UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA DA COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

Aplicação de Métodos de Computação Flexível em Navegação Autônoma de Veículos

por Marco Antonio Assfalk de Oliveira

orientador Prof. Dr. Fernando Antonio Campos Gomide

Dissertação submetida à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas, para preenchimento dos pré-requisitos parciais para obtenção do Título de Mestre em Engenharia Elétrica.

Agosto de 1995

Resumo

A navegação autônoma de veículos é um exemplo bem conhecido e típico de problemas de controle autônomo. Este tipo de controle envolve um ambiente muito complexo e desestruturado. A grande quantidade de parâmetros descarta o uso de um modelo matemático do ambiente. Por isso, o controle autônomo exige métodos de controle que se adaptam constantemente ao ambiente e que utilizam informações locais. O uso de tais métodos incorre em duas considerações: localidade (*situatedness*) e o problema da referência (*frame-of-reference*). A primeira questão está relacionada com a complexidade do ambiente de operação, a qual não permite a previsão do comportamento global do ambiente. Assim, o agente autônomo deverá ser capaz de agir baseado em informações parciais, provenientes de sua situação local. O segundo problema está ligado ao conhecimento utilizado no projeto do agente (conhecimento *a priori*). Este conhecimento pré-inserido limita a adaptabilidade do agente ao impor tanto o ponto de vista do projetista (o qual é baseado em sensores e modos de processamento bem diferentes dos do agente) quanto o conhecimento limitado sobre o ambiente deste projetista. Ambos os problemas são resolvidos pelo uso de um método de controle adaptativo e localizado.

As propostas mais promissoras provém da área de computação flexível. Computação flexível engloba um conjunto de métodos, desde teoria dos sistemas nebulosos a sistemas evolucionários. Os paradigmas principais são as redes neurais, sistemas nebulosos e um conjunto de métodos conhecido como raciocínio probabilístico. Este último inclui sistemas evolucionários, teoria da complexidade e teoria do caos, entre outros. O poder da computação flexível vem da simbiose de seus múltiplos métodos. Estes sistemas híbridos oferecem as vantagens dos seus componentes e compensam mutuamente as falhas destes componentes. As características mais interessantes incluem auto-organização, processamento de informações imprecisas e capacidade de aprendizagem.

Neste trabalho propomos um método de controle autônomo que combina uma rede neural, teoria de sistemas nebulosos e um algoritmo genético. A sinergia destes três paradigmas de computação oferece uma estrutura computacional paralela e robusta com fácil

inserção/extração de conhecimento e capaz de aprendizagem não-supervisionada. O método proposto foi aplicado ao problema de navegação autônoma de veículos em um ambiente simulado. Os resultados demonstram que as limitações observadas podem ser atribuídas à quantidade e tipo de conhecimento utilizado na escolha dos parâmetros (Sensores e representação de conhecimento). O método de navegação em si provou ser robusto e confiável, desenvolvendo comportamentos com desempenho comparável aos de um agente projetado heuristicamente.

O desenvolvimento futuro deste trabalho será focalizado na investigação de métodos auto-organizados para reconhecimento dos dados ambientais releventes e nos problemas de otimização do conhecimento *a priori* e da sensibilidade do método de aprendizagem às especificações dos sensores. Estas pesquisas têm por objetivo libertar o agente de limitações impostas pelo projetista e aumentar a adaptabilidade do agente.

Abstract

Autonomous vehicle navigation is a well-known and typical example of an autonomous control problem. This type of control problem involves a very complex, unstructured environment. The great amount of parameters preclude the use of a mathematical model of the environment. Thus, autonomous control demands constantly adapting, locally situated control methods, giving rise to two fundamental design issues: situatedness and frame-of-knowledge. The first deals with the impossibility of predicting the global behaviour of the environment. Instead the autonomous agent must be able to act upon local and limited data. The latter problem refers to the knowledge used in the design of an agent (*a priori* knowledge). Such pre-installed experience limits the agent's adaptability by imposing the designer's point of view (based on sensors and processing modes far different from those of the agent) and limited knowledge of the environment. Both issues are addressed by implementing an adaptive locally-aware control method.

One of the most promising approaches comes from the *soft computing* field. Soft computing involves a spectrum of methods, ranging from fuzzy system theory to evolutionary systems. The main bulwarks are neural networks, fuzzy systems and a number of methods known as probabilistic reasoning. The latter includes evolutionary systems, cellular automata, complexity theory and chaos theory, among others. The power and promise of soft computing emerge from the symbiosis of its many paradigms. These hybrid synergetic systems offer the strengths of their components while cross-compensating the components' drawbacks. The most interesting features include auto-organization, imprecise data handling and unsupervised learning capabilities.

In this work we propose an autonomous control method that combines a neural network, fuzzy system theory and a genetical algorithm. The synergy of these three soft computing paradigms offers a parallel robust computing structure with easily extractable/insertable knowledge and capable of unsupervised learning. The proposed method was applied to the autonomous vehicle navigation problem in a simulated environment. The results show that most of the limitations are due to the amount and type of knowledge used

in the choice of the vehicle parameters (sensors and knowledge representation). The navigation method itself proved to be robust and reliable, developing behaviours comparable to those of a hand-crafted agent.

Future work will focus on the issues of the optimum *a priori* knowledge levels, the sensivity of the learning method to sensor specifications, and on the development of self-organizing methods for relevant environment data recognition. These efforts are geared towards freeing the agent from human-imposed limiting factors while enhancing its adaptability.

Agradecimentos

Gostaria de agradecer aos meus pais e aos meus irmãos pelo amor, carinho, suporte e atenção e compreensão que sempre me deram. Agradeço ao Prof. Dr. Fernando Gomide pela orientação e à UNICAMP pela oportunidade de realizar este trabalho de pesquisa. Agradeço aos meus colegas dentro e fora do LCA pela interação, humor e ótima companhia, os quais tornaram profícua a minha estadia em Campinas. Agradeço à turma da Sala 10 pelas conversas e discussões significativas e ao Maurício pelas conversas e *talks* com/sem sentido e pelo apoio indispensável na implementação da tese. Agradeço ao Prof. Giarola e a Sra. Giarola pelos seus conselhos e amizade. Agradeço às inúmeras pessoas que influiram e contribuiram neste trabalho. Peço desculpas por não citar cada pessoa, por temer deixar alguém de fora.

"A experiência nunca erra.
É tão somente o vosso julgamento que erra ao prometer a si mesmo resultados que não decorrem das vossas experiências."

Leonardo da Vinci (circa 1510).

Conteúdo

RESUMO i

ABSTRACT iii

AGRADECIMENTOS v

CONTEÚDO v

LISTA DE FIGURAS viii

LISTA DE TABELAS xiv

1 Introdução 1

1.1 Motivação 1

1.2 Objetivos 3

1.3 Organização da Tese 3

2 Fundamentos: Computação Flexível 5

2.1 Introdução 5

2.2 Lógica Nebulosa e Sistemas Nebulosos 6

2.2.1 Introdução 6

2.2.2 Conceitos 7

2.2.3 Controlador Nebuloso 16
2.3 Redes Neurais (Artificiais) 26
2.3.1 Introdução 26
2.3.2 Conceitos 28
2.3.3 Classificação das Redes Neurais 30
2.3.4 Perceptron 33
2.3.5 Redes Auto-organizadas 37
2.3.6 Redes Neuronebulosas 42
2.4 Sistemas Evolutivos 46
2.4.1 Conceitos 47
2.4.2 Algoritmo Básico 52
2.5 Considerações Adicionais 62
2.5.1 Embasamento matemático 62
2.5.2 Operadores secundários 65
2.6 Resumo 67

3 Navegação Autônoma de Veículos **69**

3.1 Introdução 69
3.2 O Problema de Navegação Autônoma 69
3.3 Abordagens para Navegação Autônoma 71
3.3.1 Navegação por planejamento 74
3.3.2 Navegação reativa 75
3.3.3 Navegação Comportamental 82
3.4 Um Método de Navegação Autônoma 89
3.4.1 Parâmetros Utilizados 89
3.4.2 Caracterização da Rede Neuronebulosa 90
3.4.3 Caracterização do Algoritmo Genético 93

3.5 Resumo 95

4 Resultados de Simulação **97**

4.1 Introdução 97

4.2 Análise de Resultados 98

4.3 Resumo 127

5 Conclusão **129**

BIBLIOGRAFIA **131**

Lista de Figuras

2.1 Ilustração do conceito de convexidade. 15

2.2 Exemplo de uma variável lingüística (distância). 16

2.3 Diagrama de blocos de um controlador nebuloso. 16

2.4 Exemplo da determinação de uma variável nebulosa de controle. 19

2.5 Representação gráfica de dois tipos de inferência: acima, tipo Mamdani e abaixo, tipo Larsen. 23

2.6 Representação gráfica de dois tipos de inferência para antecedentes tipo singleton nebuloso: acima, tipo Mamdani e abaixo, tipo Larsen. 24

2.7 Métodos de fuzzificação. 25

2.8 Comparação entre um neurônio biológico e um artificial. 27

2.9 Modelo de um neurônio. 28

2.10 Tipos de Funções de transferência: degrau, sigmóide e gaussiana. 30

2.11 Representação de limiar como peso e conexão de valor fixo. 30

2.12 Exemplo de uma rede não-recorrente. 31

2.13 Exemplo de uma rede recorrente. 32

2.14 Esquemas de Treinamento Supervisionado. 35

2.15 Diagrama de um Sistema de Controle por Realimentação. 36

2.16 Método de Widrow para treinar controladores baseados em redes neurais. . 37

2.17 Exemplo de rede neural competitiva. 38

2.18 Representação vetorial da aprendizagem competitiva. 39

2.19 Uma rede de Kohonen bidimensional. 41
2.20 A função tipo gaussiana de distância entre neurônios para um dimensão. . . 41
2.21 Rede Neuronebulosa proposta por Figueiredo *et al.* 44
2.22 Função de ativação dos neurônios da camada de *matching*. 45
2.23 Busca genética em um espaço de busca com dois máximos (múltiplos pontos de busca). 48
2.24 Busca por gradiente. 49
2.25 Possível cromossomo para uma rede neural. 49
2.26 Exemplo de uma operação de mutação. 50
2.27 Exemplo de *crossover*. 51
2.28 Fluxograma básico de um Algoritmo Genético. 51
2.29 Exemplo de Crossover por segmentos. 59
2.30 Exemplo de Crossover multipontual 60
2.31 Exemplo de Crossover por máscara. 61
2.32 Escalamento linear utilizando os desempenhos máximo e médio como referência. 65
2.33 O problema do mínimo negativo no escalamento linear max-mid. 66
2.34 Escalamento linear utilizando os desempenhos médio e mínimo como referência. 66
2.35 Computação flexível: paradigmas básicos. 68

3.1 Classificação dos Métodos de Navegação. 72
3.2 Diagrama de blocos de um sistema baseado em planejamento. 75
3.3 Navegador baseado em redes neurais de Verschure *et al.* 77
3.4 Ambiente de operação e utilização do navegador neural de Verschure. 78
3.5 Arquitetura empregada por Almassy et Vinkhuyzen. 79
3.6 Exemplo de funções de pertinência. 81
3.7 Diagrama esquemático de um sistema comportamental. 83
3.8 Exemplo de um sistema de navegação comportamental. 84

3.9 Esquema da Máquina aprimorada de Estados Finitos. 85

3.10 Exemplo de uma arquitetura subsumption: rede parcial de um autômato de estados finitos para a locomoção de um robô hexápide. 86

3.11 Exemplo de arquitetura subsumption extendida. 87

3.12 Exemplo de arquitetura formado por Neuronal Group Selection. 88

3.13 Caracterização do Veículo utilizado: sensores de obstáculos. 90

3.14 Caracterização do Veículo utilizado: sensor de alvo. 91

3.15 Tipos de Ambientes de Operação. 92

3.16 Construção das Funções de Pertinência pela Decodificação do Cromossomo. 94

4.1 Funções de pertinência utilizados no navegador de referência. 99

4.2 Percurso executado pelo veículo usando o navegador de referência no ambiente tipo Labirinto. 100

4.3 Percurso executado pelo veículo usando o navegador de referência no ambiente tipo Estrela. 100

4.4 Percurso executado pelo veículo usando o navegador de referência no ambiente tipo Vazio. 100

4.5 Percurso executado pelo veículo usando o navegador de referência no ambiente tipo Estrela2. 101

4.6 Ambiente tipo labirinto utilizado na avaliação da população de redes neuro-nebulosos na primeira experiência. 101

4.7 Funções de pertinência evoluidos na primeira simulação. 102

4.8 Evolução de Desempenho do Navegador avaliado em ambiente tipo Labirinto. 103

4.9 Percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador evoluido em ambiente tipo Labirinto (ver texto). 103

4.10 Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador evoluido no ambiente tipo Labirinto. 103

4.11 Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido em ambiente tipo Labirinto. 104

4.12 Percurso no ambiente tipo Estrela2 executado pelo veículo usando o navegador evoluido em ambiente tipo Labirinto. 104

4.13 Ambiente tipo Estrela utilizado para a avaliação da população de redes neuronebulosas durante a evolução. 105

4.14 Gráfico da evolução do desempenho dos controladores ao longo das gerações. 106

4.15 Funções de pertinência evoluidos na segunda simulação. 106

4.16 Percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador evoluido em ambiente tipo Estrela. 107

4.17 Outro percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador evoluido em ambiente tipo Estrela. 107

4.18 Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido em ambiente tipo Estrela. 107

4.19 Novo percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido em ambiente tipo Estrela. 108

4.20 Percurso no ambiente tipo Estrela2 executado pelo veículo usando o navegador evoluido em ambiente tipo Estrela. 108

4.21 Ambiente tipo Vazio utilizado para a avaliação da população de redes neuronebulosas durante a evolução. 109

4.22 Gráfico da evolução do desempenho dos controladores ao longo das gerações na terceira simulação. 110

4.23 Funções de pertinência evoluidos na terceira simulação. 110

4.24 Percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador evoluido em ambiente tipo Vazio. 111

4.25 Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador evoluido no ambiente tipo Vazio. 111

4.26 Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido em ambiente tipo Vazio. 111

4.27 Novo percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido em ambiente tipo Vazio. 112

4.28 Gráfico da evolução do desempenho dos controladores ao longo das gerações na quarta simulação. 113

4.29 Funções de pertinência evoluidos na quarta simulação. 113

4.30 Percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador evoluido na sequencia de ambientes Vazio-Estrela. 114

4.31 Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador evoluido na sequencia de ambientes Vazio-Estrela. 114

4.32 Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador evoluido na sequencia de ambientes Vazio-Estrela. 114

4.33 Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido na sequencia de ambientes Vazio-Estrela. 115

4.34 Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido na sequencia de ambientes Vazio-Estrela. 115

4.35 Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido na sequencia de ambientes Vazio-Estrela. 116

4.36 Evolução do desempenho do rastreador de alvos. 118

4.37 Evolução do desempenho do navegador que obstáculos. 118

4.38 Funções de pertinência obtidas pela evolução do navegador que obstáculos. 119

4.39 Funções de pertinência obtidas pela evolução do rastreador de alvos. 119

4.40 Percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador que evita obstáculos gerado pelo algoritmo genético. Observe a predominância do comportamento de evitar obstáculos. 119

4.41 Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador que evita obstáculos gerado pelo algoritmo genético. 120

4.42 Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador que evita obstáculos gerado pelo algoritmo genético a procura de outro alvo. 120

4.43 Percurso executado no ambiente tipo Vazio pelo veículo usando o navegador que evita obstáculos gerado pelo algoritmo genético a procura do primeiro alvo (canto superior esquerdo). Observe a predominância do comportamento de evitar obstáculos. 121

4.44 Percurso executado no ambiente tipo Estrela2 pelo veículo usando o navegador que evita obstáculos gerado pelo algoritmo genético a procura do primeiro alvo. 121

4.45 Percurso executado no ambiente tipo Estrela2 pelo veículo usando o navegador que evita obstáculos gerado pelo algoritmo genético a procura de outro alvo (canto superior direito). 122

4.46 Percurso executado no ambiente tipo Labirinto pelo veículo usando o navegador "rastreador de alvos" gerado pelo algoritmo genético a procura do primeiro alvo (lateral esquerda). Observe a colisão decorrente da especialização. 122

4.47 Percurso executado pelo veículo usando o navegador "rastreador de alvos" gerado pelo algoritmo genético a procura do segundo alvo (canto superior direito) no ambiente tipo Labirinto. Ocorre nova colisão. 122

4.48 Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador "rastreador de alvos" gerado pelo algoritmo genético a procura do primeiro alvo (quadrante esquerdo superior). 123

4.49 Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador "rastreador de alvos" gerado pelo algoritmo genético a procura de outro alvo (canto superior direito). 123

4.50 Percurso executado no ambiente tipo Estrela2 pelo veículo usando o navegador "rastreador de alvos" gerado pelo algoritmo genético. 123

4.51 Arquitetura do navegador proposto reformulado como um sistema comportamental tipo Neuronal Group Selection. 124

4.52 Percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador composto. 125

4.53 Percurso no ambiente tipo Estrela executado pelo veículo usando o navegador composto. 125

4.54 Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador composto. 126

4.55 Percurso no ambiente tipo Estrela2 executado pelo veículo usando o navegador composto. 126

Lista de Tabelas

2.1 Exemplo de uma Relação nebulosa discreta. 10

2.2 Modus Tollens Generalizado 21

2.3 Modus Ponens Generalizado 21

2.4 Critérios intuitivos relacionando os termos Pre1 e Cons para um dado Pre2 em GMP 22

2.5 Critérios intuitivos relacionando os termos Pre1 e Cons para um dado Pre2 em GMT 22

2.6 Pares de Operadores (implicação, conectivo *AND*) de Melhor Desempenho . 23

Capítulo 1

Introdução

1.1 Motivação

A navegação autônoma de veículos é um exemplo típico de problemas de controle autônomo. Esta classe de problemas é caracterizada por uma complexidade devido ao ambiente de operação envolvendo múltiplos parâmetros que variam dinamicamente. A pesquisa em navegação autônoma visa desenvolver métodos e modelos que permitam a operação não-supervisionada de máquinas em situações como:

- ambientes hostis à presença humana;
- ambientes que não permitem o controle a distância, devido a fatores diversos (interferência na telemetria, atraso intolerável na comunicação entre operador e veículo etc.);
- desempenho de tarefas repetitivas porém com parâmetros variantes que impedem a modelagem.

A ausência do operador humano na malha de controle exige a utilização de métodos adaptativos e inteligentes. As características humanas de controle como a robustez, capacidade de aprendizagem e adaptação e flexibilidade devem ser emuladas ou embutidas no controlador autônomo.

Duas questões principais devem ser atendidas pelos métodos de controle autônomo:

– a questão da localidade (*situatedness*);

– o problema de referência (*frame*).

A questão da localidade é relacionada com a complexidade do ambiente de operação. Esta complexidade não permite em geral, a obtenção de um modelo completo do ambiente, pois a combinação dos parâmetros possíveis é vasta. Assim, um método de controle autônomo deverá ser capaz de agir com informações parciais sobre seu ambiente de operação, ou seja, com dados provenientes da sua situação local no espaço-tempo. Os métodos tradicionais baseados em planejamento de trajetórias não possuem esta capacidade, pois dependem de um modelo interno do mundo real. Esta representação simbólica interna é ba maioria das vezes, incompleta, necessitando de atualizações constantes. A atualização constante de um modelo complexo impede a reação em tempo hábil do sistema, ou na melhor hipótese, o levará a agir baseado em dados ultrapassados. O melhor modelo do mundo real é o próprio mundo. Desta forma, um método de navegação autônoma deverá "localizado", i.e., basear suas ações nos estímulos provindos do ambiente de operação e não em um modelo idealizado interno.

O problema de referência está ligado à fonte e à forma do conhecimento adquirido ou embutido no sistema autônomo. O projeto de um método de navegação involve sempre a inserção de conhecimento externo, vindo principalmente de projetistas humanos. Este conhecimento é gerado pela interação do humano com o problema de navegação a ser solucionado. Sua forma estará, portanto, intrinsicamente ligada à visão humana do problema e aos sensores humanos. Como os sensores e o modo de processamento do agente autônomo diferem daqueles dos humanos, a visão e a referência de conhecimento humanos não se adequam ao agente autônomo.O conhecimento fornecido *a priori* representa um visão estática e parcial do ambiente. Como o problema de navegação autônoma possui parâmetros dinâmicos, o "modelo" traçado por este conhecimento se defasará rapidamente, exceto se formar uma "janela" de conhecimento, construído a partir de estímulos/dados locais. Em resumo, o agente autônomo deverá depender o mínimo possível de conhecimento fornecido *a priori*, abstraindo as informações relevantes do ambiente de operação pela interação entre o próprio agente e o mundo.

Os métodos tradicionais de controle não podem ser aplicados a esta classe de problemas pois necessitam de um modelo bastante preciso do veículo e não possuem meca-

nismos para lidar com a complexa dinâmica de um ambiente de operação típico do controle autônomo.

A *computação flexível* apresenta-se como uma proposta promissora para a navegação autônoma. A computação flexível fundamenta-se em três itens básicos: redes neurais artificiais, sistemas nebulosos e o raciocínio probabilístico, este último englobando a teoria do caos, os algoritmos evolucionários, raciocínio Bayesiano etc.

A utilização conjunta destes itens fornece um sistema híbrido atendendo aos requisitos do controle autônomo. Este aspecto simbiótico induz três características principais: mecanismos de aprendizagem não-supervisionada, tolerância a dados imprecisos e robustez.

1.2 Objetivos

O objetivo deste trabalho é o de desenvolver e analisar a viabilidade de um método de controle autônomo baseado nos três paradigmas de computação flexível: redes neurais, sistemas nebulsos e algoritmos genéticos.

Três aspectos serão avaliados: capacidade de aprendizagem não supervisionada (oferecida pelo algoritmo genético), facilidade de extração de conhecimento aprendido e desempenho do método no problema de navegação autônoma, quando comparado com um sistema baseado em heurística.

1.3 Organização da Tese

A dissertação está organizada de acordo com os seguintes capítulos:

- Após esta introdução o Capítulo 2 fornece uma descrição suscinta das técnicas de computação flexível utilizadas no trabalho;
- No Capítulo 3 discute-se os aspectos relevantes do problema de controle autônomo, enfocando a navegação autônoma, descrição sucinta dos métodos existentes e exposição do método alternativo desenvolvido;

- A seguir são apresentados e discutidos os resultados de simulação do método proposto em diversos ambientes;
- Finalmente o Capítulo 5 apresenta a conclusão, incluindo uma avaliação do método de navegação autônoma desenvolvido e a sugestão de tópicos relevantes para futura pesquisa.

Capítulo 2

Fundamentos: Computação Flexível

2.1 Introdução

Computação flexível difere da computação tradicional por tolerar imprecisão, incertezas e informações parciais. Esta tolerância é explorada visando obter soluções robustas e de baixo custo computacional para problemas considerados intratáveis por técnicas tradicionais. Atualmente os principais constituintes da computação flexível são os sistemas nebulosos, as redes neurais e o raciocínio probabilístico. Este último incorpora autômatos celulares, teoria do caos, algoritmos genéticos e alguns pontos da teoria da aprendizagem. A computação flexível não é apenas a junção destes diversos métodos, mas sim uma simbiose destes. As diferentes características de cada método computacional se complementam, compensando as deficiências de cada parte e reforçando suas vantagens comuns.

Neste capítulo examinaremos três métodos de computação flexível: as redes neurais, os sistemas nebulosos e os algoritmos genéticos.

Estas técnicas formam a base do método de navegação autônoma de veículos proposto nesta dissertação.

2.2 Lógica Nebulosa e Sistemas Nebulosos

2.2.1 Introdução

2.2.1.1 O que é lógica nebulosa?

A lógica multivalorada é uma extensão do lógica Booleana. Ela é baseada na teoria dos conjuntos nebulosos (*fuzzy sets*, [50]), a qual, por sua vez, é uma extensão da teoria dos conjuntos. A teoria dos conjuntos nebulosos admite um variação gradual da pertinência de um elemento a um conjunto (graus de pertinência), em contraponto à teoria tradicional, a qual admite apenas a pertinência ou não-pertinência. A teoria dos conjuntos tradicional passa ser denominada *crisp* e se torna uma instância da teoria dos conjuntos nebulosos.

Este conceito de pertinência parcial possibilita a descrição de eventos inexatos, como a altura de uma pessoa e/ou subjetivos, como a beleza de um pássaro, além de solucionar paradoxos da lógica booleana, como o do barbeiro e o do mentiroso de Creta e problemas tipo *sorites*, como a do monte de areia. Os paradoxos da lógica booleana decorrem do prin'cipio do meio excluído, ou seja, A $\cap$ não-A $= \phi$.

Os conceitos da teoria de conjuntos nebulosos permitem representar objetos de forma similar àquela utilizada na linguagem e o raciocínio humano.

2.2.1.2 Possibilidade vs. Probabilidade

O conceito de possibilidade é muito confundido com o conceito de probabilidade. Argumenta-se que ambos fornecem uma medida de certeza sobre um evento e seriam portanto, equivalentes.

A teoria da possibilidade difere conceitualmente da teoria probabilística. Enquanto a possibilidade de um evento mede o grau de pertinência deste evento a uma certa classe, a probabilidade de um evento fornece uma medida sobre a ocorrência ou não de um evento, face a um conjunto de experiências realizadas com aquele evento.

Seja, por exemplo, um conjunto de garrafa de conteúdo desconhecido e dentre

estas, uma garrafa com um rótulo. Considere dois rótulos diferentes:

- rótulo probabilístico: Garrafa A, $p_{\text{potável}}(A) = 0.9, p \in [0,1]$
- rótulo possibilístico: Garrafa B, $\lambda_{\text{potável}}(B) = 0.9, \lambda_{\text{potável}}(x) \in [0,1]$

O rótulo probabilístico afirma que, de n amostras retiradas do conjunto de garrafas, $0.9n$ amostras (garrafas) contiveram um líquido potável. Esta medida se aplica a todas as outras garrafas da população. O rótulo possibilístico afirma que o conteúdo da garrafa rotulada pertence à classe de líquidos potáveis com grau 0.9. Esta medida se aplica apenas a esta garrafa.

