Bonissone (2000).

Trabalhos demonstrando essa grande sinergia entre as duas metodologias são fatores motivantes para a busca por transformações entre elas. A constatação e a consolidação de transformações entre elas resulta em uma mudança na modelagem de soluções que as utilizam, conforme poderá ser visto no decorrer deste texto.

Uma abordagem mais abrangente a respeito de CF pode ser encontrada em Nguyen e Walker (1996). Já Goldberg (1989) é uma referência que descreve detalhadamente diferentes variantes de CE. Definições acerca dos diversos tipos de sistemas híbridos envolvendo técnicas de IC podem ser encontradas em Goonatilake e Khebbal (1994).

3 Modelos para a Busca por Relações

Neste capítulo é feita uma breve explicação sobre o Diagrama Gajski-Kuhn, a partir do qual foram propostos novos modelos a serem utilizados na busca pelas relações entre as técnicas de IC, com base no que foi proposto em Andrade (2002). O modelo atual, chamado de diagrama Andrade-Caversan-Fialho também é apresentado.

3.1 Diagrama Gajski-Kuhn (GK)

O Diagrama Gajski-Kuhn (GK), proposto em Gajski e Kuhn (1983) com o objetivo de manipular sistemas digitais em diferentes níveis de abstração, representa uma boa alternativa para a busca de relações, devido às suas características. Tal diagrama é composto por círculos concêntricos, que caracterizam níveis cuja abstração aumenta do círculo mais interno até o mais externo, conforme a figura 3.1.

Os seus 3 eixos constituintes podem ser descritos da seguinte forma, de acordo com Gajski e Kuhn (1983):

- Comportamental/Funcional: descreve o comportamento temporal e funcional de um sistema. Pode ser visto como o eixo algorítmico, com foco nas estruturas de controle necessárias para a sequência de manipulação de dados do sistema.
- Estrutural: todo sistema é composto por subsistemas. Neste eixo é representada a estrutura do modelo computacional, com os diferentes subsistemas e as suas interconexões, para cada nível de abstração. Pode ser visto como a ponte entre o eixo comportamental e o eixo físico.
- Físico/Geométrico: representa as propriedades físicas/geométricas do sistema e seus

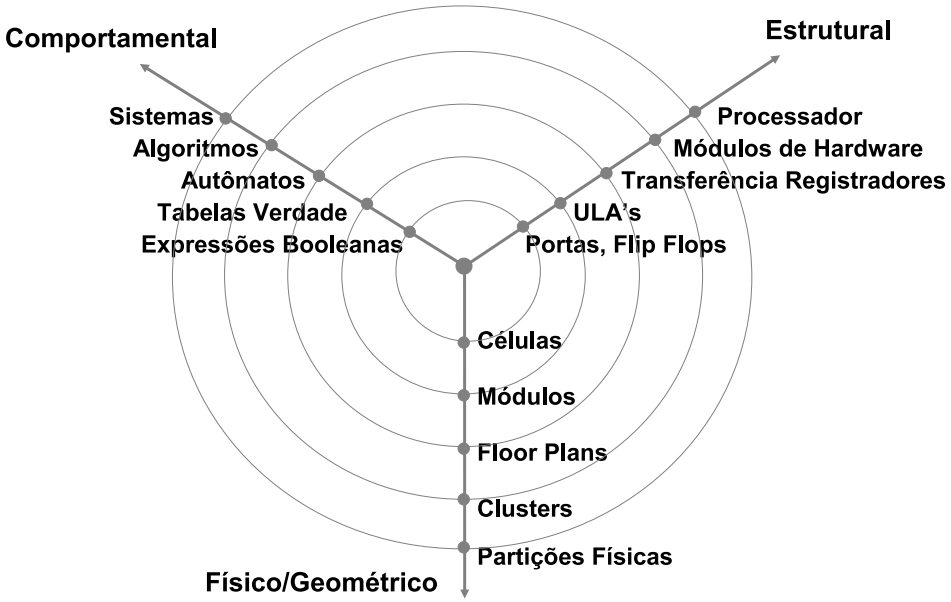


Figura 3.1: Diagrama GK, reproduzido de Wagner, Jansh-Pôrto e Weber (1988).

subsistemas, com informações sobre o tamanho, formato, posicionamento físico, etc.

As intersecções entre os eixos e os círculos são chamadas de *pontos notáveis*, e representam pontos discretos bem definidos dos sistemas digitais, servindo como base para definir os níveis de abstração equivalentes (CAVERSAN; ANDRADE, 2005). Quanto mais afastado do centro do diagrama, maior o nível de abstração do ponto. A definição dos pontos notáveis pode ser diferente, de acordo com o problema a ser solucionado, atentando apenas para o fato de que a idéia principal do Diagrama GK em relação ao conteúdo de cada eixo e aos níveis de abstração deve ser mantida.

Um dos fatores motivantes para a utilização deste diagrama nesta linha de pesquisa é a analogia que pode ser feita entre os seus eixos e as metodologias de IC, apresentada na figura 3.2, conforme Andrade (2002). Para essa correlação proposta entre os eixos do Diagrama GK e as técnicas de IC, foram consideradas as seguintes similaridades:

- Computação Nebulosa - Eixo Comportamental: trabalha com variáveis e regras linguísticas, que podem ser vistas como expressões do comportamento / funcionamento do sistema. Sistemas nebulosos podem ser representados como uma "caixa preta", onde o comportamento está exposto, mas a estrutura da plataforma computacional que o implementa está escondida.

- Computação Neural - Eixo Estrutural: trabalha com camadas de neurônios, tomando explicitamente como base a estrutura do cérebro humano, podendo assim ser associada ao eixo estrutural. Sistemas neurais podem ser representados como

uma "caixa branca", na qual a estrutura está exposta, mas o comportamento está escondido.

- Computação Evolutiva - Eixo Físico: trabalha com representações matemáticas das propriedades dos indivíduos, manipuladas através de operadores genéticos. As propriedades físicas são expostas, mas nada se sabe sobre a sua estrutura ou seu comportamento. Por esse motivo, os sistemas evolutivos podem ser denominados como uma "caixa cinza", e a técnica pode ser associada ao Eixo Físico.

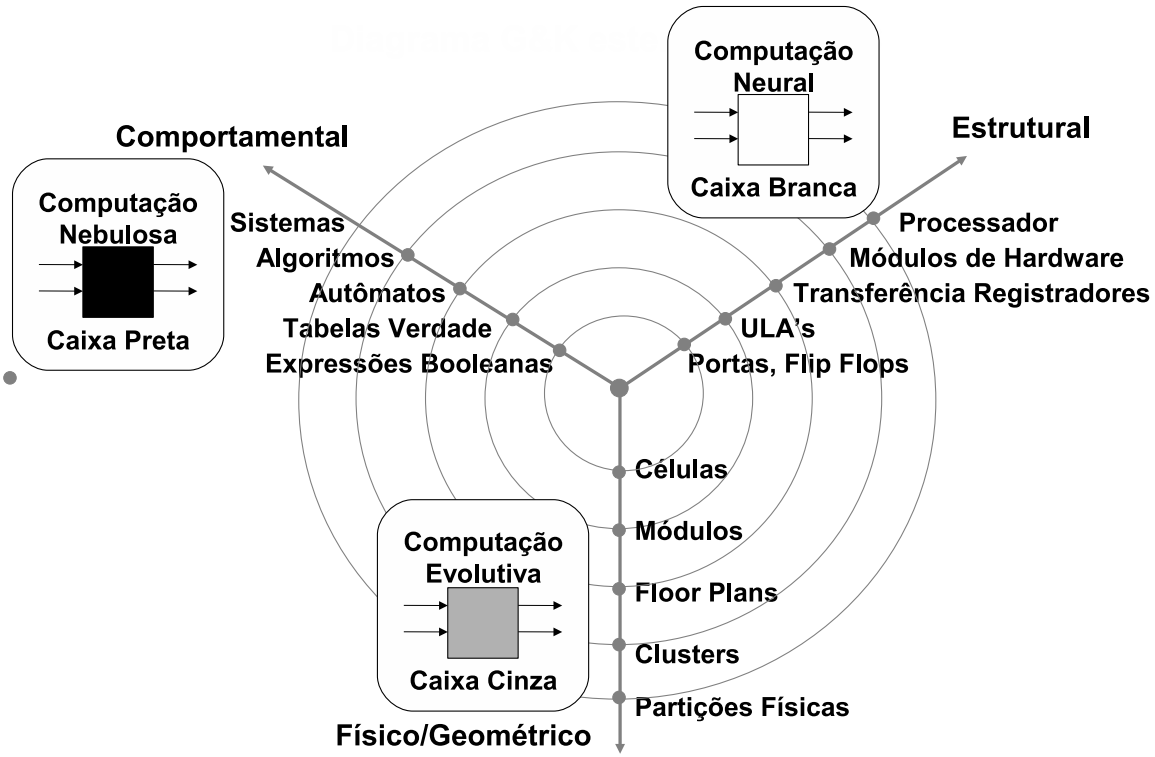


Figura 3.2: Eixos do Diagrama GK e as técnicas de IC.

Outro fator que motiva a sua utilização é a existência do conceito de transporte, que pode ser realizado entre eixos e pontos notáveis diferentes. Um exemplo desse transporte é a chamada "síntese funcional", que propõe transportar o projeto do eixo comportamental, representado com algoritmos, para níveis de abstração inferiores do próprio eixo comportamental, como máquinas de estados ou equações lógicas, ou ainda para o eixo estrutural, no nível de transferência de registradores ou portas lógicas (WAGNER; JANSH-PôRTO; WEBER, 1988).

Com cada eixo representando uma das metodologias de IC dessa linha de pesquisa, e com cada círculo concêntrico representando os diferentes níveis de abstração das metodologias, a definição das operações entre os pontos notáveis das técnicas de IC pode ser

vista como o tópico principal desta pesquisa, ou seja, essas operações seriam justamente as relações e transformações buscadas.

Para explorar as relações utilizando tal diagrama, a primeira idéia que se teve foi a busca por pontos notáveis que tivessem alguma analogia com os pontos notáveis originais. Durante o trabalho de Caversan e Andrade (2005), o primeiro eixo analisado foi o Comportamental, bem como a sua técnica correspondente, a CF.

Assim, o ponto "Sistemas" do eixo comportamental foi relacionado ao Sistema de Inferência Nebuloso (SIN); o ponto "Algoritmos" teve como representantes os Algoritmos Nebulosos; o ponto "Autômatos" também pôde ser diretamente relacionado com os Autômatos Nebulosos.

O ponto "Tabela-Verdade" foi relacionado diretamente com a tabela-verdade da CF, na qual é apresentado o conjunto de variáveis linguísticas de entrada e saída com os seus possíveis valores linguísticos. E o ponto "Expressões Booleanas" foi relacionado com as expressões nebulosas, propostas em Zadeh (1965), que possuem operadores e relacionamentos bem definidos.

Através da análise desse primeiro eixo notou-se que, ao tentar elencar apenas os pontos de IC que tivessem alguma relação com os pontos do diagrama original, muitos elementos de grande importância para a metodologia em questão não estariam representados no diagrama.

Por isso, abandonou-se essa idéia e partiu-se para a busca de pontos relacionados às metodologias de IC, sem se importar com as suas relações com os pontos notáveis originais do diagrama GK, o que levou à construção do diagrama Andrade-Caversan e, posteriormente, da sua extensão para o diagrama Andrade-Caversan-Fialho, que serão descritos na sequência.

3.2 Diagrama Andrade-Caversan (AC)

Em Caversan (2006), os principais fatores das metodologias foram elencados, sem a preocupação com a equivalência em relação aos pontos notáveis do diagrama GK, mas continuando com a proposta em 3 eixos. Estabeleceram-se os pontos notáveis dos eixos da CF e da CN. A CE não foi considerada, por não fazer parte do escopo daquele trabalho. Essa alternativa recebeu o nome de diagrama Andrade-Caversan (AC).

No caso da CF, foram considerados os seguintes elementos:

- Conjunto Nebuloso: trata-se do elemento primário para a criação dos sistemas nebulosos, que toma como base a teoria de conjuntos nebulosos.
- Modificador Nebuloso: é aplicado sobre um número nebuloso, modificando a sua função de pertinência e, por consequência, o seu comportamento.
- Variável Linguística: variável que assume valores de números nebulosos, utilizada na modelagem linguística dos sistemas nebulosos.
- Regra Linguística: regra necessária para a realização da inferência nebulosa, constituída de antecedentes e conseqüentes linguísticos no formato "SE antecedente(s) ENTÃO consequente(s)".
- Sistema de Inferência Nebuloso: trata-se do sistema nebuloso completo, com as capacidades de representação e processamento de conhecimento na forma linguística.
- Autômato Nebuloso: são autômatos que não trabalham com transições abruptas entre estados, e sim transições graduais, nebulosas (REYNERI, 1997), cujo estado não é bivalente, mas sim com diversos graus de pertinência diferentes.
- Algoritmo Nebuloso: com base na teoria de conjuntos nebulosos, são algoritmos criados com declarações linguísticas (ZADEH, 1968, 1973).

Mesmo a CN não fazendo parte do escopo deste trabalho, faz-se necessário descrever brevemente os pontos que foram estabelecidos para o seu eixo:

- Função de Transferência: é utilizada pelos neurônios, para gerar os seus sinais de saída.
- Peso Sináptico: atenua ou amplifica o valor de entrada do neurônio.
- Neurônio Artificial: elemento básico das Redes Neurais Artificiais, recebe um valor de entrada atenuado ou amplificado pelo Peso Sináptico, integra este valor através da Função de Transferência, e libera esse valor como saída.
- Camadas de Neurônios: tratam-se de conjuntos de neurônios que trabalham paralelamente, recebendo a entrada e passando a saída para as camadas subsequentes.
- Rede Neural Artificial: trata-se do sistema neural completo, que pode ter uma ou várias camadas, cada uma contendo um ou mais neurônios.

- RNAs Conectadas: tratam-se de conjuntos de RNAs interconectadas, que muitas vezes são utilizadas em problemas que podem ser divididos em sub-tarefas, ficando cada uma das redes responsável por resolver uma dessas sub-tarefas.

O trabalho apresentado em Caversan (2006) foi o primeiro desta linha de pesquisa. Além de propor o diagrama AC, conseguiu-se com sucesso extrair algumas transformadas por aproximação entre as metodologias citadas.

Através dos estudos de CF e CN, verificou-se que os conjuntos nebulosos influenciam nos centros das funções de ativação e no formato da função de transferência. Os centros das funções podem ser representados como parte dos pesos das entradas. Desse modo, os conjuntos nebulosos exercem influência sobre os pesos sinápticos e sobre as funções de transferência.

Um neurônio artificial pode avaliar o potencial de ativação de cada regra nebulosa, com as entradas, pesos e função de transferência do neurônio sendo modelados como o antecedente da regra. Dado que um SIN é formado por um conjunto de regras, e que estas podem ser representadas por neurônios, este pode ser modelado como uma RNA, com um neurônio para cada regra na primeira camada e uma camada adicional para o cálculo da defuzificação.

Esta possibilidade de conversão de um SIN para uma RNA foi explorada com sucesso na prática. O problema escolhido (navegação de robô) teve a sua solução primeiramente resolvida através de um SIN. Através das transformações aproximadas implementadas, do SIN para uma RNA, o problema também foi resolvido com sucesso. As transformações citadas podem ser vistas no diagrama AC apresentado na figura 3.3.

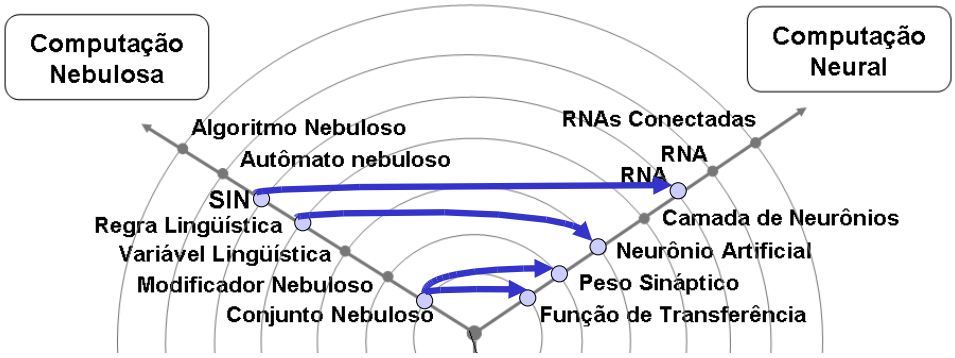


Figura 3.3: Transporte de elementos entre os eixos do Diagrama AC.

3.3 Diagrama Andrade-Caversan-Fialho (ACF)

Dadas as metodologias que compõem o objeto de estudo deste trabalho, fez-se necessário a extensão do diagrama AC, com a inclusão dos pontos notáveis da CE, e também de modificações no eixo da CF. O estágio atual do diagrama estendido, batizado de Andrade-Caversan-Fialho (ACF), pode ser visualizado na figura 3.4.

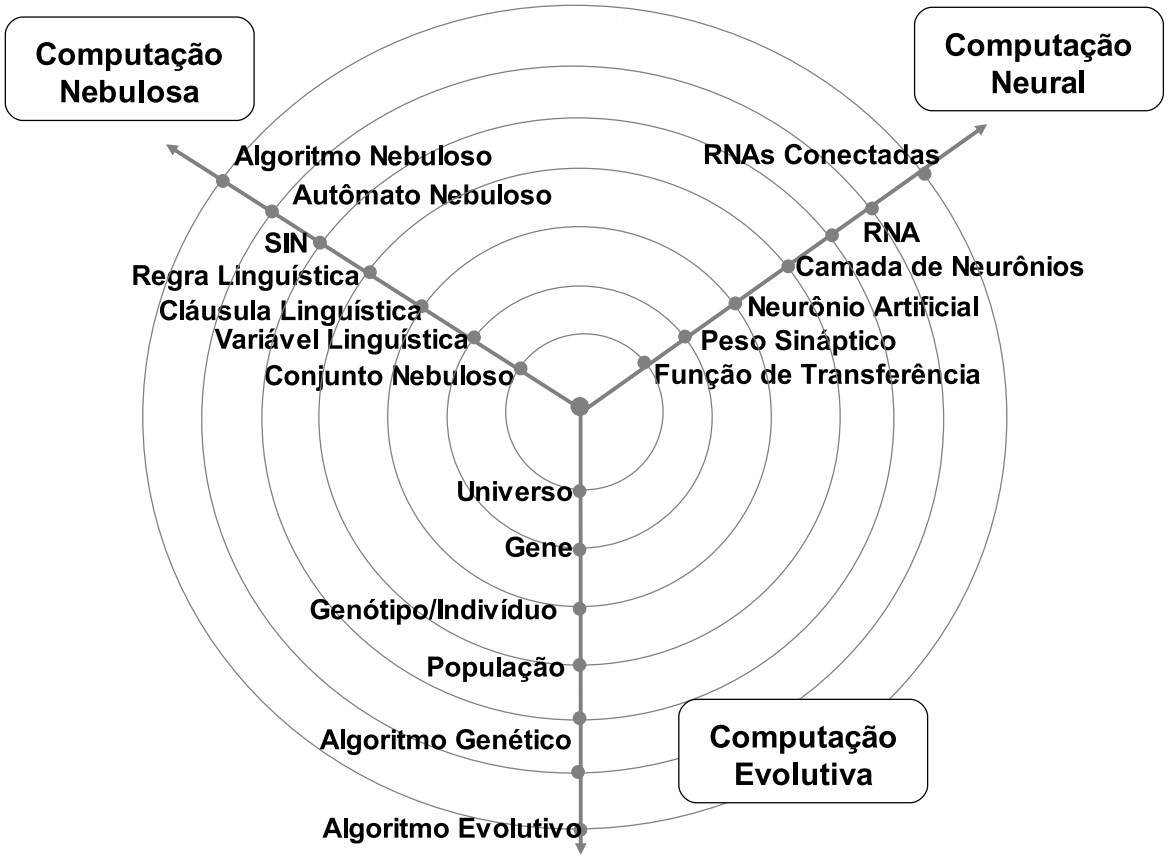


Figura 3.4: Estágio atual do diagrama Andrade-Caversan-Fialho.