Examinemos o conteúdo destas duas garrafas rotuladas. Digamos que em ambas encontremos ácido clorídrico (HCl). Ora, o rótulo possibilístico terá que ser modificado, enquanto o rótulo probabilístico continua válido (supondo-se a inclusão desta garrafa no espaço de amostras).

2.2.1.3 Tópicos abordados

O restante desta seção está organizado do seguinte modo: uma breve introdução aos conceitos e definições da teoria dos conjuntos nebulosos, a qual estabelece uma base para a compreensão do controlador nebuloso descrito em seguida. Ao final tecemos alguns comentários sobre sistemas nebulosos.

2.2.2 Conceitos

Apresentaremos nesta seção as definições e conceitos básicos necessários à compreensão da teoria dos conjuntos nebulosos e do funcionamento de um controlador nebuloso. Para um estudo mais detalhado, consulte as referências [50], [32], [23], [29] nas quais foi baseada esta seção.

2.2.2.1 Definições Básicas

Seja o espaço de objetos U composto de elementos u. U é denominado o *universo de discurso* e pode ser descrito por $U = \{u\}$. Um conjunto nebuloso A é um conjunto de

pares ordenados

$$A = (u, \chi(u)), u \in U$$

onde $\chi(u)$ é o grau de pertinência de u em A e geralmente pertence a $[0,1]$. $\chi(u) = 0$ indica a não-pertinência de u em A e $\chi(u) = 1$ a pertinência completa de u em A.

Existem ainda outras formas de representar um conjunto nebuloso. Para um conjunto nebuloso A discreto e finito

$$A = \chi_A(u_1)/u_1 + \chi_A(u_2)/u_2 + \cdots + \chi_A(u_n)/u_n \tag{2.1}$$

ou

$$A = \sum_{j=1}^{n} \chi_A(u_j)/u_j \tag{2.2}$$

e no caso de um conjunto A contínuo:

$$\int_U \chi_A(u_j)/u_j \tag{2.3}$$

O suporte de um conjunto nebuloso A é o conjunto de elementos u pertencentes a A, nos quais $\chi_A(u) > 0$. Um *singleton* nebuloso é um conjunto nebuloso cujo suporte é um único elemento u no universo de discurso U com $\chi_A(u) = 1$. O(s) *ponto(s) de crossover* de um conjunto nebuloso A são os elementos u em U cuja pertinência em A é dada por $\chi_A(u) = 0.5$.

2.2.2.2 Operações sobre Conjuntos Nebulosos

Sejam A e B dois conjuntos nebulosos definidos em U com funções de pertinência χ_A e χ_B, respectivamente. Define-se as operações de união, interseção, complementação e produto cartesiano da seguinte forma:

– União:

$$A \cup B = \{(u, \chi_{A \cup B}(u)) \mid \chi_{A \cup B}(u) = max(\chi_A(u), \chi_B(u))\} \tag{2.4}$$

– Interseção:

$$A \cap B = \{(u, \chi_{A\cap B}(u)) \mid \chi_{A\cap B}(u) = min(\chi_A(u), \chi_B(u))\} \tag{2.5}$$

– Complemento:

$$\bar{A} = \{(u, \chi_{\bar{A}}(u)) \mid \chi_{\bar{A}}(u) = 1 - \chi_A(u)\} \tag{2.6}$$

– Produto Cartesiano:

Dados os conjuntos nebulosos $A_1, A_2, \ldots, A_n$ definidos em $U_1, U_2, \ldots, U_n$, respectivamente, o produto cartesiano $A_1 \times A_2 \times \cdots \times A_n$ no espaço $U_1 \times U_2 \times \cdots \times U_n$ é dado por:

$$\begin{aligned} A_1 \times A_2 \times \cdots \times A_n \quad = \quad & \{((u_1, u_2, \ldots, u_n), \chi_{A_1\times A_2\times\cdots\times A_n}(u_1, u_2, \ldots, u_n)) \mid \\ & \chi_{A_1\times A_2\times\cdots\times A_n}(u_1, u_2, \ldots, u_n) = \\ & \min(\chi_{A_1}(u_1), \chi_{A_2}(u_2), \cdots, \chi_{A_n}(u_n))\} \end{aligned}$$

2.2.2.3 Relações Nebulosas

Uma *relação nebulosa* R entre os elementos de um conjunto X e os elementos do conjunto Y é um subconjunto nebuloso do produto cartesiano $X \times Y$. R é caracterizado pela função de pertinência $\chi_R(x, y)$ e é expressa por

$$R = \int_{X\times Y} \chi_R(x, y)/(x, y), \qquad x \in X, y \in Y \tag{2.7}$$

De modo genérico, uma relação n-ária R é um subconjunto de $X_1 \times X_2 \times \cdots \times X_n$ caracterizada por

$$R = \int_{X_1\times X_2\times\cdots\times X_n} \chi_R(x_1, \cdots, x_n)/(x_1, \cdots, x_n) \qquad x_i \in X_i, i = 1, \cdots, n \tag{2.8}$$

Relações discretas também podem ser representadas, de modo conveniente, por matrizes. Exemplo:

Tabela 2.1: Exemplo de uma Relação nebulosa discreta.

Representação matricial da Relação Nebulosa Discreta $R_{A\to B}$				
$R_{A\to B}$	x_1	x_2	x_3	x_4
y_1	0	0	0.3	0.4
y_2	0.1	1.0	0.3	0.9
y_3	0	0.4	0	0
y_4	0.9	0.4	0.6	0
y_5	0	0.7	0.2	0.4

$x_i \in A$ and $y_i \in B$

Composição Sup-estrela

Seja uma relação R em $X \times Y$ e uma relação S em $Y \times Z$. A *composição* de R e S é uma relação denotada por $R \circ S$ e definida como sendo:

$$R \circ S = \int_{X\times Z} \sup_y \left(\chi_R(x,y) * \chi_S(y,z)\right)/(x,z) \tag{2.9}$$

onde: *sup* denota o supremo e * pode ser qualquer operador do tipo T-norma (apresentados em 2.2.2.4), por exemplo min($\wedge$) ou produto algébrico ($\cdot$, definido em [0,1]).

Os operadores de composição mais comuns são: sup-min [50], sup-product [24], sup-produto limitado [38], sup-produto drástico [38], sendo os dois primeiros as formas mais utilizadas em controladores nebulosos.

Para relações discretas, a composição de relações toma a forma de uma multiplicação matricial, na qual as operações de adição e multiplicação são substituidas respectivamente, por operações de máximo e norma triangular.

Exemplo: Sejam as relações nebulosas discretas $R_{A\to B}$ e $S_{B\to C}$ dadas abaixo.

$$R = \begin{bmatrix} 0 & 0.9 & 0.3 & 0.2 \\ 0.1 & 0.4 & 0.3 & 0.7 \\ 0.3 & 0.8 & 0.5 & 0.1 \\ 0.9 & 0.2 & 0.3 & 0.1 \end{bmatrix} \quad S = \begin{bmatrix} 0.2 & 0.3 & 0.1 & 0.9 \\ 0.3 & 0.8 & 0 & 0.6 \\ 0.4 & 0.7 & 0.3 & 0.1 \\ 0.9 & 0.6 & 0.3 & 0.2 \end{bmatrix}$$

A composição $R \circ S$ será dada pela seguinte matriz:

$$R \circ S = \begin{bmatrix} \max(\min(0,0.2),\cdots,\min(0.2,0.9)) & \cdots & \max(\min(0,0.9),\cdots,\min(0.2,0.2)) \\ \vdots & \ddots & \vdots \\ \max(\min(0.9,0.2),\cdots,\min(0.1,0.9)) & \cdots & \max(\min(0.9,0.9),\cdots,\min(0.1,0.2)) \end{bmatrix}$$

$$= \begin{bmatrix} 0.3 & 0.8 & 0.3 & 0.6 \\ 0.7 & 0.6 & 0.3 & 0.4 \\ 0.4 & 0.8 & 0.3 & 0.6 \\ 0.3 & 0.3 & 0.3 & 0.9 \end{bmatrix}$$

Propriedades das Relações Nebulosas

- $R_1 \circ (R_2 \circ R_3) = (R_1 \circ R_2) \circ R_3$
- $R_3 \circ (R_1 \cup R_2) = (R_3 \circ R_1) \cup (R_3 \circ R_2)$
- $R_3 \circ (R_1 \cap R_2) \subseteq (R_3 \circ R_1) \cap (R_3 \circ R_2)$

2.2.2.4 Funções de Implicação Nebulosas

As funções nebulosas são uma forma de relação nebulosa. Existem basicamente três tipos de funções nebulosas, de acordo com [32]: funções de conjunção, disjunção e implicação. Estas funções serão definidas a seguir, após a apresentação das normas e co-normas T.

Normas triangulares

Uma norma triangular * é uma função comutativa, associativa e monotônica não-decrescente $* : [0,1] \times [0,1] \rightarrow [0,1]$, com as condições de contorno $X * 0 = 0$ e $X * 1 = X$. Esta definição engloba os operadores produto drástico, interseção, produto algébrico, e produto limitado, dados abaixo. A norma triangular mais restrita é a interseção e a menos restrita é o produto drástico. Sejam $x, y \in [0,1]$, então define-se:

interseção	$x \wedge y = \min(x, y)$
produto drástico	$x \cap y = \begin{cases} x & y = 1 \\ y & x = 1 \\ 0 & x, y < 1 \end{cases}$
produto limitado	$x \odot y = \max(0, x + y - 1)$
produto algébrico	$x.y = xy$

Co-normas triangulares

Uma co-norma triangular $\dot{+}$ é uma função comutativa, associativa e monotônica não-decrescente $\dot{+}: [0,1]\times[0,1] \rightarrow [0,1]$, com as condições de contorno $X\dot{+}1 = 1$ e $X\dot{+}0 = X$. Esta definição engloba as funções união, soma algébrica, soma limitada, soma drástica e soma disjunta, dadas abaixo. Sejam $x, y \in [0,1]$, então define-se:

união	$x \vee y = \max(x, y)$
soma algébrica	$x\hat{\dot{+}}y = x + y - xy$
soma limitada	$x \oplus y = \min(1, x + y)$
soma drástica	$x \cup y = \begin{cases} x & y = 0 \\ y & x = 0 \\ 1 & x, y > 0 \end{cases}$
soma disjunta	$x\Delta y = \max[\min(x, 1 - y), min(1 - x, y)]$

Disjunção

A disjunção nebulosa é definida para todo $u \in U$ e $v \in V$ por

$$A \rightarrow B = A\dot{+}B = \int_{U\times V} \chi_A(u)\dot{+}\chi_B(v)/(u, v)$$

onde $\dot{+}$ é uma co-norma triangular.

Conjunção

A conjunção nebulosa é definida para todo $u \in U$ e $v \in V$ por

$$A \rightarrow B = A * B = \int_{U\times V} \chi_A(u) * \chi_B(v)/(u, v)$$

onde * é uma norma triangular.

Implicação

A implicação nebulosa está associada com cinco famílias de funções de implicação nebulosas. Aqui, como antes, * representa uma norma triangular e $\dot{+}$ uma co-norma triangular.

– Implicação material:

$$A \rightarrow B = (\text{not}A)\dot{+}B$$

– Cálculo proposicional:

$$A \rightarrow B = (\text{not}A)\dot{+}(A * B)$$

– Cálculo proposicional estendido:

$$A \rightarrow B = (\text{not}A \times \text{not}B)\dot{+}B$$

– Generalização do Modus Ponens:

$$A \rightarrow B = \sup\{c \in [0,1],\ A * c \leq B\}$$

– Generalização do Modus Tollens:

$$A \rightarrow B = \inf\{t \in [0,1],\ B\dot{+}t \leq A\}$$

Entre as múltiplas formas de implicação nebulosa geradas pela associação dos diversos tipos de normas e co-normas triangulares com as cinco definições de implicação nebulosa, destacam-se as seguintes, por sua freqüência de uso:

– regra de implicação nebulosa tipo mínimo de Mamdani

$$R_c = A \times B = \int_{U\times V} \chi_A(u) \wedge \chi_B(v)/(u,v)$$

– regra de implicação nebulosa tipo produto de Larsen

$$R_p = A \times B = \int_{U\times V} \chi_A(u)\chi_B(v)/(u,v)$$

– regra de implicação nebulosa aritmética de Zadeh

$$R_a = (\text{not } A \times V) \oplus (U \times B) = \int_{U\times V} 1 \wedge (1 - \chi_A(u) + \chi_B(v))/(u,v)$$

– regra de implicação nebulosa tipo maxmin de Zadeh

$$R_m = (A \times B) \cup (\text{not } A \times V) = \int_{U\times V} (\chi_A(u) \wedge \chi_B(v)) \vee (1 - \chi_A(u))/(u,v)$$

– implicação nebulosa tipo seqüência padrão

$$R_s = A \times V \rightarrow U \times B = \int_{U \times V} (\chi_A(u) > \chi_B(v))/(u,v)$$

onde

$$\chi_A(u) > \chi_B(v) = \begin{cases} 1 & \chi_A(u) \leq \chi_B(v) \\ 0 & \chi_A(u) > \chi_B(v) \end{cases}$$

– implicação nebulosa Booleana

$$R_b = (\text{not } A \times V) \cup (U \times B) = \int_{U \times V} (1 - \chi_A(u)) \vee (\chi_B(v))/(u,v)$$

– implicação nebulosa de Goguen

$$R_\delta = A \times V \rightarrow U \times B = \int_{U \times V} (\chi_A(u) \gg \chi_B(v))/(u,v) = \begin{cases} 1 & \chi_A(u) \leq \chi_B(v) \\ \frac{\chi_B(u)}{\chi_A(v)} & \chi_A(u) > \chi_B(v) \end{cases}$$

2.2.2.5 Variáveis Lingüísticas

Um número nebuloso F em um universo de discurso contínuo U é um conjunto nebuloso F normal e convexo em U, ou seja:

$$\text{normal:} \qquad \max_{u \in U} \chi(u) = 1 \tag{2.10}$$

$$\text{convexo:} \quad \chi_F(\lambda u_1 + (1-\lambda)u_2) \geq min(\chi_F(u_1), \chi_F(u_2)), \quad u_1, u_2 \in U, \lambda \in [0,1] \tag{2.11}$$

A Figura 2.1 ilustra o conceito de convexidade.

As variáveis lingüísticas podem ser definidas por meio de um número nebuloso ou por meio de termos lingüísticos. Pode-se representar uma variável lingüística por uma quíntupla:

$$(x, T(x), U, G, M)$$

onde x é o nome da variável, $T(x)$ é o conjunto de termos de x, ou seja, o conjunto de nomes relacionados aos valores lingüísticos de x, sendo cada valor um número

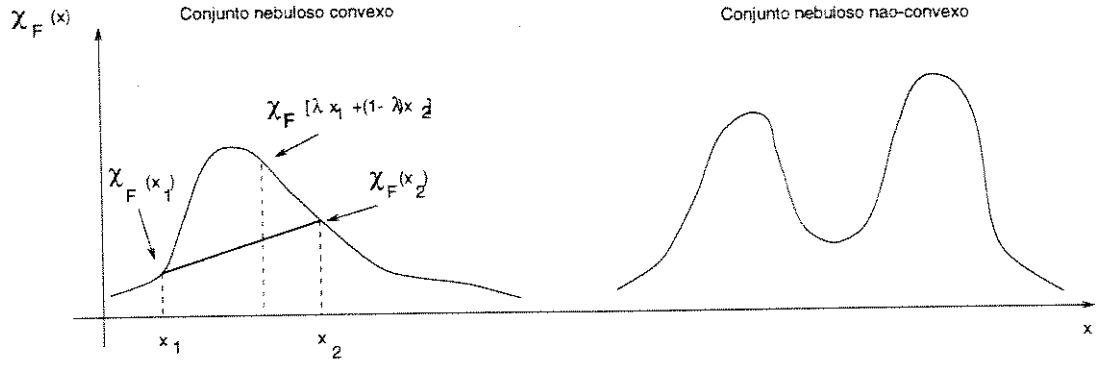


Figura 2.1: Ilustração do conceito de convexidade.

nebuloso definido em U, G é regra sintática geradora dos nomes dos valores x, M é uma regra semântica associando cada valor com seu significado.

Exemplo: Seja a variável lingüística denominada *distância*. Um dos possíveis conjuntos de termos T(distância) seria:

$$T(\text{distância}) = \{muito\ perto, perto, mediana, longe, muito\ longe, \cdots\}$$

Cada termo em T(distância) é caracterizado por um conjunto nebuloso no universo de discurso $U = [0, 1000]$. Uma possível regra semântica M poderia associar o rótulo "longe" com o significado "distância acima de 700 metros", "mediana" com "distância em torno de 500 m" e "perto" com "distância abaixo de 200 m". Estes termos podem ser caracterizados por conjuntos nebulosos cujas funções de pertinência estão mostradas na Figura 2.2.

Conectivos

O conectivo lingüístico **and** utilizado em regras com múltiplas entradas e/ou saídas é associado tanto à operação de disjunção quanto à conjunção.

O conectivo lingüístico **also** é utilizado na agregação de um conjunto de regras. Tipicamente o operador associado a este conectivo é a união. A Tabela 2.6 mostra pares apropriados de operadores (implicação, conectivo), de acordo com os critérios das tabelas

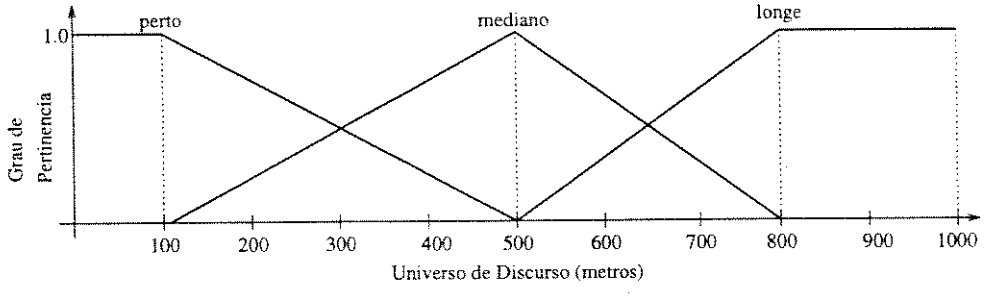


Figura 2.2: Exemplo de uma variável lingüística (distância).

2.5 e 2.2.3.3. Os resultados foram apresentados em [Lee 73] e são baseados no controle de uma planta com atraso de primeira ordem.

2.2.3 Controlador Nebuloso

2.2.3.1 Estrutura

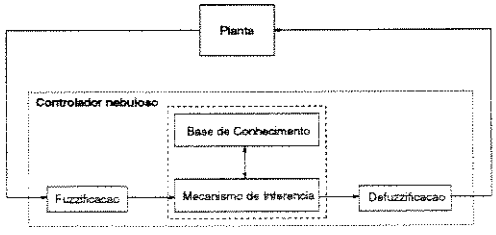


Figura 2.3: Diagrama de blocos de um controlador nebuloso.

Um controlador nebuloso é um controlador que utiliza regras nebulosas para controlar um processo. O uso de regras nebulosas e de variáveis lingüísticas confere ao controlador nebuloso várias vantagens, entre as quais podemos citar:

- simplificação do modelo do processo;
- redução de custos através do uso de sensores de menor precisão;
- relacionamento mais próximo com o usuário;

- implementação física mais simples;
- satisfação de múltiplos objetivos de controle;
- fácil incorporação do conhecimento de especialistas humanos.

Os dados provenientes dos sensores e os sinais s serem enviados aos atuadores de um sistema de controle são de natureza não-nebulosa. Portanto faz-se necessário uma interface entre o controlador nebuloso e o processo a ser controlado. Esta interface é implementada sob as formas de um *fuzzificador* e um *defuzzificador* (veja figura 2.3). O fuzzificador converte dados crisp em valores nebulosos e o defuzzificador atua de modo contrário, convertendo valores nebulosos em valores *crisp*.

O correto funcionamento deste filtros depende da correta caracterização dos universos de discurso das variáveis de entrada e de saída, sob a forma de parâmetros contidos no *base de conhecimento*.

O cerne do controlador nebuloso consiste da *base de conhecimento* e do *mecanismo de inferência*.

A base de conhecimento se divide em uma *base de dados* e uma *base de regras*. A base de dados contém as informações que especificam o modo através do qual o controlador interage com o processo controlado. Nela estão contidos o domínio e as partições dos universos de discurso das variáveis de entrada e das de saída, bem como a quantização utilizada, caso o universo de discurso seja discreto. A base de regras contém a estratégia de controle representada por relações (estado, ação), expressas por regras nebulosas do tipo:

$$\text{if } (x_1 \text{ is } A_1) \text{ and } \cdots \text{ and } (x_n \text{ is } A_n) \text{ then } (y_1 \text{ is } B_1) \text{ and } \cdots \text{ and } (y_m \text{ is } B_m) \quad (2.12)$$

onde $(x_1, \cdots, x_n)$ são as variáveis ling'úísticas associadas às entradas e $(A_1, \cdots, A_n)$ são rótulos lingüísticos associados às variáveis de entrada. $(y_1, \cdots, y_n)$ são as variáveis linguísticas de saída e $(B_1, \cdots, B_n)$ são os termos lingüísticos associados às variáveis de saída.

Nos itens seguintes cada componente do diagrama do controlador nebuloso será descrito em maior profundidade.

2.2.3.2 Base de conhecimento

A base de conhecimento é composta por duas partes: a base de dados e a base de regras lingüísticas. A base de dados contém as definições das variáveis nebulosas utilizadas pelas regras lingüísticas, i.e., os universos de discurso, as partições e os rótulos lingüísticos associados de cada variável. Já a base de regras contém a estratégia de controle sob a forma de regras lingüísticas.

Base de Dados

A base de dados contém todas as informações pertinentes ao modo como o controlador interage com o ambiente no qual está inserido. Estas informações englobam aquelas associadas à especificação de qualquer controlador (quantidade, qualidade e tipo de variáveis de entrada e saída) mais aquelas específicas a sua natureza nebulosa (particionamento, funções de pertinência, termos lingüísticos, método de implicação e de composição, forma de fuzzificação, forma de defuzzificação).

O universo de discurso de uma variável nebulosa define o domínio desta, ou seja, os valores não nebulosos u sobre os quais esta variável está definida. No exemplo mostrado na figura 2.4, o universo de discurso é o intervalo [-10, 100] graus centígrados.

O particionamento do universo de discurso associa intervalos do universo de discurso a termos lingüísticos utilizados nas regras de controle. Na figura 2.4 os termos *quente*, *morno*, *frio*, *fresco*, *gelado* são associados à conjuntos nebulosos definidos nos subintervalos [40, 100], [25, 43], [-5, 12], [10, 27] e [-10, -3], respectivamente. A partição do universo de discurso e a atribuição de termos adequados às partições são processos subjetivos e geralmente refletem o conhecimento existente sobre a variável. Por exemplo, a atribuição do termo lingüístico *gelado* ao intervalo [40,50] é possível mas não recomendável pois não reflete o conhecimento existente sobre o assunto. A função de pertinência relaciona um dado ponto de cada partição a um valor de pertinência, a qual indica o grau de pertinência daquele ponto ao conjunto representado por aquele termo. As funções de pertinência possuem várias formas, sendo as mais comuns o triângulo e o trapézio, entre as funções lineares e a gaussiana entre as funções contínuas.

A escolha dos métodos de implicação e composição será discutido no ítem seguinte, Base de Regras.

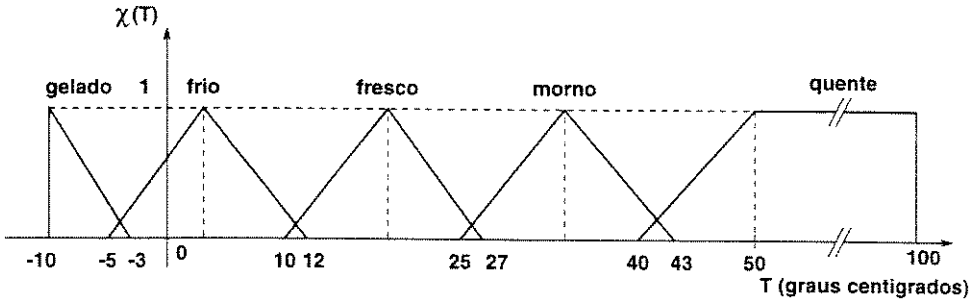


Figura 2.4: Exemplo da determinação de uma variável nebulosa de controle.

A determinação destas especificações é normalmente norteado pela experiência de um ou mais humanos familiarizados com o processo. A configuração inicial nem sempre é a melhor. A especificação satisfatória pode ocorrer após vários ciclos de teste. Para processos sobre o controle dos quais pouco se conhece, seja por sua dinâmica muito complexa, seja por ser seu recente surgimento, utiliza-se métodos de aprendizagem artificial, como as redes neurais e os algoritmos genéticos (veja as seções seguintes). Durante aprendizagem, os subintervalos são descritos por termos meta-lingüísticos. Estes termos são simbólicos e não possuem significado lingüístico definido, servindo apenas como ligação entre as premissas das regras e as variáveis nebulosas. Este artifício se faz necessário devido aos constantes rearranjos das partições. Ao longo da aprendizagem um termo do tipo "GRANDE POSITIVO" poderá ser deslocado para uma posição heurísticamente atribuído ao termo "MÉDIO NEGATIVO", por exemplo. A atribuição de termos significativos lingüísticamente só poderá ocorrer após a estabilização das partições.

Base de Regras

A base de regras especifica o comportamento desejado do controlador, sob forma de um conjunto de regras nebulosas, em conjunto com os métodos de composição e inferência utilizados por estas regras.

As regras nebulosas são relacões nebulosas, representadas por funções de implicação nebulosas e expressas lingüísticamente de duas formas:

– Regras Descritivas

São caracterizadas pela identificação de um estado e a escolha de um consequente associado àquele estado. Possuem a forma:

$$\text{if } (x_1 \text{ is } A_1) \text{ and } \cdots \text{ and } (x_n \text{ is } A_n) \text{ then } (y_1 \text{ is } B_1) \text{ and } \cdots \text{ and } (y_m \text{ is } B_m) \tag{2.13}$$

onde $x_1, \cdots, x_n$ são variáveis (lingüísticas) nebulosasas de entrada, $A_1. \cdots, A_n$ são os termos (*labels*) lingüísticos associadas a partições do universo de $x_1, \cdots, x_n$ respectivamente e $y_1, \cdots, y_m$ são variáveis (lingüísticas) nebulosas de saída, sendo $B_1, \cdots, B_m$ os termos (*labels*) lingüísticos associadas a partições do universo de $y_1, \cdots, y_m$ respectivamente.

– Regras Preditivas

Esta espécie de regra se assemelha mais ao raciocínio humano. Escolhe-se uma ou um conjunto de ações de controle baseado numa predição dos seus efeitos sobre um conjunto relevante de parâmetros. Possuem a forma:

$$\text{if } (u \text{ is } C_i \rightarrow (x \text{ is } A_i \text{ and } y \text{ is } B_i)) \text{ then } u \text{ is } C_i \tag{2.14}$$

onde u é a ação de controle sob avaliação, x e y são os parâmetros de avaliação e C_i, A_i e B_i são os estados nebulosos associados a esta regra e às variáveis u, x e y, respectivamente.

A conseqüência C'_i derivada a partir de uma regra R_i é obtida através da aplicação da composição sup-estrela, juntamente com uma das diversas formas da função nebulosa de implicação e do conectivo " and". A partir de uma regra do tipo

$$\text{if } x_1 \text{ is } A_i \text{ and } x_2 \text{ is } B_i \text{ then } z \text{ is } C_i$$

e desde que $x_1 \text{is} A'$ e $x_2 \text{is} B'$, a conseqüência $z = C'_i$ é determinada por:

$$C'_i = (A', B') \circ R_i, \tag{2.15}$$

$$\begin{aligned} \chi_{R_i} &= \chi_{(A_i \text{ and } B_i \rightarrow C_i)}(u, v, w) \\ &= [\chi_{A_i}(u) \text{ and } \chi_{B_i}(v)] \rightarrow \chi_{C_i}(w) \end{aligned} \tag{2.16}$$

A escolha de uma determinada forma de implicação pode ser baseada na satisfação dos critérios contidos na tabela 2.2.3.3. Lee [32] determinou que as funções de implicação que melhor satisfazem tais critérios são: a implicação tipo seqüência padrão R_s, a implicação tipo mínimo de Mamdani R_c e a implicação tipo produto de Larsen R_p.

2.2.3.3 Mecanismo de Inferência

O mecanismo de inferência é baseado na regra de inferência composta sup-estrela e nas definições de implicação nebuloso e dos conectivos " and" e " also". O disparo das regras é simultâneo e em paralelo. Existem duas regras de inferência nebulosa: a Modus Ponens Generalizada (GMP) (tabela 2.3) e a Modus Tollens Generalizada (GMT) (tabela 2.2) . Os controladores nebulosos utilizam a forma Modus ponens, portanto será esta a forma referida no restante desta seção.

Tabela 2.2: Modus Tollens Generalizado

Modus Tollens Generalizado (GMT)	
premissa 1:	y is B'
premissa 2:	if x is A then y is B
conseqüência:	x is A'

Tabela 2.3: Modus Ponens Generalizado

Modus Ponens Generalizado (GMP)	
premissa 1:	x is A'
premissa 2:	if x is A then y is B
conseqüência:	y is B'

Seja $R = \cup_{i=1}^{n} R_i$ o conjunto de n regras do tipo " if x_1 is A_i and x_2 is B_i then z_i is C_i" da base de regras. Então a conseqüência nebulosa C' será dada por:

$$C' = (A', B') \circ \bigcup_{i=1}^{n} R_i$$

$$= \bigcup_{i=1}^{n} ((A', B') \circ R_i)$$

Tabela 2.4: Critérios intuitivos relacionando os termos Prel e Cons para um dado Pre2 em GMP

	x is A' (Prel)	y is B' (Cons)
Critério 1	x is A	y is B
Critério 2A	x is muito A	y is muito B
Critério 2B	x is muito A	y is B
Critério 3A	x is mais-ou-menos A	y is mais-ou-menos B
Critério 3B	x is mais-ou-menos A	y is B
Critério 4A	x is não A	y is desconhecido
Critério 4B	x is não A	y is não B

Tabela 2.5: Critérios intuitivos relacionando os termos Prel e Cons para um dado Pre2 em GMT

	y is B' (Prel)	x is A' (Cons)
Critério 5	y is não B	x is não A
Critério 6	y is não muito B	x is não muito A
Critério 7	y is não mais-ou-menos B	x is não mais-ou-menos A
Critério 8A	y is B	x is desconhecido
Critério 8B	y is B	x is A

$$= \bigcup_{i=1}^{n} ([(A') \circ (A_i \rightarrow C_i)] + [(B') \circ (B_i \rightarrow C_i)]) \tag{2.17}$$

Existem vários procedimentos de inferência, cada qual associada à T-norma utilizada na regra de composição sup-estrela. Na figura 2.5 mostramos a interpretação geométrica de dois procedimentos, o de Larsen (operador produto algébrico) e o de Mamdani (operador min). Quando as variáveis de entrada A' e B' são *singletons* nebulosos, a inferência toma a forma mostrada na figura 2.6.

2.2.3.4 Fuzzificação

O processo de fuzzificação converte um valor *crisp* u em um valor nebuloso w, utilizando as informações de funções de pertinência e a partição do universo de discurso das variáveis associadas àquele valor *crisp*. Existem vários métodos de fuzzificação.

Tabela 2.6: Pares de Operadores (implicação, conectivo *AND*) de Melhor Desempenho

Implicação	Conectivo *also*
$R_c R_p R_{bp} R_{dp}$	$\cup \hat{+} \oplus \cup \delta$
R_a	$\cap \odot \cap$
R_m	-
$R_s R_\delta R_g$	$\cap \odot \cup^a$
R_b	$\dot{\odot} \cup$

^{a}

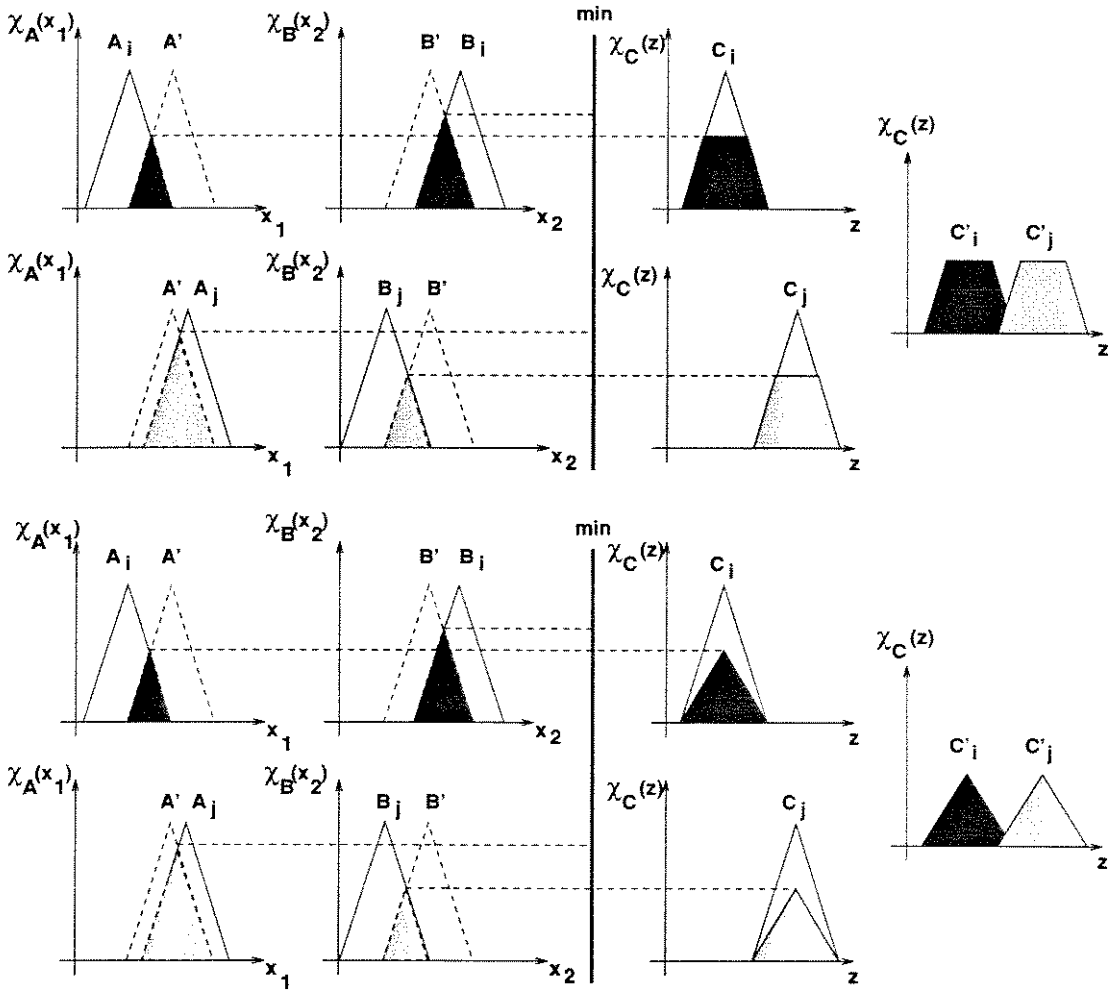


Figura 2.5: Representação gráfica de dois tipos de inferência.

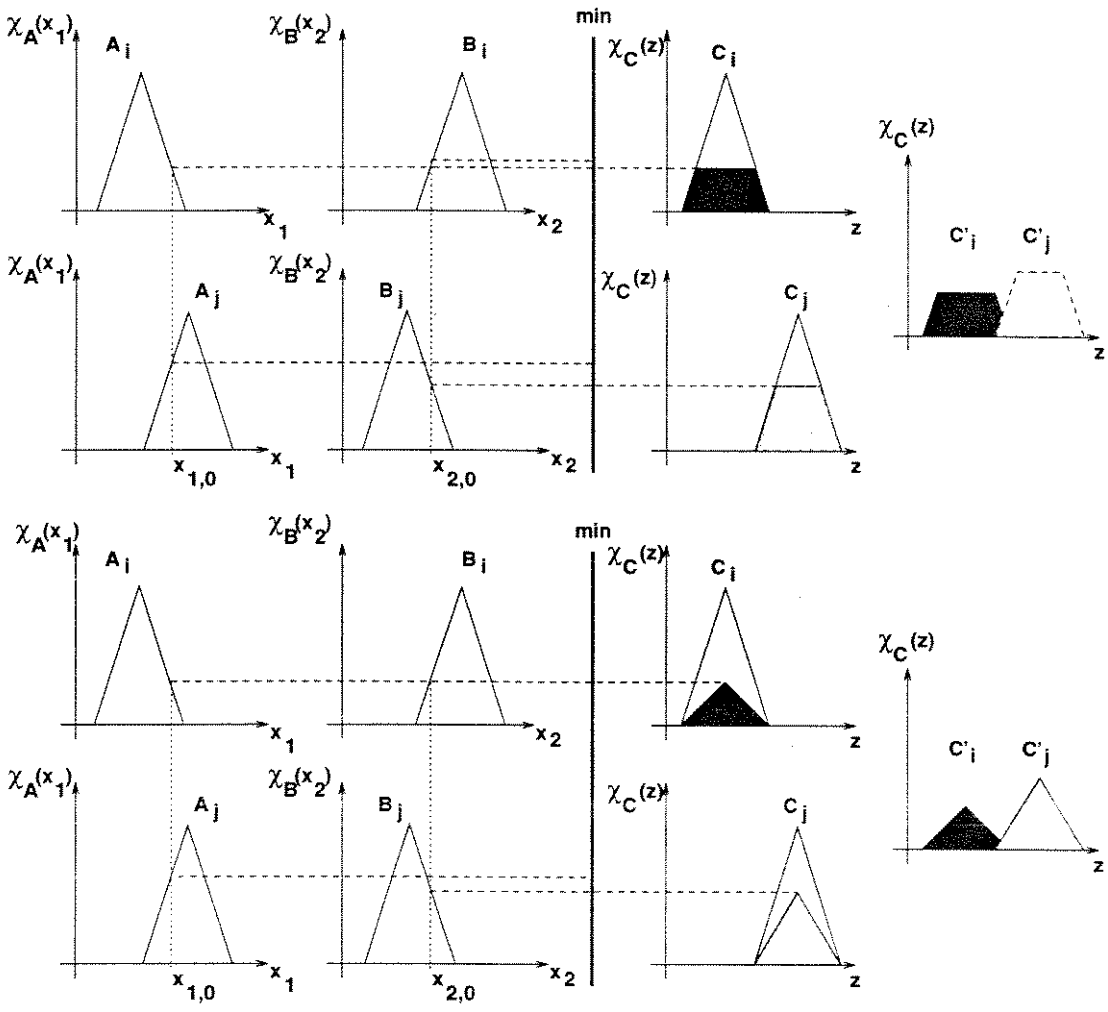


Figura 2.6: Representação gráfica de dois tipos de inferência para antecedentes tipo singleton nebuloso.

– tipo singleton: o valor *crisp* u_0 é convertido em um conjunto nebuloso A com uma função de pertinência $\chi_A(u)$ nula exceto em u_0, onde $\chi_A(u_0) = 1$. É o método mais utilizado (veja parte superior da figura 2.7).

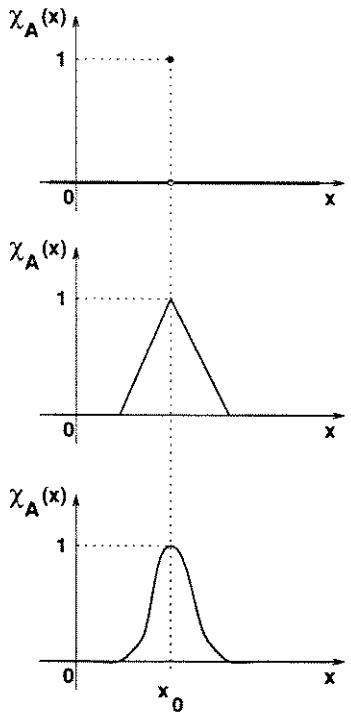


Figura 2.7: Métodos de fuzzificação.

– tipo estocástico: usado quando os valores observados a serem fuzzificados estão acrescidos de ruído. Neste caso, deve-se preservar a informação probabilística gerada pelo ruído na fuzzificação. O valor *crisp* é transformado em um conjunto nebuloso A cuja função de pertinência incorpore tanto a média quanto o desvio padrão dos valores do conjunto de dados associados ao valor *crisp* original. A função de pertinência será então simétrica em relação ao valor médio e com largura proporcional ao desvio padrão observado. Duas funções-exemplo são mostradas na parte inferior da figura 2.7.

– tipo híbrido: este tipo poderá ser utilizado quando se dispõe de informações possibilísticos e estocásticos acerca dos valores a fuzzificar.

2.2.3.5 Defuzzificação

A defuzzificação converte um conjunto nebuloso em um um número ou ação não-nebulosa. Esta conversão é um mapeamento do espaço de ações nebulosas de controle definido sobre o universo de discurso das saídas para o espaço de ações não-nebulosas (*crisp*). A ação *crisp* resultante deverá ser representativo da distribuição de possibilidade da ação nebulosa. Existem vários métodos de defuzzificação de número nebulosos. Os mais comuns são: método do máximo, método da média dos máximos e a de centro de área.

- Método do Máximo

  O número *crisp* gerado por este método é o ponto no qual a distribuição de possibilidade da ação nebulosa de controle atinge seu valor máximo.

- Método da Média dos Máximos

  Esta estratégia gera um valor *crisp* representando o valor médio de todos as variáveis nebulosas cujas funções de pertinência atingem o valor máximo. Para um universo discreto, o valor defuzzificado é dado por:

$$z_0 = \sum_{j=1}^{l} \frac{w_j}{l} \tag{2.18}$$

  onde w_j é o valor de suporte, no qual a função de pertinência atinge o valor máximo $\chi_z(w_j)$ e l é a quantidade destes valores de suporte.

- Método do Centro de Área O valor defuzzificado é o centro de area da distribuição de possibilidade da ação de controle. Para um universo de discurso discreto, tem-se:

$$z_0 = \frac{\sum_{j=1}^{l} \chi_z(w_j) w_j}{\sum_{j=1}^{n} \chi_z(w_j)} \tag{2.19}$$

  onde n é a quantidade de níveis de quantização da saída.

2.3 Redes Neurais (Artificiais)

2.3.1 Introdução

Redes neurais artificiais são estruturas paralelas de computação, baseados nos modelos matemáticos das estruturas neurais existentes nos seres biológicos. O paradigma

das redes neurais artificiais busca a implementação da inteligência artificial através da emulação da arquitetura do cérebro.

A arquitetura de uma rede neural artificial é composta de um conjunto de processadores simples interconectados. Os processadores são baseados em um modelo simplificado do neurônio biológico e estão interconectados por sinapses. Estas sinapses se assemelham às sinapses biológicas, realizando operações (geralmente uma espécie de modulação) sobre a informação que flui através delas. A figura 2.8 mostra uma comparação entre uma rede biológica e uma rede neural artificial.

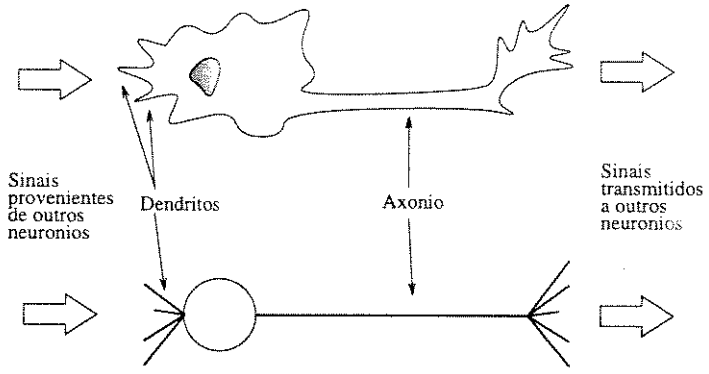


Figura 2.8: Comparação entre um neurônio biológico e um artificial.

O primeiro modelo matemático de um neurônio foi elaborado por McCulloch e Pitts em 1943 [36]. Este modelos iniciais tinham por objetivo a análise do comportamento do neurônio. Para tanto, consideravam os vários mecanismos biofísicos e eletroquímicos apresentados pelo neurônio biológico. Não se cogitava a utilização destes modelos complexos na simulação do comportamento global de uma rede de neurônios.

O primeiro modelo matemático de neurônio elaborado com a aplicação específica de adaptar as propriedades de aprendizagem e processamento paralelo apresentados pelas redes neurais biológicos surgiu com Rosenblatt em 1958 [43]. Este modelo de rede neural, denominado *perceptron* foi utilizado na aprendizagem de funções lógicas, tais como as operações de AND e OR.

Em 1969, Minsky [37] provou que os perceptrons eram capazes de resolver apenas

os problemas de separação linear. A função lógica XOR, por exemplo, não poderia ser aprendida pela rede. Este trabalho marcou o início de uma fase de pouca pesquisa nesta área.

As redes neurais resuscitaram em 1986, com o trabalho de Rumelhart [44]. Rumelhart propôs a inclusão de um camada intermediária, entre as camadas de entrada e saída. Com isto, demonstrou que esta nova rede neural, a perceptron de múltiplas camadas (*multi-layer perceptron*) é capaz de resolver problemas de separação não-linear.

Houve então uma explosão de arquiteturas, métodos de aprendizagem e aplicações. As arquiteturas e os métodos de aprendizagem geralmente são interdependentes. As aplicações são as mais diversas, desde o sensoriamento remoto até a produção de alimentos, passando pela navegação de veículos e pelo reconhecimento de caracteres.

As seções seguintes apresentam conceitos básicos sobre neurônios e redes neurais artificiais, exemplos de redes neurais supervisionadas, exemplos de redes neurais auto-organizadas e por fim uma rede neural neuronebulosa, nesta ordem. Este texto foi baseado em [31], [21] e [11].

2.3.2 Conceitos

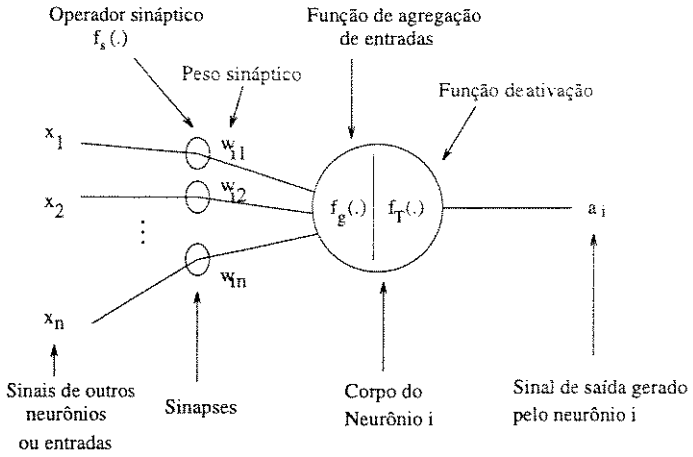


Figura 2.9: Modelo de um neurônio.

O modelo genérico de um neurônio possui a estrutura disposta na figura 2.9. O neurônio recebe sinais provenientes de sensores ou outros neurônios através de conexões de entrada, análogos aos dendritos do neurônio biológico. Os sinais gerados pelo neurônio são transmitidos através das conexões de saída, análogo ao axônio biológico.

As conexões de entrada são dotados de sinapses, os quais modulam o sinal recebido. A modulação é representada por um operador sináptico f_s e por um peso sináptico, w_{ij} (i indica o neurônio o qual está recebendo o sinal e j indica a origem do sinal). O operador sináptico mais utilizado é o produto algébrico:

$$f_s(x_j) = w_{ij}x_j \tag{2.20}$$

O processamento neural dos sinais recebidos é modelado por duas funções: a de agregação de entradas f_g e a função de ativação f_T.

A função de agregação f_g funde os sinais de entrada, fornecendo um valor representativo do conjunto de sinais. Em geral, utiliza-se a somatória:

$$f_g = \sum_{j=1}^{n} f_s(x_j) \tag{2.21}$$

A função de ativação f_T produz um sinal de saída a_i a partir do sinal agregado. Três funções de ativação comuns estão mostradas na figura 2.10. Da direita para a esquerda temos: a função gaussiana, a função sigmóide e a função degrau. Sua forma é não-linear, contendo um ponto de corte ou limiar (veja fig. 2.10). Este formato visa reproduzir o comportamento do neurônio biológico, o qual se torna ativo apenas quando for estimulado para além de um limiar. No caso da função gaussiana, o neurônio estará ativo para estímulos contidos no intervalo $[x_l - \Delta x_l, x_l + \Delta x_l]$.

A função sigmóide é dada por:

$$f_T = \frac{1}{1 - e^{x - x_l}} \tag{2.22}$$

onde x_l é o valor de limiar da função.

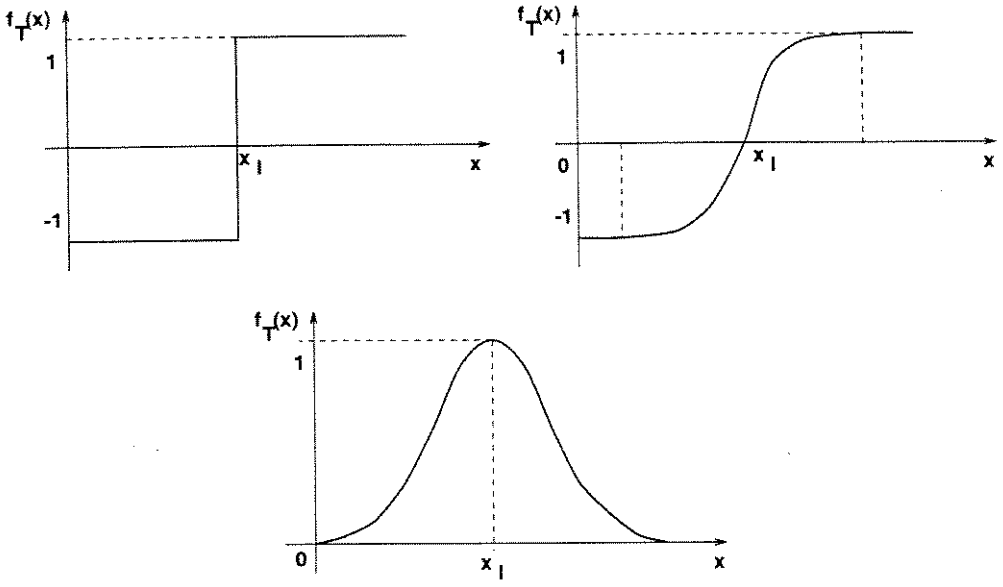


Figura 2.10: Tipos de funções de ativação: degrau, sigmóide e gaussiana.

O valor de limiar x_l pode ser representada por uma conexão adicional de peso x_l ligado a uma entrada de valor fixo -1 (fig 2.11).

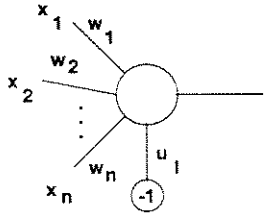


Figura 2.11: Representação de limiar como peso e conexão de valor fixo.