O fato de um ponto estar no mesmo nível de outro não reflete que existam relações e transformadas entre tais pontos. Ou seja, os pontos da CE foram posicionados valendo-se apenas dos seus respectivos níveis de abstração dentro da CE, sem tomar o cuidado de posicioná-los no mesmo nível de pontos de CF com os quais foram estudadas transformações.

No eixo da CF, as únicas alterações realizadas foram: o ponto "Modificador Nebuloso" foi retirado, por não terem sido evidenciadas transformações a partir do mesmo; e o ponto "Cláusula Linguística" foi inserido. Uma cláusula é uma associação de uma variável linguística com um dos seus conjuntos nebulosos, comumente na forma "X is A". O antecedente e o consequente de uma regra linguística é constituído de uma ou mais cláusulas linguísticas.

Para o eixo da CE, foram considerados os seguintes elementos:

- Universo: normalmente, cada gene pode receber um valor dentre uma escala de valores, ou seja, cada gene tem o seu universo de discurso. Exemplificando, um gene de uma codificação binária pode receber apenas os valores 0 ou 1, enquanto um gene de uma codificação que representa as distâncias entre cidades deve sempre receber valores maiores do que 0.
- Gene: trata-se do elemento primário de um AG, que pode representar diretamente uma característica do indivíduo, ou fazer parte de um conjunto de genes que defina uma característica.
- Genótipo/Indivíduo: trata-se do vetor de genes que representam o indivíduo/solução completa para o problema, o objeto a ser evoluído.
- População: trata-se do conjunto de indivíduos, o *pool* de soluções-candidatas.
- Algoritmo Genético: trata-se da variante de algoritmo evolutivo, já descrita na seção 2.2.4 deste texto.
- Algoritmo Evolutivo: representa qualquer algoritmo que utilize a teoria da evolução como fundamentação para a busca e otimização de soluções.

3.4 Considerações Finais do Capítulo

Dada a metodologia empírica desta pesquisa, um modelo faz-se necessário para direcionar a busca pelas transformações entre as metodologias de IC. A partir deste pode-se representar as técnicas e os seus elementos constituintes, e assim ter uma visão diferente a respeito das possíveis transformações entre os elementos. Em poucas palavras, tal modelo pode ser definido como um ferramental de investigação das transformações entre as metodologias.

O diagrama ACF foi o modelo utilizado para nortear esta pesquisa. A partir da análise do mesmo e do estudo das metodologias, hipóteses de transformações entre as técnicas puderam ser vislumbradas e posteriormente consolidadas, conforme apresentado no próximo capítulo.

4 Busca de Transformações entre CE e CF

Este capítulo tem como objetivo apresentar conceitos importantes para a busca por transformações realizada. Os caminhos que essa pesquisa já trilhou e que foram por algum motivo abandonados são brevemente apresentados, com o caminho utilizado na sequência do trabalho sendo também descrito. Além disso, algumas transformações entre CF e CE encontradas na literatura são descritas e analisadas.

4.1 Relações Isomórficas

A linha de pesquisa proposta em Andrade (2002), como um todo, objetiva encontrar relações isomórficas entre as técnicas de IC. Por esse motivo, faz-se necessário definir o conceito de isomorfismo.

Isomorfismos podem ser vistos como relações de equivalência e/ou transformação. Resumindo, afirmar que uma técnica X é isomórfica de uma técnica Y significa que cada componente da técnica X possui UM componente equivalente na técnica Y, e que existem funções de transformação de ida-e-volta entre os dois domínios, sendo sempre possível retornar ao domínio original, preservando-se a associação unívoca entre os elementos dos domínios considerados.

Um exemplo deste tipo de relação é comumente visto quando se deseja resolver um problema no domínio do tempo. Dado a alta complexidade das equações deste domínio, através da transformada de Laplace, transporta-se a solução para o domínio da frequência, no qual integrações e derivações tornam-se, respectivamente, multiplicações e divisões; resolve-se o problema neste último domínio; e então retorna-se com a resposta para o domínio original através da transformada inversa.

Outro exemplo conhecido de presença de isomorfismo está na relação entre a Teoria Clássica de Conjuntos, a Lógica Proposicional e a Álgebra de Boole. As equivalências entre as 3 teorias estão demonstradas na tabela a seguir, reproduzida de Andrade (2002).

Tabela 4.1: Exemplo de isomorfismo entre sistemas lógicos formais

Teoria de Conjuntos	Álgebra Booleana	Lógica Proposicional
$\wp(X)$	B	$\Im(V)$
$\cup$	$+$	$\vee$
$\cap$	$\cdot$	$\wedge$
$\sim$	$\sim$	$\sim$
X	1	1
$\emptyset$	0	0
$\subseteq$	$\leq$	$\Rightarrow$

A existência de isomorfismos entre as metodologias de IC, com as quais seja possível realizar uma transformação completa e transparente de ida e volta de um domínio para outro, é o objetivo maior desta linha de pesquisa. Com relações desse tipo, seria possível transportar toda uma solução modelada em CE, por exemplo, para o domínio da CF, resolvê-la utilizando CF, e então trazer a resposta de volta para o domínio da CE, caso necessário.

A princípio não existem evidências suficientes que suportem a existência dessa transformação completa de ida-e-volta entre as técnicas em questão. Para chegar a esse nível de formalismo, primeiramente é preciso tentar definir relações e transformações ponto-a-ponto, partindo dos componentes básicos, subindo o nível de abstração gradativamente, buscando, dessa forma, atingir o nível de sistema completo. É nessa busca pontual que se concentrou a realização deste trabalho, de acordo com as abordagens que serão citadas a seguir.

4.2 Substituição de Técnica

A CF possibilita lidar, principalmente, com problemas de classificação, modelagem, controle e representação de conhecimento. A CN também possui as características de classificação e controle. Por existir estas intersecções entre as áreas de atuação das duas metodologias, torna-se possível o estudo das relações entre as mesmas através da solução de um único problema utilizando cada uma delas de forma independente.

Em Caversan (2006), o problema da navegação de robôs autoguiados foi solucionado utilizando CF, depois o mesmo problema foi solucionado utilizando CN e, a partir do

estudo das suas soluções, foi proposto e executado com sucesso um algoritmo de transformação automática aproximada da solução em CF (Sistema de Inferência Nebuloso) para a solução em CN (Rede Neural Artificial).

Dado o sucesso obtido no trabalho citado, para o estudo das relações entre CE e CF, essa foi a primeira abordagem a ser levantada. Uma busca por técnicas de ambas as metodologias que apresentassem a mesma natureza, ou seja, que pudessem resolver o mesmo tipo de problemas, foi realizada, conforme apresentado a seguir.

4.2.1 CF Para Aprendizado

Inicialmente, tentou-se trazer a CF para o domínio de aprendizado/busca pela função da CE, através da utilização de alguma técnica de CF que apresentasse a característica de aprendizado. Propostos em Procyk e Mamdani (1979), os Controladores Nebulosos Auto-Organizáveis (CNAOs) são controladores/classificadores nebulosos que possuem a capacidade de criar e/ou alterar o seu comportamento de acordo com um retorno provido.

Apesar dos artigos seminais de Zadeh (1965, 1973) constituírem a base teórica dos CNAOs, verificou-se que a CF é utilizada apenas na construção do controlador. Os mecanismos propostos para o aprendizado do mesmo "não fazem uso da noção de conjuntos nebulosos e, em teoria, poderiam ser utilizados com qualquer tipo de controlador", conforme Procyk e Mamdani (1979). Ou seja, o aprendizado não tem nada de CF, apenas o controlador é nebuloso.

Como conclusão, com esta alternativa, não estaria sendo relacionada uma técnica de CE com uma técnica de CF, mas sim com uma outra técnica, a qual pode ser definida como uma camada que provê aprendizado à CF, fugindo do escopo do trabalho.

4.2.2 CE Para Classificação

A segunda tentativa foi o caminho inverso: encontrar alguma técnica de CE que apresentasse as capacidades de classificação/controle da CF. Propostos em Holland et al. (1986), os Sistemas Classificadores Genéticos chamaram a atenção não só pelo nome, mas também por um de seus criadores, o "pai" dos AGs - John Holland. Em suma, tratam-se de sistemas classificadores com capacidade de aprendizado.

Entretanto, assim como na tentativa anterior, o que foi verificado é que a CE é utilizada nestes sistemas apenas para a sua geração e/ou manutenção. O classificador/controlador de fato não tem nada de evolutivo, podendo inclusive ser nebuloso.

Assim, da mesma forma, não estaria sendo relacionada uma técnica de CF com uma técnica de CE, mas sim com um sistema classificador que apresenta como provedor de aprendizado a CE, mas que, durante a sua função de classificação/controle, não pode ser considerado evolutivo.

4.2.3 Análise Crítica

Como conclusão, ainda não foram evidenciadas intersecções entre as áreas de atuação que contemplam as técnicas de CF e CE. Tratam-se de metodologias originalmente propostas com naturezas muito distintas e, a princípio, não seria possível realizar a solução de um único problema utilizando cada uma das técnicas.

Algumas das interseções entre as áreas de atuação das metodologias de IC podem ser visualizadas na figura 4.1. A região de interseção entre CE e CF permanece vazia neste momento, assim como a área comum entre as 3 metodologias.

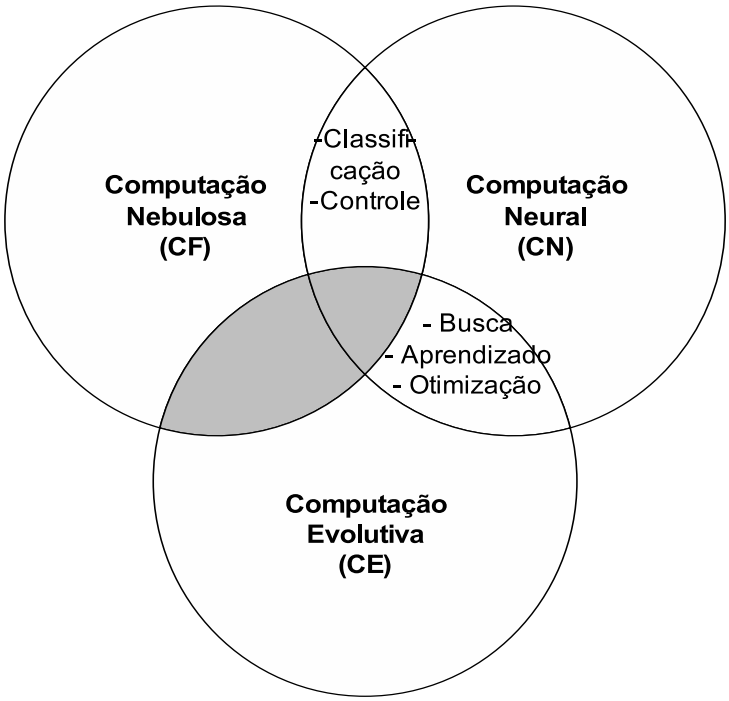


Figura 4.1: Algumas intersecções entre as áreas de atuação das metodologias de IC.

É interessante observar que existe a possibilidade de se estudar relações entre a CE e a Computação Neural (CN) desta forma, dado que a CN também apresenta as características de otimização/busca pela função em sua fase de treinamento. Esta terceira sub-pesquisa da linha está sendo executada por um grupo de estudo do Laboratório de Engenharia de Conhecimento da Escola Politécnica da Universidade de São Paulo (KNOMA, 2007).

4.3 Sistemas Híbridos

Dado que não é possível uma metodologia substituir a outra na solução de determinado problema, uma alternativa encontrada para o estudo das relações foi a análise de sistemas que utilizam características de uma das metodologias para cobrir falhas da outra, ou para acrescentar suas características à outra. Esse tipo de sistema comumente é classificado como híbrido, mas trata-se de uma classificação subjetiva. Os "híbridos" buscados por esta pesquisa vão além de simples combinações, buscando a integração do poder de abstração da CF e da capacidade de inovação da CE através de transformações pontuais entre as técnicas.

Uma das combinações mais utilizadas envolvendo CE e CF é a utilização da primeira para adicionar a característica de aprendizado à segunda. Como exemplo de aplicação, pode-se citar o controle nebuloso de um robô móvel. Dado que o ambiente do robô pode sofrer variações no decorrer do tempo, e dada a frequente imprecisão ou incompletude das informações providas pelos sensores, a modelagem manual de um Controlador Lógico Nebuloso (CLN) robusto se torna uma tarefa difícil. Para esta tarefa, os AGs podem ser utilizados, substituindo os especialistas, sem necessitar de um modelo detalhado do mundo no qual o robô estará inserido.

Em Hoffmann e Pfister (1997) são apresentadas duas abordagens diferentes para se desenvolver/otimizar um CLN utilizando AGs. Utilizando esses exemplos, aliados à taxonomia de sistemas inteligentes híbridos apresentada em Goonatilake e Khebbal (1994), é possível distinguir quais sistemas híbridos são úteis para esta pesquisa e quais não o são, o que será apresentado a seguir.

4.3.1 Híbridos Inter-Comunicativos - Troca de Informações

Na primeira abordagem citada, o AG é utilizado para otimizar alguns parâmetros do CLN, como por exemplo, os fatores de escala dos conjuntos nebulosos e/ou os formatos das funções de pertinência. O AG é um componente externo que configura os parâmetros do CLN e recebe um retorno do mesmo, conforme ilustrado na figura 4.2.

Nesse caso, trata-se de um sistema híbrido que utiliza as duas metodologias de forma independente e cooperativa, apenas trocando informações entre elas, não realizando nenhum tipo de transformação, o que não traz nenhuma contribuição para a linha de pesquisa deste trabalho.

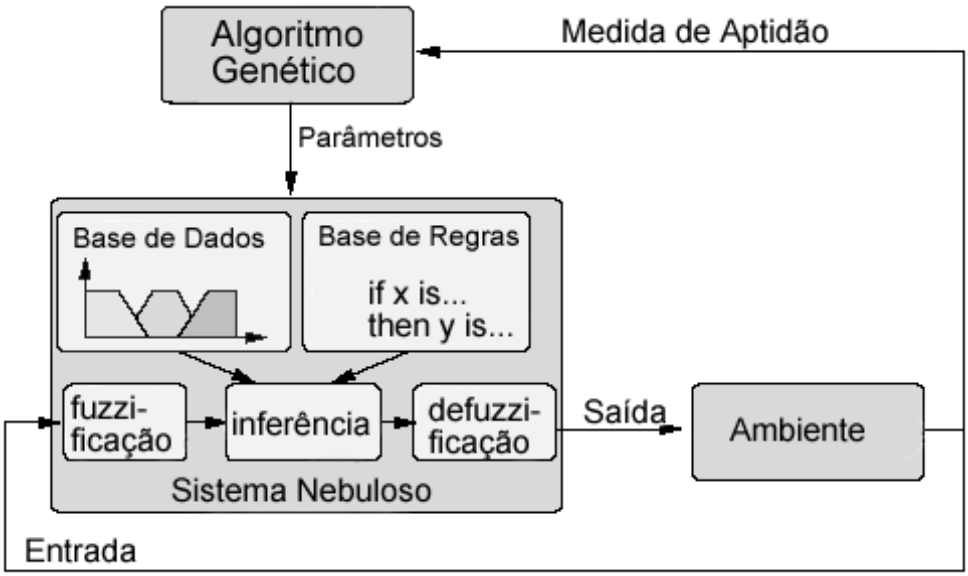


Figura 4.2: Troca de informações entre um AG e um CLN.

4.3.2 Híbridos Polimórficos - Transformação de Componentes

A segunda abordagem descrita em Hoffmann e Pfister (1997) é um exemplo do que esta linha de pesquisa busca. Ao invés de utilizar os componentes de forma independente, pontos de CF são codificados como indivíduos de CE. Nesse estudo é demonstrado que, através de diferentes técnicas de codificação, é possível transformar desde cláusulas linguísticas até mesmo uma base inteira de regras, na forma de elementos da CE.

Uma ilustração desta situação está exposta na figura 4.3, na qual uma Base de Conhecimento (Base de Dados + Base de Regras) é transformada em uma população de indivíduos evolutivos. O mecanismo de inferência nebulosa é anexado ao cálculo da medida de aptidão, integrando assim os poderes de ambas as metodologias. A partir do momento que a população atinge um nível de aptidão satisfatório, ela pode ser transportada de volta para o seu domínio original na forma de Base de Conhecimento.

De acordo com Rezende (2003), os Híbridos Polimórficos têm como motivação "descobrir novas funcionalidades de uma técnica e entender como diferentes técnicas podem se relacionar". Esse é o tipo de integração útil para esta linha de pesquisa, sendo o caminho seguido por este trabalho. A partir da análise das possibilidades no diagrama ACF e da revisão bibliográfica envolvendo esse tipo de aplicação híbrida, algumas transformação puderam ser elencadas, as quais serão apresentadas a seguir.

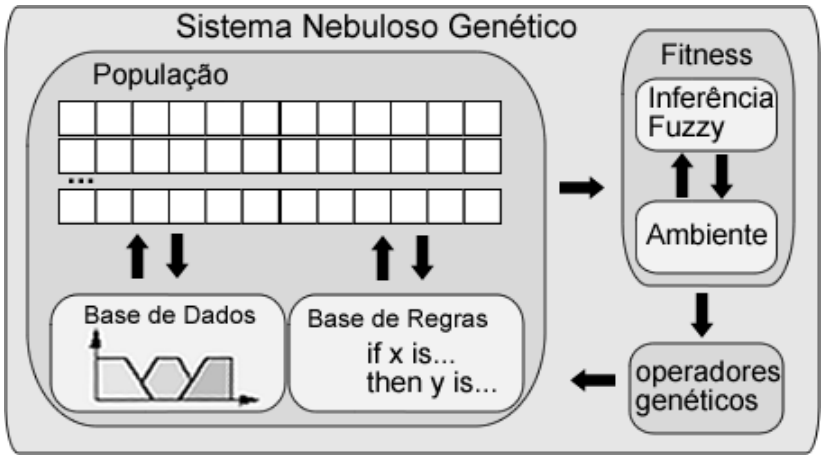


Figura 4.3: Sistema integrando CE e CF através de transformações.

4.4 Transformações entre CE e CF

Existem diversos artigos abrangendo a utilização conjunta das duas metodologias em questão, com uma gama deles apresentando transformações de diferentes elementos da CF para elementos da CE, com o objetivo de tê-los criados e/ou otimizados.

É importante destacar que, para a totalidade dos trabalhos revisados, não é possível avaliar no domínio da CE o quão aptos (*fitness*) são os indivíduos que representam elementos da CF. Para avaliá-los como solução para o problema, é necessário transportá-los de volta para o domínio nebuloso, para realizar o processo de inferência e dessa forma medir a diferença entre o comportamento obtido e o desejado. Ou seja, a existência da transformada de volta para o domínio da CF é essencial.

Diversos tipos de possíveis transformações serão brevemente descritos a seguir, com as transformações apresentadas nos trabalhos de Shi, Eberhart e Chen (1999) e Hoffmann e Pfister (1997) sendo analisadas de uma forma mais detalhada.