2.3.3 Classificação das Redes Neurais

As redes neurais podem ser classificadas em 3 grupos por duas características básicas: topologia e tipo de aprendizagem.

A *topologia* de uma rede neural é determinada pelo sentido de fluxo de dados através das conexões da mesma:

- não-recorrente: os sinais se propagam em uma única direção, sendo recebidos por um grupo de neurônios receptores, passando por grupos intermediários de neurônios até atingir o grupo de neurônios de saída. Não existem conexões de realimentação, ou seja, conexões entre as saídas de um grupo de neurônios e as entradas dos neurônios do mesmo grupo ou de grupos anteriores. O *Multilayer Perceptron* é um exemplo de rede neural não-recorrente.

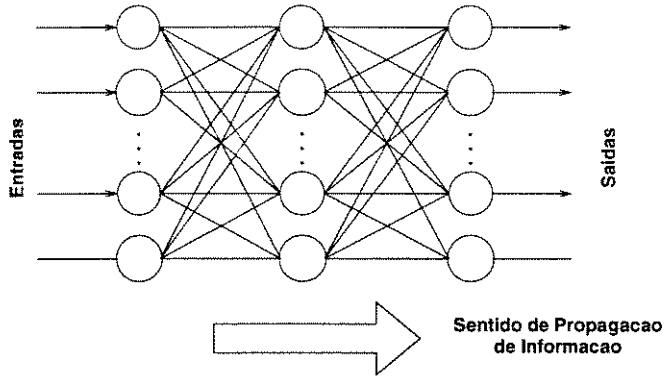


Figura 2.12: Exemplo de uma rede não-recorrente.

- recorrente: redes recorrentes são aqueles com conexões de realimentação. Em algumas instâncias deste classe de redes, os pesos sinápticos são determinados *a priori* e não mais alterados. Os sinais de entrada, ao serem aplicados á rede, provocam uma mudança de estado que eventualmente estabilizam em estados de equilíbrio. É o caso da rede de Hopfield [19].

O conhecimento de uma rede neural está distribuído nas suas sinapses, na forma de pesos sinápticos. A *aprendizagem* de uma rede neural consiste no ajuste destes pesos. Existem dois métodos genéricos de aprendizagem: a supervisionada e a não-supervisionada.

A aprendizagem *supervisionada* utiliza um conjunto de pares de dados na forma (entrada(s) $x_{i,j}$, saída(s) desejada(s) $a^*_{k,j}$), onde $i = 1, \cdots, n$ (n é número de entradas); $k =$

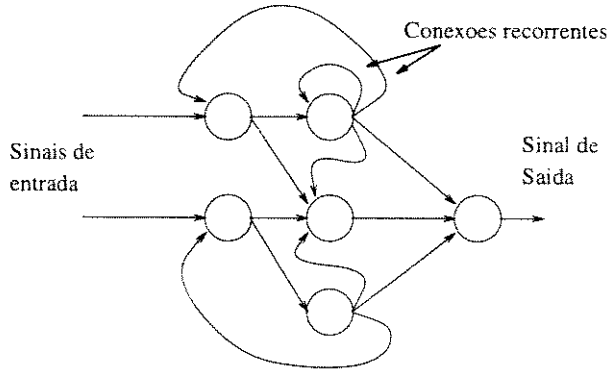


Figura 2.13: Exemplo de uma rede recorrente.

$1, \cdots, m$ (m é o número de saídas); $j = 1, \cdots, p$ (p é o número de pares de aprendizagem), os quais informam a rede qual o comportamento desejado. O ajuste dos pesos é feito com auxílio de uma medida de erro, obtido da comparação entre as saídas $a_{k,j}$ apresentadas pela rede neural e as saídas desejadas $a^*_{k,j}$, referentes às entradas $x_{i,j}$. Algoritmos supervisionados são geralmente baseados em técnicas tipo gradiente.

Na aprendizagem *não-supervisionada* o sistema não dispõe de dados sobre as saídas desejadas, devendo reconhecer padrões existentes no conjunto de dados (de entrada). A rede neural aprende a responder de modo semelhante a conjuntos de entradas semelhantes. Como exemplos desta classe de aprendizagem temos a rede de Kohonen e as redes competitivas [27] [45].

Outros algoritmos de aprendizagem são baseados na regra de aprendizagem de Hebb, proposto em [16]. O princípio básico é reforçar as conexões entre neurônios que se ativam simultaneamente quando a rede é estimulada.

Sejam i e j os neurônios ativos e $w_{i,j}$ o peso da sinapse entre estes. Então a regra de Hebb toma a forma:

$$\Delta w_{i,j} = \gamma a_i a_j \tag{2.23}$$

onde γ é uma constante positiva denominada *razão/fator de aprendizagem.*

Uma segunda regra utiliza a diferença entre o valor desejado da saída a_i^* de um neurônio i e a saída obtida a_i para ajustar os pesos sinápticos $\Delta w_{i,j}$ entre este neurônio i e os neurônios anteriores j:

$$\Delta w_{i,j} = \gamma(a_i^* - a_i)a_j \tag{2.24}$$

Esta regra é conhecida como sendo a regra de *Widrow-Hoff* ou regra *delta.*

Serão vistos a seguir redes neurais representativas das classes expostas acima.

2.3.4 Perceptron

O perceptron multicamada (*multiple layer perceptron (MLP)*) é uma extensão do perceptron, através da inclusão de camadas adicionais entre a de entrada e a camada de saída. Estas camadas intermediárias são denominadas camadas *escondidas* (*hidden*) ou internas (*internal*) e aumentam a complexidade da rede.

Demonstrou-se em [20] que uma rede neural com uma camada escondida e funções de ativação não-lineares são capazes de aproximar com precisão arbitrária, qualquer função com um número finito de descontinuidades.

Como o perceptron, o MLP é do tipo feedforward, ou seja, o fluxo de dados se faz em apenas um sentido, da camada de entrada até a camada de saída.

O treinamento do MLP utiliza uma forma generalizada da regra delta:

$$\Delta w_{i,j} = \gamma \delta_i^p a_j^p \tag{2.25}$$

onde $\Delta w_{i,j}$ é o peso da sinapse entre os neurônios i e j, γ é o fator de aprendizagem, a_j^p é a saída do neurônio j ao estímulo p apresentado à rede e δ_i^p é a medida de erro da saída do neurônio i, dado pela expressão:

$$\delta_i^p = \begin{cases} (d_i^p - a_i^p)F_T'(i_i^p) & \text{se o neurônio } i \text{ for da camada de saída} \\ F'(i_i^p)\sum_{h=1}^{N_o} \delta_h^p w_{h,i} & \text{caso contrário} \end{cases} \tag{2.26}$$

onde F' é a derivada primeira da função de transferência do neurônio i e i_i^p é o estímulo agregado recebido pelo neurônio i quando a rede neural foi alimentada com o conjunto p de estímulos.

O treinamento da rede preceptron multicamadas ocorre em duas etapas:

- apresentação do padrão de entradas à rede e propagação destes estímulos pela rede até a camada de saída.
- cálculo da medida de erro entre a saída obtida e a desejada e a retropropagação (*backpropagation*) desta medida pela rede, com o ajuste dos pesos sinápticos através da aplicação da regra delta generalizada.

Existem dois esquemas básicos de treinamento por gradiente, diferindo no modo como ajustam os pesos sinápticos (figura 2.14). No fluxograma esquerdo (a), faz-se o ajuste dos pesos após apresentar cada conjunto p de estímulos $x_i^p (p = 1, \cdots, P)$. No fluxograma direito(b), utiliza-se um erro acumulado para modificar os pesos após a apresentação de todos os P conjuntos de sinais de entrada. Em ambos os casos, o ajuste dos pesos determina o final de uma iteração.

O primeiro método, quando converge, pode levar a uma especialização da rede nos padrões iniciais ou finais. Uma proposta de solução involve a mudança da ordem de apresentação dos padrões à cada iteração de ajuste dos pesos.

Um outro fenômeno a ser considerado no uso do algoritmo de aprendizagem por gradiente é a relação entre a razão de convergência e o fator de aprendizagem . Para um fator de aprendizagem baixo a convergência é lenta, enquanto que para um fator alto oscilações podem ocorrer devido aos mínimos locais, os quais podem impedir a convergência. Um meio de evitar este dilema é adicionar um termo de *momento* à regra delta generalizada, correlacionando o ajuste do peso com o ajuste anterior:

$$\Delta w_{i,j}(t+1) = \gamma \delta_i^p a_j^p + \alpha \Delta w_{i,j}(t) \tag{2.27}$$

onde t é um índice indicando a iteração atual do algoritmo.

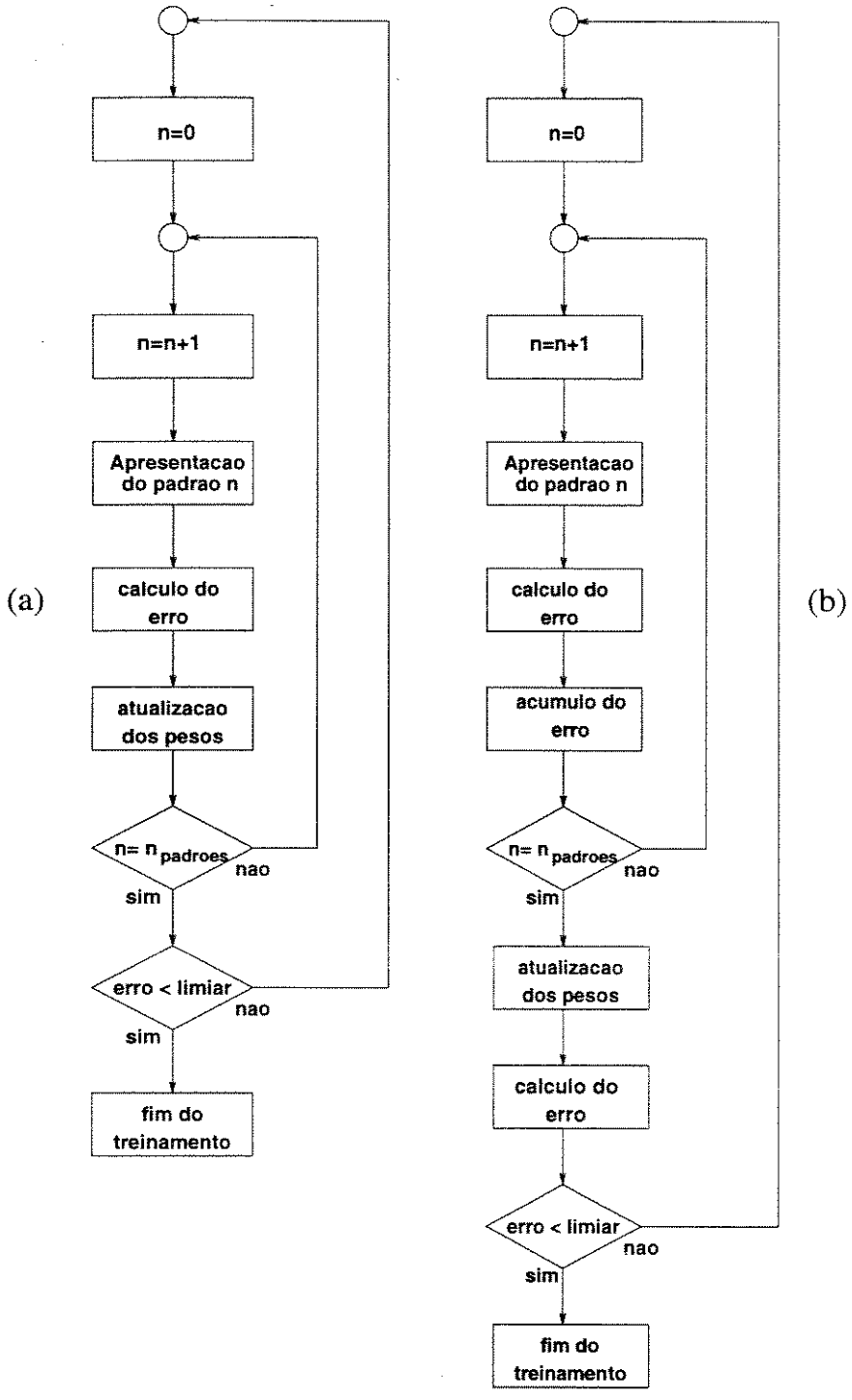


Figura 2.14: Esquemas de Treinamento Supervisionado.

2.3.4.1 O Algoritmo de Busca por Gradiente

Os algoritmos de gradiente utilizam a direção de maior inclinação da função de erro para ajustar os pesos sinápticos. Apesar de garantir a maior variação possível dos pesos em busca do mínimo local, a nova direção pode apresentar uma componente negativa em relação à direção tomada na iteração anterior, levando a um retrocesso quanto à minimização do erro ao longo daquele vetor anterior.

O algoritmo de busca por *conjugado gradiente* [17] evita esta ineficiência determinando um conjunto de vetores ortogonais entre si, de modo que a minimização do erro por um destes vetores não afete a minimização obtida ao longo dos vetores anteriores. Esta modificação no algoritmo de gradiente se baseia na minimiazação de um sistema quadrático com n graus de liberdade através de n buscas sucessivas ao longo das direções conjugadas.

2.3.4.2 Aplicações

As redes neurais supervisionadas têm encontrado larga aplicação no reconhecimento de padrões. O NetTalk desenvolvido por Rosenberg e Sejnowski [46] é capaz de transformar texto escrito em texto falado de modo natural.

O uso das redes supervisionadas em controle de processos é dificultado pela estrutura empregada no controle realimentado (figura 2.15). Dispõe-se apenas de uma associação entre as entradas da rede controladora e as saídas desejadas da *planta*, quando são necessários as saídas desejadas do controlador (usados no cálculo da medida de erro e posterior ajuste dos pesos da rede).

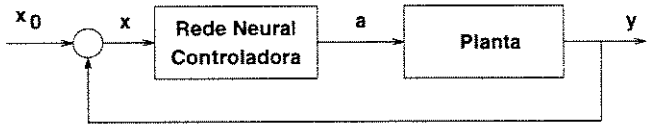


Figura 2.15: Diagrama de um Sistema de Controle por Realimentação.

Foram propostas várias soluções. Widrow propôs a utilização de uma segunda rede perceptron multicamada. Esta rede auxiliar é treinada para emular o comportamento

da planta. Uma vez treinada, seus pesos sinápticos são fixos. Em seguida é utilizada no treinamento da rede de controle para retropropagar a medida de erro até a rede de controle (figura 2.16). Outros autores como Ichikawa e Sawa propuseram o uso de algoritmos alternativos de aprendizagem, como os algoritmos genéticos [22].

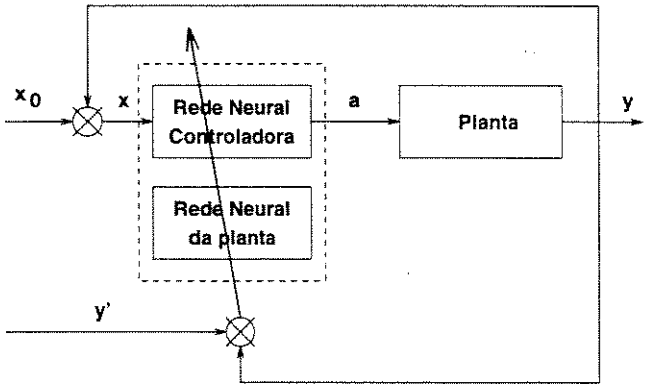


Figura 2.16: Método de Widrow para treinar controladores baseados em redes neurais.

2.3.5 Redes Auto-organizadas

As redes auto-organizadas não necessitam de pares de dados (entrada, saída desejada) para a aprendizagem, pois identificam os padrões existentes no conjunto de dados de entrada através de mecanismos de competição entre neurônios.

Os neurônios competem entre si pelo direito de responder a um determinado padrão de estímulos. As redes auto-organizadas são utilizadas em quatro tipos de aplicações:

- clustering: determinar e classificar os possíveis agrupamentos (*clusters*) de dados existentes no conjunto de dados de entrada.
- quantização vetorial: determinar a discretização ótima de um espaço contínuo n-dimensional de entrada, representada por um vetor.
- redução dimensional: dado um subconjunto de dados contido em um subespaço de dimensionalidade menor que o conjunto total de dados, determinar o mapeamento ótimo que preserva a variância dos dados de entrada.

– extração de características: o sistema deverá extrair subconjuntos existentes nos sinais de entrada.

Existem vários métodos de treinamento de redes auto-organizadas, como as algoritmos competitivos [45] e os de Kohonen [27], as quais serão apresentadas nas seções seguintes.

2.3.5.1 Redes Competitivas

As redes competitivas aprendem a responder a padrões existentes no conjunto fornecido de dados de entrada. Cada neurônio da camada de saída corresponde a um possível sinal de saída.

Seja uma rede competitiva como aquela mostrada na figura 2.17. Tanto os padrões de entrada quanto os pesos sinápticos são normalizados.

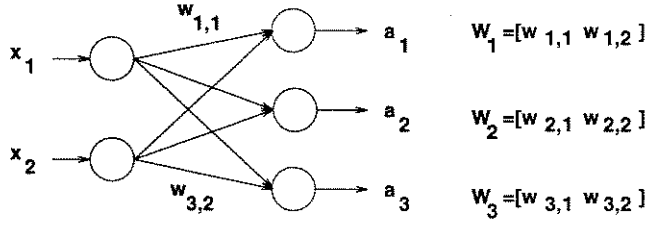


Figura 2.17: Exemplo de rede neural competitiva.

Após alimentar a rede com o conjunto de entradas X^p $(p = 1, \cdots, P)$ calcula-se as saídas $a_i = \sum_j w_{i,j} x_j$. O neurônio com o maior valor de saída a_k

$$\forall i \neq k \ : \ a_i < a_k \tag{2.28}$$

é declarado o *vencedor*. Os pesos sinápticos ligados ao neurônio vencedor são reforçados, enquanto os demais são enfraquecidos:

$$w_k(t+1) = \frac{w_k(t) + \gamma(x(t) - w_k(t))}{\|w_k(t) + \gamma(x(t) - w_k(t))\|} \tag{2.29}$$

O efeito deste ajuste pode ser visto na figura 2.18. Os conjuntos de pesos associados a cada neurônio e os padrões de entrada estão representados por vetores. Está ilustrado o caso de uma rede competitiva com duas entradas apenas. Nesta representação, para um dado padrão apresentado à rede, o neurônio *vencedor* será aquele cujo vetor peso associado está mais próximo do vetor de entradas representando o padrão apresentado. O ajuste dos pesos sinápticos ligados ao neurônio vencedor desloca o vetor-peso para mais perto do vetor-padrão. Após sucessivas iterações, os vetores-peso serão deslocados para próximo do centro de massa dos agrupamentos de estímulos existentes no conjunto de dados de entrada (vetores pontilhados). Assim, ao ser alimentado com um padrão, a rede responderá ativando o neurônio cujo vetor-pesos estiver mais próximo do agrupamento ao qual o padrão alimentado pertence, classificando-o.

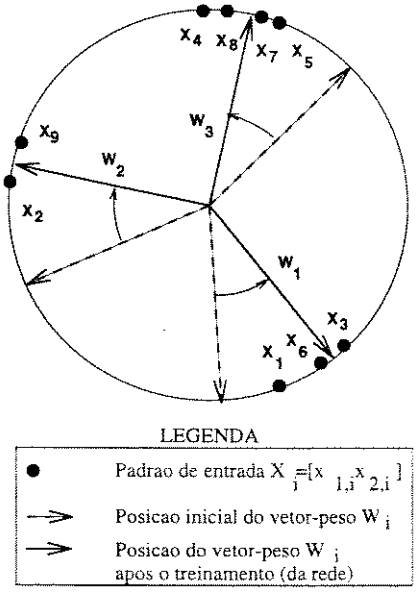


Figura 2.18: Representação vetorial da aprendizagem competitiva.

O treinamento competitivo faz com que os pesos convirjam para os centros de massa dos padrões existentes no conjunto de dados, com exceção do caso abaixo [45].

Para espaços do conjunto de dados de entrada de alta dimensão, poderá ocorrer que um ou mais vetores-peso w_i nunca vençam e não sejam utilizados. Existem várias propostas para resolver esta situação. Uma alternativa utiliza algumas amostras do conjunto

de dados de entrada para inicializar os pesos da rede neural. A segunda propõe ajustar os pesos dos neurônios não-vencedores, de modo a afastá-los do *cluster* associado com o neurônio vencedor. Este afastamento é realizado com a equação abaixo:

$$w_l(t+1) = w_l(t) + \gamma'(x(t) - w_l(t)) \tag{2.30}$$

onde $\gamma' < \gamma$ é denominado o *fator de aprendizagem residual (leaky learning rate).*

Um terceiro método introduz uma medida de acertos: quanto menos vezes um neurônio vence, maior será sua chance de vencer. Este método é denominado *aprendizagem competitiva sensível a freqüência* [45].

2.3.5.2 Rede de Kohonen

A rede de Kohonen [27],[28] é um extensão da rede competitiva, porém possui aplicações diferentes. As redes de Kohonen executam um mapeamento que preserva no espaço-imagem, a estrutura topológica existente no espaço-domínio. Este tipo de comportamento também é observado em sistemas biológicos, os quais mapeam estímulos semelhantes em áreas próximas do cérebro. É o caso das áreas sensoriais e motoras do córtex.

A topologia de uma rede de Kohonen é semelhante a de uma rede competitiva, definida pelo fato de sua camada de saída ser estruturada. Cada neurônio exercerá influência sobre seus neurônios vizinhos, espelhando a relação de semelhança existente entre amostras vizinhos do conjunto de sinais de entrada.

A dimensão de uma rede de Kohonen, dada pela dimensão da estrutura de sua camada de neurônios de saída, deverá ser igual ou maior que a dimensão da estrutura a ser mapeada. Assim, para mapear um politopo de ordem M, existente num espaço de dimensão N (N > M), a ordem da rede de Kohonen deverá ser de pelo menos M. A figura 2.19 mostra uma rede de Kohonen estruturada para mapear dados bidimensionais. Os vizinhos do neurônio indicado por uma cruz são os neurônios preenchidos..

A distância entre dois neurônios q e k pode ser expressa por uma função gaussiana da forma $e^{-\sqrt{(q_x-k_x)^2+(q_y-k_y)^2}}$, onde (q_x, q_y) e (k_x, k_y) são as coordenadas dos dois neurônios. Em uma dimensão, a função se torna $h(q,k) = e^{-(q-k)^2}$, onde q e k indicam a

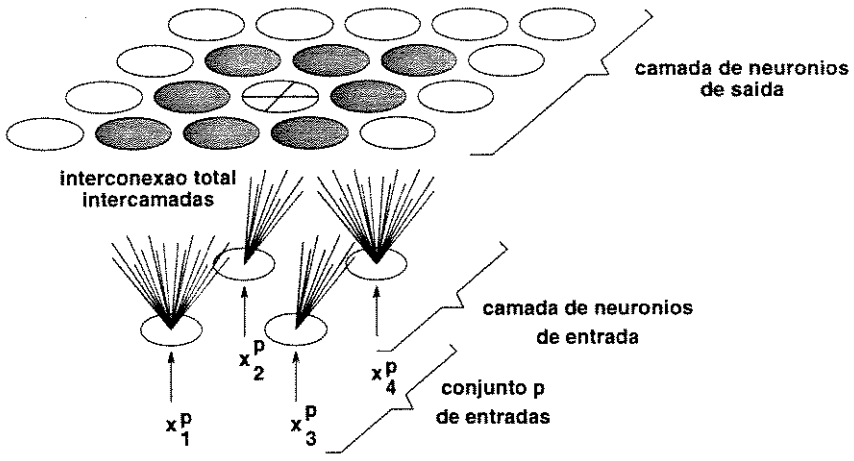


Figura 2.19: Uma rede de Kohonen bidimensional.

posição dos neurônios q e k na camada de saída. O gráfico desta função está disposto na figura 2.20.

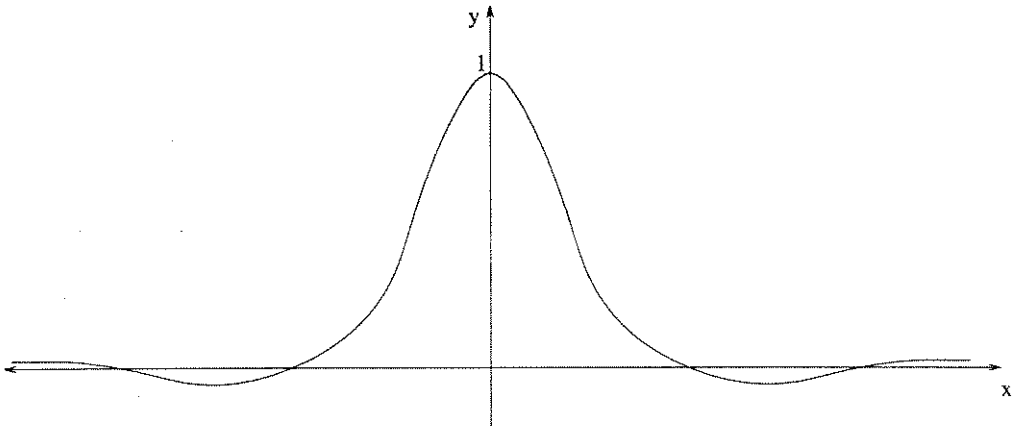


Figura 2.20: A função tipo gaussiana de distância entre neurônios para um dimensão.

O processo de treinamento é semelhante ao das redes competitivas. Apresenta-

se um vetor de sinais de entrada para a rede e determina-se o neurônio vencedor. O ajuste dos pesos ocorre no neurônio vencedor e nos neurônios em sua vizinhança, de acordo com a equação:

$$w_q(t+1) = w_q(t) + \gamma h(q,k)\,(x(t) - w_q(t))\,,\ \forall q \in S \tag{2.31}$$

onde k é o neurônio vencedor, γ é o fator de aprendizagem e $h(q,k)$ é a função distância entre o neurônio vencedor k e seu vizinho q.

Este tipo de rede neural tem sido muito utilizado no processamento de imagens [26] e de fala [25].

2.3.6 Redes Neuronebulosas

No âmbito de controle autônomo, as redes neuronebulosas podem ser classificadas em duas categorias: as que emulam um controlador nebuloso e aquelas que "apenas" utilizam operadores nebulosos.

O primeiro tipo implementa todo o processo de controle nebuloso em sua estrutura, desde a fuzzificação e a inferência até a defuzzificação. O segundo tipo utiliza operadores nebulosos como ferramentas, descrevendo mecanismos neurobiológicos com o auxílio de operadores nebulosos [10].

Nesta seção será apresentado apenas um representante da primeira classe, pois esta será utilizada como base do navegador proposto nesta dissertação e por ser o primeiro tipo o mais comum..

A rede neuronebulosa proposta por Figueiredo *et al* em [11] apresenta um mapeamento biunívoco entre os pesos sinápticos de um rede neural e o conjunto de regras lingüísticas e variáveis nebulosas de um controlador nebuloso.

Esta rede neuronebulosa basea-se em um conjunto de entradas nebulosas do tipo:

X_1 is $\breve{A}_1$ and X_2 is $\breve{A}_2$ and $\cdots$ X_m is $\breve{A}_m$

e em um conjunto de regras com o formato:

$$\begin{cases} \text{rule 1:} & \text{IF } X_1 \text{ is } A_1^1 \text{ and } \cdots X_2 \text{ is } A_2^1 \text{ and } \cdots X_m \text{ is } A_m^1 \text{ then } y \text{ is } g^1 \\ \text{rule 2:} & \text{IF } X_1 \text{ is } A_1^2 \text{ and } \cdots X_2 \text{ is } A_2^2 \text{ and } \cdots X_m \text{ is } A_m^2 \text{ then } y \text{ is } g^2 \\ & \vdots \\ \text{rule N:} & \text{IF } X_1 \text{ is } A_1^N \text{ and } \cdots X_2 \text{ is } A_2^N \text{ and } \cdots X_m \text{ is } A_m^N \text{ then } y \text{ is } g^N \end{cases} \tag{2.32}$$

onde: X_j é a variável nebulosa j, $\breve{A}_j$ é o conjunto nebuloso de entrada j, A_j^i é o conjunto nebuloso i do antecedente j, y é o consequente e g^i é um valor real.

Os universos de discurso dos m antecedentes são discretos e compostos de q_j intervalos discretos x_k, $j = 1, \cdots, m$ e $k = 1, \cdots, q_j$. As funções de pertinência são discretizados em níveis, sendo que cada intervalo discreto x_k possui apenas um valor de pertinência z_k para cada função de pertinência j.

Seja um valor numérico u do k-ésimo intervalo x_k. Então, para uma determinada função de pertinência j o grau de pertinência de u será dado por:

$$\phi_{z(u)} = Z(u) = Z\left(x_k\right) = z_k \tag{2.33}$$

onde: $Z(.)$ é a função de pertinência do conjunto nebuloso Z e x_k é dado acima.

O mapeamento entre o universo de discurso multidimensional das entradas e o universo de saída y ocorre em três etapas:

- *(matching)*: calcula-se a medida de possibilidade P_j^i entre os conjuntos nebulosos de entrada A_j' e os dos antecedentes A_j^i para cada regra i e antecedente j:

$$P_j^i = max_k \left(min\left(a_{jk}', a_{jk}\right)\right) \tag{2.34}$$

  para cada intervalo discreto k. a_{jk} é o valor de pertinência no intervalo k do antecedente j e a_{jk}' é um *singleton* indicando a existência ou não de uma entrada A_j' no intervalo k.

- *agregação de antecedentes*: calcula-se o grau de ativação de cada regra i do seguinte modo:

$$H^i = min_j\left(P_j^i\right), j = 1, \cdots, m. \tag{2.35}$$

– *agregação de regras*: calcula-se o consequente y como uma soma ponderada dos consequentes das regras:

$$y = \frac{\sum_{i=1}^{n} H^i g^i}{\sum_{i=1}^{n} H^i} \tag{2.36}$$

A rede neuronebulosa de Figueiredo é do tipo *feedforward*, possui cinco camadas e utiliza neurônios nebulosos [42]. Nos neurônios nebulosos as funções sinápticas, de agregação de entradas e de ativação são operadores nebulosos.

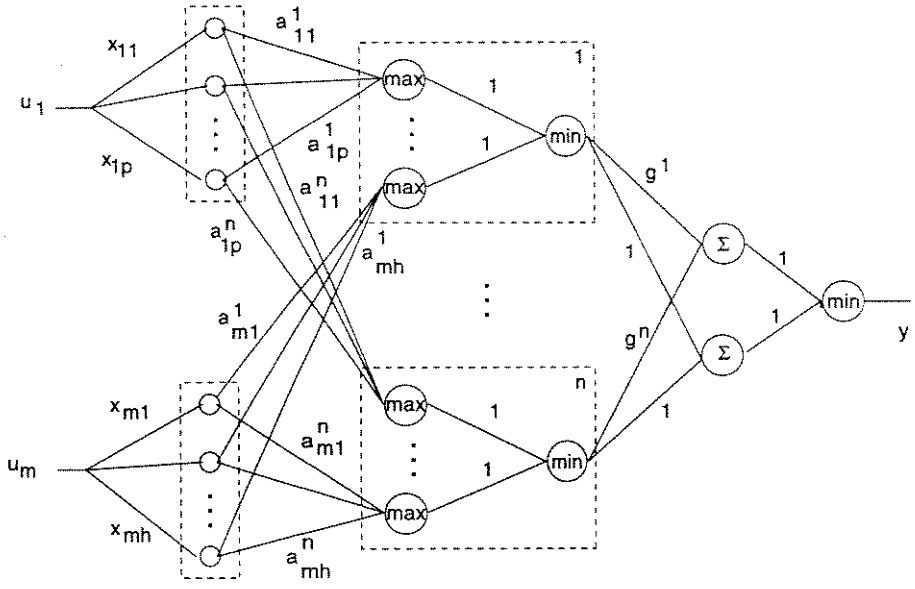


Figura 2.21: Rede Neuronebulosa proposta por Figueiredo *et al.*

A primeira camada é responsável pela etapa de fuzzificação. Ela é composta de m grupos de m_q neurônios. Cada grupo corresponde a uma variável nebulosa, similar aos antecedentes das regras. Cada neurônio possui apenas uma entrada e representa um intervalo nebuloso discreto do universo de discurso associado. Todos os neurônios de um dado grupo j recebe o mesmo sinal u_j. Cada neurônio k deste grupo j fuzzifica este sinal, transformando-o em um *singleton* nebuloso a_{jk} e o transmite para a segunda camada (veja figura 2.21).

O operador sináptico dos neurônios poderá ser de qualquer tipo, contanto que não modifique o sinal de entrada. Os pesos sinápticos das conexões de entrada são unitários.

O operador de agregação é a soma algébrica e a função de ativação pode ser dada por (veja figura 2.22):

$$\phi(u_j) = \begin{cases} 1 & , u_j \in [x_v, x_w] \\ 0 & , \text{caso contrário.} \end{cases} \tag{2.37}$$

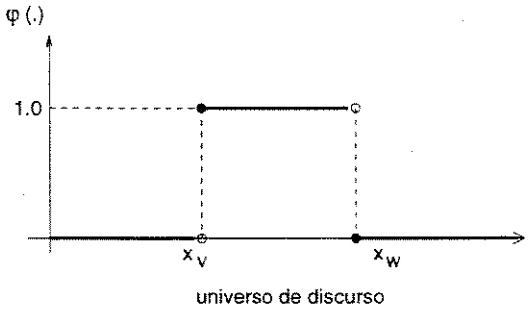


Figura 2.22: Função de ativação dos neurônios da camada de *matching*.

A segunda e terceira camadas implementam conjuntamente a segunda etapa do processo de inferência. Para cada regra i existem:

- g neurônios na segunda camada, cada qual correspondendo a um antecedente e
- um neurônio na terceira camada.

Os neurônios da segunda camada correspondentes ao antecedente f são ligados somente a cada neurônio de entrada l associado com o universo de discurso do antecedente f. O peso sináptico destas conexões é a^i_{fl}. O operador sináptico entre a primeira e segunda camadas é a norma-T min e o operador de agregação de entradas é a norma-S max. A função de ativação apenas repassa a saída da função de agregação.

A terceira camada é composta de n neurônios, os quais calculam o grau de ativação H^i de cada regra i. O operador sináptico entre a segunda e terceira camadas é o produto algébrico com pesos unitários.

A quarta camada é composta de dois neurônios apenas, os quais calculam o numerador e denominador da equação 2.36. O neurônio responsável pelo numerador está conectado a cada e todo neurônio da terceira camada através do peso g^i e do operador

sináptico produto algébrico. O neurônio denominador também está conectado a todos os neurônios da terceira camada por sinapses com operador produto algébrico, porém com pesos unitários. O operador de agregação de entradas de ambos os neurônios da quarta camada é a soma algébrica. Novamente a função de ativação apenas repassa o resultado da agregação de entradas.

A quinta camada possui um neurônio apenas, o qual calcula o quociente entre suas duas entradas. Nas sinapses entre esta e a quarta camadas utiliza-se o produto algébrico com peso unitário.

Gomide e Rocha apresentam em [15] um modelo de neurônio que implementa os operadores de máximo e mínimo.

É interessante notar a relação direta entre a estrutura e os pesos sinápticos da rede e a base de regras nebulosas. Este mapeamento bidirecional permite tanto a fácil inclusão de regras nebulosas em uma rede existente quanto a fácil extração de regras a partir de uma rede neural já treinada.

Figueiredo e Pedrycz propuseram o método do gradiente para a aprendizagem desta rede. Este método apresenta deficiências quando aplicadas a uma rede neuronebulosa pois necessita da forma derivada das funções de ativação dos neurônios. Os operadores nebulosos empregados nesta rede não possuem derivada conhecida, forçando o uso de aproximações [11].

2.4 Sistemas Evolutivos

Sistemas evolutivos são essencialmente sistemas inspirados nos mecanismos biológicos e evolutivos de seleção natural, recombinação e mutação genética. O paradigma foi proposto por J. Holland em sua tese [18] para a modelagem da evolução natural. Após seu uso por David Goldberg [14] na otimização de um complexo de gasodutos, começou a ser aplicado nas mais variadas áreas, tais como otimização de sistemas, teoria de jogos e aprendizagem de máquina. Nesta seção será enfocada sua aplicação em sistemas de controle. Um estudo mais abrangente é apresentado em [14] [12], entre outros.

2.4.1 Conceitos

No processo de seleção natural dispõe-se de uma população de seres. O objetivo desta população é, em essência, promover a própria sobrevivência. Porém, devido a pressões do meio ambiente, apenas os mais aptos sobrevivem. Estes se acasalam e geram um nova população, provavelmente mais adaptada àquele ambiente do que a geração anterior. Este ciclo vida/sobrevivência/geração é contínuo e leva, ao longo das gerações, ao aparecimento de um indivíduo melhor adaptado àquele ambiente. Como o meio ambiente sofre constantes mudanças, o processo de seleção natural é contínuo.

Sistemas evolutivos aplicam estes mecanismos à soluçao de problemas, tipicamente os de otimização. A população passa a ser uma população de soluções possíveis do problema de otimização. O papel de selecionador, desempenhado pelo meio ambiente no processo natural, é preenchido pelo problema e suas restrições. A cada geração novas soluções são criadas e avaliadas. A miscigenação dos sobreviventes (os mais aptos) tende a gerar soluções de melhor desempenho. Interrompe-se o ciclo quando uma solução for considerada satisfatória ou quando se passar de um número pré-determinado de gerações.

Um sistema evolutivo possui três características fundamentais: uso de uma população de soluções, uso de uma forma codificada de solução e uso de operadores de recombinação.

A população de soluções pode ser considerada como sendo uma amostra de pontos do espaço multidimensional de soluções possíveis. Neste ponto de vista, as soluções ótimas estariam localizadas nos pontos de máximo (ou mínimo, caso seja considerada a otimização como a minimização). O uso de uma população de soluções, ao invés de apenas uma solução, permite ao sistema evitar mínimos/máximos locais. Assim, mesmo com dados iniciais que pouco caracterizam a solução ótima, é possível chegar a uma solução quase ótima (veja figuras 2.23 e 2.24 abaixo).

Os indivíduos são avaliados por uma função de desempenho f. Esta função indica o quanto esta solução se aproxima da solução ideal. Esta medida não está restrita às características internas do indivíduo, como o são, por exemplo, os algoritmos tipo gradiente, e.g., *backpropagation*, os quais dependem da diferencabilidade das funções de ativação nos neurônios de uma rede neural. Os critérios de medida podem avaliar características de

operação do indivíduo, tais como *overshoot* ou tempo de estabilização (eq. 2.38), permitindo uma maior flexibilidade de medida e uma especificação mais precisa da solução desejada.

$$f = \frac{p_1}{(overshoot^2)} + \frac{p_2}{t_{estab}} \tag{2.38}$$

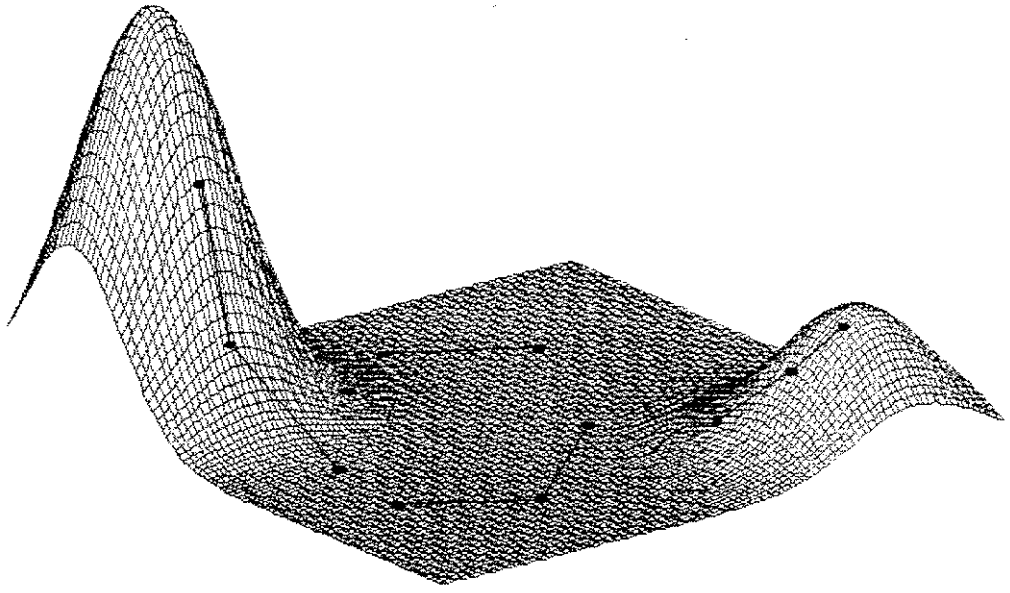


Figura 2.23: Busca genética em um espaço de busca com dois máximos (múltiplos pontos de busca).

Cada indivíduo é representado por um código, o *cromossomo*. O cromossomo contém as características do indivíduo associado. No caso de uma rede neural, o cromossomo poderá conter, por exemplo, as ligações entre os neurônios e/ou os pesos de cada sinapse (veja Figura 2.25). O cromossomo também pode ser visto como sendo as coordenadas do indivíduo no espaço de soluções. O tipo de formato utilizado pode variar desde um simples vetor de bits até um programa em LISP, por exemplo.

Assim como o genoma deve ser decodificado para gerar o ser vivo por ele representado, o cromossomo decodificado resultará na solução-candidato. Esta forma decodificada será avaliada pela função de desempenho f em uma ou mais situações simuladas de

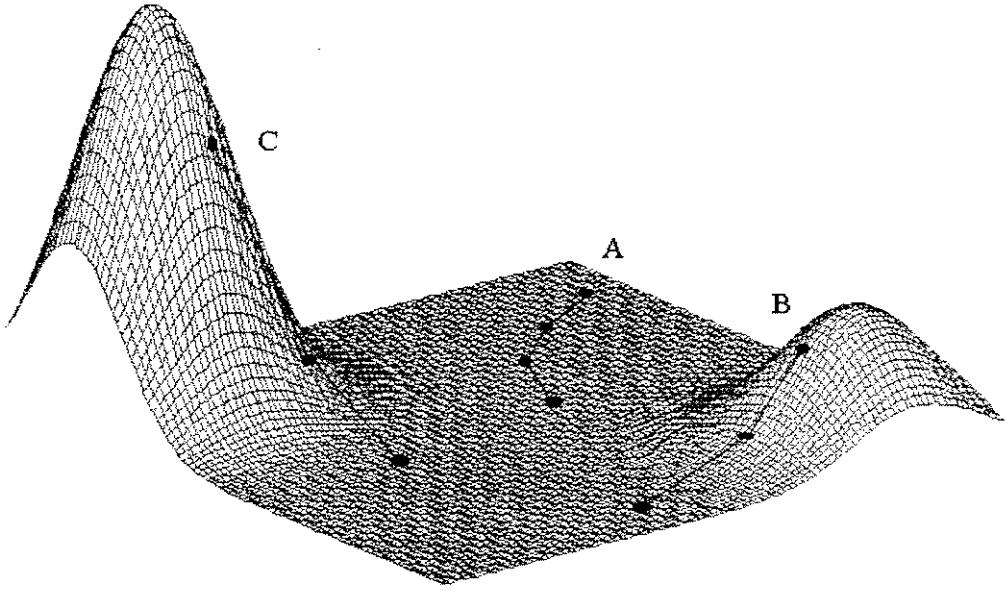


Figura 2.24: Busca por gradiente. Caso A: ponto inicial ruim levando a uma busca demorada. Caso B: ponto inicial próximo ao menor máximo levando a solução sub-ótima. Caso C: Solução ótima.

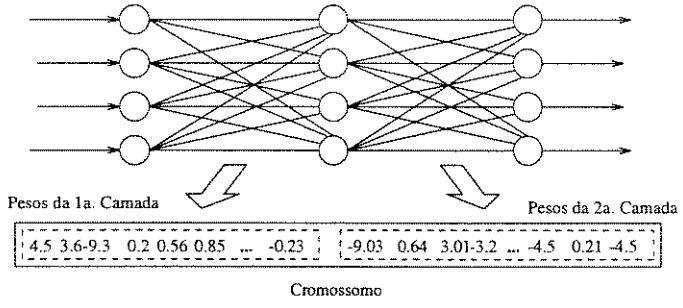


Figura 2.25: Possível cromossomo para uma rede neural.

operação.

Após a avaliação da população, seleciona-se conjuntos de indivíduos para a formação de uma nova população. A seleção é feita baseada no valor de desempenho de cada indivíduo. Os melhores indivíduos possuem maior probabilidade de seleção. Existem vários métodos de seleção, os quais serão vistos mais adiante.

A nova população pode ser formada a partir da anterior por três processos: diretamente com mutação, por recombinação ou por recombinação e mutação.

O processo direto com mutação forma um novo indivíduo acrescentando um ruído aos elementos do cromossomo do indivíduo original. Este acréscimo de ruído (mutação) ocorre com uma probabilidade p_{mut}. Este processo é análogo à partenogênese biológica, no qual um indivíduo se divide em dois iguais ou quase iguais.

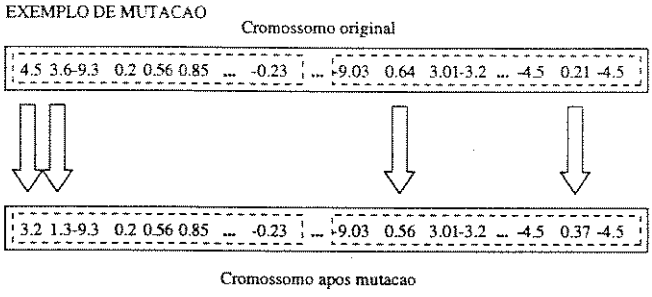


Figura 2.26: Exemplo de uma operação de mutação.

A recombinação é um mecanismo estocástico que forma indivíduos novos a partir da troca de segmentos entre os cromossomos dos indivíduos selecionados. A troca ocorre com probabilidade P_{xover}. Os indivíduos originais são denominados indivíduos-pais e os indivíduos novos os filhos. Com este operador espera-se obter indivíduos de desempenho maior, partindo do princípio dos *building blocks*. Este princípio estabelece uma ligação direta entre o valor de desempenho de um indivíduo e seu cromossomo. Espera-se com a recombinação, agrupar segmentos de cromossomos que levam a bons desempenhos, com o intuito de formar indivíduos de desempenho ainda maior.

Uma vez formada a nova população, o processo de avaliação, seleção e formação é repetido. Cada ciclo completo é denominado uma *geração*. O processo é interrompido

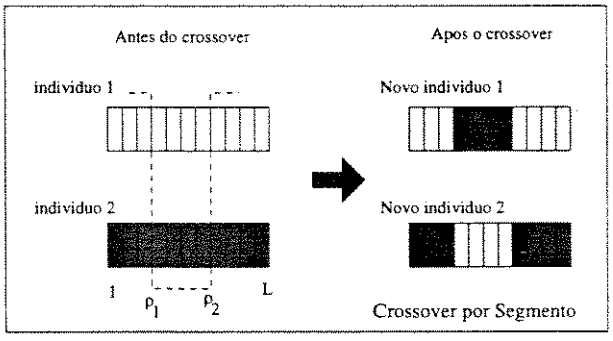


Figura 2.27: Exemplo de *crossover*.

quando se obtiver um indivíduo satisfatório ou quando se passar de uma quantidade de gerações pré-determinada.

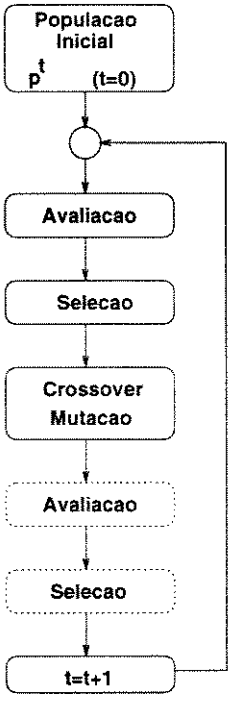


Figura 2.28: Fluxograma básico de um Algoritmo Genético.

De acordo com Fogel [12] atualmente existem dois tipos de sistemas evolu-

cionários: a programação evolucionária/genética e os algoritmos genéticos. Estas diferem entre si pelo tipo de operador de recombinação utilizado.

A programação genética utiliza um operador estocástico que gera novos indivíduos através da adição de ruído aos cromossomos dos indivíduos-pais. Foi empregado por Koza [30] e Reynolds [41] na geração de agentes autônomos, codificados como programas na linguagem LISP. Os programas gerados foram capazes de, entre outras coisas, jogar o jogo PacMan (TM) ([30]) e navegar um robo por um ambiente desconhecido (em simulação) ([30], [41]).

O algoritmo genético utiliza operadores mais complexos que promovem a troca de partes do cromossomo entre os indivíduos-pais. São denominados operadores de recombinação *crossover*. Os algoritmos genéticos foram empregados por vários autores ([39], [22]) no papel de algoritmos de aprendizagem por otimização *off-line* de parâmetros.

Na próxima seção será apresentado, com mais detalhe, o funcionamento do algoritmo genético.

2.4.2 Algoritmo Básico

O algoritmo genético pode ser abstraído pela entidade [47]:

$$GA = (p^0, I, \lambda, L, f, s, c, m, T, A)$$

onde:

$p^0 = (\mathbf{a}_1^0, \cdots, \mathbf{a}_\lambda^0)$	população inicial
$= \left(\begin{bmatrix} a_{1,1} \\ \vdots \\ a_{1,L} \end{bmatrix}, \cdots, \begin{bmatrix} a_{\lambda,1} \\ \vdots \\ a_{\lambda,L} \end{bmatrix} \right) \in I^\lambda$	
$I = A^L$	codificação dos cromossomos
$\lambda \in N$	tamanho da população
$L \in N$	comprimento do cromossomo

$f : I \rightarrow R$	função de desempenho
$s : I^{\lambda} \rightarrow I$	operação de seleção de pais
$c : I^2 \rightarrow I^2$	operação de *crossover*
$m : I \rightarrow I$	operação de mutação
$T : I^{\lambda} \rightarrow A$	critério de término
$A = \{A_1, A_2, \cdots, A_n\}$	alfabeto de ordem n

Nesta forma, o objetivo do algoritmo genético é achar um vetor p_i^k que maximize a função de desempenho f.

Nos itens seguintes serão apresentados cada aspecto do algoritmo genético básico.

Cromossomo

Os sistemas evolutivos lidam com grupos de indivíduos, denominados *populações*. Cada indivíduo da população representa uma possível solução ao problema em consideração. Este indivíduo é manipulado em uma forma codificada, o *cromossomo*. O cromossomo contém as características operacionais do indivíduo-solução ao qual está associado. Como os operadores de recombinação e mutação operam apenas com a forma codificada do indivíduo, a forma daqueles depende do tipo de código utilizado. Tecendo uma analogia com a seleção biológica, o cromossomo seria o *genótipo*, enquanto que a decodificação deste resulta no *fenótipo*.

Existem diversas formas de codificar os parâmetros e características das soluções. Em geral estas formas refletem a natureza da solução. No caso das redes neurais, estas podem ser representadas por uma matriz de números reais, sendo os elementos a_{ij}^0 e a_{ij}^1desta matriz os pesos sinápticos entre a primeira e a segunda camadas e entre a segunda e a terceira camadas, respectivamente.