4.4.1 Base de Dados (BD)

Supondo uma situação em que as regras de um Controlador Lógico Nebuloso (CLN) podem ser facilmente extraídas de especialistas; porém, a obtenção das partições das variáveis linguísticas do problema em questão é muito difícil, devido à alta granularidade do problema ou outro motivo qualquer. Ou seja, a Base de Regras (BR) está acessível, mas a Base de Dados (BD) não.

Uma solução apropriada para este caso é a utilização de AGs para criar a BD ou aprimorá-la, caso uma versão inicial seja provida. Abrangendo ou não transformações,

tal abordagem é utilizada com sucesso em diversos trabalhos, como por exemplo, Karr e Gentry (1993), Herrera e Magdalena (1997), Cordon, Herrera e Jesus (2000), Shi, Eberhart e Chen (1999).

Envolvendo a criação e/ou aprimoramento da BD, o objeto a ser evoluído, com algumas hipóteses de transformações, pode ser:

- Um Conjunto Nebuloso
  - Transformação de Limites do Conjunto Nebuloso para Gene e Conjunto Nebuloso para Indivíduo - se o alvo é a criação ou o aprimoramento de um Conjunto Nebuloso, cada Gene pode representar os seus limites, com o Indivíduo consequentemente representando o Conjunto, e a População contendo o "pool" de candidatos a formato ótimo para tal Conjunto.
- Uma Variável Linguística
  - Transformação de Conjunto Nebuloso para Genes e Variável Linguística para Indivíduo - se o objeto a ser criado ou aprimorado é uma Variável Linguística, cada Conjunto Nebuloso pode ser representado por um conjunto de Genes, com o Indivíduo representando a Variável Linguística, e a População sendo o "pool" de candidatas à melhor constituição da Variável Linguística.
  - Transformação de Conjunto Nebuloso para Indivíduo e Variável Linguística para População - outra alternativa é cada Indivíduo representar um dos seus Conjuntos Nebulosos, com a População consistindo na própria Variável.
- A Base de Dados (BD)
  - Transformação de Conjunto Nebuloso para Genes, Variável Linguística para Indivíduo e BD para População - já se o objetivo é criar ou aprimorar uma BD inteira, um conjunto de Genes pode representar um Conjunto Nebuloso, a Variável Linguística pode ser transformada para a forma de Indivíduo, e a População pode representar toda a BD.
  - Transformação de Variável Linguística para Genes e BD para Indivíduo - Outra alternativa é a BD inteira ser transformada em um Indivíduo, com a população representando o "pool" de candidatas à Base de Dados.

4.4.1.1 Shi, Eberhart e Chen (1999)

Neste trabalho, com o objetivo de criar uma Base de Dados (BD) através do AG, a abordagem utilizada foi a última citada, na qual transforma-se o BD inteiro para a forma de Indivíduo, com a População representando vários candidatos à BD ótimo.

Para tal, deve-se assumir que o número de Variáveis Linguísticas de entrada e de saída são conhecidos, assim como o número de Conjuntos Nebulosos pertencentes a cada Variável e a faixa de valores (limites) de cada Variável. Ou seja, é sempre necessário um certo conhecimento *a priori* sobre o domínio do problema a ser resolvido.

Dado que tais números são fixos, o que é transformado de fato para a forma de Genes são os Conjuntos Nebulosos, com cada Conjunto sendo representado por três Genes: o ponto inicial x_1, o ponto final x_2 e o identificador (número inteiro) do tipo de função de pertinência do Conjunto Nebuloso em questão, com 6 possíveis formatos previamente definidos.

Dessa forma, cada indivíduo é um vetor de inteiros, com a quantidade de Genes definida pela soma do número de Conjuntos Nebulosos de cada Variável Linguística, multiplicado por três. Os Conjuntos Nebulosos são representados nesse vetor de acordo com a ordem das suas respectivas Variáveis Linguísticas no sistema e pela sua ordem dentro da Variável Linguística.

A transformação da CF para a CE, a princípio, não é realizada e nem mencionada, dado que a População inicial é criada baseando-se no conhecimento *a priori* mencionado. Mas a transformação da CE para a CF é realizada intensivamente durante o processo de evolução. A cada vez que o *fitness* de um Indivíduo precisa ser calculado, cada três Genes são transformados para a forma de um Conjunto Nebuloso, atribuído a sua respectiva Variável Linguística.

Tal transformação é realizada por equações definidas sobre x_1 e x_2, de acordo com o *tipo* da função de pertinência. Por exemplo, caso a função de pertinência de determinado Conjunto Nebuloso seja Triangular, os seus pontos constituintes são calculados de acordo com o conjunto de equações (4.1).

$$f_{triangle} = \begin{cases} 0, & \text{if } x < x_1, \\ 2\frac{x-x_1}{x_2-x_1}, & \text{if } x_1 \leq x \leq \frac{x_2+x_1}{2}, \\ 2\frac{x_2-x}{x_2-x_1}, & \text{if } \frac{x_2+x_1}{2} < x \leq x_2, \\ 1, & \text{if } x > x_2. \end{cases} \tag{4.1}$$

4.4.2 Base de Regras (BR)

Uma outra situação ideal para a aplicação de AGs na criação e otimização de elementos nebulosos seria o contrário da anterior, ou seja, a BD está acessível, mas a Base de Regras (BR) não, devido à ausência de especialistas ou outro motivo qualquer. Tal abordagem é utilizada com sucesso em diversos trabalhos, como por exemplo, Herrera e Magdalena (1997), Hoffmann e Pfister (1997), Shi, Eberhart e Chen (1999).

Envolvendo a criação e/ou aprimoramento da BR, o objeto a ser evoluído, com algumas possíveis transformações elencadas, pode ser:

- Uma Cláusula Linguística
  - Transformação de Variável Linguística com o seu Conjunto Nebuloso correspondente para a forma de Genes e de Cláusula Linguística para Indivíduo.
- Uma Regra Linguística
  - Transformação de Cláusula Linguística para Genes e Regra para Indivíduo.
  - Transformação de Cláusula para Indivíduo e Regra para População.
- A Base de Regras (BR)
  - Transformação de Cláusula Linguística para Genes, Regra para Indivíduo e BR para População - a abordagem de cada Indivíduo representar uma Regra é comumente conhecida como "Michigan-style" (HOLLAND; REITMAN, 1978).
  - Transformação de Regra para Genes e Base de Regras para Indivíduo - já a abordagem de cada Indivíduo representar uma BR completa recebe o nome de "Pittsburgh-style" (SMITH, 1980).

4.4.2.1 Shi, Eberhart e Chen (1999)

Neste trabalho, com o objetivo de criar uma Base de Regras (BR) através do AG, a abordagem utilizada foi a "Pittsburgh-style", com cada Indivíduo representando uma BR inteira, e com a População consequentemente contendo vários candidatos de BR para um problema de classificação.

Para tal, assume-se que o número de Variáveis Linguísticas de entrada e de saída são conhecidos, assim como o número de Conjuntos Nebulosos pertencentes a cada Variável. Ou seja, assim como na criação da BD, é necessário um conhecimento *a priori* sobre o domínio do problema. Além disso, mesmo não tendo idéia da quantidade de Regras ideal para o sistema, um número máximo também precisa ser assumido.

Cada Indivíduo é um vetor de inteiros, com a quantidade de Genes definida pelo número máximo de Regras, multiplicado pelo número de Variáveis Linguísticas (n_vars) do sistema. Cada Gene representa uma Cláusula, com cada n_vars Genes representando uma Regra. O i^{esimo} Gene contém o identificador do Conjunto Nebuloso associado à i^{esima} Variável Linguística do sistema. O sinal de negativo (-) representa a negação (operador NOT), enquanto que o "zero" é usado para representar a ausência da Variável Linguística na Regra.

Se a Base de Dados possui 3 Variáveis Linguísticas de entrada e 1 de saída, e cada Variável possui 5 Conjuntos Nebulosos associados, um exemplo de transformação de Regra para Indivíduo seria o seguinte:
"if input_var_2 is its_fuzzy_set_3 and input_var_1 is not its_fuzzy_set_5 then output_var_4 is its_fuzzy_set_4" $\Rightarrow$ *"-5; 3; 0; 4"*

Dado que a ordem original das Cláusulas não é preservada nessa transformação, não é possível representar os operadores entre as Cláusulas. Por conta disso, apenas Regras do tipo Mamdani são consideradas, de acordo com a taxonomia apresentada na seção 2.1.4.

4.4.2.2 Hoffmann e Pfister (1997)

Neste outro trabalho, com o mesmo objetivo e a mesma abordagem "Pittsburgh-style", um método de representação/transformação diferente é apresentado. Todas as Variáveis Linguísticas e os Conjuntos Nebulosos do sistema são etiquetadas com identificadores numéricos. Cada Cláusula é representada por um par de Genes: o identificador da Variável Linguística e o identificador do Conjunto Nebuloso associado.

Dessa forma, se a Base de Dados possui 3 Variáveis Linguísticas de entrada e 1 de

saída, e cada Variável possui 5 Conjuntos Nebulosos associados, um exemplo de transformação de Regra para Indivíduo seria o seguinte:
"if input_var_2 is its_fuzzy_set_3 and input_var_1 is not its_fuzzy_set_5 then output_var_4 is its_fuzzy_set_4" ⇒ *"2, 3; 1, -5; 4, 4"*

Em relação à abordagem de Shi, Eberhart e Chen (1999), algumas vantagens podem ser destacadas: as Variáveis Linguísticas ausentes na Regra não precisam ser representadas; e a ordem das Cláusulas é preservada, possibilitando (como uma contribuição ao trabalho) a inserção de mais um Gene representando o operador entre as Cláusulas.

Entretanto, esta representação utiliza o dobro de espaço em memória para armazenar cada Cláusula; e devido às Variáveis Linguísticas ausentes nas Regras, Indivíduos de tamanhos diferentes podem existir na População, o que resulta na necessidade de desenvolvimento de operadores genéticos mais complexos.

É importante destacar que em ambas as abordagens, apesar de previamente definido um número máximo de regras, é possível que ao transformar o Indivíduo para a forma de Base de Regras, tal Base contenha menos Regras do que o número estabelecido. Isso acontece devido a possibilidade de Regras impraticáveis serem geradas durante o processo evolutivo, como por exemplo, Regras sem nenhum antecedente ou consequente.

4.4.3 Base de Conhecimento (BC)

A partir das transformações citadas acima, abrangendo a Base de Dados (BD) e a Base de Regras (BR), pode-se afirmar que é possível transformar uma Base de Conhecimento completa (BD + BR) para a forma de um Indivíduo evolutivo, para que a mesma seja criada, aprimorada, etc.

Para alcançar este objetivo, basta realizar a concatenação das representações da BR e da BD já expostas. O trabalho apresentado em Shi, Eberhart e Chen (1999) também explora essa idéia.

Com essa lista de transformações propostas entre as duas metodologias, é possível atualizar o diagrama ACF, para representá-las, conforme pode ser visto na figura 4.4.

4.5 Considerações Finais do Capítulo

A CE e a CF são metodologias criadas para propósitos de natureza diferente, não possuindo intersecções entre suas áreas de atuação. A CF representa uma função, enquanto a

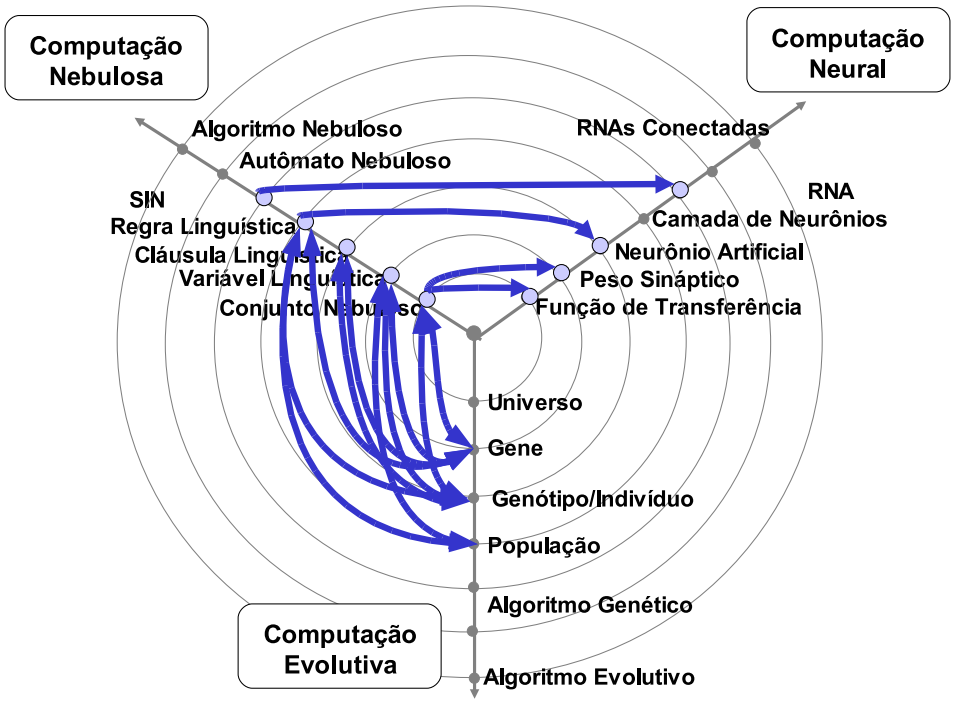


Figura 4.4: Transporte de elementos entre os eixos do Diagrama ACF.

CE realiza a busca pela função. Por conta disso, não é possível resolver um único problema utilizando ambas as metodologias, assim como foi feito na pesquisa abrangendo CF e CN (CAVERSAN, 2006). Uma alternativa é a análise de sistemas híbridos que apresentem, em algum nível de abstração, transformações entre os dois domínios.

Alguns trabalhos envolvendo esse tipo de hibridismo foram encontrados e estudados. As transformações analisadas, que podem ser vistas como métodos de codificação para os AGs, envolvem basicamente quatro pontos notáveis do eixo da CF no diagrama ACF - Variáveis Linguísticas e seus respectivos Conjuntos Nebulosos (Base de Dados); e Regras Linguísticas com as suas respectivas Cláusulas Linguísticas (Base de Regras) - sendo transformados para os pontos Gene, Indivíduo e População da CE. A concatenação dos dois tipos de representação leva à uma transformação de mais alto nível, da Base de Conhecimento (BD + BR) em Indivíduo.

Apesar de todas as relações analisadas ocorrerem no nível da representação, e não abrangerem neste momento todos os pontos notáveis de ambas as metodologias, tratam-se de fato de transformações entre os domínios. Algumas delas foram consolidadas, por meio de uma plataforma computacional, a ser descrita no próximo capítulo. Nesta plataforma foram realizados experimentos computacionais, apresentados e analisados na sequência, demonstrando a aplicabilidade e a utilidade das mesmas.

5 Plataforma Computacional para Exploração das Relações

Neste capítulo é apresentada a plataforma computacional criada especificamente para a exploração de transformações entre as técnicas de Inteligência Computacional. A modelagem é descrita, os pacotes especificados são apresentados.

Para a realização dos experimentos computacionais, visando a validação das transformações analisadas no capítulo anterior, uma plataforma computacional foi construída. Tal construção iniciou-se no trabalho de Caversan (2006), que apresentou todos os passos de especificação simplificada de um software, conforme proposto em Pressman (1995), batizando-a de Unified Computational Intelligence Plataform (UCIP). Dada a necessidade de estratificação das metodologias, o paradigma de desenvolvimento orientado a objetos foi escolhido. Um protótipo englobando as técnicas de CF e CN foi especificado e implementado com sucesso. A parte evolutiva também foi especificada, mas sem o devido detalhamento, dado que esta não estava presente no escopo do trabalho citado. Tomando um caminho de desenvolvimento similar ao modelo espiral, a idéia para esta plataforma é a de ser útil durante toda a pesquisa na linha de IC, sendo aprimorada de forma incremental conforme a necessidade.

Este trabalho propõe uma extensão desta plataforma e, por isso, faz-se necessário que a especificação dos requisitos, propostas em Caversan (2006), seja resumidamente citada, o que será feito a seguir. Depois disso, as alterações e os incrementos propostos serão apresentados.

5.1 Especificação dos Requisitos

Com o objetivo principal de possibilitar experimentos envolvendo as metodologias de IC e as suas possíveis relações, a UCIP teve como metas intermediárias a criação de um

núcleo básico para cada uma das metodologias, atentando para o fato das suas respectivas construções serem realizadas de forma hierárquica e bem estratificada, para possibilitar também a modelagem de soluções utilizando apenas alguns dos seus componentes.

Com essa especificação, torna-se possível a construção de sistemas fiéis a cada uma das metodologias, assim como também sistemas híbridos, através da utilização de apenas algumas classes de cada uma das metodologias combinadas em prol de um resultado melhor. Além disso, é possível realizar experimentos de transporte de componentes de cada uma das técnicas para as outras.

Na figura 5.1 são apresentados os estados que uma instância da plataforma pode assumir, conforme a especificação proposta. Exemplificando, uma solução modelada na forma de uma técnica de CF pode ser transformada para uma técnica de CE ("CF > CE") ou de CN ("CF > CN"), caso tal transformação já tenha sido especificada e implementada.

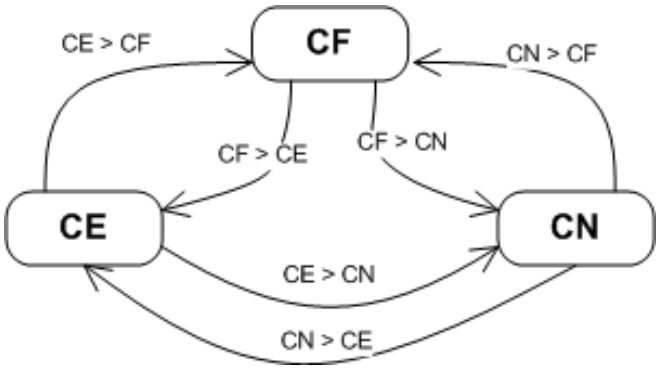


Figura 5.1: Diagrama de estados de uma instância da UCIP, reproduzido de Caversan (2006).

Para a criação da UCIP, definiu-se que os componentes devem ser especificados de acordo com os pontos notáveis propostos no diagrama ACF, podendo até mesmo serem divididos em classes menores em caso de necessidade, mas nunca agrupados em classes comuns, dado que isso causaria perda da característica de simulação e análise da plataforma.

Os pacotes disponibilizados pela plataforma, assim como o diagrama de casos de uso da mesma, são apresentados na figura 5.2. Intelligent Computational Model (ICM) foi o nome dado ao pacote que representa um modelo unificado de IC, do qual a CF, a CN e a CE são especializações; FC foi o nome utilizado para o pacote de Computação Nebulosa; NC para Computação Neural; EC para Computação Evolutiva; e MT para o pacote *Model Transformation*, no qual estão localizadas as rotinas de transporte.

O pacote ICM e as suas especializações devem possuir as funcionalidades de: cons-

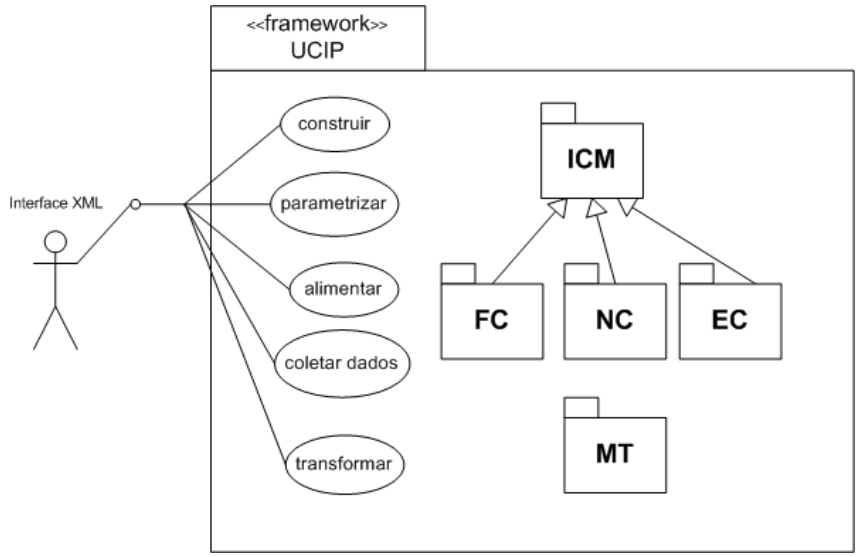


Figura 5.2: Diagrama de casos de uso e componentes da plataforma UCIP, reproduzido de Caversan (2006).

trução de um modelo funcional através de seus componentes; parametrização do modelo de acordo com a necessidade; alimentação através da entrada de dados; e coleta de informação pela saída de dados.

A outra funcionalidade presente no diagrama, "transformar", está presente no pacote Model Transformation, que têm por utilidade o transporte de um ou mais componentes de um caso particular do pacote ICM para outro. Para possibilitar o estudo de relações entre alguns pontos notáveis, é desejável que cada componente de um determinado modelo possa ser transportado independentemente.

Foram propostas duas abordagens diferentes para a utilização da plataforma. A primeira sugere que cada técnica suportada pela UCIP deve oferecer um protocolo que permita a sua utilização através de uma parametrização simples, ou seja, através de dados de entrada, saída e outros parâmetros necessários. Após parametrizado, o sistema receberá as entradas e fornecerá as saídas, possibilitando assim a integração com sistemas diferentes.

A segunda abordagem sugere a construção do modelo através de programação. Dado que a UCIP é composta de um conjunto de componentes, divididos hierarquicamente em classes, uma maneira de utilizá-la é através da instanciação, ou mesmo da extensão destes componentes. Trata-se da alternativa a ser utilizada para a análise de relações entre as técnicas, dado que isso precisa ser feito ponto (notável) a ponto.

5.2 Especificação do Projeto

Para a representação da parte estática do projeto, diagramas de classes foram propostos; e para representar a sua funcionalidade, foram utilizados diagramas de atividades, conforme sugerido em Booch, Rumbaugh e Jacobson (2000). Em prol da simplificação dos diagramas, apenas os elementos mais relevantes da modelagem foram apresentados.

Cada um dos pacotes citados na seção anterior foi descrito em Caversan (2006) através do seu respectivo modelo lógico, e dos diagramas de classe e de atividade. Não foram necessárias alterações na especificação do pacote da CF, dado que os pontos notáveis modificados no diagrama ACF já estavam presentes e discretizados no seu diagrama de classes. Dado que esse pacote foi utilizado na totalidade dos experimentos realizados, a sua descrição, juntamente com o seu modelo lógico e os seus diagramas de classe e de atividades, foram reproduzidos de Caversan (2006), com a autorização do autor, e anexados a este trabalho (Anexo A). O pacote NC não será descrito, por não fazer parte do escopo deste trabalho. A especificação do pacote EC foi toda refeita, com um maior nível de detalhamento, e será apresentada na sua forma atual. O pacote MT também será brevemente descrito. O pacote ICM será suprimido, dado que o atual estágio da linha de pesquisa ainda não contempla uma generalização das 3 metodologias.

5.2.1 Pacote *Evolutionary Computing* (EC)

Para a construção de um modelo evolutivo (*EvolutionaryModel*), conforme ilustrado na figura 5.3, é necessário um mecanismo para codificar (*Encoder*) os parâmetros da solução a ser evoluída em um indivíduo (*Individual*), uma solução-candidata para o problema, formada por genes (*Gene*), existindo casos em que cada gene representa diretamente uma característica da solução, e outros em que um conjunto de genes representa uma única característica.

Assim como na natureza, só os indivíduos mais aptos sobrevivem, e por isso faz-se necessário um mecanismo para o cálculo da aptidão (*FitnessScore*), que avalia o indivíduo como solução para o problema. O conjunto de indivíduos formam a população (*Population*), sobre a qual é exercida a seleção (*Selection*) dos pais que gerarão descendentes. Selecionados os pais, a prole é gerada através da atuação de operadores evolutivos (*Operator*), dentre os quais destacam-se a mutação (*Mutation*), o cruzamento (*CrossOver*), e a reprodução (*Reproduction*). Com os descendentes gerados, é necessário definir a população que passará para a próxima geração, através da substituição (*Replacement*) de

indivíduos menos aptos da população atual pelos filhos mais aptos gerados. Atingido o critério de parada, os genes são decodificados (*Decoder*) para os parâmetros da solução, ou para a solução propriamente dita.

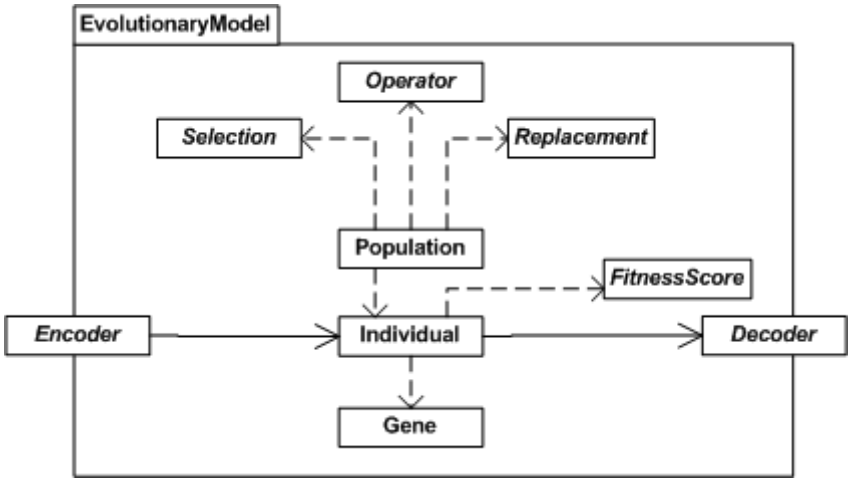


Figura 5.3: Modelo lógico do pacote para realização de CE

O diagrama de classes proposto para esta metodologia é apresentado na figura 5.4. Os diferentes métodos de codificação, seleção e substituição, e as variantes dos operadores de mutação e cruzamento, presentes no diagrama, são os mais comumente encontrados na literatura referente a Algoritmos Genéticos (AGs). É importante atentar para o fato de que, conforme a necessidade, é muito simples adicionar um novo método de codificação, seleção ou substituição; ou uma nova variante de operador de mutação ou de cruzamento; ou até mesmo um novo operador evolutivo pode ser criado; sendo necessário alterar apenas o componente em questão.

A execução de um algoritmo evolutivo ocorre conforme descrito na seção 2.2.1, o que pode ser visualizado na figura 5.5, na forma de um diagrama de atividades.

Assim como o pacote FC foi validado, durante o trabalho de Caversan (2006), através da solução do problema de navegação do robô, para obter-se a certeza da corretude do modelo proposto para a CE, o pacote foi implementado, e dois problemas foram resolvidos utilizando AGs: o problema das N-Rainhas (*N-Queens*) e o problema do caixeiro-viajante (*TSP*). Ambos são classificados como problemas Não-Polinomiais, ou seja, problemas que não apresentam soluções determinísticas que sejam executadas em tempo polinomial, restando como opção a utilização de algoritmos não determinísticos, como por exemplo, as variantes da CE. Além disso, ambos apresentam um espaço enorme de soluções, e uma função de aptidão facilmente calculada, o que representam mais características típicas de problemas solucionados utilizando CE.

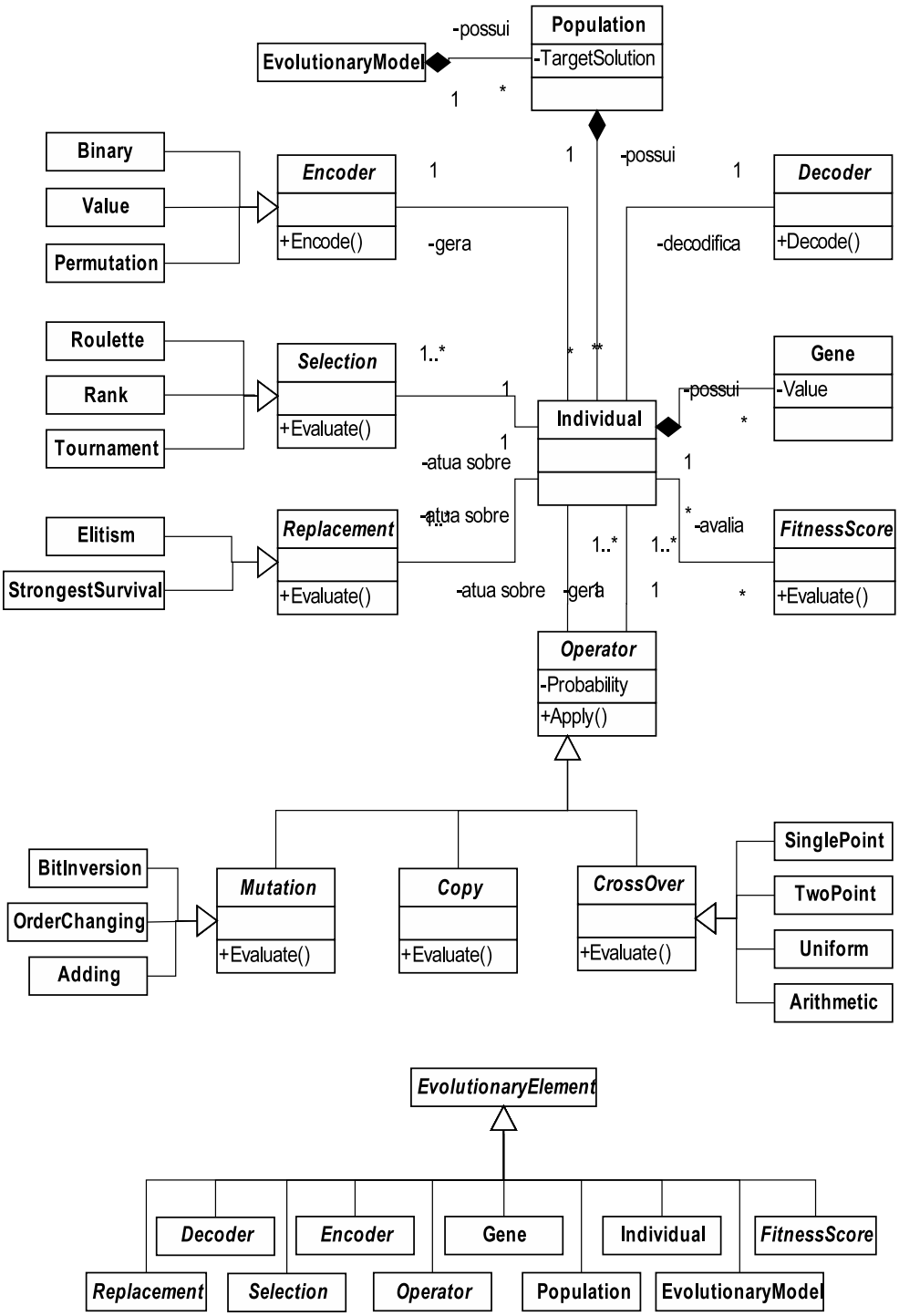


Figura 5.4: Diagrama de classes do pacote para realização de CE

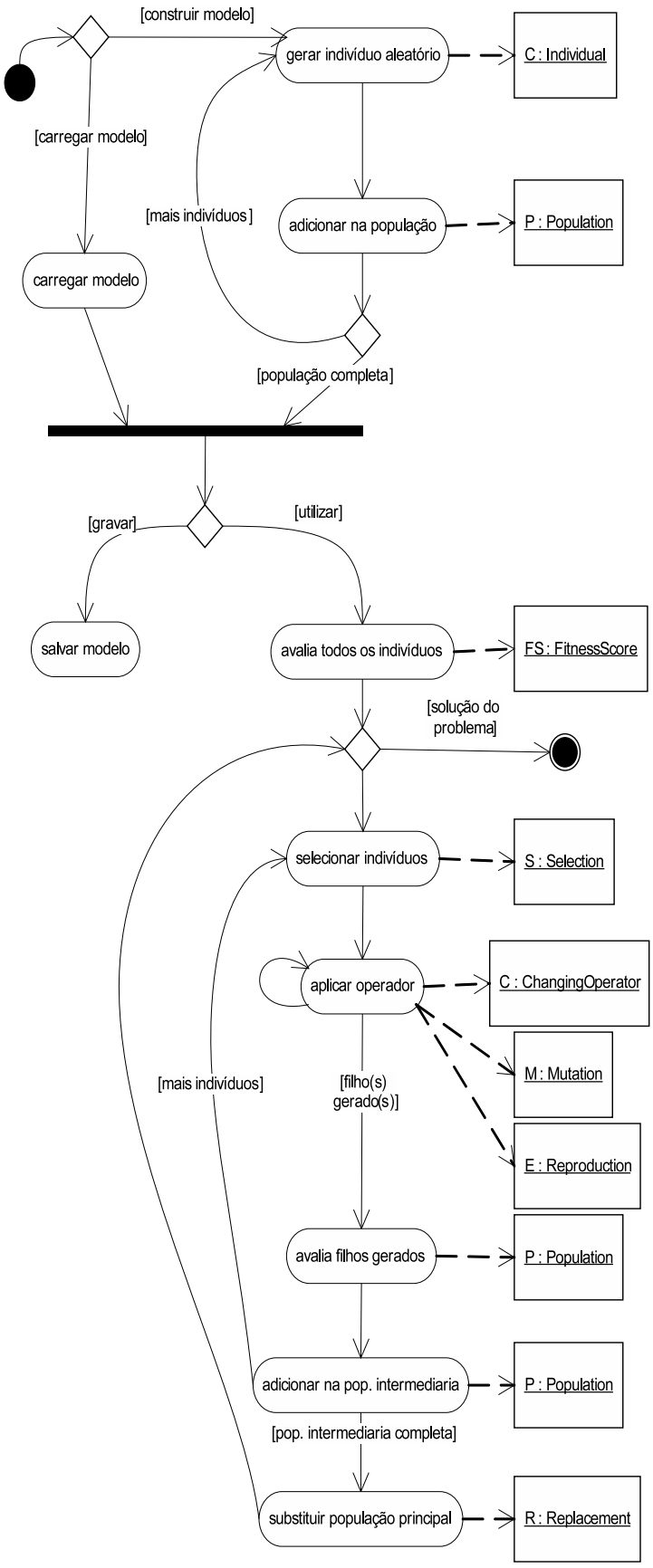


Figura 5.5: Diagrama de atividades representando a operação com CE

O problema das N-Rainhas necessita posicionar n rainhas em um tabuleiro de tamanho n por n, de tal forma que nenhuma rainha possa atacar a outra. As rainhas podem realizar os movimentos das regras do jogo de xadrez, ou seja, podem se mover para cima, baixo, esquerda, direita e diagonais. Dessa forma, a solução ótima não pode ter nenhuma rainha na mesma linha, coluna ou diagonal de outra. Baseado em Crawford (1992), utilizou-se uma codificação na qual cada indivíduo possui n genes, com o valor do gene i representando a linha em que se encontra a rainha na coluna i do tabuleiro, conforme exemplo na figura 5.6.

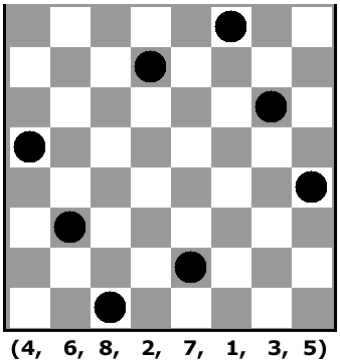


Figura 5.6: Exemplo de tabuleiro e da sua respectiva codificação evolutiva.

O problema do caixeiro-viajante talvez seja o mais velho e melhor estudado problema na área chamada de otimização combinatória, de acordo com Potvin (1996). Em poucas palavras, um caixeiro-viajante (*traveling salesman*) precisa passar por n cidades, sem repetir nenhuma cidade, retornando ao final para a sua cidade de origem. O problema consiste em encontrar a menor rota possível para a viagem do caixeiro. A codificação utilizada foi a de permutação: cada cidade é identificada por um número, e a ordem com que os números estão dispostos nos genes do indivíduo representa a ordem com que as cidades serão visitadas.

5.2.2 Pacote *Model Transformation* (MT)

Um pacote para realizar as transformações os três domínios da IC, batizado de *Model Transformation* (MT), foi proposto em Caversan (2006), com o seu diagrama de classes exposto na figura 5.7.

Tal pacote possui as classes abstratas *NeuralTransform, FuzzyTransform, EvolutionaryTransform*, além da classe ancestral comum *ModelTransform*. Essas classes são utilizadas para a criação de outras classes específicas para as transformações propostas. Classes implementadas a partir da classe abstrata *EvolutionaryTransform*, por exemplo,

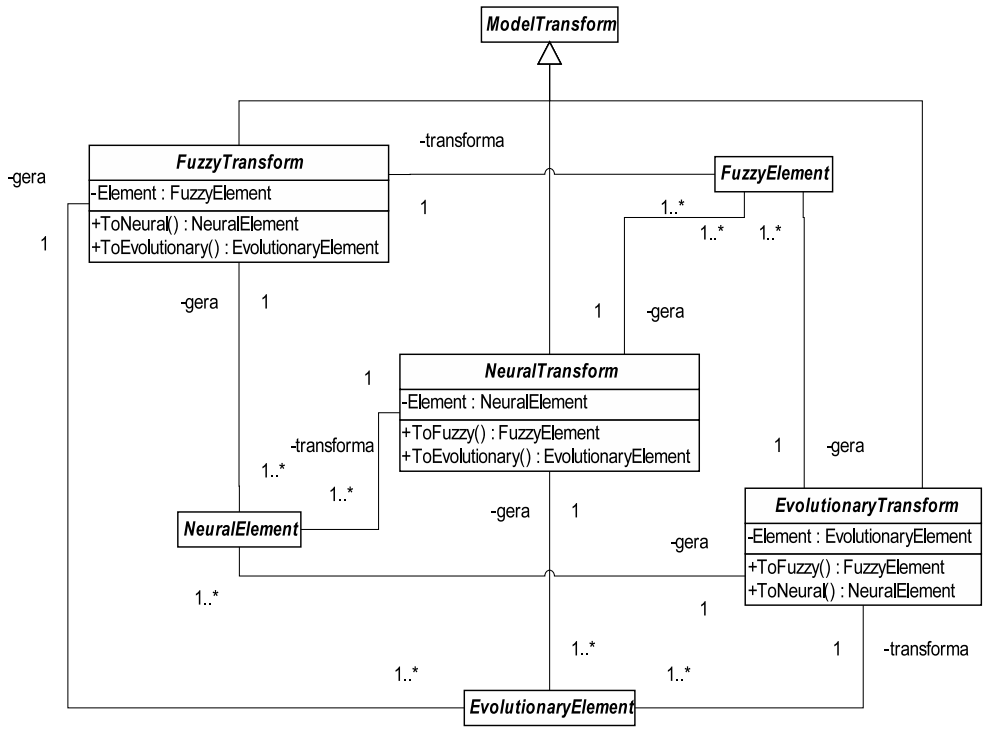


Figura 5.7: Diagrama de classes do pacote MT, adaptado de Caversan (2006).

podem abranger transformações de elementos evolutivos para elementos neurais (*ToNeural*) ou para elementos nebulosos (*ToFuzzy*).