Goldberg [14] apresenta uma representação em vetores de elementos binários. Esta representação simplifica as operações de recombinação e mutação mas requer um processo por vezes intricado de decodificação. Outra vantagem apresentada seria a da maior proporção de esquemas por unidade de representação oferecido pelo representação binária. Por unidade de representação entenda-se um elemento do vetor-cromossomo. Para um alfabeto com k símbolos temos E=$(k+1)^l$ esquemas, onde l é o comprimento do vetor. $k+1$ pois inclui-se o símbolo *don't care*, representando qualquer símbolo. Para a representação

binária temos a proporção de $\frac{3^l}{2^l} > \frac{5^{(l/2)}}{4^{l/2}}$ de um alfabeto com quatro símbolos.

No entanto, Antonisse [2] mostrou que a complexidade do alfabeto leva a uma maior riqueza do espaço de busca, contrariando a afirmação de Goldberg. Antonisse baseia-se numa interpretação menos restrita dos *don't cares*. Ao invés de considerar apenas *don't cares* do tipo "0 ou símbolo", resultantes da extensão de "zero ou um", Antonisse observa que em alfabetos mais complexos, ocorre uma multiplicidade de *don't cares*, como " A ou B", " A ou C", " A ou D", " B ou C", " B ou D" e " C ou D" para o alfabeto A, B, C, D.

A simplicidade da estrutura binária é ao mesmo tempo sua maior vantagem e maior desvantagem. Ao passo que os mesmos operadores de recombinação e mutação poderão ser utilizados para diversos problemas, sem necessidade de mudança, serão necessários esquemas elaborados de decodificação.

Este inconveniente não ocorre quando for utilizado um cromossomo (genótipo) mais próximo à forma decodificada (fenótipo) do indivíduo. Um exemplo deste método é a codificação utilizada por Koza [30] na otimização de agentes autônomos simulados. Koza utilizou programas em LISP como cromossomos. Como os programas em LISP não precisam ser decodificados para a sua avaliação, pode-se afirmar que o genótipo se tornou o fenótipo. Porém, considerando agentes autônomos como um conjunto de comportamentos apresentados pelo agente, pouco importando a forma como são implementados, divorciam-se novamente o genótipo e o fenótipo.

Avaliação

A avaliação da população é realizada por duas funções: a função objetivo e a função de desempenho.

Função Objetivo

A função objetivo representa os critérios a serem preenchidos pela solução-candidata, quantificando a medida de satisfação do individuo sob consideração ao problema proposto. Os parâmetros para a avaliação de uma rede neural podem ser, por exemplo, o número de neurônios utilizados na(s) camada(s) escondida(s), a proporção de padrões classificados corretamente ou ainda o tempo de subida do processo controlado ante uma entrada degrau. A função objetivo fornece as diretrizes da busca no espaço de soluções. A função objetivo é utilizada para avaliar todos os indivíduos da população e atribuir a

cada um deles uma medida de quão satisfatório é o seu desempenho. Esta medida é calculada comparando-se o comportamento do indivíduo sendo avaliado com o comportamento considerado desejado.

Exemplo 1: Seja a otimização de um controlador de um motor. Algumas características desejáveis deste controlador seriam *overshoot* nulo, tempo de subida instantânea e curto período de estabilização.

Exemplo 2: Suponha que se quer obter um agente que jogue "squash". A tarefa do controlador seria manter uma bola dentro de uma quadra bidimensional com três lados fechados e um aberto. O agente seria uma raquete, a qual pode se movimentar lateralmente apenas, no lado aberto da quadra. Alguns parâmetros da função objetiva poderiam ser o número de rebatidas, a distância da raquete à bola quando esta sair da quadra e a energia gasta pela raquete, representada, por exemplo, pela distância total percorrida entre rebatidas.

Função de Desempenho

Na função objetivo existem alguns parâmetros que devem ser minimizados e outros maximizados para se obter um indivíduo que atenda as especificações desejadas. O operador de seleção manipula apenas quantidades não-negativas, portanto utiliza-se a função de desempenho para transformar o valor dado pela função objetivo em um valor não-negativo. Observe os exemplos abaixo.

Exemplo 1: No caso do controlador citado no Exemplo 1 do item anterior, poderíamos ter a seguinte função de desempenho:

$$f(a_i^t) = \frac{1}{oshoot} + \frac{1}{(t_{\text{rise}})^2} + \frac{1}{(t_{\text{estab}})^2}$$

onde a_i^t é o indivíduo sendo avaliado, $oshoot$, t_{rise} e t_{estab} são o *overshoot*, tempo de subida e tempo de estabilização apresentados pelo indivíduo, respectivamente.

Exemplo 2: A função de desempenho do agente apresentado no Exemplo 2 do item anterior poderia ser:

$$f(a_i^t) = p_1 * (n_{\text{rebatidas}}) + p_2 * (dist_{max} - dist_{bola-raq}) + \frac{p_3}{(energos)^2}$$

onde a_i^t é o indivíduo sendo avaliado, $n_{\text{rebatidas}}$ é o número de bolas rebatidas, $dist_{max}$ é a maior distância possível entre a bola e a raquete, no evento da primeira sair da quadra, $dist_{bola-raq}$ é a distância entre a bola e a raquete, quando aquela sai da quadra e $energos$ é a distância total percorrida entre as duas últimas rebatidas. p_1, p_2 e p_3 são pesos.

Operador de Seleção

O operador de seleção s produz uma população intermediária $p'^t = (a'^t{}_1, \ldots, a'^t{}_\gamma)$ a partir da população $p^t = (a^t{}_1, \ldots, a^t{}_\lambda)$ na geração t. Esta população intermediária p'^t será objeto dos operadores de mutação e recombinação, levando à formação da população da geração seguinte, p^{t+1}. A seleção utiliza o valor de desempenho de cada indivíduo e o compara ao daquele do restante da população. Os melhores indivíduos possuirão maior probabilidade de seleção e portanto, existirão em p'^t em maior quantidade. A probabilidade de um indivíduo a_i^t com desempenho f_i é dado por $p_{s,i} = f_i / \sum_{j=1}^{n} f_j$. O número esperado de cópias de a_i^t na população p'^t é dada por $n_i = p_{s,i} * n$.

Existem vários tipos de seleção. Cada um destes é composto por um esquema de transição entre a população atual p^t e a população seguinte p^{t+1} e um mecanismo de amostragem da população atual.

Os esquemas de transição giram em torno do fator de *gap (G)*. O *gap* indica a quantidade de indivíduos vindos intactos da população atual p^t na população seguinte p^{t+1}. O *gap* é expresso por uma proporção entre a quantidade de novos indivíduos e a população total. Por indivíduos novos entende-se indivíduos gerados por recombinação/mutação e que não existiam na população p^t. Por exemplo, um *gap* G=0.8 indica que 80% da população p^{t+1} será formada por novos indivíduos. Os esquemas de seleção diferem entre si pelo valor fixo de *gap* utilizado.

Existe também o *gap dinâmico*. Neste esquema, a população da geração seguinte p^{t+1} é gerada pela aplicação do operador de seleção à uma segunda população intermediária p''^t, composta pela união da população atual p^t e a população gerada pelos operadores de recombinação e mutação p^{*t}, i.e., $p''^t = p^t \cup p^{*,t}$. Este esquema não utiliza um valor fixo de *gap*, deixando-o flutuar de acordo com o resultado do mecanismo de amostragem.

O mecanismo de amostragem determina a relação entre o valor de desempenho do indivíduo e a sua probabilidade de seleção. Existem vários métodos, alguns dos quais

são descritos a seguir.

– amostragem estocástica com substituição

Esta é a estratégia mais difundida. Também conhecida por algoritmo da roleta ou de Monte Carlo. De [47] temos a seguinte definição:

Qualquer $a'^t_i = a^t{}_q = s(p^t)$ em p'^t é selecionado por um número real aleatório α, tal que

$$0 \leq \alpha \leq \sum_{j=1}^{\lambda}(p_{s,j})$$

O índice q é obtido de

$$q = min\left\{k \mid \forall k \in \{1,\ldots,\lambda\},\ \underline{\text{s.t.}}\ \alpha_i \leq \sum_{k=1}^{\lambda} p_{s,k}\right\} \qquad (2.39)$$

Seja por exemplo a população $p^t = \{(01101,4),(01011,2),(10111,2),(01110,3)\}$ onde as duplas são definidas por (indivíduo, valor de desempenho). Suponha ainda que tenha se obtido o número aleatório $\alpha_1 = 6$. Então, o indivíduo selecionado será o segundo (01011,2).

– amostragem determinística

A probabilidade de seleção é determinada de modo idêntico à amostragem estocástica com substituição. O número de cópias de cada indivíduo é dado por $n_i = \text{int}(p_{s,i} * n)$. Se a população intermediária não for preenchida pelas cópias, ordena-se a população p^t em ordem descrescente pelo valor $\text{frac}(p_{s,i} * n)$ e completa-se a população p^tcom os primeiros indivíduos desta lista.

– amostragem estocástica por resto sem substituição

Este método se distingue da amostragem determinística pelo método usado para completar a população intermediária p'^t. Aqui a população é completada usando amostragem por roleta (veja acima) baseada na parte fracionária de $(p_{s,k} * n)$ ao invés de $p_{s,k}$, na equação 2.39.

- amostragem estocástica por resto com substituição

  Novamente a diferença entre esta variante e as outras reside no método como se completa a população intermediária. Nesta amostragem a parte fracionária de $(p_{s,k} * n)$ é utilizada como a probabilidade de sucesso em um ensaio de Bernoulli. Se o ensaio tiver sucesso, o indivíduo insere outra cópia na população. Este processo é repetido para todos os indivíduos de p^t até completar p'^t.

Estas variantes do operador de seleção foram comparados por Brindle [3] em aplicações envolvendo a otimização de funções. Nesta pesquisa, o método de amostragem estocástica por resto se mostrou o melhor.

Operação de Crossover

A operação de crossover promove a troca de informação entre dois ou mais indivíduos de uma população. Estes indivíduos são denominados *pais*. A troca resultará na formação de um ou mais novos indivíduos com características herdadas dos indivíduos pais. O objetivo desta intercâmbio é a otimização da população e por conseguinte da solução buscada, através da união de características favoráveis de indivíduos com desempenho acima da média. Estas características estão codificadas no cromossomo de cada indivíduo. De modo mais formal, o crossover pode ser descrito do seguinte modo [47], para um par de indivíduos pais:

Para qualquer par de cromossomos selecionados a partir da população p^t, um valor associado, $0 \leq p_c \leq 1$ é gerado aleatoriamente. Se p for maior que o limiar pré-definido de crossover p_c, o operador de crossover é aplicado ao par de cromossomos.

Existem vários tipos de crossovers, sejam utilizando mais de dois pais, sejam utilizando formas diferentes de troca. Serão descritos três dos tipos mais utilizados deste último grupo. As descrições são de crossovers utilizando dois indivíduos pais.

- crossover por segmento

  Produz uma população intermediária p'^t a partir da população p^t como definido em (2.40). As posições de corte em um par de cromossomos são selecionados aleatoriamente. O par de cromossomos é dividido em $cs + 1$ segmentos por estes cs pontos e recombinados trocando-se estes segmentos entre os cromossomos (veja figura 2.29).

$$\left\{ \begin{array}{c} \mathbf{a}'^t_i \\ \mathbf{a}'^t_{i+1} \end{array} \right\} = c_{tp}\left(\left\{ \begin{array}{c} \mathbf{a}^t_i \\ \mathbf{a}^t_{i+1} \end{array} \right\} \right), \ \forall i \in \{1, 3, \cdots, 2k+1, \cdots, \lambda - 1\}$$

$$= \left\{ \begin{array}{c} \left[a_{(i+1),1}, a_{1,(i+1)}, \cdots, a_{(i+1),\rho_1}, a_{i,(\rho_1+1)}, \cdots, \right. \\ \left. a_{i,\rho_2}, a_{(i+1),(\rho_2+1)}, \cdots, a_{(i+1),L} \right]^T \\ \left[a_{i,1}, a_{i,2}, \cdots, a_{i,\rho_1}, a_{(i+1),(\rho_1+1)}, \cdots, \right. \\ \left. a_{(i+1),\rho_2}, a_{i,(\rho_2+1)}, \cdots, a_{i,L} \right]^T \end{array} \right\} \qquad (2.40)$$

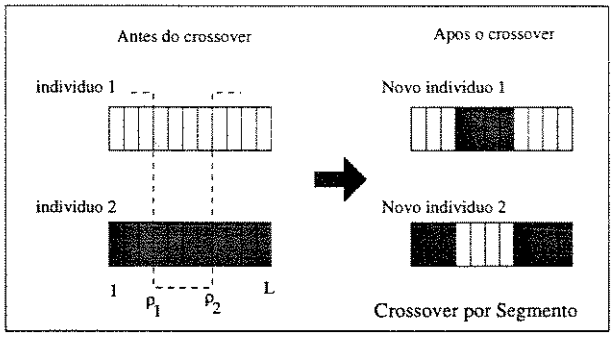


Figura 2.29: Exemplo de Crossover por segmentos.

– crossover multiponto

Esta segunda variedade de crossover produz um população intermediária p'^t a partir da população p^t de acordo com (2.41), onde $0 \leq p_k, p_{mp} \leq 1$. Nesta estratégia cpp posições são escolhidas aleatoriamente para serem trocadas entre o par de cromossomos (veja Fig.2.30). A troca ocorrerá em uma destas posições se, para um valor aleatório p_k, gerado para aquela posição $a_{i,k}$, $\rho_k \geq \rho_{mp}$. onde ρ_{mp} é uma probabilidade pré-definida de limiar.

$$\left\{ \begin{array}{c} \mathbf{a}'^t_i \\ \mathbf{a}'^t_{i+1} \end{array} \right\} = c_{mp}\left(\left\{ \begin{array}{c} \mathbf{a}^t_i \\ \mathbf{a}^t_{i+1} \end{array} \right\} \right), \ \forall i \in \{1, 3, \cdots, 2k+1, \cdots, \lambda - 1\}$$

$$= \bigcup_{k=1}^{L} \left(z_k = \left\{ \begin{bmatrix} a_{i,k} \\ a_{(i+1),k} \end{bmatrix}, \text{ se } \rho_k \geq \rho_{mp} \right. \right. \\ \left. \left. \begin{bmatrix} a_{(i+1),k} \\ a_{i,k} \end{bmatrix}, \text{ se } \rho_k < \rho_{mp} \right. \right) \tag{2.41}$$

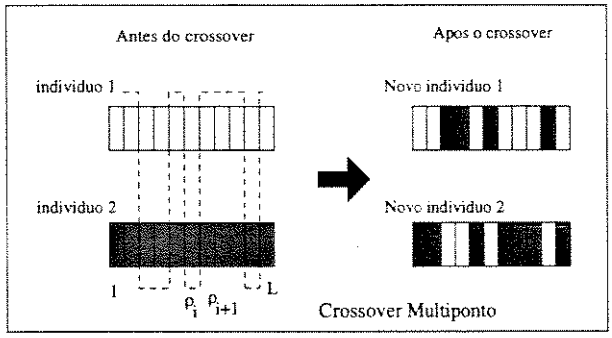


Figura 2.30: Exemplo de Crossover multipontual

– crossover por máscara

Este tipo de crossover produz uma população p'^t a partir da população p^t com o auxílio de uma máscara (veja eq. (2.42)). A máscara é um vetor binário de comprimento L. L valores reais, $0 \leq r_j \leq 1 (j = 1, 2, 3, \ldots, L)$ são gerados aleatoriamente. Se r_j for maior ou igual a um valor pré-definido de limiar, $\rho_{ma} \in [0, 1]$, o valor do j-ésimo elemento na máscara será 1. Caso contrário será 0 (eq. 2.43).

$$ma = \bigcup_{j=1}^{L} \{([1], \text{ se } r_j \geq \rho_{ma}) \text{ ou } ([0], se r_j < \rho_{ma})\} \tag{2.42}$$

O crossover por máscara pode ser definido pela expressão (2.43). A máscara definirá os locais a serem trocados ($r_j = 1$) ou não ($r_j = 0$) para todos os pares de cromossomos.

$$\begin{Bmatrix} \mathbf{a}'^t_i \\ \mathbf{a}'^t_{i+1} \end{Bmatrix} = c_{mp} \left(\begin{Bmatrix} \mathbf{a}^t_i \\ \mathbf{a}^t_{i+1} \end{Bmatrix} \right), \forall i \in \{1, 3, \cdots, 2k+1, \cdots, \lambda - 1\}$$

$$= \bigcup_{k=1}^{L} \left(z_k = \left\{ \begin{array}{l} \left[\begin{array}{c} a_{i,k} \\ a(i+1),k \end{array} \right], \text{ se } ma_k = 0 \\ \left[\begin{array}{c} a_{(i+1),k} \\ a_{i,k} \end{array} \right], \text{ se } ma_k = 1 \end{array} \right. \right) \tag{2.43}$$

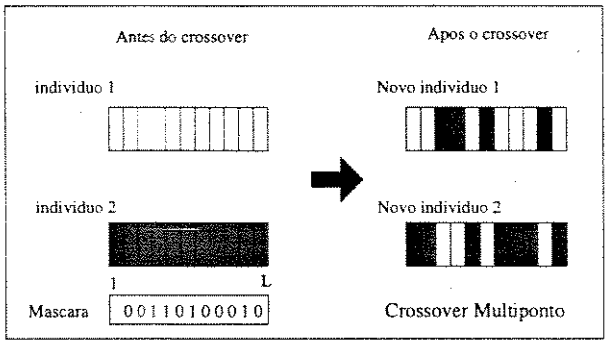


Figura 2.31: Exemplo de Crossover por máscara.

Operador de Mutação O operador de mutação é geralmente aplicado à população intermediária p'^t formada pelo operador de recombinação. Na definição abaixo, a qual está enunciada em função de uma população genérica p, ter-se-ia $p = p'^t$.

Para todo cromossomo a_i na população p, um valor real associado, $0 \leq \rho_i \leq 1$ é gerado aleatoriamente. Se ρ_i for menor que um limiar pré-definido de mutação ρ_m, a operação de mutação é aplicada ao cromossomo a_i. A mutação acrescentará ruído ao símbolo em uma posição aleatória do cromossomo, mudando-a para outro símbolo.

Uma interpretação alternativa de mutação aplica a operação de mutação a todas as posições do cromossomo, com probabilidade $\rho'_m = 1 - \sqrt[L]{1 - \rho_m}$, onde ρ_m, a probabilidade de mutação para o cromossomo, é definida como sendo a probabilidade pelo menos uma mutação no cromossomo.

2.5 Considerações Adicionais

2.5.1 Embasamento matemático

O algoritmo genético explora o espaço de soluções utilizando uma amostra de pontos deste espaço. Estas amostras são instâncias de objetos mais genéricos denominados *esquemas*. Os esquemas podem ser considerados como classes de indivíduos, as quais representam um ou mais características genéricas daquela classe de indivíduos-solução. Exemplo: O esquema H=1***00* engloba, entre outros, os indivíduos $a_1 = 1110000$ e $a_2 = 1001001$ e apresenta as características codificadas nos seus dígitos fixos. Por outro lado, o indivíduo $a_3 = 1101010$ representa vários esquemas, como $H_1 = *101010$, $H_2 = *1***10$, $H_3 = 110*0*0$.

Os esquemas possuem representação semelhante à dos indivíduos, diferindo apenas no alfabeto empregado. Este alfabeto A^* é um extensão do alfabeto original A (por exemplo, o alfabeto binário $A = 0, 1$), acrescentando um símbolo adicional, *, denominado de símbolo coringa (*don't care*). O símbolo *don't care* representa uma elemento ambíguo, podendo ser substituido por qualquer um dos símbolos do alfabeto A. Para alfabetos de ordem $O(A) > 2$, existirão vários símbolos *don't care*, representando um possível intercâmbio entre 2 símbolos ((*1=A ou *1=B) e (*2=C ou *2=D)), 3 símbolos ((*1=A ou *1=B ou *1=C)) e assim por diante.

Na discussão abaixo serão utilizados um alfabeto original binário $A = 0, 1$. O alfabeto estendido será então $A^* = 0, 1, *$ e o cromossomo composto por um vetor destes símbolos.

Dois parâmetros caracterizam um esquema H: a distância $\delta(H)$ e a ordem o(H).

A ordem de um esquema o(H) é definido como sendo o número de posições do cromossoma com símbolos fixos. Exemplo: Seja o esquema H=*10***1. A ordem deste exemplo é o(H)=3 pois existem três posições com símbolos fixos.

A distância de um esquema é definido como sendo a diferença entre a primeira e a última posição fixa. Exemplo: Seja o esquema H=*10***1. A distância é $\delta(H) = 6 - 1 = 5$.

Serão mostrados a seguir os efeitos dos operadores de seleção, recombinação e

mutação sobre a quantidade de cópias de cada esquema.

Se a população da geração seguinte p^{t+1} for formada apenas pela seleção de indivíduos da população anterior p^t, o número de cópias de um determinado esquema H será a soma das cópias alocadas para todos os indivíduos englobados por aquele esquema:

$$n\frac{f(H)}{\sum f_i} = \sum_{j=1}^{m} \frac{f(a_i(t))}{\sum f_i}$$

Portanto, sendo $m(H,t)$ a quantidade de cópias do esquema H na população da geração t, a quantidade de cópias do esquema H na população da geração seguinte é dada por:

$$m(H,t+1) = m(H,t)n\frac{f(H)}{\sum f_i} = m(H,t)\frac{f(H)}{(\bar{f})}$$

onde $\bar{f}$ é o desempenho médio da população.

Da equação acima, retira-se que a quantidade de cópias dos esquemas com desempenho acima da média cresce ao longo das gerações enquanto que a de cópias dos esquemas abaixo da média decresce ao longo das gerações. De fato, assumindo que $t_0 = 0$ e que um dado esquema H permaneça com um desempenho acima da média de um valor $c\bar{f}$, com c constante, obter-se-á:

$$\begin{aligned} m(H,t+1) &= m(H,t)\frac{\bar{f}+c\bar{f}}{\bar{f}} = (1+c)m(H,t) \\ m(H,t) &= m(H,0)(1+c)^t \end{aligned} \tag{2.44}$$

Ora, esta é um equação de uma progressão geométrica que se aproxima da função exponencial. Portanto, deduz-se que o algoritmo genético, através da seleção, aloca um número exponencialmente crescente de cópias para esquemas de desempenho acima da média.

O operador de crossover age como disruptor de esquemas pois promove a divisão e a troca de segmentos de cromossomos entre indivíduos. Seja p_c a probabilidade de se

aplicar crossover a um determinado indivíduo. Então, a probabilidade de sobrevivência p_s de um esquema H é dado por:

$$p_s \geq 1 - p_c \frac{\delta(H)}{l-1}$$

onde l é o comprimento total do cromossomo.

Assim, após o crossover,$m(h, t+1)$ se torna:

$$m(H, t+1) \geq m(H, t) \frac{f(H)}{(\bar{f})} \left(1 - p_c \frac{\delta(H)}{l-1}\right)$$

Da equação acima deduz-se que esquemas de curta distância e baixa ordem são menos suscetíveis a desaparecerem por crossover. Deve-se levar este fato em conta ao determinar o código a ser utilizado no cromossomo.

A mutação alterará um esquema apenas se incidir sobre as posições fixas do esquema. Desta forma, os esquemas de baixa ordem possuirão uma probabilidade maior de escaparem intactos da mutação. Seja p_m a probabilidade de mutação em cada uma das posições do cromossomo. Então a probabilidade de sobrevivência de um esquema é dada por:

$$(1 - p_m)^{o(H)} \approx 1 - o(H)p_m, \; p_m \ll 1$$

Por conseguinte, desprezando os termos com produto de $p_m * p_c$, a quantidade de cópias de um esquema H após as operações de seleção, crossover e mutação se torna:

$$m(H, t+1) = m(H, t) \frac{f(H)}{\bar{f}} \left[1 - p_c \frac{\delta(H)}{l-1} - o(H)p_m\right]$$

Desta equação final inferimos que as operações de recombinação e mutação pouco interferem na convergência para os melhores indivíduos. Os esquemas de alto desempenho, distância curta e de baixa ordem continuam recebendo pelo menos um número exponencialmente crescente de cópias a cada geração.

2.5.2 Operadores secundários

Escalamento

Nas gerações iniciais da busca poderá ocorrer uma convergência prematura para um máximo local, devido ao esquema de seleção. Para evitar o surgimento deste fenômeno utiliza-se um operador secundário denominado escalamento. Este operador redistribui as probabilidades de seleção da população em torno do valor médio de desempenho apresentado pela população. Os indivíduos com desempenho igual à média da população passam a receber uma cópia na população intermediária, enquanto o indivíduo de maior desempenho recebe duas cópias. Os indivíduos restantes recebem uma quantidade intermediária de cópias, de acordo com seu desempenho (figura 2.32). Eventualmente esta distribuição poderá levar à alocação de quantidades negativas de cópias (veja figura 2.33). Neste caso, a função linear de redistribuição é redefinida em relação ao indivíduo de desmepenho médio, o qual continua recebendo uma cópia e em relação ao indivíduo de menor desempenho que não recebe cópia alguma (veja 2.34).

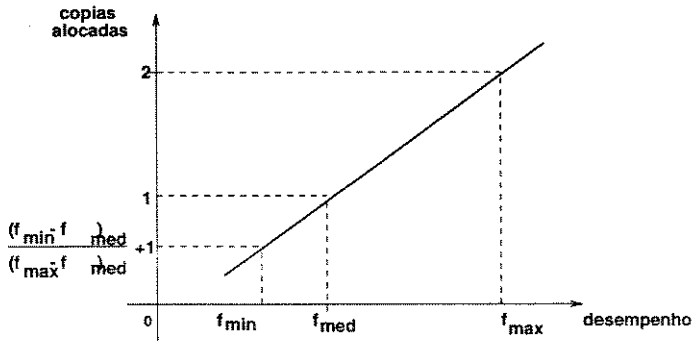


Figura 2.32: Escalamento linear utilizando os desempenhos máximo e médio como referência.