5.3 Considerações Finais do Capítulo

Com a especificação de cada uma das metodologias propostas bem divididas em classes, com cada componente fundamental ou ponto notável em uma classe diferente, a plataforma UCIP possibilita analisar cada ponto de forma isolada ou mesmo o sistema todo e, o mais importante para esta pesquisa, simular as transformações entre componentes de diferentes classes e/ou de diferentes metodologias.

A modelagem original do pacote EC foi estendida e detalhada. Apesar do foco atual desse pacote serem os AGs, a especificação proposta serve para o desenvolvimento de soluções utilizando qualquer técnica evolutiva, bastando para tal a inserção dos tipos de codificação e operadores evolutivos empregados pela técnica.

Dois problemas foram solucionados utilizando-se implementações a partir do pacote EC. Além de se validar a corretude do pacote, colocou-se à prova também a característica de extensibilidade da plataforma, dado que para ambos os problemas, diferentes tipos de codificação, cruzamento e mutação tiveram que ser implementados.

6 Experimentos Computacionais

Do anteriormente exposto, constata-se que algumas das transformações descritas no capítulo anterior foram validadas. Esta validação deu-se pela realização de experimentos práticos, nos quais diferentes Bases de Conhecimento foram criadas e modificadas por meio de AGs, visando diferentes objetivos. Dado que este problema foi resolvido, são aqui apresentados os diferentes experimentos realizados e a análise argumentativa extensiva de seus resultados.

Para realizar os experimentos de transporte entre os dois domínios, a Base de Conhecimento (BC) foi escolhida como o elemento nebuloso a ser transformado para a forma de Indivíduo genético. Apesar de não aparecer como um ponto notável do diagrama ACF, tal elemento abrange os seguintes pontos: Regras e Cláusulas Linguísticas (Base de Regras); e Variáveis Linguísticas e Conjuntos Nebulosos (Base de Dados). Ou seja, ao consolidar as transformações envolvendo a BC, são consequentemente simulados e validados os transportes em todos os níveis de abstração citados. Como o próprio nome do elemento já diz, trata-se de todo o conhecimento disponível para a realização da inferência nebulosa.

A seguir será realizada uma breve descrição do problema atacado. Na sequência serão apresentadas as transformações implementadas junto aos experimentos computacionais realizados.

6.1 Navegação de Robôs Autoguiados

Atualmente, existem inúmeras áreas nas quais robôs têm se apresentado úteis, como por exemplo, na segurança e patrulhamento de residências e propriedades em geral; na supervisão de pessoas que necessitam de cuidados especiais, como idosos e doentes; na localização e até mesmo no resgate de pessoas em escombros e locais de difícil acesso; no acesso e exploração de locais onde o ser humano não poderia chegar (planetas muito

distantes, vulcões, etc); e nas mais variadas tarefas do dia-a-dia, como cortar a grama do jardim, entre outras.

Para um robô realizar algumas dessas tarefas, um dos requisitos mais importantes é a sua navegação autônoma, ou seja, ele deve se locomover sem a intervenção humana em ambientes para os quais ele não foi especificamente construído (SAFFIOTTI, 1997).

Tais ambientes são normalmente cheios de incertezas e imprecisões, as quais podem ser intrínsecas aos sensores do robô, ou advindas de condições externas (iluminação, ruídos, etc). Por esse motivo, quando a precisão não é de fato uma prioridade, trata-se de um problema de controle tipicamente nebuloso.

Assim sendo, em Caversan (2006), esse foi o problema escolhido para a implementação e validação do pacote FC, para o qual utilizou-se um Controlador Lógico Nebuloso (CLN) para controlar um robô reativo autoguiado. Foi criado um ambiente no qual o comportamento do robô pôde ser simulado, tendo como comportamento-objetivo a sua locomoção em um terreno desconhecido evitando colisões com os obstáculos.

Para a simulação, foi definido que o robô estaria sozinho no ambiente, sem nenhum outro tipo de objeto se movimentando. O robô simulado foi modelado com três sensores, medindo a distância entre a sua posição atual e os obstáculos, sendo um para a frente (F), outro para a esquerda (*left* - L) e outro para a direita (*right* - R). Essas são as entradas do sistema, que possui uma única saída, o ângulo de alteração na direção do robô (θ), o qual pode variar de -40° to $+40^o$. A sua velocidade é sempre constante. Essas restrições visaram uma simplificação dos mecanismos de sensoreamento e inferência, dado que inicialmente o objetivo principal era apenas validar o pacote FC. Posteriormente, esse mesmo problema foi empregado na consolidação dos transportes entre CF e CN. Uma ilustração do mesmo pode ser vista na figura 6.1.

Neste trabalho, a navegação de robôs autoguiados também foi o problema utilizado na condução dos experimentos entre CF e CE, conforme apresentado a seguir.

6.2 Criação e Aprimoramento de Bases de Conhecimento (BC)

A CF pura não possui nenhum mecanismo de aprendizado ou aprimoramento. Ou seja, dado que têm-se um problema para o qual deseja-se aplicar uma solução nebulosa, é necessária a presença de um especialista para a especificação da BC sobre o domínio do problema. Não havendo um especialista disponível, uma solução possível é a utilização de

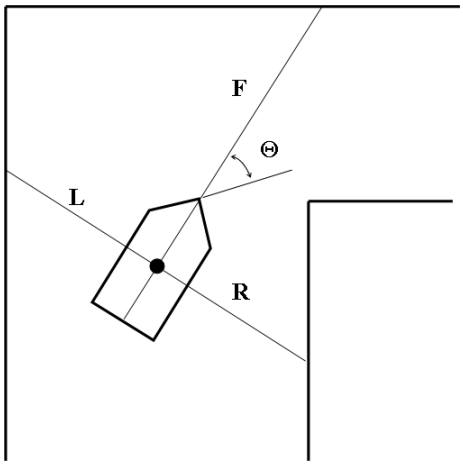


Figura 6.1: Robô autoguiado com os seus sensores (F, L e R) e a saída (θ), reproduzido de Caversan (2006).

outras técnicas para prover aprendizado à CF, como por exemplo, os Algoritmos Genéticos (AGs).

Com base nisso, como uma aplicação para a validação das transformações analisadas, foi proposta a implementação de um algoritmo capaz de criar e aprimorar uma BC utilizando AGs. Os experimentos realizados tiveram como foco o problema da navegação de robôs autoguiados, mas abordagem desenvolvida pode ser utilizada na modelagem de qualquer tipo de classificador ou controlador nebuloso.

Para a realização desta tarefa, faz-se necessária a realização intensa de transportes da CE para a CF. As transformações implementadas foram adaptadas a partir da abordagem descrita em Shi, Eberhart e Chen (1999) (ver seção 4.4). Para atingir o nível de abstração da BC, várias transformações pontuais mais elementares tiveram que ser implementadas, as quais serão apresentadas a seguir.

Dado que em Caversan (2006) é apresentada uma solução de CF que resolve de forma satisfatória o problema de navegação do robô, na qual a BC foi extraída de especialistas, um conhecimento a respeito da solução pôde ser aproveitado: o número de variáveis linguísticas de entrada e de saída, o número de conjuntos nebulosos de cada variável linguística e o número de regras do sistema. Nos experimentos de aprimoramento foi possível aproveitar também toda a BC dessa solução conhecida, inserindo-na na população inicial do processo evolutivo. Com isso, pôde-se validar também transformações da CF para a CE.

A seguir, serão apresentadas algumas adaptações feitas em relação à abordagem utilizada como base, serão descritos detalhes pertinentes ao Algoritmo Genético aplicado à solução proposta, e na sequência serão apresentadas as transformações implementadas

juntamente aos experimentos realizados.

6.2.1 Ambiente de Simulação

Em Caversan (2006), um ambiente de simulação para o prolema de navegação do robô autoguiado foi desenvolvido. Através desse ambiente, é possível criar toda uma Base de Conhecimento (BC) para o Controlador Lógico Nebuloso utilizado no direcionamento do robô, simular o comportamento do robô através da utilização dessa BC, salvar a BC criada em um arquivo XML, ou mesmo carregar uma BC já existente. O mapa no qual o robô deve se locomover pode ser desenhado à mão-livre ou carregado a partir de um arquivo BMP. Além disso, estão implementadas nesse ambiente as transformações entre CF e CN, desenvolvidas durante o trabalho citado. Uma explicação mais detalhada a respeito da sua versão original pode ser encontrada no mesmo.

Para a realização dos experimentos pertinentes à esta pesquisa, tal ambiente foi estendido. A versão atual da sua interface principal pode ser vista na figura 6.2. Toda a coluna mais à direita da interface, com o título "Evoluir Modelo Fuzzy", foi adicionada ao ambiente.

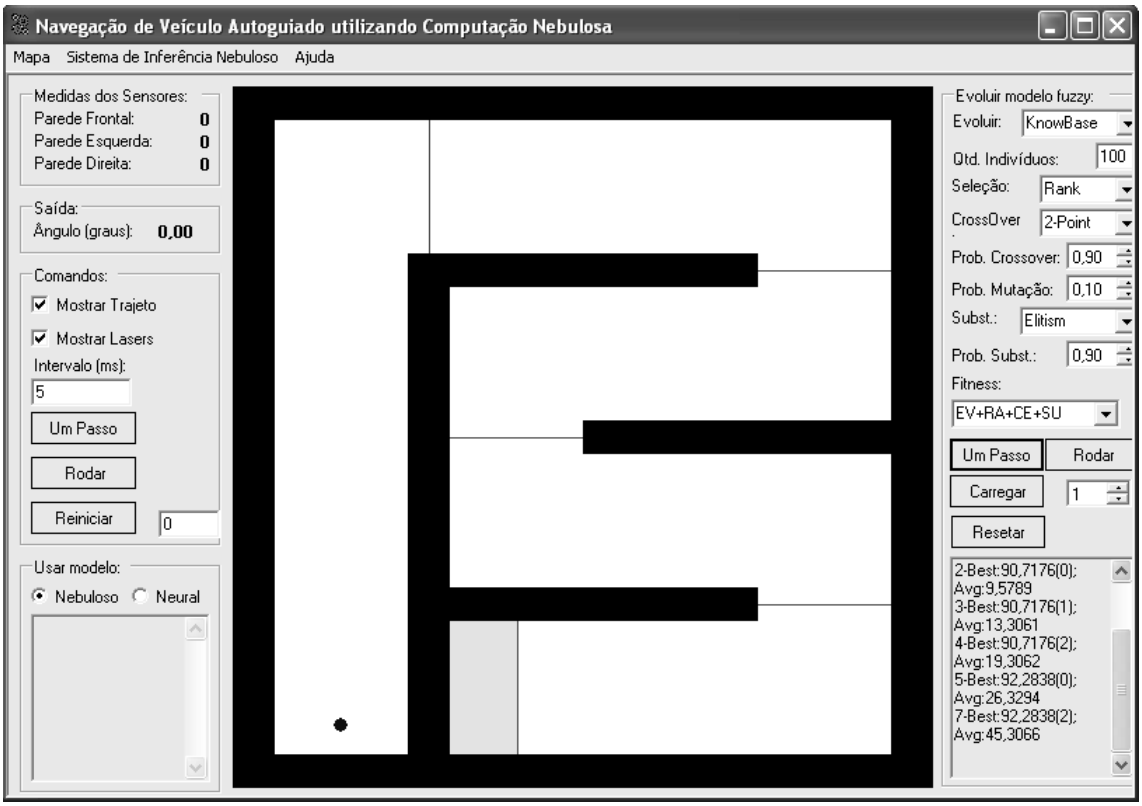


Figura 6.2: Interface principal do ambiente de simulação estendido.

Nesta coluna de controles adicionada, é possível manipular diversos parâmetros per-

tinentes ao Algoritmo Genético implementado, sendo eles:

- "Evoluir": é possível escolher se o objeto a ser evoluído é a Base de Conhecimento inteira, ou apenas a Base de Regras ou a Base de Dados.
- "Qtd. Indivíduos": definição do tamanho da população.
- "Seleção": operador de Seleção empregado, podendo ser a seleção por Roleta ou por Posicionamento (*Ranking*).
- "CrossOver": operador de Cruzamento empregado, podendo ser o cruzamento de um ponto ou o de dois pontos.
- "Prob. CrossOver": taxa de ocorrência do Cruzamento.
- "Prob. Mutação": taxa de ocorrência da Mutação. Não existe seleção para o tipo de operador de mutação empregado por existir apenas uma opção, implementada com características especiais, conforme descrito na seção 6.2.4.
- "Subst.": operador de Substituição empregado, podendo ser a substituição por Elitismo ou a "Strongest Survival" (os mais fortes dentre as populações pai e filho sobrevivem).
- "Prob. Subst.": no caso da substituição por elitismo, a porcentagem de pais que serão substituídos pelos melhores filhos na população principal deve ser definida.
- "Fitness": de acordo com o experimento a ser realizado, é possível escolher qual dentre os cálculos de Fitness implementados será utilizado.

Além disso, tal interface possibilita os seguintes comandos:

- "Um Passo": realiza um "passo evolutivo", ou seja, uma nova população de indivíduos é gerada.
- "Rodar": realiza quantos passos evolutivos forem necessários até que uma determina condição de parada seja atingida, ou até que o mesmo botão seja apertado novamente.
- "Carregar": possibilita, a qualquer momento do processo evolutivo, transformar qualquer um dos indivíduos da população atual para a forma de Base de Conhecimento, para então simular o seu comportamento.

- "Resetar": zera o processo evolutivo, para que uma nova simulação possa ser realizada.

A partir desse ambiente foi possível testar diferentes valores para os parâmetros citados do AG. Os parâmetros selecionados e utilizados nos experimentos serão descritos na sequência.

6.2.2 Abordagem Baseada em Shi, Eberhart e Chen (1999)

A partir da abordagem proposta em Shi, Eberhart e Chen (1999), algumas adaptações foram definidas, para adequá-la ao foco principal dos experimentos deste trabalho, a validação das transformações entre os dois domínios.

Tomando-se como base a solução conhecida, existem três variáveis de entrada (sensores - *"front", "left" e "right"*), com cada uma possuindo 3 conjuntos nebulosos associados (distância - *"near", "medium" e "far"*), cujo valor pode variar entre 0 e 350. A variável de saída (ângulo - θ), possui 7 conjuntos nebulosos associados, sendo eles: "very negative", "negative", "little negative", "zero", "little positive", "positive" e "very positive", com o valor podendo variar entre -40 e 40. Além disso, foram implementadas 4 formas diferentes de funções de pertinência - "Triangular Fuzzy Number" (TFN), "Left Fuzzy Number" (LFN), "Flat Fuzzy Number" (FFN) e "Right Fuzzy Number" (RFN), sendo as últimas três funções trapezoidais. Com isso, para a BD têm-se um espaço de busca com o seguinte tamanho: $(350 * 350 * n_formas)^{n_conjs_entrada} * (80 * 80 * n_formas)^{n_conjs_saida} \approx (350 * 350 * 4)^9 * (80 * 80 * 4)^7 \approx 10^{82}$.

Em relação à Base de Regras (BR), o número máximo de regras foi definido como 7, assim como na solução conhecida. De acordo com a abordagem proposta em Shi, Eberhart e Chen (1999), cada regra é representada por n_vars genes, sendo $n_vars = 4$ neste caso. Foi declarada como "impraticável" a existência de duas regras com os mesmos antecedentes e consequentes diferentes em uma mesma Base de Regras (BR). Como cada cláusula pode conter o identificador do conjunto nebuloso associado à variável, ou um identificador com o sinal de negativo na frente (operador NOT) ou o zero (ausência da variável na regra), para as variáveis de entrada são 7 possíveis situações, enquanto que para a variável de saída são 15. Com isso, o tamanho do espaço de busca referente à BR foi calculado: $((7 * 7 * 7)!/((7 * 7 * 7) - n_regras)!) * (15^{n_regras}) \approx ((7 * 7 * 7)!/((7 * 7 * 7) - 7)!) * (15^7) \approx 10^{25}$.

Dado que a Base de Conhecimento (BC) é constituída pela BD e pela BR, multi-

plicando os dois valores obtidos, obteve-se o tamanho total do espaço de busca da BC para esse problema: $10^{82} * 10^{25} = 10^{107}$. Trata-se de um espaço de busca muito grande. Como o foco principal dos experimentos é a validação das transformações entre os dois domínios, tentou-se reduzir ao máximo esse espaço, fazendo uso do conhecimento disponível através da solução conhecida, para que os resultados pudessem ser atingidos o mais rápido possível.

Primeiramente, os conjuntos nebulosos tiveram as suas funções de pertinência fixadas da seguinte forma: o primeiro conjunto nebuloso de cada variável foi fixado como "RFN", o último como "LFN", e os intermediários como "FFN". Apenas com essa alteração, obteve-se uma boa redução no espaço de busca referente à BD, que passou a ser $(350 * 350)^9 * (80 * 80)^7 \approx 10^{71}$.

Além disso, como na solução conhecida não foi necessário o emprego do operador NOT em nenhuma regra, ele também foi retirado do espaço de buscas da BR. Com essa restrição, o espaço de busca referente à BR teve seu tamanho reduzido para: $((4 * 4 * 4)!/((4 * 4 * 4) - 7)!) * (7^7) \approx 10^{18}$

Com isso, conseguiu-se obter uma boa redução no espaço de busca de toda a BC, de 10^{107} seguindo fielmente Shi, Eberhart e Chen (1999), para $10^{71} * 10^{18} = 10^{89}$ com as adaptações/reduções propostas.

A partir dessas definições, o número total de genes de cada indivíduo, representando uma BC completa para esse problema, foi obtido. São 16 conjuntos nebulosos sendo representados por dois genes cada (ponto inicial e ponto final); e 7 regras com 4 cláusulas cada uma, sendo um gene para representar cada cláusula. Com isso, têm-se: 60 genes, 32 para representar a BD e 28 para a BR.

6.2.3 Inicialização

O tamanho da população foi definido em 100 indivíduos. Cada indivíduo foi gerado aleatoriamente, usando a informação nebulosa existente: o número de variáveis de entrada (3) e de saída (1), o número de conjuntos nebulosos de cada variável (3, 3, 3, 7), os limites dos conjuntos nebulosos (entradas: [0;350], saída: [-40;40]), e o número máximo de regras (7).

6.2.4 Operadores Genéticos

O operador de seleção empregado foi o "Posicionamento (Ranking)". Tal operador faz com que indivíduos que dificilmente seriam selecionados pelo método da Roleta, por apresentarem uma aptidão muito baixa, tenham uma chance maior de serem escolhidos para gerar prole, acarretando em uma diversidade populacional maior.

Depois de transformados para o domínio evolutivo, os elementos nebulosos se tornam simples vetores de números inteiros. Por conta disso, não foi necessária a especificação de um operador de cruzamento especial, podendo ser utilizado qualquer tipo de operador tradicional, como os apresentados na seção 2.2.4.2. O operador utilizado nos experimentos foi o com 2 pontos de cruzamento, aplicado a uma taxa de 90%.

O operador de mutação, entretanto, teve que ser "especial". Visando auxiliar na diversidade populacional, a mutação não foi aplicada apenas em um simples gene, mas sim em um conjunto de genes representando um conjunto nebuloso ou uma regra linguística (dependendo da região sorteada para mutação no indivíduo), substituindo o conjunto ou a regra por outro elemento do mesmo tipo gerado aleatoriamente. A mutação foi aplicada a uma taxa de 10%.

Depois da criação de cada novo indivíduo, um método de verificação de consistência foi necessário na região contendo a BD (primeiros 32 genes do indivíduo). Conjuntos nebulosos com ponto final menor do que o ponto inicial poderiam ser gerados pelo processo de reprodução, bastando apenas inverter os seus valores para solucionar o problema.