Distribuição

O operador de distribuição limita o número de de indivíduos da população em uma dada região do espaço de soluções. O objetivo é evitar a convergência prematura a um solução sub-ótima, forçando a formação de novos indivíduos em outra região. O mecanismo empregado para tal é decrementar o valor de desempenho de um dado indivíduo proporcio-

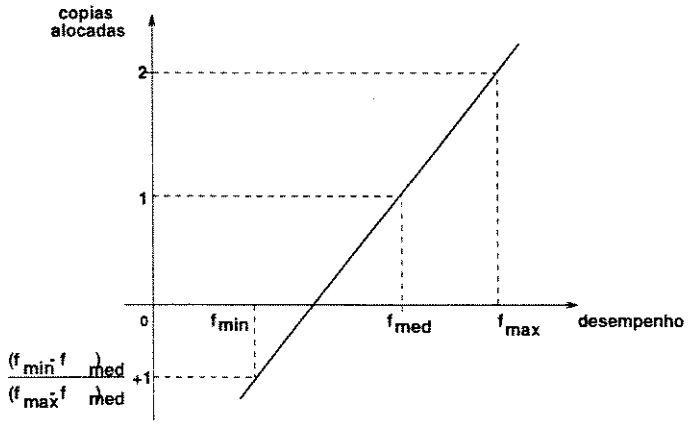


Figura 2.33: O problema do mínimo negativo no escalamento linear max-mid.

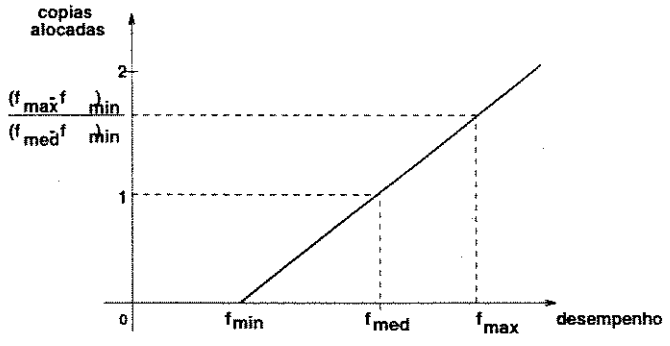


Figura 2.34: Escalamento linear utilizando os desempenhos médio e mínimo como referência.

nalmente ao número de indvíduos genéticamente semelhantes a este. A semelhança genética indica a proximidade no espaço de busca. Assim, quanto mais indivíduos convergirem para a mesma região, menor será o desempenho médio da região, os indivíduos daquela região terão menor probabilidade de sobreviver e se multiplicar, evitando a convergência da população. O análogo evolucionário é a limitação de recursos de uma dada região. Seja uma região com muitos recursos (alto desempenho associado). Esta região irá atrair muitos indivíduos (convergência). Porém o aumento da população regional levará a menos recursos por indivíduo (distribuição levando a diminuição do valor de desempenho), desencorajando a continuada migração de indivíduos àquela região.

Uma forma de implementar este operador secundário é dividir o fitness de cada indivíduo $a_i(t)$ da população pelo número de indivíduos semelhantes geneticamente a este indivíduo:

$$f(a_i(t)) = \frac{f(a_i(t))}{N_{\text{semelhantes}}} \tag{2.45}$$

A medida de semelhança genética entre um indivíduo e outro pode ser dada, por exemplo, pela proporção entre o número de gens iguais e o número total de gens.

2.6 Resumo

Neste capítulo foram apresentadas três técnicas básicas da computação flexível: as redes neurais, os sistemas nebulosos e um dos métodos de raciocínio probabilístico, os algoritmos genéticos (figura 2.35).

No próximo capítulo será apresentado e caracterizado o problema da navegação autônoma de veículos. Consideraremos alguns dos métodos de navegação já existentes, para em seguida propor uma técnica de navegação, baseada nos três paradigmas de computação discutidos nas seções precedentes.

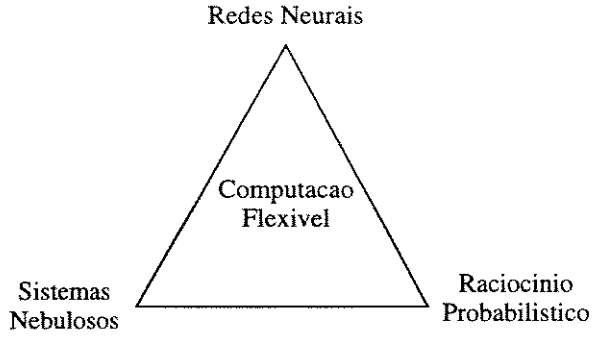


Figura 2.35: Computação flexível: paradigmas básicos.

Capítulo 3

Navegação Autônoma de Veículos

3.1 Introdução

Este capítulo enfoca o problema da navegação autônoma, ressaltando suas características e dificuldades. Uma classificação dos métodos existentes, bem como uma breve sinopse das mais relevantes são apresentados. Ao final, é apresentado um método baseado na computação flexível, conforme o capítulo anterior.

3.2 O Problema de Navegação Autônoma

Podemos definir o problema de navegação autônoma sucintamente do seguinte modo:

Sejam um ambiente de operação A contendo um conjunto de objetivos T e um conjunto não-vazío de obstáculos O e um veículo V com um conjunto de sensores S. Considerar-se-á este veículo V como sendo um veículo autoguiado *ou seja, capaz de navegação autônoma, se, utilizando apenas as informações sobre o ambiente A fornecidos pelos sensores S, o veículo V for capaz de, através de um algoritmo de controle conveniente C que não contenha componente humano H algum, atingir os objetivos T sem se chocar com um ou mais obstáculos O.*

O problema de navegação autônoma possui aparente simplicidade por ser facil-

mente resolvido por um ser humano. Deve-se considerar porém os enormes recursos dos quais um humano padrão dispõe, como por exemplo, memória, capacidade de abstração, aprendizagem não-supervisionada e riqueza de informações colhidas através de seus sensores.

A resolução de qualquer problema passa obrigatoriamente por trê fases: a caracterização da mesma, a determinação dos requisitos mínimos para a solução mais simples e a determinação da solução em si.

O problema de navegação autônoma engloba três partes: o agente/veículo, os sensores utilizados e o ambiente. O sucesso da solução, ou seja, o algoritmo desenvolvido, dependerá da correta caracterização destes componentes.

Existem dois tipos de ambientes de operação: o mundo real e simulações do mundo real. A operação no "mundo real" exige a implementação física do agente e adiciona critérios como tempo de resposta e tolerância a ruído ao problema. Um erro de hardware é muito mais difícil de localizar e corrigir que um erro de software. Por outro lado, o uso do "mundo real" evita o trabalho da construção de uma simulação que possa emular o mundo real de maneira satisfatória e possíveis falhas de controle originados da transição da simulação para a aplicação prática. O uso de uma simulação permite a rápida modificação e re-adequação do algoritmo de navegação a imprevistos ou erros de caracterização. Suas desvantagens residem na formação de uma simulação cujas características se aproximam tal ao mundo real que permitam o uso do algoritmo de controle sem modificações práticas. A escolha do tipo de ambiente é restrita pelos recursos disponíveis, sejam computacionais, financeiros, ou de disponibilidade de hardware ou de tempo.

Uma vez escolhido o ambiente, passa-se a modelar a interação entre o ambiente e o veículo. Este processo de modelagem incorre na escolha dos sensores adequados e nas ações que o veículo possa tomar.

Os sensores devem ser em número suficiente e de tipos adequados para permitir a navegação do veículo. Um veículo com poucos sensores poderá não diferenciar entre situações e tomar a ação errada. Por exemplo, um veículo com apenas um sensor de distância fornecendo a distância ao obstáculo mais próximo certamente colidirá por não possuir informações sobre a posição relativa do obstáculo. Um objeto a 1 cm a sua frente não será diferenciado de um objeto a 1cm a sua esquerda. Neste caso, o navegador necessita ainda

de informações sobre a direção ao obstáculo.

A escolha dos sensores é ainda mais crítica quando o algoritmo a ser usado não possui capacidade de aprendizagem. Neste caso, o conhecimento embutido no agente reflete o conhecimento do(s) projetista(s) sobre o ambiente e a sua percepção do que constitui um comportamento de navegação correto. Esta percepção é bastante dificultada quando se usa sensores de natureza diversa da dos sensores humanos. Os sensores mais comuns indicam a distância e posição de objetos em relação ao veículo (sensores infra-vermelhos de proximidade, sensores ultrassônicos de distância, câmeras de video).

Os atuadores default utilizados em navegação autônoma determinam o movimento do veículo: direção e velocidade. O mecanismo pelo qual se efetua tais mudanças no movimento dependerá do tipo de impulsão empregado pelo robô.

Se ele possuir duas rodas motrizes traseiras e uma roda direcionador frontal, basta mudar o ângulo da roda frontal para mudar o sentido de movimento do agente. O movimento de ré pode ser obtido invertendo-se o sentido de movimento das rodas motrizes. A velocidade do veículo será proporcional à velocidade angular das rodas.

No caso de um robô com duas rodas motrizes independentes apenas, a direção de movimento poderá ser modificada fazendo com que uma roda gire mais rapidamente que a outra. Assim, para virar para a esquerda, a roda direita girará mais rapidamente que a esquerda.

Para robôs providos de pernas, a mudança de direção exige um algoritmo mais elaborado de coordenação motora, a qual dependerá evidedentemente, da geometria das pernas.

3.3 Abordagens para Navegação Autônoma

Os métodos de navegação autônoma podem ser organizados por dois critérios: filosofia empregada e características operacionais.

No âmbito da filosofia de projeto, identificam-se três tipos de sistemas de controle autônomo: sistemas comportamentais, sistemas reativos e sistemas planejadores.

A utilização de características operacionais para a classificação dos navegadores autônomos permite uma vasta quantidade de categorias. Estabelecemos como parâmetros relevantes:

- uso de memória;
- capacidade de aprendizagem e adaptabilidade;
- uso de informações locais ou globais.

A figura abaixo mostra uma classificação de alguns métodos existentes pelos dois últimos critérios operacionais acima.

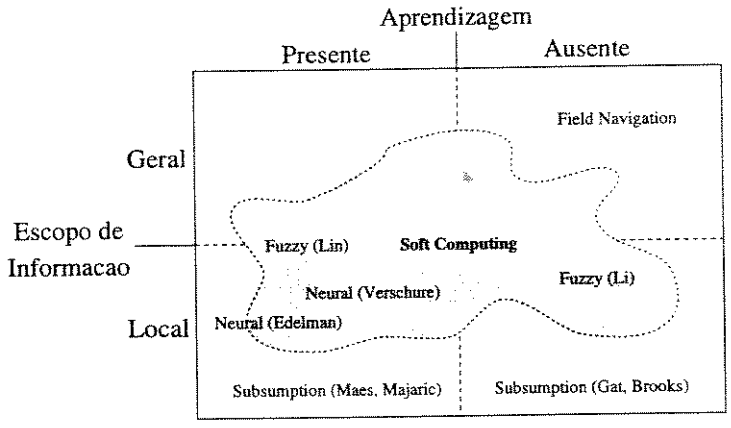


Figura 3.1: Classificação dos Métodos de Navegação.

A memória está diretamente atrelada com a aprendizagem, sendo uma condição necessária mas não suficiente. A memória consiste de um mecanismo de recuperação de eventos passados. Existem diversos formas de memória: genética, ontológica, e a cultural/social.

- A memória *genética* possui significado restrito, existindo apenas quando existe um processo de aprendizagem ou otimização evolucionária. Ela consiste do conjunto de pares (genoma, desempenho) da população. Uma vez interrompido processo evolucionário e selecionado o indivíduo de melhor desempenho, pode-se dizer que todo o conhecimento gerado pela evolução foi concentrado no genoma deste indivíduo.

- A memória *ontológica* é a memória dita "convencional". Aparece sob diversas formas: como locais pré-determinados para a armazenagem de dados, como parâmetros modificáveis pelo próprio agente, ou quando o sistema é dinâmico e causal. Neste último caso a memória consiste de todo o comportamento passado do sistema. Como exemplos existem os pesos das sinapses das redes neurais ou o banco de conhecimento de um sistema especialista.
- A memória *cultural/social* pode ocorrer apenas em ambientes com múltiplos agentes com capacidade de comunicação entre si. O conhecimento está armazenado de modo distribuído, de modo parcial ou total na memória ontológica de cada agente.

Em todos os casos a memória liberta o agente da localidade temporal, i.e., o agente não está restrito aos estímulos existentes naquele instante.

Como apenas alguns métodos de navegação autônoma não fazem uso de memória em suas variadas formas, os métodos atuais de navegação serão apresentados e classificados em função de capacidade de aprendizagem e do escopo de seus sensores.

A aprendizagem não é condição necessária para a operação autônoma. Os agentes construídos por Brooks [6], Gat [13] e Li [34] não possuem mecanismos de aprendizagem e navegam perfeitamente em ambientes diversos. Entretanto, podemos dizer que estes modelos são menos autônomos do que os dotados de aprendizagem, pois necessitam de mais informações colhidas *a priori* e neles embutidas pelos projetistas humanos. De fato, os mecanismos de aprendizagem apresentam também a possibilidade de melhor adaptação do agente ao ambiente, através da capacidade de auto-modificação.

Um método sem aprendizagem poderá ser atualizado, através do acoplamento de um algoritmo de aprendizagem conveniente. De fato, esta agregação geralmente ocorre como uma segunda etapa no desenvolvimento dos algoritmos de controle autônomo. Como foi discutido no Capítulo 1, esta tática nem sempre leva aos melhores resultados.

Um agente autônomo utiliza informações locais quando necessita apenas de dados (colhidos pelos seus sensores) sobre o ambiente em seu entorno.

Os agentes que utilizam apenas informações locais para desempenhar tarefas são mais flexíveis. Primeiro porque não requerem a caracterização de todo o ambiente

de operação, reduzindo a quantidade de informação a ser processada e segundo porque podem ser utilizados em um maior número de ambientes, pois ambientes macroscopicamente distintos podem ser localmente semelhantes.

A adaptabilidade de um controlador é uma medida da variação de seu desempenho dado uma variação nas variáveis de operação. As mudanças podem ocorrer tanto no controlador quanto no processo controlado. Algumas alterações usuais no controlador incluem variações na calibração dos sensores, aumento de ruído no sinal amostrado e/ou falha parcial de hardware/software. O processo controlado poderá ter suas características de operação alteradas por variações climáticas, alterações nos seus componentes ou pelo uso em condições fora das especificações iniciais.

Um processo autônomo deverá ser necessariamente adaptativo, pois deverá ser capaz de lidar com perturbações no seu ambiente de operação sem a intervenção de outro agente (humano).

A seguinte discussão dos métodos existentes de navegação autônoma seguirá a classificação por filosofia, comentando sobre os parâmetros relevantes indicados acima quando necessário.

3.3.1 Navegação por planejamento

A navegação por planejamento baseia suas ações na correlação entre um modelo interno do ambiente de operação e o ambiente real. Todas as características consideradas relevantes são pré-determinadas durante o projeto e incluídas na representação interna. Os sensores são responsáveis pela contínua atualização deste modelo de mundo. O agente utiliza este modelo interno para planejar suas ações. O diagrama da figura 3.2 ilustra este tipo de sistema.

A constante atualização do modelo interno representa uma parte considerável do processamento em um sistema baseado em planejamento. Em um ambiente muito complexo, esta necessidade de atualização poderá inviabilizar a reação em tempo hábil a eventos externos. Existe ainda a questão da exatidão do modelo interno. Se a representação interna não refletir os aspectos realmente relevantes do ambiente, as decisões nela baseadas estarão comprometidas.

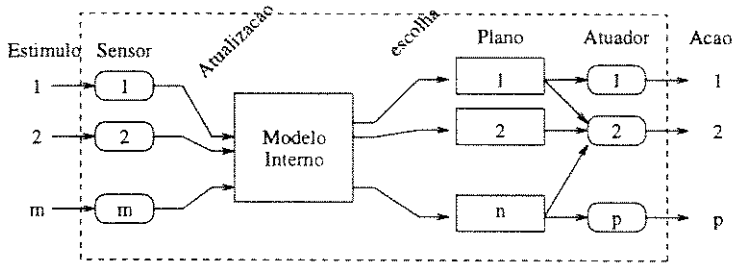


Figura 3.2: Diagrama de blocos de um sistema baseado em planejamento.

O exemplo típico de sistemas baseados em planejamento são os sistemas especialistas [49]. Nestes todo o conhecimento é originado e formulado por um especialista humano. Assim o sistema possui essencialmente a visão de mundo humano. Esta base de dados está geralmente formulada em termos de regras lógicas e baseada em um conjunto de símbolos. Se o ambiente de operação do sistema não for bem caracterizado, ou seja, se o conjunto de símbolos provar ser insuficiente, não será garantido a correta operação do sistema. A agregação de um mecanismo de aprendizagem será de alguma valia somente se for capaz de gerar novos símbolos. Os sistemas baseados em planejamento também dependem criticamente da correta tradução dos estímulos externos captados pelos sensores nos símbolos pré-determinados. Isto somente ocorrerá se todos os estímulos possíveis forem antecipados na formulação do modelo interno. Tal fato é restritivo quando se trata de ambientes desestruturados e com grande variação de parâmetros. Este é o caso do problema de navegação autônoma.

3.3.2 Navegação reativa

A navegação reativa associa um conjunto de ações a cada conjunto possível de estímulos. Estas relações são compactamente representadas por pares do tipo (estímulos, ações). Esta forma de processamento é localizada pois considera apenas os estímulos ambientais atuais provindos do ambiente circundante, colhidos por sensores. A agregação de estados internos (memória) à arquitetura reativa, sob a forma de *estímulos* internos permite uma globalização temporal da estratégia de navegação. Os *estímulos* internos forneceriam um modo de comparação do estado atual com experiências passadas. Exemplos típicos

desta arquitetura são os sistemas nebulosos e as redes neurais mais simples.

Navegadores Neurais

Os navegadores neurais atuam como uma função não-linear distribuida, a qual mapeia estímulos a saídas. Este mapeamento é implementado por uma rede de processadores simples, os neurônios, interligados por conexões providos de sinapses. Os neurônios reagem a estímulos gerando um sinal de saída conforme a expressão:

$$a = f_T\,(f_g(entradas)) \tag{3.1}$$

onde a é a saída do neurônio, entradas são os sinais que chegam ao neurônio, já modulados pelas sinapses, f_g é a função de agregação e f_T é a função de ativação, geralmente uma função não-linear provida de saturação, como a função sigmóide.

Os navegadores neurais são providos de memória e de aprendizagem. A memória está distribuida nos pesos sinápticos e a memorização ocorre no processo de aprendizagem. Um rede neural aprende a associar estímulos com ações através da modificação dos pesos sinápticos. Existem vários algoritmos de aprendizagem, os quais podem ser divididos em duas categorias: os supervisionados e os não-supervisionados.

Os esquemas de aprendizagem supervisionados dependem do fornecimento de dados sobre o comportamento desejado do navegador, em termos dos possíveis estímulos experimentados por este. A complexidade do problema de navegação autônomo, representada pela vasta gama de situações possíveis, inviabiliza o uso de mecanismos da prendizagem supervisionados, pois não há como determinar a desempenho do sistema a cada instante.

Os métodos de aprendizagem não-supervisionada utilizam critérios mais genéricos para a otimização dos pesos sinápticos de uma rede neural. Como exemplo, as redes de Kohonen utilizam como parâmetros de aprendizagem a proximidade espacial de neurônios e seu grau de ativação face a estímulos. São fortalecidos as conexões entre sensores e neurônios com intensa ativação e enfraquecidos as conexões dos demais neurônios.

Duas propostas de navegadores neurais serão considerados: a de Verschure et al por sua simplicidade e a de Almassy por usar um algoritmo genético como mecanismo de aprendizagem.

Verschure et al. utilizaram uma rede neural de estrutura simples e aprendizagem por condicionamento [49].

A figura 3.3 ilustra a estrutura do navegador neural desenvolvido por Verschure. Três tipos de sensores são utilizados: um medidor de distâncias em 37 direções distintas, um indicador de colisões e outro da direção na qual está o objetivo. Cada sensor é associado a um conjunto de neurônios. Estes neurônios estimulam os neurônios motores, responsáveis por 5 tipos de movimento pré-determinados (desvio grande a esquerda, desvio pequeno a esquerda, em frente, desvio pequeno a direita e desvio grande a direita) e interagem entre si. Os neurônios de colisão inibem os neurônios associados à direção do alvo e os neurônios do medidor de distâncias estimulam tanto os neurônios de colisão quanto os neurônios de direção do alvo. Duas ações de reflexo são embutidas: virar para a direita quando houver colisão à esquerda e virar para a esquerda quando houver colisão à direita do veículo.

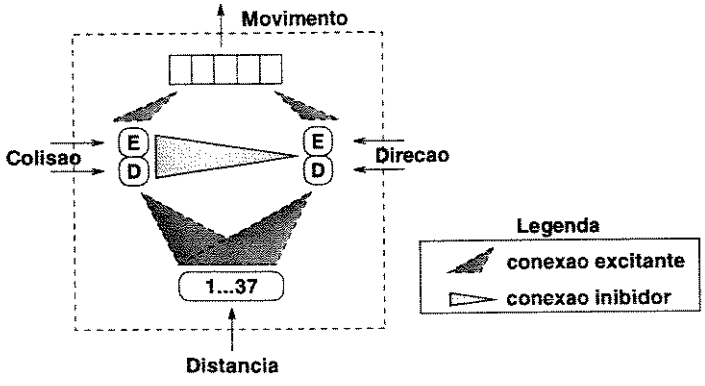


Figura 3.3: Navegador baseado em redes neurais de Verschure *et al.*

A aprendizagem é baseada na associação entre padrões sensoriais e colisões, utilizando uma forma da regra de Hebb:

$$\Delta K_{ij}^{\lambda\gamma} = \frac{1}{N}\left(\epsilon^{\lambda\gamma} s_i^{\lambda} s_j^{\lambda} - \eta s'^{\lambda} K_{ij}^{\lambda}\right) \tag{3.2}$$

onde K_{ij}^{λ} é o peso sináptico a modificar, situado na conexão entre o neurônio i do grupo (λ)de neurônios receptores e o neurônio j do grupo de neurônios de origem (γ), N é o número de neurônios ligados do grupo de origem, $\epsilon^{\lambda\gamma}$ é o coeficiente de aprendizagem

das conexões entre os grupos mencionados, s_j^{γ} é a saída (estado) do neurônio j do grupo (γ), s_i^{λ} é a saída (estado) do neurônio i do grupo (λ), s'^{λ} é a atividade média do grupo (λ), η é o coeficiente de esquecimento.

Demonstrou-se através de várias simulações, a capacidade do veículo de navegar em ambientes com muitos obstáculos (veja 3.4) e a emergência de comportamentos não definidos previamente, como a associação da proximidade de obstáculos como a colisão, levando ao desvio de obstáculos antes da colisão. Não foi avaliado o desempenho deste navegador em ambientes mais complexos como aquele contendo um obstáculo em " U ". São passíveis de crítica a pré-fixação das conexões entre os sensores e neurônios e a necessidade de determinação experimental dos coeficientes do algoritmo de auto-organização.

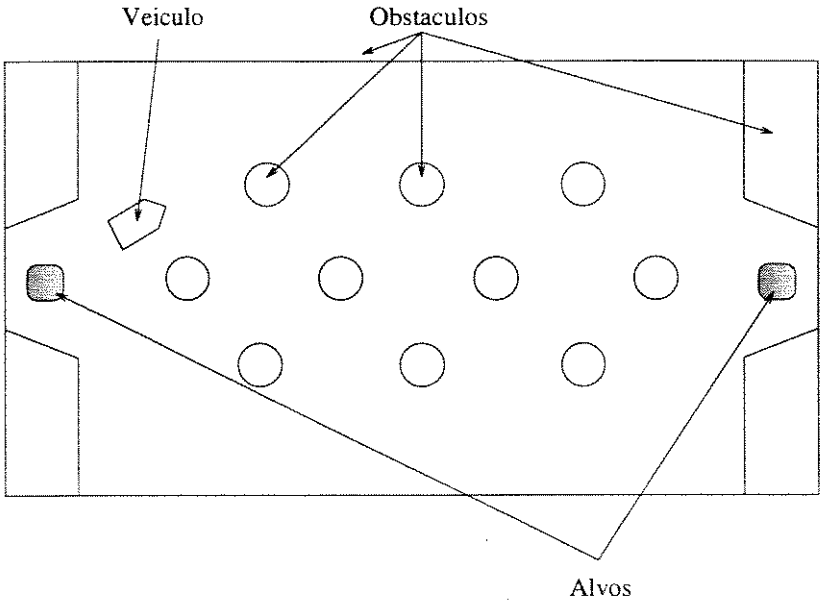


Figura 3.4: Ambiente de operação e utilização do navegador neural de Verschure.

Almassy e Vinkhuysen [1] expandiram o navegador neural de Verschure, acrescentando um sensor de movimento e mais um mecanismo de aprendizagem, um algoritmo genético. Os experimentos visavam o estudo da surgimento de comportamento grupal de robôs, daí o detetor de movimentos, na verdade um detetor de outros agentes. O algoritmo genético foi empregado para determinar os valores ótimos dos parâmetros de meca-

nismo Hebbiano de aprendizagem. A arquitetura utilizada está disposta na figura 3.5. As conexões inibidoras estão marcadas com uma flecha com circunferência, sendo as flechas simples conexões excitatórias.