Da mesma forma, na região relativa à BR (últimos 28 genes do indivíduo), mais de uma regra com o mesmo conjunto de antecedentes e consequentes diferentes poderiam existir no indivíduo gerado. Se tal caso era encontrado, a primeira regra era mantida, enquanto as outras contendo os mesmos antecedentes eram substituídas por novas regras geradas aleatoriamente. Uma outra possível solução para isso seria declarar as regras repetidas como "impraticáveis", deixando a base com um número menor de regras.

O cálculo da aptidão empregado será explicado na sequência, de acordo com o experimento realizado. Também para preservar a diversidade na população, um indivíduo idêntico a outro já avaliado recebia uma penalidade de 50% na sua medida de aptidão.

Toda essa preocupação referente à diversidade da população se deve à grande possibilidade da solução cair em ótimos locais nesse enorme espaço de busca. Ou seja, existem várias possíveis Bases de Conhecimento que atendem "bem" aos requisitos da função de aptidão, enquanto outras em menor quantidade atendem "melhor". Trata-se de uma pre-

venção para a convergência prematura, a qual ocorria frequentemente nos experimentos iniciais.

Por fim, o operador de substituição empregado foi o do elitismo a uma taxa de 10%. Ou seja, os 10% melhores pais são sempre copiados para a população seguinte, sendo ela completada pelos 90% melhores filhos. Dessa forma, não corre-se o risco de se perder os melhores indivíduos obtidos até então.

6.2.5 Transformações Implementadas CE ⇒ CF

As transformações da CE para a CF são utilizadas muitas vezes durante o processo evolutivo da abordagem proposta, ocorrendo em três situações distintas:

1. Avaliação: A cada vez que é necessário avaliar a aptidão de um indivíduo, o mesmo é transformado de volta para a CF, para que o processo de inferência seja realizado, e o comportamento obtido possa então ser comparado com o desejado. Todo indivíduo criado tem a sua aptidão avaliada pelo menos uma vez na "vida", ou seja, trata-se de uma transformação que ocorre intensamente durante o processo de evolução.

2. Simulação: Através do ambiente criado, é possível parar o processo de evolução a qualquer momento, e simular o funcionamento de qualquer um dos indivíduos, sendo para isso necessário transportá-los para o domínio nebuloso.

3. Produção: Ao final do processo evolutivo, o melhor indivíduo gerado pode então ser transportado para o domínio nebuloso para, finalmente, ser utilizado como solução para o problema.

No pacote MT foi criada uma classe *EvolutionaryTransform*, na qual foi implementado o algoritmo 3, utilizado para transformar um Indivíduo genético em uma Base de Conhecimento nebulosa. Este algoritmo recebe como parâmetros: o Indivíduo a ser transformado, o número de variáveis linguísticas (N_Vars) e o número de conjuntos nebulosos associados a cada variável (N_Sets[]).

Em tal algoritmo, alguns métodos são utilizados, sendo eles:

- BD.AdicionaVariavel(string Nome): adiciona uma variável à BD com o nome passado como parâmetro.

- V.AdicionaConjunto(string Nome, ConjuntoNebuloso CN): adiciona à variável V o Conjunto Nebuloso CN com o nome passado como parâmetro.

Algoritmo 3 Conversão de Indivíduo para Base de Conhecimento Nebulosa

```
método ConverterParaNebuloso(Ind:Individuo, N_Vars:int, N_Sets:int[]):BaseConhecimento
   i, j, pos: int;
   forma : string;
   BD: BaseDados;
   BR: BaseRegras;
   BC: BaseConhecimento;
início

   //Base de Dados
   i := j := pos := 0;
   enquanto (i < N_Vars) faça
      BD.AdicionaVariavel(``Variavel '' + i);
      enquanto (j < N_Sets[i]) faça
         se (j = 0) então  //se for o primeiro conjunto
            forma := ``RFN'';
         senão se (j = N_Sets[i] - 1) //se for o último conjunto
            forma := ``LFN'';
         senão //se forem os conjuntos intermediarios
            forma := ``FFN'';
         fim se
         BD.Variavel[i].AdicionaConjunto(``CN '' + j,
                   ConverterParaNebuloso(Ind[pos], Ind[pos+1], forma));
         j := j + 1;
         pos := pos + 2;
      fim enquanto
      i := i + 1;
      j := 0;
   fim enquanto

   //Base de Regras
   i := pos;
   enquanto (i < Ind.N_Genes) faça
      BR.AdicionaRegra(``Regra '' + i);
      j := 0;
      enquanto (j < N_Vars) faça
         se (Ind[pos] > 0)
            BR.Regra[i].AdicionaClausula(BD.Variavel[j].Nome,
                                 BD.Variavel[j].Conjunto[Ind[pos]].Nome);
         fim se
         pos := pos + 1;
         j := j + 1;
      fim enquanto
      i := i + N_Vars;
   fim enquanto

   BC.BaseDados := BD;
   BC.BaseRegras := BR;
   ConverterParaNebuloso := BC;
fim.
```

- BR.AdicionaRegra(string Nome): adiciona uma regra à BR com o nome passado como parâmetro.

- R.AdicionaClausula(string NomeVar, string NomeCN): adiciona uma cláusula à Regra R, associando a variável NomeVar com o conjunto NomeCN. No momento da adição, faz-se a distinção entre antecedentes e consequentes de acordo com a variável. No contexto desta pesquisa, este comando realiza a transformação de Genes para Cláusulas Linguísticas, consequentemente transformando também Genes em Regras.

Além disso, no algoritmo 3 é empregado também outro método de transformação de CE para CF, no qual os pontos x_1, x_2 e a $forma$ da sua função de pertinência, que são Genes no domínio evolutivo, são transformados para Conjuntos Nebulosos, conforme apresentado no algoritmo 4.

Algoritmo 4 Construtor de Genes para Conjunto Nebuloso

```
método ConverterParaNebuloso(x1: double, x2: double, forma: string):ConjuntoNebuloso
   CN : ConjuntoNebuloso;
início:
   se (forma = ‘‘FFN’’)
      CN.m1 := x1;
      CN.m2 := x1 + ((x2 - x1) / 4);
      CN.m3 := x2 - ((x2 - x1) / 4);
      CN.m4 := x2;
   senão se (forma = ‘‘TFN’’)
      CN.m1 := x1;
      CN.m2 := ((x1 + x2) / 2);
      CN.m3 := x2;
      CN.m4 := 0;
   senão //RFN ou LFN
      CN.m1 := x1;
      CN.m2 := x2;
      CN.m3 := 0;
      CN.m4 := 0;
   fim se
   CN.Forma = forma;
   ConverterParaNebuloso := CN;
fim.
```

6.2.6 Transformações Implementadas CF ⇒ CE

Dado que têm-se disponível uma Base de Conhecimento (BC) funcional disponível, o conhecimento contido na mesma pode ser aproveitado pelo processo evolutivo, o que auxilia bastante na performance do AG. Para tal, a BC existente é transformada para a forma de Indivíduo e inserida na população inicial. Dessa forma, o processo evolutivo já se inicia com um “bom” ponto no espaço de buscas, cujas características serão transmitidas

por herança para as próximas gerações. Essa é a única situação levantada na qual se faz necessário o transporte da CF para a CE.

De acordo com Andrade (2007), enquanto estão na CF, os conjuntos nebulosos da forma "FFN" são representados por 4 pontos ($m1$, $m2$, $m3$ e $m4$), enquanto os conjuntos da forma "TFN" precisam de 3 pontos, e as formas "RFN" e "LFN" necessitam apenas de dois pontos para serem representados, conforme pode ser visto na figura 2.2. No domínio evolutivo, os conjuntos nebulosos são representados pelo ponto inicial x_1 e pelo ponto final x_2. Dessa forma, a transformação deles para o eixo evolutivo ocorre da seguinte forma: para os "LFNs" e "RFNs", $x_1 = m_1$ e $x_2 = m_2$; para os "TFNs", $x_1 = m_1$ e $x_2 = m_3$; enquanto que para os "FFNs", $x_1 = m_1$ e $x_2 = m_4$.

Para transportar as regras linguísticas para o domínio evolutivo, é necessário que cada variável receba um identificador numérico, assim como os seus respectivos conjuntos nebulosos. O identificador da variável representa o índice do gene no qual o identificador do conjunto nebuloso associado na cláusula é inserido.

No pacote MT, foi adicionada à já existente classe *FuzzyTransform* (CAVERSAN, 2006) um método para realizar a transformação da Base de Conhecimento para Indivíduo, cujo pseudo-código é apresentado no Algoritmo 5.

Este algoritmo recebe como parâmetro apenas a Base de Conhecimento. Nele, alguns métodos são utilizados, sendo eles:

- BD.Variaveis: retorna uma lista ordenada das Variáveis presentes na BD.
- BD.N_Variaveis: retorna o número de Variáveis presentes na BD.
- V.ConjuntosNebulosos: retorna uma lista ordenada dos Conjuntos Nebulosos presentes na variável V.
- V.Indice: retorna o identificador numérico da variável V.
- CN.mx: retornam os 4 pontos que constituem um Conjunto Nebuloso.
- CN.Forma: retorna a forma da função de pertinência do Conjunto Nebuloso.
- CN.Indice: retorna o identificador numérico do conjunto CN.
- BR.Regras: retorna uma lista ordenada das Regras Linguísticas presentes na BR.
- R.Clausulas: retorna uma lista das Cláusulas (antecedentes e consequentes) que constituem a regra R.

Algoritmo 5 Conversão de Base de Conhecimento Nebulosa para Indivíduo

```
método ConverterParaEvolutivo(BC: BaseConhecimento): Individuo
   Ind: Individuo;
   pos: int;
início

   //Base de Dados
   para cada (Variavel V em BC.BaseDados.Variaveis)
      para cada (ConjuntoNebuloso CN em V.ConjuntosNebulosos)
         Ind[pos++] := CN.m1;
         se (CN.Forma = ‘‘FFN’’) então
            Ind[pos++] := CN.m4;
         senão se (CN.Forma = ‘‘TFN’’) então
            Ind[pos++] := CN.m3;
         senão
            Ind[pos++] := CN.m2;
         fim se
      fim para
   fim para

   //Base de Regras
   para cada (Regra R em BC.BaseRegras.Regras)
      para cada (Clausula C em R.Clausulas)
         Ind[pos + C.Variavel.Indice] := C.ConjuntoNebuloso.Indice;
      fim para
      pos := pos + BD.N_Variaveis;
   fim para

   ConverterParaEvolutivo := Individuo;
fim.
```

6.3 Experimentos Realizados

Após ter definido o problema a ser atacado, ter proposto uma abordagem a ser utilizada na sua solução e ter estabelecido os parâmetros pertinentes ao processo evolutivo, uma série de experimentos foi realizada.

Primeiramente, o foco era apenas a *criação* automática de Bases de Conhecimento (BCs) utilizando AGs. A partir de algumas constatações, atentou-se para a possibilidade de utilizar a abordagem desenvolvida também para *aprimorar* BCs existentes.

Diferentes experimentos foram então realizados, utilizando a mesma base de implementação, tendo como única diferença o modo como a aptidão de cada indivíduo é calculada. Tais experimentos são apresentados a seguir, seguindo a ordem de realização dos mesmos no decorrer do trabalho, juntamente com as constatações obtidas a partir da realização de cada um.

6.3.1 Criação 1 - Comparação com Solução Existente

Tendo à disposição uma solução criada por especialistas, a primeira abordagem elencada, para validar as transformações criando uma BC automaticamente através do AG, foi a comparação com os resultados provenientes da solução existente dos especialistas.

Um arquivo com 1058 amostras de entradas e saídas foi gerado a partir da solução funcional existente, representando o comportamento "ideal" a ser buscado pelo processo evolutivo. Cada indivíduo gerado era avaliado com base nesses casos de teste. Quanto mais parecidas fossem as respostas do robô gerado em relação aos resultados "ideais", maior era a nota de aptidão deste indivíduo.

Com o ângulo podendo ter 80 possíveis valores (de -40° a $+40^o$), e possuindo 1058 casos de teste, a diferença máxima total entre a saída obtida e a desejada pôde ser obtida: $80 * 1058 = 86400$. Com isso, a aptidão dos indivíduos para este experimento foi calculada da seguinte forma:

$$ind.FitnessScore = 100 - \frac{\sum_{1...1058} |ObtainedAngle - DesiredAngle| * 100}{86400}$$

Através de diferentes simulações, foi verificado na prática que o AG demoraria demais para encontrar uma solução com 100% de aptidão de acordo com o cálculo proposto. Foram realizadas várias simulações, percorrendo até 5000 gerações, e o máximo que se atingiu foi 99%. O motivo para isso é que exite apenas um, ou pouquíssimos pontos do espaço de busca (10^{89}) que gerem resultados iguaís à solução dos especialistas.

Dos indivíduos gerados com aptidão alta, muitos conseguiam cumprir o requisito de se locomoverem evitando obstáculos, como por exemplo o indivíduo cujo trajeto é apresentado na figura 6.3b.

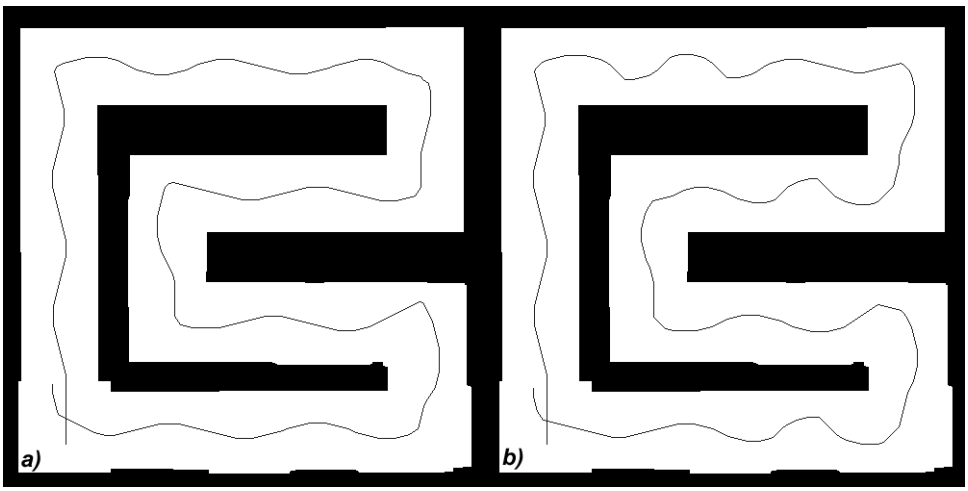


Figura 6.3: Trajetórias obtidas a partir de BCs criadas de forma diferente: a) Especialistas b) Algoritmo Genético.

Porém , o fato de um indivíduo apresentar uma aptidão de 99%, por exemplo, não assegura que o mesmo não vá colidir com obstáculos. Esse 1% de tomadas de decisões diferentes pode estar acumulado em poucos casos de teste, resultando em diferenças drásticas nesses casos, as quais podem levar à ocorrência de colisões.

6.3.2 Criação 2 - Evitar Colisões (EV)

A partir dos resultados obtidos com o primeiro experimento, constatou-se na prática que avaliar o quão parecido determinado indivíduo é em relação à solução dos especialistas não representa uma boa medida de aptidão, apresentando casos em que indivíduos recebem uma boa "nota" mas falham em cumprir o requisito de evitar colisões. Por esse motivo, decidiu-se implementar um cálculo de aptidão que avaliasse de fato o quão bem o robô gerado evitava colisões.

O mapa utilizado era um "circuito", no qual um "bom" robô poderia ficar dando voltas e mais voltas. Foi observado que diferentes soluções geradas com o primeiro experimento, assim como a solução dos especialistas, precisavam de aproximadamente 500 passos para completar uma volta no circuito.

A partir dessa observação, foi definido que, partindo sempre de um mesmo ponto no mapa, uma BC com 100% de aptidão para o problema deveria resultar em um robô que desse 500 passos sem colidir com obstáculos. Assim sendo, o comportamento do indivíduo

gerado era simulado na prática, e o seu cálculo da aptidão foi calculado da seguinte forma:

$$ind.FitnessScore = 100.0 * \frac{n_passos}{500}$$

Dessa forma, assegurou-se que todos os indivíduos com 100% de aptidão conseguiam de fato atender o requisito de se locomoverem sem colidirem com obstáculos. Diversas "boas" BCs foram geradas, as quais resultavam nos mais diferentes trajetos.

Entretanto, foi observado que com este único comportamento-objetivo, alguns indivíduos gerados cumpriam o requisito, mas se locomovendo em círculos, não cobrindo a área total do mapa, como apresentado na figura 6.4.

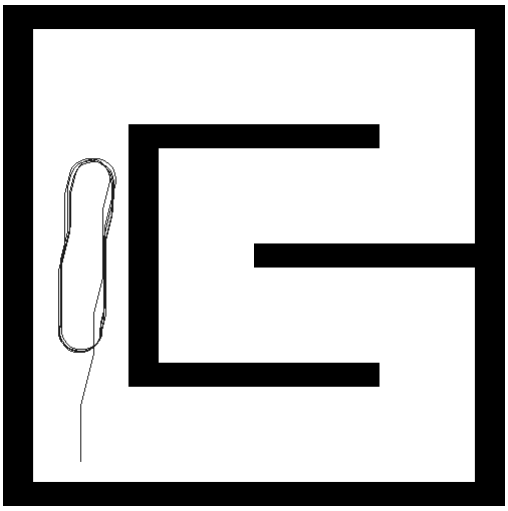


Figura 6.4: Mapa original, com robô se locomovendo em círculos.

6.3.3 Análise Crítica - Experimentos de Criação

Com a realização desses experimentos envolvendo a criação de BCs, foram validadas a aplicabilidade e a utilidade das transformações de um modo geral, demonstrando na prática a possibilidade da criação automática de uma BC a partir dos AGs.

Entretanto, apenas com esses experimentos, não foi possível justificar a utilidade da abordagem proposta em relação ao problema da navegação dos robôs autoguiados, dado que as soluções geradas atingiam o mesmo comportamento já obtido pelos especialistas, "empatando", não demonstrando nenhuma vantagem em relação à solução existente.

Além disso, atentando para os últimos resultados obtidos, os indivíduos "circulantes" gerados, apesar de atenderem ao requisito de evitar colisões, não demonstram um comportamento comumente desejado quando se tem todo um mapa a ser percorrido.

Para atender a essas duas questões levantadas, notou-se a necessidade de acrescentar

mais objetivos ao comportamento do robô, ou seja, *aprimorar* a BC, adicionando ao processo evolutivo novas metas a serem buscadas. Nesse momento foi então implementada a transformação da CF para a CE, que até então não havia sido necessária. A BC existente é transformada para um indivíduo e inserida na população inicial do processo evolutivo. Três experimentos diferentes abrangendo o *aprimoramento* de BCs foram realizados, os quais serão apresentados a seguir.

6.3.4 Aprimoramento 1 - EV + Regiões-Alvo (RA)

Visando eliminar a geração de indivíduos "circulantes" e promover a geração de indivíduos que percorressem a totalidade dos corredores existentes no mapa, o "circuito" do mapa original foi fechado, e o final do percurso foi marcado como região a ser alcançada pelo robô, acrescentando essa meta aos comportamentos buscados pelo processo evolutivo. O cálculo da aptidão ficou então baseado no cumprimento desses dois objetivos: evitar colisões e atingir a região marcada.

Porém, com apenas uma região-alvo a ser alcançada, a avaliação não possuía "granularidade" suficiente para medir o quão perto determinada solução tinha chegado da região-alvo. Ou a solução atingia o alvo (100%) ou não atingia (0%). Uma solução para isso seria fazer uso de geometria para calcular as distâncias. Porém, dado que o mapa é uma espécie de labirinto, tal cálculo seria muito complexo. Outra solução mais simples encontrada foi o acréscimo de mais regiões-alvo ao mapa, sendo estas "metas intermediárias" a serem cumpridas - foram inseridas 4.

Para realizar aprimoramentos, é necessário acumular as fórmulas empregadas no cálculo de aptidão de cada experimento. Dessa forma, uma BC com 100% de aptidão para o problema deveria resultar em um robô que desse 500 passos sem colidir com obstáculos e atingisse todas as regiões-alvo do mapa. A aptidão de cada indivíduo foi calculada então da seguinte forma:

$$ind.FitnessScore = 100 * \frac{n_passos}{500} * \frac{n_regioes}{5}$$

Várias BCs diferentes que cumpriam esses requisitos foram geradas. Um exemplo de um indivíduo gerado é apresentado na figura 6.5. As regiões intermediárias são as linhas finas atravessando o mapa; a região final é representada pelo retângulo mais claro.

Com o objetivo de demonstrar mais vantagens em relação à BC existente, e mais justificativas para as transformações propostas, foram realizados mais dois experimentos acumulando "*aprimoramentos*", conforme apresentado a seguir.

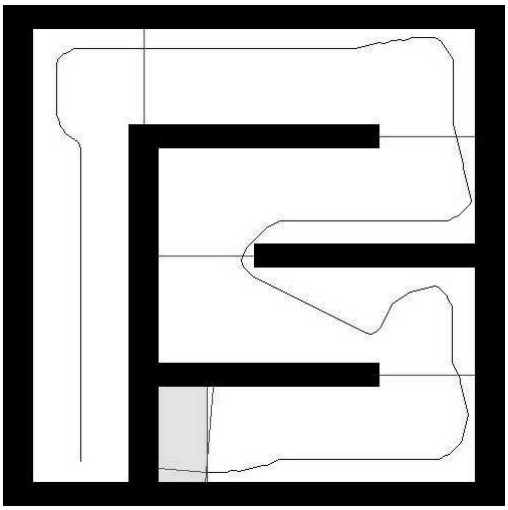


Figura 6.5: Mapa estendido, com um robô atingindo todas as regiões-alvo.

6.3.5 Aprimoramento 2 - EV + RA + Centralizador (CE)

Apesar de atingir todas as regiões-alvo, alguns dos robôs gerados passavam muito perto dos obstáculos. Dadas as imprecisões e incertezas do mundo real, um robô real com tal comportamento poderia colidir com algum obstáculo. Para evitar esse tipo de acidente, mais um objetivo foi adicionado: se locomover o mais no centro do corredor quanto possível. Tal objetivo pode ser atingido minimizando a cada passo a diferença entre as medidas dos sensores da direita e da esquerda do robô.

Para calcular a aptidão neste caso, é necessário conhecer o melhor ou o pior caso possível, podendo ser usado o que for mais simples de ser calculado. Para este mapa, 451 é a maior diferença que pode ser obtida entre as medidas dos sensores esquerdo e direito, o que pode ocorrer em apenas duas regiões distintas do mesmo. Apesar de impossível de acontecer, a diferença acumulada máxima das diferenças entre os sensores seria obtida caso a diferença entre as medidas dos sensores fosse de 451 em todos os passos, ou seja, o pior dos casos seria $451 * n_passos$. Com isso, a aptidão para esse experimento foi calculada assim:

$$ind.FitnessScore = fitness_anterior * (1 - \frac{\sum_{i=1}^{n_passos} |dist_L_i - dist_R_i|}{451 * n_passos})$$

Como é improvável a geração de um indivíduo cuja soma das diferenças entre os sensores seja *zero*, ou seja, um indivíduo que ande exatamente no meio do corredor em todos os passos, outra condição de parada deve ser utilizada, já que o cálculo da aptidão provavelmente não atingirá os 100%.

Utilizando o limite de 1000 gerações como condição de parada para o processo evolu-

tivo, um exemplo de um "bom" indivíduo gerado pode ser visto na figura 6.6.

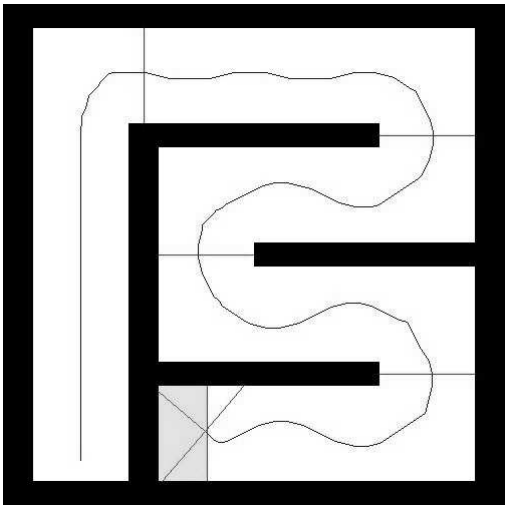


Figura 6.6: Robô "centralizado".

6.3.6 Aprimoramento 3 - EV + RA + CE + Suavizador (SU)

Várias BCs diferentes foram geradas atendendo aos requisitos de evitar colisões, atingir as regiões-alvo e andar no centro dos corredores. Entretanto, alguns desses indivíduos gerados executavam mudanças de direção muito abruptas, o que não seria possível na grande maioria de modelos reais de robô. Assim, outro comportamento-objetivo foi adicionado: fazer mudanças de direção o mais suave possível.

Tal comportamento pode ser atingido minimizando as diferenças de ângulo entre cada passo subsequente. Como a maior diferença de ângulo entre passos subsequentes pode ser de 80°, o pior caso seria $80 * n_passos$. Assim, a aptidão dos indivíduos para este experimento foi calculada da seguinte forma:

$$ind.FitnessScore = fitness_anterior * (1 - \frac{\sum_{i=2}^{n_passos} |\theta_i - \theta_{i-1}|}{80 * n_passos})$$

Utilizando o mesmo limite de 1000 gerações como condição de parada para o processo evolutivo, um exemplo de um "bom" indivíduo gerado pode ser visto na figura 6.7.

6.4 Considerações Finais do Capítulo

A extração do conhecimento necessário a respeito do problema muitas vezes se apresenta como uma tarefa complexa, por vários motivos: pode não haver especialistas no domínio do problema; podem existir especialistas, mas que não queiram ou não consigam ajudar;

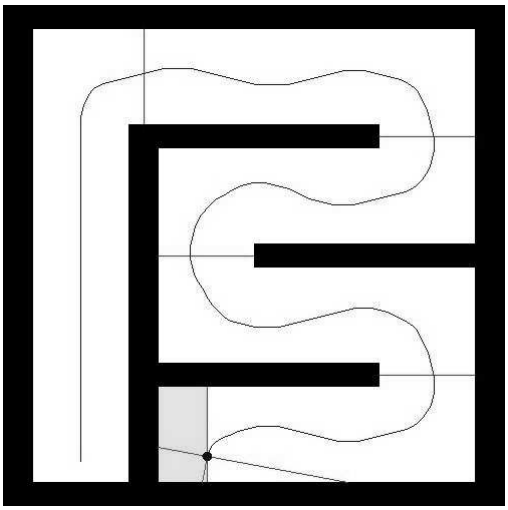


Figura 6.7: Robô "suavizado".

dentre outros. Nesses casos, um mecanismo que possibilite a extração automática desse conhecimento se faz muito útil.

A abordagem apresentada cumpre esse papel. Com a utilização das transformações propostas, é possível aplicá-la na criação e aprimoramento de BCs para qualquer tipo de problema de classificação ou controle nebuloso, bastando para isso um pouco de conhecimento sobre o domínio do problema, e uma métrica para avaliar a aptidão de cada indivíduo gerado. Em Cordon et al. (2004) é apresentada uma vasta revisão bibliográfica a respeito da criação e aprimoramento de BCs através de AGs.

É importante citar que quanto mais se conhece sobre o problema, mais informações podem ser aproveitadas para filtrar o espaço de buscas de soluções, o que influencia diretamente na eficiência e na performance do Algoritmo Genético (AG). Os parâmetros, como por exemplo, o tamanho da população, os tipos de operadores genéticos empregados, as taxas com as quais eles ocorrem, etc, também influenciam o comportamento dos AGs. Não foi realizada uma análise profunda a respeito do impacto de cada parâmetro do AG no processo evolutivo. Tal análise pode ser objeto de futuros estudos, podendo ser utilizado como ferramental o próprio ambiente de simulação desenvolvido.

Experimentos foram realizados, nos quais uma Base de Conhecimento inteira foi automaticamente criada, de acordo com os comportamentos desejados. Para atingir os resultado apresentados, transformações automáticas ocorreram intensivamente entre os dois domínios.

Além da criação, o aprimoramento de BCs existentes por meio de transformações entre os domínios também foi consolidado através dos experimentos. Com a abordagem

desenvolvida, torna-se possível inserir novos "comportamentos desejados" ao controlador/classificador em questão, bastando para isso alterar o cálculo de aptidão empregado, de acordo com os novos objetivos a serem cumpridos. Esta solução pode ser muito útil quando novas situações precisam ser tratadas.

Apesar de terem sido apresentadas agrupadas, as transformações envolvendo a Base de Regras (BR) e a Base de Dados (BD) podem ser facilmente desmembradas. Dessa forma, a abordagem desenvolvida pode ser aproveitada para outras situações, criando ou aprimorando apenas a BR ou apenas a BD.

A partir dos experimentos realizados, foi possível validar as transformações analisadas, demonstrando a utilidade delas na modelagem do conhecimento especialista. A solução desenvolvida pode ser empregada na prática em uma gama de aplicações do mundo real.

7 Considerações Finais

O desenvolvimento de soluções inspiradas pelos conceitos da inteligência biológica é algo relativamente novo: os estudos em RNAs (CN) se iniciaram na década de 50, a CF foi proposta em meados da década de 60, e os AGs (CE) uma década mais tarde. Desenvolvidas por pesquisadores diferentes e em épocas diferentes, cada uma dessas metodologias possuía o seu papel bem definido. Entretanto, conforme foi aumentando o conhecimento a respeito dos seus pontos fortes e fracos, tais técnicas passaram a ser utilizadas em conjunto, com uma suprindo as falhas da outra, o que é popularmente chamado como "sistemas híbridos".

A sinergia apresentada entre as técnicas constituintes da IC é tão "visível" que acabou por levantar questões, tais como: "Serão estas técnicas de IC representantes de modelos para manifestações distintas de um mesmo tipo de fenômeno físico/biológico?", "Serão estas abordagens passíveis de tratamento por um mesmo tipo de modelagem computacional?", "Será que existe, ou pode ser criada, alguma técnica que represente uma generalização dessas três abordagens?", etc.

Este trabalho contribui para esta linha, no que tange às transformações entre o domínio nebuloso e o domínio evolutivo. Apesar de serem metodologias com naturezas distintas, foram propostos com sucesso alguns transportes entre elas, os quais foram consolidados através de experimentos práticos. A cada transformação validada entre as metodologias de IC, dá-se mais um passo rumo à constatação da existência de um modelo unificado.

7.1 Cumprimento dos Objetivos Propostos

O diagrama Andrade-Caversan-Fialho foi proposto para direcionar a pesquisa, e várias possíveis abordagens para a busca dessas transformações foram analisadas. O caminho escolhido foi o estudo dos sistemas híbridos que empregam transportes entre as duas metodologias em questão.

Transformações oriundas da literatura foram organizadas e sistematizadas, sendo separadas em dois grandes grupos, o dos elementos constituintes da Base de Dados e o dos elementos constituintes da Base de Regras, sendo todos, por sua vez, componentes da Base de Conhecimento.

Uma plataforma computacional existente (UCIP) foi estendida, para abranger os experimentos utilizando CE. A partir disso, um protótipo com as transformações propostas foi implementado e experimentos foram realizados. Diferentes resultados foram alcançados, conforme apresentado no capítulo 6.

No decorrer de todo o trabalho, foram apresentados argumentos que sustentam a existência de transformações entre os dois domínios. Principalmente no capítulo 4, no qual foram elencadas possíveis transformações e abordagens, e no capítulo 6, onde a utilidade das transformações propostas foi colocada em prova, através da solução de um problema prático.

7.2 Contribuições do Trabalho

A partir dos objetivos cumpridos, algumas contribuições podem ser destacadas. A primeira delas concerne a organização e sistematização de boa parte do conhecimento a respeito das transformações entre CE e CF em um único trabalho.

O modelo proposto, batizado de diagrama Andrade-Caversan-Fialho, representa uma ferramenta de auxílio à investigação das relações, o qual pode continuar sendo utilizado nos estudos subsequentes desta linha de pesquisa. Tal modelo se torna útil na explanação da linha de pesquisa como um todo, e na comunicação do progresso obtido até então, reunindo em um único diagrama todas as transformações analisadas.

A plataforma UCIP foi estendida, com a modelagem e a implementação parcial do pacote evolutivo. Apesar desse pacote ter sido criado objetivando a análise de transformações entre os domínios, o modo genérico e estratificado como foi especificado permite que através do mesmo sejam derivadas as mais diversas soluções evolutivas.

Além do pacote evolutivo, novas transformações foram acrescentadas ao pacote *ModelTransformation* existente, com as quais é possível transportar desde um Conjunto Nebuloso até mesmo uma BC completa (BD: variáveis linguísticas e respectivos conjuntos + BR: cláusulas e regras linguísticas) em um Indivíduo evolutivo e vice-versa.

Os experimentos realizados mostraram na prática a contribuição obtida pela aplicação

das transformações. A abordagem proposta possibilita a utilização da CE para "aprender" BCs automaticamente, criando-as a partir do zero, ou mesmo aprimorando-as, sendo para essas tarefas necessário pouquíssimo conhecimento a respeito do domínio do problema e o mais importante: uma métrica para avaliar cada indivíduo gerado.

Em relação ao aprimoramento de BCs existentes, além de possibilitar um ajuste fino para a obtenção de uma tomada de decisão mais apurada, é possível acrescentar às mesmas novos "comportamentos desejados", o que mostra-se muito útil quando é necessário o tratamento de novas situações. Pode-se dizer, então, que a abordagem proposta atingiu o seguinte nível:

1. Deseja-se, por exemplo, que um Controlador Lógico Nebuloso (CLN) existe trate uma situação determinada;
2. Implementa-se o tratamento de tal situação no cálculo de aptidão do AG;
3. Transforma-se a BC atual do CLN para a forma de um Indivíduo genético, inserindo-se o mesmo na População inicial, sendo os demais Indivíduos gerados aleatoriamente, de acordo com o conhecimento existente sobre o problema
4. Executa-se o processo de aprendizado evolutivo;
5. Quando algum dos Indivíduos atinge um nível de aptidão satisfatório, transforma-se o mesmo de volta para o domínio nebuloso;
6. Obtém-se então, automaticamente, o CLN aprimorado, com o qual a nova situação pode ser tratada de forma satisfatória.

Apesar dos experimentos terem sido realizados sobre um CLN aplicado à navegação de um veículo auto-guiado, a solução obtida é genérica, podendo ser utilizada na criação ou aprimoramento de BCs para qualquer tipo de Sistema de Inferência Nebuloso. Têm-se com isso uma vasta gama de aplicações do mundo real para as quais a abordagem apresentada neste trabalho se faz útil na extração do conhecimento.

7.3 Trabalhos Futuros

A partir da realização deste trabalho, originou-se uma lista de possibilidades de pesquisa e trabalhos futuros:

- Análise da influência dos parâmetros do AG na criação e aprimoramento de BCs. Utilizando como plataforma o próprio ambiente de simulação criado, uma análise profunda acerca de cada parâmetro e da sua influência no processo evolutivo pode levar ao estabelecimento de valores com os quais o processo evolutivo para esse tipo de problema se torne muito mais eficiente.

- Realização de experimentos envolvendo outras aplicações de CF. Apesar de ser definido como uma solução genérica que pode ser usada em qualquer tipo de sistema especialista, faz-se importante realizar experimentos envolvendo outras aplicações de CF, a partir dos quais podem ser levantadas outras necessidades e/ou limitações para a aplicação da abordagem proposta.

- Estudo e possível transporte de outros elementos entre a CE e a CF. Outras relações entre as duas metodologias podem ser exploradas, como por exemplo entre os modificadores e operadores nebulosos e os operadores evolutivos. Em Herrera e Lozano (1998), por exemplo, operadores de T-Norma e T-Conorma são utilizados como operadores de um AG.

- Exploração envolvendo outras técnicas evolutivas. Dentre as técnicas de CE, apenas os AGs foram analisados, mas a Programação Genética, por exemplo, também é muito utilizada em conjunto com técnicas de CF, podendo a partir daí serem extraídas outras relações.

- Exploração de relações entre a CE e a CN. Dando continuidade na linha de pesquisa, podem ser exploradas relações entre a CE e a CN. Tal estudo pode encontrar relações de equivalência até mesmo a nível de "sistema", dado que a CN também apresenta em sua fase de treinamento a característica de "busca pela função".

Referências

ANDRADE, M. T. C. de. *Uma contribuição à pesquisa em inteligência computacional.* Tese (Livre Docência) — Escola Politécnica da Universidade de São Paulo, São Paulo, 2002.

ANDRADE, M. T. C. de. *Notas de aula PCS5711 - Computação Fuzzy.* Escola Politécnica da Universidade de São Paulo. Disponível em: <http://www.pcs.usp.br/∼pcs5711>. Acesso em: 7 de Março de 2007.

ARNONE, S.; DELL'ORTO, M.; TETTAMANZI, A. Toward a fuzzy government of genetic populations. *IEEE Conference on Tools with Artificial Intelligence*, v. 6, p. 585–591, 1994.

BONISSONE, P. Hybrid soft computing systems: Where are we going? *European Conference on Artificial Intelligence*, Berlin, v. 14, p. 739–746, 2000.

BONISSONE, P. P. Soft computing: the convergence of emerging reasoning technologies. *Soft Computing—A Fusion of Foundations, Methodologies and Applications*, v. 1, n. 1, p. 6–18, 1997.

BONISSONE, P. P.; CHEN, Y.-T.; GOEBEL, K.; KHEDKAR, P. S. Hybrid soft computing systems: Industrial and commercial applications. *Proceedings of the IEEE*, v. 87, n. 9, p. 1641–1667, 1999.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *UML: guia do usuário.* Rio de Janeiro: Editora Campus, 2000.

CAVERSAN, F. L. *Exploração de relações entre técnicas simbólicas e conexionistas da inteligência computacional.* Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, 2006.

CAVERSAN, F. L.; ANDRADE, M. T. C. Exploração de relações isomórficas entre técnicas simbólicas e conexionistas da inteligência computacional. *INFOCOMP Journal of Computer Science*, v. 4, n. 1, p. 19–25, 2005.

CLARK, D. Interview: deep thoughts on Deep Blue. *IEEE Expert*, v. 12, n. 4, p. 31, 1997.

COBO, J. E. A. *Considerações e modelos arquiteturais para prototipagem rápida em hardware de modelos nebulosos.* Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, São Paulo, 2000.

CORDON, O.; GOMIDE, F.; HERRERA, F.; HOFFMANN, F.; MAGDALENA, L. Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Systems*, v. 141, p. 5–11, 2004.

CORDON, O.; HERRERA, F.; JESUS, M. J. del. Evolutionary approaches to the learning of fuzzy rule-based classification systems. Boca Raton: CRC Press, Inc., 2000.

CORDON, O.; HERRERA, F.; LOZANO, M. *A classified review on the combination fuzzy logic-genetic algorithms bibliography*. Technical Report DECSAI-95129. Granada. 1995.

_____. On the bidirectional integration of genetic algorithms and fuzzy logic. *Online Workshop on Evolutionary Computation*, v. 2, p. 13–17, 1996.

CRAWFORD, K. D. Solving the n-queens problem using genetic algorithms. *Proceedings of the ACM/SIGAPP Symposium on Applied Computing*, ACM Press, New York, NY, USA, p. 1039–1047, 1992.

FOGEL, D. B. An introduction to simulated evolutionary computation. *IEEE Transactions on Neural Networks*, v. 5, n. 1, p. 3–14, 1994.

FOGEL, D. B.; FUKUDA, T.; GUAN, L. Scanning the special issue/technology on computational intelligence. *Proceedings of the IEEE*, v. 87, n. 9, p. 1415–1422, 1999.

FOGEL, L. J.; OWENS, A. J.; WALSH, M. J. *Artificial intelligence through simulated evolution*. New York: John Wiley & Sons, Inc., 1966.

GAJSKI, D.; KUHN, R. H. New VLSI tools. *IEEE Computer*, v. 16, p. 11–14, 1983.

GOLDBERG, D. E. *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley, 1989.

GOONATILAKE, S.; KHEBBAL, S. *Intelligent hybrid systems*. New York, NY, USA: John Wiley & Sons, Inc., 1994.

HERRERA, F.; LOZANO, M. *Fuzzy genetic algorithms: issues and models*. Technical Report DECSAI-98116. Granada. 1998.

HERRERA, F.; MAGDALENA, L. Genetic fuzzy systems: a tutorial. *Seventh International Fuzzy Systems Association World Congress*, Prage, v. 13, p. 93–121, 1997.

HOFFMANN, F.; PFISTER, G. Evolutionary design of a fuzzy knowledge base for a mobile robot. *International Journal of Approximate Reasoning*, v. 17, n. 4, p. 447–469, 1997.

HOLLAND, J. *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Press, 1975.

HOLLAND, J.; HOLYOAK, K.; NISBET, R.; THAGARD, P. *Induction: process of inference, learning, and discovery*. Cambridge, MA: MIT Press, Bradford Books, 1986.

HOLLAND, J. H.; REITMAN, J. S. Cognitive systems based on adaptive algorithms. In: WATERMAN, D. A.; HAYES-ROTH, F. (Ed.). *Pattern-directed Inference Systems*. New York: Academic Press, 1978.

JANG, J.-S. R. *Neuro-fuzzy modeling: architecture, analisys and applications*. Tese (Doutorado) — University of California, Berkeley, 1992.

KARR, C.; GENTRY, E. Fuzzy control of pH using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, v. 1, p. 46–53, feb 1993.

KNOMA. *Laboratório de Engenharia de Conhecimento*. Disponível em: <http://www.knoma.pcs.usp.br/>. Acesso em: 19 de Fevereiro de 2007.

KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*. Cambridge, MA: The MIT Press, 1992.

LEE, C. C. Fuzzy logic in control systems: fuzzy logic controller - part I. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 20, n. 2, p. 404–418, 1990.

_____. Fuzzy logic in control systems: fuzzy logic controller - part II. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 20, n. 2, p. 404–418, 1990.

NGUYEN, H. T.; WALKER, E. A. *A first course in fuzzy logic*. Boca Raton: CRC Press, Inc., 1996.

OBITKO, M. *An introduction to genetic algorithms with Java applets*. Disponível em: <http://cs.felk.cvut.cz/∼xobitko/ga/>. Acesso em: 19 de Fevereiro de 2007.

PEDRYCZ, W.; GOMIDE, F. *An introduction to fuzzy sets: analysis and design*. Cambridge, Massachusetts: The MIT Press, 1998.

POTVIN, J.-Y. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, v. 63, n. 3, p. 337–370, jun 1996.

PRESSMAN, R. S. *Engenharia de software*. São Paulo: Makron Books, 1995.

PROCYK, T. J.; MAMDANI, E. H. A linguistic self-organising process controller. *Automatica*, v. 15, p. 15–30, jan 1979.

RECHENBERG, I. *Cybernetic solution path of an experimental problem*. UK: Royal Aircraft Establishment, 1965.

REYNERI, L. M. An introduction to fuzzy state automata. *International Work-Conference on Artificial and Natural Neural Networks*, Springer-Verlag, London, UK, p. 273–283, 1997.

REZENDE, S. O. *Sistemas inteligentes: fundamentos e aplicações*. Barueri, SP: Editora Manole, 2003.

SAFFIOTTI, A. The uses of fuzzy logic for autonomous robot navigation: a catalogue raisonné. *Soft Computing Research journal*, v. 1, n. 4, p. 180–197, 1997.

SHI, Y.; EBERHART, R.; CHEN, Y. Implementation of evolutionary fuzzy systems. *IEEE Transactions on Fuzzy Systems*, v. 7, n. 2, p. 109–119, 1999.

SMITH, S. F. *A learning system based on genetic adaptive algorithms*. Tese (Doutorado) — University of Pittsburgh, 1980.

TSAKONAS, A.; DOUNIAS, G. Hybrid computational intelligence schemes in complex domains: an extended review. *Second Hellenic Conference on Artificial Intelligence*, Springer-Verlag, London, p. 494–512, 2002.

TSOUKALAS, L. H.; UHRIG, R. E. *Fuzzy and neural approaches in engineering*. New York: Wiley-Interscience, 1997.

WAGNER, F. R.; JANSH-PôRTO, I.; WEBER, R. F. W. T. S. Métodos de validação de sistemas digitais. *Escola de Computação*, Campinas, v. 6, 1988.

WIKIPEDIA. *Evolution strategy — Wikipedia, The Free Encyclopedia*. Disponível em: <http://en.wikipedia.org/w/index.php?title=Evolution_strategy>. Acesso em: 20 de Abril de 2006.

_____. *Evolutionary programming — Wikipedia, The Free Encyclopedia*. Disponível em: <http://en.wikipedia.org/w/index.php?title=Evolutionary_programming>. Acesso em: 20 de Abril de 2006.

ZADEH, L. A. Fuzzy sets. *Information Control*, v. 8, p. 338–353, 1965.

_____. Fuzzy algorithms. *Information Control*, v. 12, p. 94–102, 1968.

_____. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, v. 3, n. 1, p. 28–44, 1973.

_____. Fuzzy logic and soft computing: issues, contentions and perspectives. *International Conference on Fuzzy Logic, Neural Nets and Soft Computing*, Iizuka, Japan, v. 3, p. 1–2, 1994.

_____. Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, v. 2, p. 103–111, 1996.

ZUBEN, F. J. V. Computação evolutiva: uma abordagem pragmática. *Jornada de Estudos em Computação de Piracicaba e Região*, Piracicaba, v. 1, p. 25–45, 2000.

Anexo A – Pacote FC - Plataforma UCIP

Neste anexo é apresentada a especificação do pacote FC, reproduzida na íntegra a partir de Caversan (2006), com a autorização do respectivo autor. A reprodução desta descrição do pacote faz-se necessária devido à importância que o mesmo terá no desenvolvimento dos experimentos, dado que ele implementa uma das duas metodologias que são objeto de estudo desta pesquisa.

Conforme ilustrado na figura A.1, a construção de um modelo nebuloso (*FuzzyModel*) envolve a escolha e combinação de determinados elementos de CF: um mecanismo de fuzificação para converter entradas categóricas em nebulosas (*Fuzzyfier*), um mecanismo de inferência (*InferenceSystem*) capaz de inferir conhecimento utilizando uma base de dados (*DataBase*) e uma base de regras (*RuleBase*). A inferência é realizada através da escolha da regra composicional de inferência (*CompositionalRule*), T-Normas (*Norm*) e T-Conormas (*CoNorm*) utilizados para avaliar as regras. Além destes, possui um mecanismo defuzificador (*Defuzzyfier*), para transformar as saídas (valores lingüísticos) em valores categóricos.

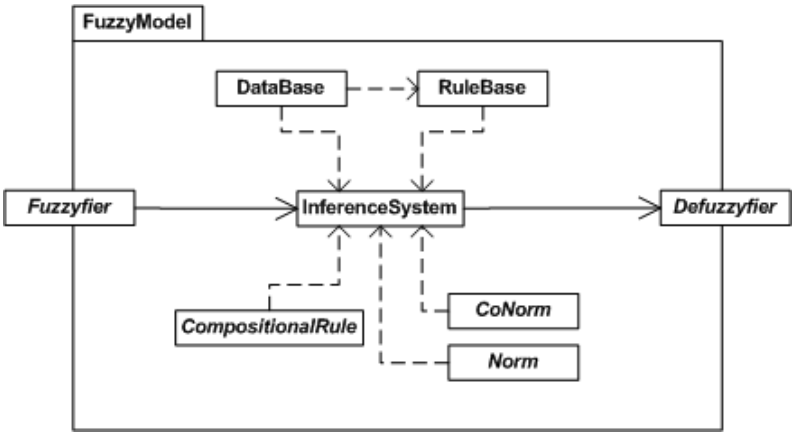


Figura A.1: Modelo lógico do pacote para realização de CF

As relações entre estas classes principais e as classes necessárias para a construção de um modelo nebuloso são apresentadas na figura A.2, o diagrama de classes do pacote FC.

A base de dados é composta de variáveis lingüísticas (*LinguisticVariable*). Estas por sua vez são compostas de etiquetas lingüísticas que são conjuntos nebulosos (*FuzzySet*), representando os valores que as variáveis podem assumir. A variável lingüística pode receber um valor categórico ou nebuloso como entrada, para calcular a respectiva pertinência em relação às suas etiquetas lingüísticas. Além de serem utilizados na composição de variáveis, os conjuntos nebulosos podem ser gerados a partir de valores categóricos, através dos fuzificadores (*Fuzzyfier*).

A base de regras é composta de regras nebulosas (*Rule*). As regras podem ser divididas em antecedentes e conseqüentes, que são compostos de cláusulas nebulosas (*Clause*), que são sentenças do tipo "X is F", sendo X uma variável lingüística e F um valor lingüístico. Estas cláusulas podem ser unidas por operadores (*Operators*). Os conectivos AND (*OperatorAND*) e OR (*OperatorOR*) operam, respectivamente, em duas cláusulas nebulosas através de T-Normas e T-Conormas. Um operador unário NOT (*OperatorNOT*) pode ser aplicado à uma cláusula, também. A composição de uma cláusula nebulosa pode utilizar-se de um modificador nebuloso (*Hedge*), como, por exemplo, "X is very F", sendo *very* um modificador nebuloso. Modificadores como VERY (*Very*), LESS (*Less*), MORE OR LESS (*MoreOrLess*) podem ser utilizados. Para simplificar a relação entre as cláusulas nebulosas e os conjuntos nebulosos, optou-se por sempre aplicar um modificador à um conjunto pertencente à uma cláusula. Quando nenhum modificador se faz necessário, o modificador EQUAL (*Equal*) deve ser utilizado.

Cada regra possui um mecanismo de cálculo do seu "potencial de ativação", responsável por calcular as pertinências das proposições, agrupá-las através dos operadores, resultando na confiança da veracidade do antecedente. Esta confiança é propagada para o conseqüente através de uma regra composicional de inferência nebulosa (*CompositionalRule*). As saídas das regras são utilizadas pelo SIN no cálculo do valor final de uma variável de saída.

O pacote permite a criação de diversos modelos nebulosos, possibilitando a extensão através dos mecanismos da orientação à objetos. Em particular, as classes abstratas *FuzzySet*, *Fuzzyfier*, *(*Hedge), *Defuzzyfier*, *Norm*, *Conorm*, *Operator* e *CompositionalRule* possibilitam a extensão natural da plataforma, utilizando herança e polimorfismo para implementar casos diferentes dos seus métodos. Todos os elementos possuem um ancestral comum, a classe *FuzzyElement*. Isto possibilita que toda classe pertencente ao domínio

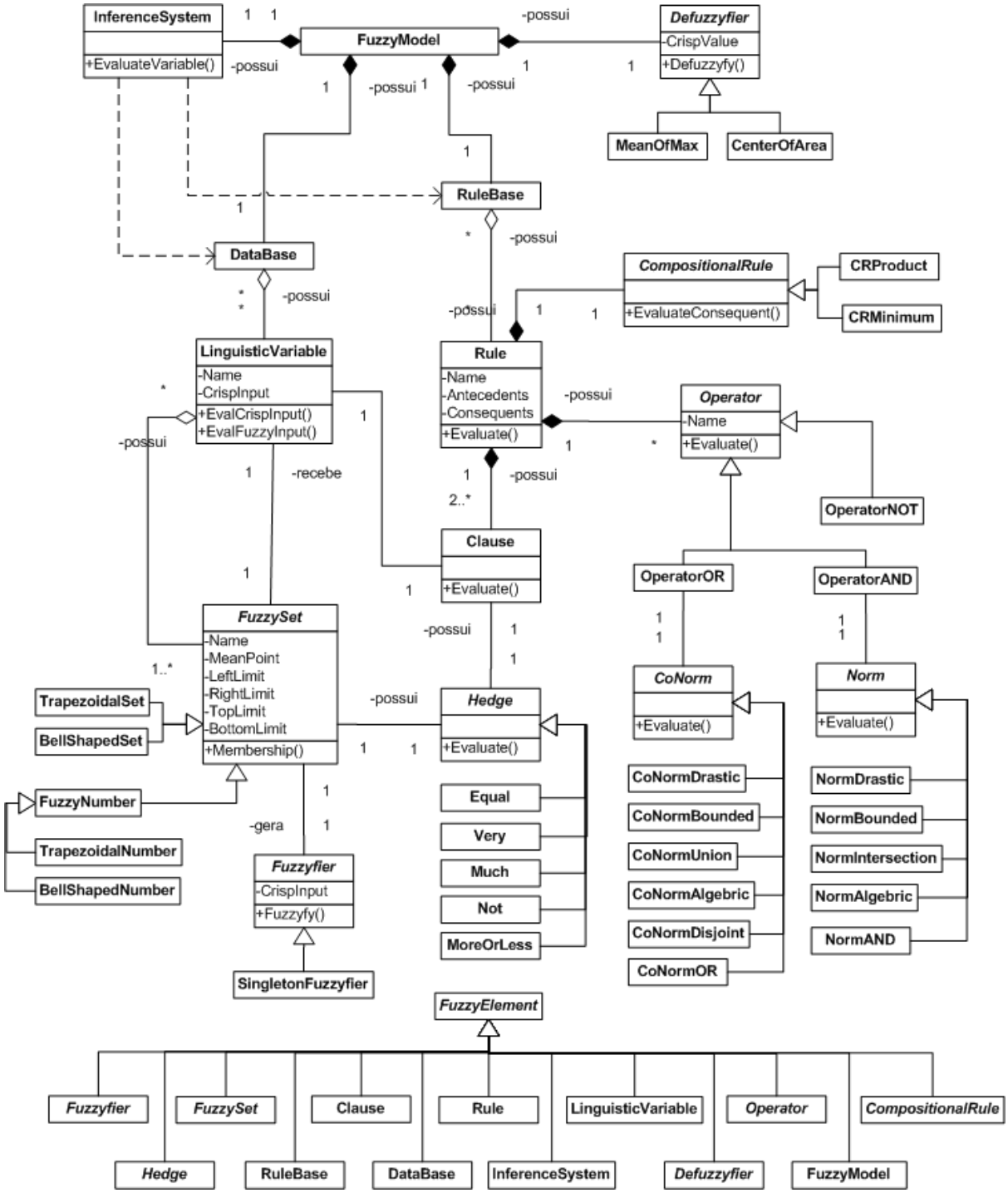


Figura A.2: Diagrama de classes do pacote para realização de CF.

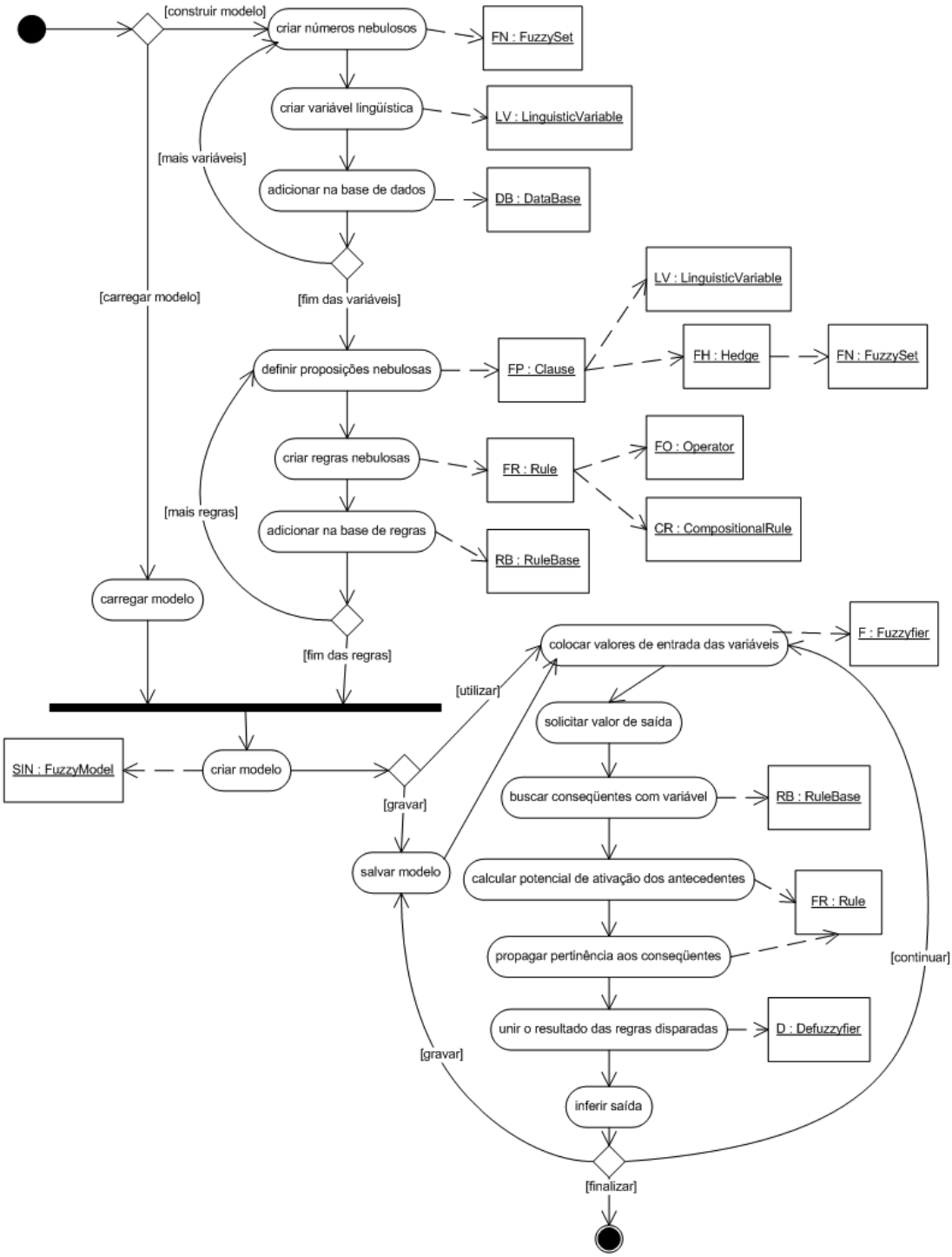


Figura A.3: Diagrama de atividades representando a operação com CF.

de CF seja tratada como um elemento nebuloso genérico, característica especialmente útil na especificação das transformações entre modelos.

A figura A.3 ilustra a seqüência de operações realizadas para construir (ou carregar), salvar e utilizar os modelos nebulosos através dos componentes do diagrama estático.

Na etapa de construção do modelo, as variáveis lingüísticas são incorporadas à base de dados, modeladas a partir do domínio do problema. As regras são criadas através do conhecimento do especialista, utilizando estas variáveis. Feito isso, a etapa de parametrização está feita e o SIN construído pode ser utilizado. Os valores categóricos da entrada são informados, e um valor de saída é solicitado. As regras que possuem tal saída no conseqüente são então avaliadas, e um método de defuzificação é aplicado para que se possa obter a saída numérica.