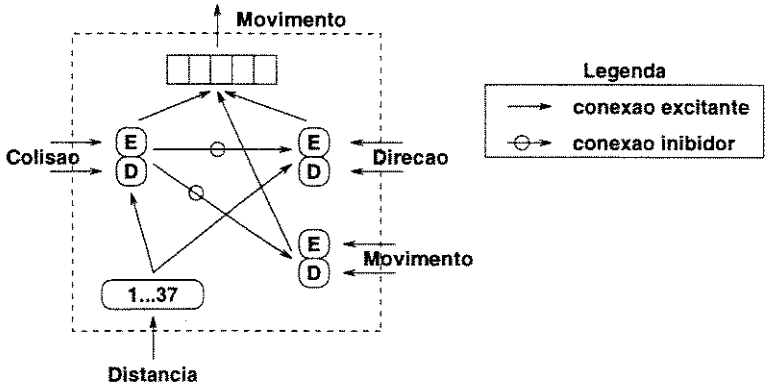


Figura 3.5: Arquitetura empregada por Almassy et Vinkhuyzen.

O algoritmo genético utilizado é de *estado contínuo*. Nesta variação os mecanismos de seleção e recombinação independem dos valores de desempenho globais da população, permitindo a evolução e avaliação concorrentes da população. Os indivíduos foram avaliados em simulações envolvendo múltiplos agentes. A função de desempenho possuia os seguintes critérios:

- alvos alcançados;
- colisões com outros agentes;
- colisões com obstáculos.

Observou-se novamente o surgimento de comportamentos novos e mais sofisticados como a tática de perseguir outros agentes para achar alvos.

As experiências de Almassy e Vinkhuysen demonstraram ser possível o desenvolvimento de dispositivos capazes de navegação autônoma através de mecanismos de auto-organização e como mínimo de interferência humana (determinação da função de desempenho).

Navegadores Nebulosos

Os sistemas nebulosos de navegação autônoma são outra instância de um sistema reativo. Estes sistemas são baseados na codificação de conhecimento na forma de relações lógicas entre variáveis nebulosas. A associação existente entre a reação dos sistema a um estímulo ambiental e o próprio estímulo é expressa por regras da forma:

Se X_1 é A_1^1 e X_2 é A_2^1 então Y_1 é B_1^1 e Y_2 é B_2^1

onde X_1 e X_2 são as variáveis nebulosas dos antecedentes, associados aos estímulos, A_1^1 e A_2^1 são conjuntos nebulosos das variáveis dos antecedentes, Y_1 e Y_2 são as variáveis nebulosas dos consequentes, associados às ações tomadas pelo navegador e B_1^1 e B_2^1 são conjuntos nebulosos das variáveis de saída. Os conjuntos nebulosos são representados, nas regras, por termos linguísticos, como PEQUENO, BAIXO, LONGE.

Uma regra típica de um navegador nebuloso, descrevendo a ação tomada pelo sistema quando prestes a colidir com um obstáculo a sua direita seria:

Se DISTANCIA-AO-OBSTACULO-A-ESQUERDA é GRANDE e DISTANCIA-AO-OBSTACULO-A-DIREITA é PEQUENA então VELOCIDADE-RODA-DIR é ALTA e VELOCIDADE-RODA-ESQ é BAIXA

A forma clara das regras nebulosas facilita a síntese de conhecimento a partir da experiência de um humano e permite a rápida implementação deste conhecimento em um sistema autônomo.

O projeto de um sistema nebuloso requer, além do banco de regras, a definição das variáveis relevantes ao problema, do intervalo de variação destas variáveis nebulosas (universo de discurso), do particionamento deste intervalo e da associação destas partições a rótulos linguísticos e a uma curva conferindo um grau de pertinência de um estímulo daquele subintervalo àquela categoria (veja figura 3.6).

As regras são ativadas em paralelo e a resposta do sistema é formado por uma composição dos consequentes das regras ativadas. Como as informações provenientes dos sensores e os sinais esperados pelos atuadores não são nebulosos, é necessária a transdução dos valores nebulosos utilizados pelo mecanismo de inferência em valores crisp (defuzzificação) e vice-versa (fuzzificação).

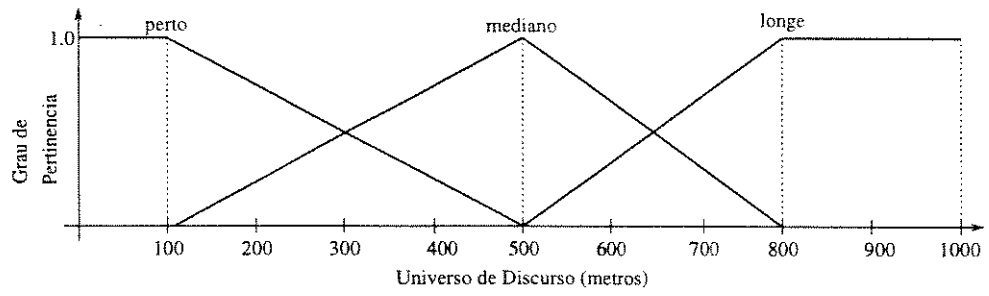


Figura 3.6: Exemplo de funções de pertinência.

As deficiências dos sistemas nebulosos residem na origem do conhecimento neles embutidos. Por não dispor de um mecanismo de aprendizagem integrado, todo o conhecimento é derivado da experiência de um humano. Esta dependência restringe a utilização do navegador a ambientes e situações já caracterizadas pelo operador. Dada a complexidade dos ambientes associados a problemas de controle autônomo, mesmo com sua capacidade de manipular dados imprecisos, tais sistemas seriam de pouca eficácia. O uso da ontologia humana também limita a gama de sensores utilizáveis àqueles semelhantes ao sensores humanos. Portanto, o sistema navegador poderá ser no máximo tão eficaz quanto um humano.

A primeira deficiência poderá ser compensada através de um algoritmo de aprendizagem. A questão dos sensores ainda não possui solução satisfatória. A escolha ainda deve ser feita pelo projetista.

Dentre os navegadores nebulosos destacam-se o de Li [34] pelo desempenho em situações complexas tipo "U" e o de Leitch [33], o qual utilizou um algoritmo genético como método de aprendizagem das regras nebulosas.

O navegador nebuloso de Li demonstra um ótimo desempenho tanto em ambientes simples como corredores quanto em ambientes mais complexos com obstáculos em "U". Li utiliza quatro sensores (distâncias a obstáculos situados a esquerda, direita e a frente do veículo, direção atual de movimento) e dois atuadores: as velocidades imprimidas às rodas esquerda e direita. As regras foram determinadas experimentalmente. Quatro comportamentos essenciais foram identificados e implementados por Li: busca de alvos, seguir paredes, evitar colisões e diminuir a velocidade em curvas e caminhos estreitos. Como as

regras compartilham as variáveis de entrada e de saída e não interagem diretamente, os quatro grupos de regras podem ser consideradas um conjunto único.

Leitch empregou um algoritmo genético para desenvolver a base de regras nebulosas. A base de conhecimento nebuloso foi codificado como uma cadeia ordenada de parâmetros: <entradas significativas><centros das funções de pertinência das entradas><centros das funções de pertinência das saídas><base das funções de pertinência><consequentes das regras>

Observe que as funções de pertinência devem ser simétricas e de formato pré-especificado. O alfabeto utilizado foi (0, 1, E), onde E é um divisor de parâmetros. Os valores numéricos são recuperados utilizando conversão binária inteira (e.g. $1011_b = 11_d$) e fracional, no caso dos consequentes (e.g. $1011_b = (1/11)_d$). Este esquema de codificação permite o crossover de cromossomas em qualquer posição da cadeia, aumentando a diversidade genética e evitando a convergência prematura.

A função objetivo utilizada é a principal deficiência do método de Leitch, pois é baseada em uma medida de erro entre a trajetória ideal e a trajetória realizada pelo navegador. Apesar destas trajetórias serem consideradas no espaço de estados, tal medida necessita de uma caracterização detalhada do ambiente.

Leitch realizou duas simulações: a navegação e a realização de uma meia-volta em um corredor. A aprendizagem do segundo comportamento foi realizado sempre da mesma posição e orientação inicial e dividido em etapas, cada qual correspondendo a uma etapa da meia-volta. A meia-volta completa foi realizada com a agregação das regras aprendidas em cada etapa.

Apesar do sucesso na realização destas tarefas, a natureza limitada dos ambientes utilizados indica a fragilidade do mecanismo de aprendizagem.

3.3.3 Navegação Comportamental

Os sistemas baseados em comportamentos podem ser considerados uma evolução dos sistemas reativos.

Comportamentos são modos distintos de operação caracterizados por conjuntos

distintos de relações entre estímulos e ações. Um módulo pode compartilhar sensores com outros módulos. Os módulos interagem, modificando os sinais de entradas fornecidos a outros módulos (veja a figura 3.7). A composição das ações de cada módulo será a ação tomada pelo sistema.

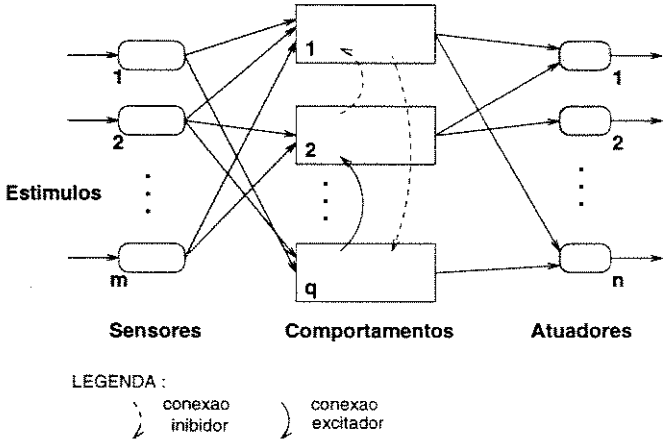


Figura 3.7: Diagrama esquemático de um sistema comportamental.

O grau de interação entre os módulos determinará se o sistema é realmente baseado em comportamentos. Se os módulos comportamentais influenciam a ação do sistema apenas através de uma agregação de suas saídas, o sistema é dito reativo, pois a distinção entre módulos é apenas formal. É o caso do navegador nebuloso de Li [34] exposto acima. A arquitetura *subsumption* de Brooks [4] apresenta módulos compartamentais que interagem entre si. A influência de um módulo sobre outro ocorre apenas como uma inibição (*subsumption*) das entradas deste por aquele. Os navegadores baseados na teoria de Seleção Neural proposta por Edelman [7] são mais sofisticados, pois permitem uma extensa interação inter e intracomporamental. A interação ocorre através de conexões inibidoras ou reforçantes entre grupos neurais distintos.

A interação entre os comportamentos e os próprios comportamentos podem ser construídos durante o projeto (*a la* Brooks) ou por meio de um mecanismo não-supervisionado de aprendizagem e adaptação (como proposto por Edelman).

O "comportamento" navegação autônoma pode ser decomposto, por exemplo,

nos seguintes comportamentos básicos:

- seguir paredes;
- evitar obstáculos;
- seguir alvo (seja este estático ou dinâmico).

Um diagrama de blocos descrevendo o comportamento acima é mostrado na figura 3.8.

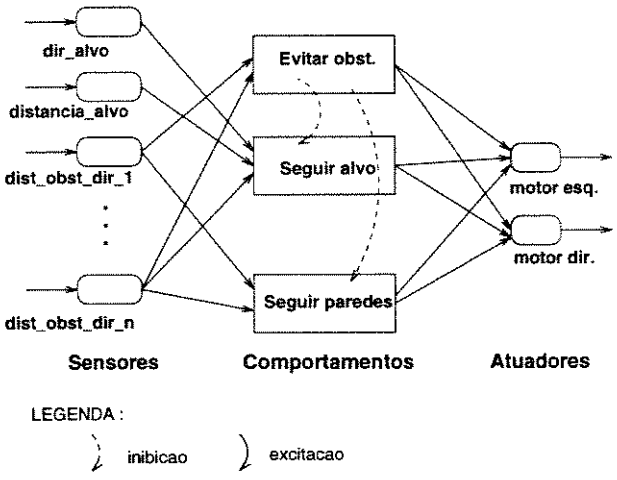


Figura 3.8: Exemplo de um sistema de navegação comportamental.

Arquitetura *Subsumption*

A arquitetura *subsumption* é um modelo baseado na interação de comportamentos (*behaviors*) proposto por Brooks [6],[4]. Estes comportamentos são implementados como segmentos de código, emulando máquinas de estado, e organizados em uma estrutura hierárquica e distribuída. A hierarquia é implementada por meio de restricões de comunicação entre módulos. Os módulos de comportamento de um dado nível hieárquico poderão se comunicar apenas com outros do mesmo nível ou pertencentes a níveis mais baixos. A operação 'e distribuída pois o comportamento emergente do agente é derivado da agregação dos comportamentos de cada módulo. A interação ocorre através de relações de *subsumption*. A operação dos módulos de comportamento é contínua e isócrona.

Os módulos algorítmicos são descritos por máquinas incrementais de estado finito. O autômato de estado finito possui um conjunto de registradores internos, entradas e saídas e um relógio. O autômato percorre um ciclo de estados (veja Fig.3.9), os quais caracterizam o comportamento apresentado por aquele módulo.

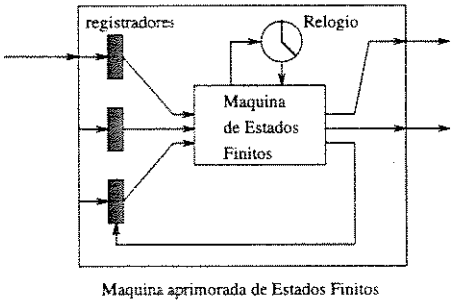


Figura 3.9: Esquema da Máquina aprimorada de Estados Finitos.

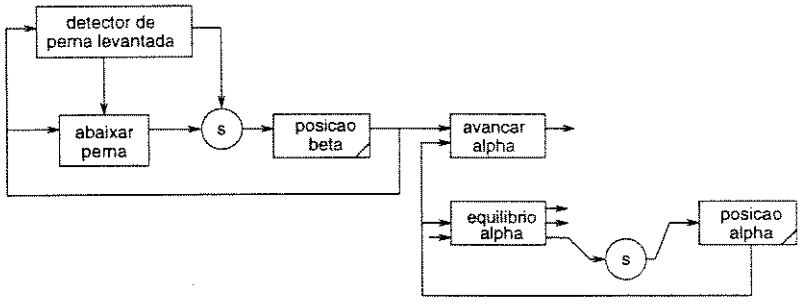
Esquema da Máquina aprimorada de Estados Finitos.

A interação entre os módulos ocorre através de conexões de *subsumption*, os quais possuem a função de transmissão de dados entre módulos e também podem servir como inibidores de módulos (veja Fig.3.10).

A possível complexidade de um agente baseado nestas máquinas incrementais de estado levou a criação de um linguagem de alto nível, a linguagem de comportamentos (*behaviour language*), pela equipe de Brooks [5], para auxiliar na definição e depuração do sistema de módulos.

Este método foi implementado fisicamente em uma série de robôs (Genghis, Attila). Este robôs se locomovem por meio de um conjunto de pernas simples e são capazes de navegar em terrenos ligeiramente acidentados e desconhecidos como o ambiente de um escritório. O sistema *subsumption* controlava a coordenação das pernas, além da navegação do robô.

Gat *et al.* criaram uma arquitetura *subsumption* aprimorada, descrita pela linguagem ALFA [13]. Esta arquitetura substitui a interação por inibição de entradas por interfaces de agregação. Estes são associados a grupos de módulos cooperantes e são responsáveis pela recepção e filtragem dos sinais passados entre os grupos. Deste modo Gat *et*



Adaptado de Luc Steels, 1994.

Figura 3.10: Exemplo de uma arquitetura subsumption: rede parcial de um autômato de estados finitos para a locomoção de um robô hexápide. As caixas indicam variáveis de estado. Caixas com um traço no canto inferior direito são autômatos de estados finitos. Os nodos "s" indicam uma relação de inibição (*subsumption*). No exemplo, a ativação do "detector de perna levantada" inibe a saída de "perna abaixada" para o autômato "posição beta".

al procuram assegurar o funcionamento de módulos de comportamento de níveis hierárquicos inferiores mesmo com a adição de níveis superiores, Fig.3.11.

Mataric desenvolveu um método de aprendizagem para a estrutura *subsumption* em [35]. Seu método utiliza uma matriz de pesos representando o grau de conexão entre os estados possíveis do agente e as ações passíveis de execução.

Neuronal Group Selection

A teoria de seleção de grupos neurais (*Neuronal Group Selection*) de Edelman também poderá ser vista como sendo um sistema comportamental [7][8][9]. Neste modelo, baseado na extensa pesquisa desenvolvida em neurofisiologia, Edelman propõe que a unidade básica do processamento neural seja um grupo de neurônios densamente interconectados e não um neurônio isolado. Estes grupos neurais são organizados em conjuntos, os *repertórios*. Cada repertório é responsável pelo processamento de um determinado conjunto de estímulos. O sistema neurobiológico é então formado a partir da interação entre repertórios através de conexões entre repertórios.

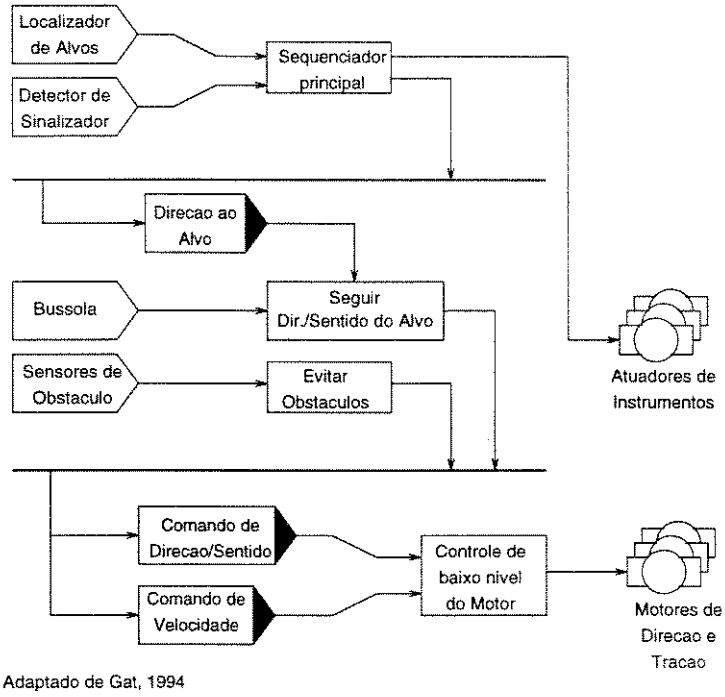


Figura 3.11: Exemplo de arquitetura subsumption extendida: rede de navegação do robô Rocky III.

A aprendizagem ocorre em três etapas: seleção formativa (*developmental selection*), seleção por experiências (*experiential selection*) e mapeamento recorrente (*reentrant mapping*). No primeiro estágio, múltiplos repertórios são formados e distribuídos de forma que existam vários repertórios para cada conjunto de sensores. Assim, haverá vários repertórios competindo entre si para responder a determinados padrões de estímulos. Esta distribuição de padrões entre repertórios ocorre na segunda etapa de aprendizagem. Aqui utiliza-se condicionamento para associar repertórios a padrões relevantes de estímulos. Em seguida os repertórios representando padrões de informações são interligados por conexões recorrentes para formar padrões de comportamento mais complexos. Estas duas últimas etapas da aprendizagem, seleção experimental e mapeamento recorrente são contínuas e ocorrem também durante a operação do sistema.

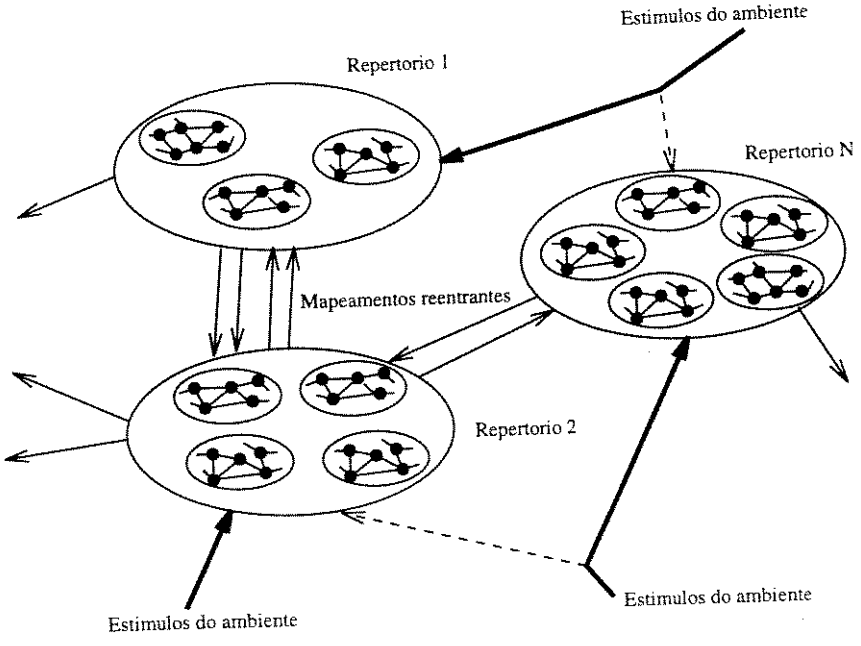


Figura 3.12: Exemplo de arquitetura formado por Neuronal Group Selection.

Este modelo sofisticado de processamento neurobiológico foi aplicado em um autômato denominado NOMAD-Darwin IV. Este robô era provido de uma câmera de vídeo, sensores infravermelhos de proximidade, um sensor de condutância atrelada a uma garra magnética e duas rodas de movimento independente . A complexa rede neural foi imple-

mentada em um computador massivamente paralelo, um nCube/10, ligado através de rádio ao robô. As experiências realizadas demonstraram que o modelo é capaz de aprender a seguir uma fonte de luz e também de diferenciar entre objetos através da associação de condutividade do objeto e cor do objeto (azul a não-condutor e vermelho a condutor).

As desvantagens observadas no autômato de Edelman são a necessidade de recursos computacionais avantajados e a relativa vagarosidade de operação [40]. Ambas as desvantagens podem ser atribuídas a complexidade computacional do modelo.

3.4 Um Método de Navegação Autônoma

O navegador proposto é um sistema híbrido, empregando uma rede neuronebulosa e um algoritmo genético. A rede neuronebulosa é responsável pelo processamento efetivo, oferecendo processamento paralelo e robusto (rede neural), fácil extração do conhecimento adquirido (dada pela relação entre a estrutura da rede e a base de regras a ela associada) e capacidade de operar com informações imprecisas (uso da lógica nebulosa). O algoritmo genético será responsável pela aprendizagem desta rede.

A rede neuronebulosa utilizada possui a arquitetura apresentada no capítulo 2.

A seguir serão descritas em detalhes as características do navegador proposto.

3.4.1 Parâmetros Utilizados

O navegador utilizado para a simulação e avaliação do método proposto possui as seguintes características:

- sensores S: compostos por um sensor de alvo, fornecendo a direção e sentido no qual se encontra o alvo e um sensor de obstáculos, fornecendo a distância ao e a direção e sentido do obstáculo mais próximo. Ambas as direções são dadas em relação à direção de movimento do veículo, Figs. 3.13 e 3.14.
- veículo V: dotado de sensores S e capaz de mudar o sentido de movimento em até 18 graus por vez. Um ângulo maior provavelmente não poderia ser executado por um veículo real. O movimento é sempre na direção e sentido atuais.

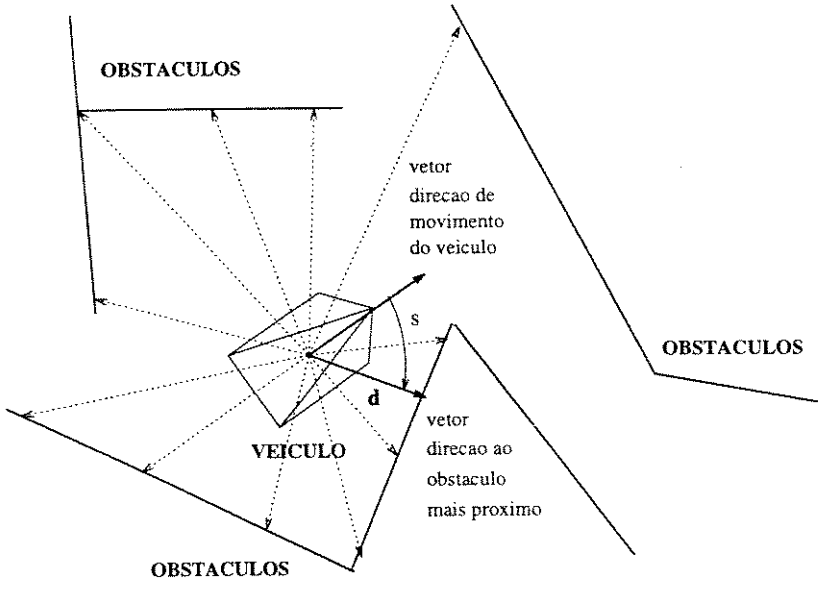


Figura 3.13: Caracterização do Veículo utilizado: sensores de obstáculos.

– ambiente A: composto de um conjunto de obstáculos retilíneos de espessura desprezível e um conjunto de alvos. Estes são apresentados aos sensores do veículo em seqüência. Um novo alvo será gerado logo após o veículo atingir o anterior. Exemplos de ambientes estão mostrados na Fig. 3.15.

3.4.2 Caracterização da Rede Neuronebulosa

O controlador neuronebuloso proposto possui três variáveis nebulosas de entrada e fornecerá um sinal não-nebuloso de controle. As variáveis de entrada correspondem aos sensores especificados no item anterior, a saber: dir_{alvo} (direção do alvo), $dist_{obst}$ (distância ao obstáculo mais próximo) dir_{obst} (direção no obstáculo mas próximo).

A variável $dist_{obst}$ possui um universo de discurso, $U_2 = [0, d_{max}]$, onde d_{max} é a maior distância possível entre um obstáculo e o veículo. Três partições são alocadas para esta variável.

A variável dir_{alvo} possui como universo de discurso, U_1, o intervalo real [-180, 180], representado os ângulos possíveis. Um ângulo à direita do sentido de movimento do agente é

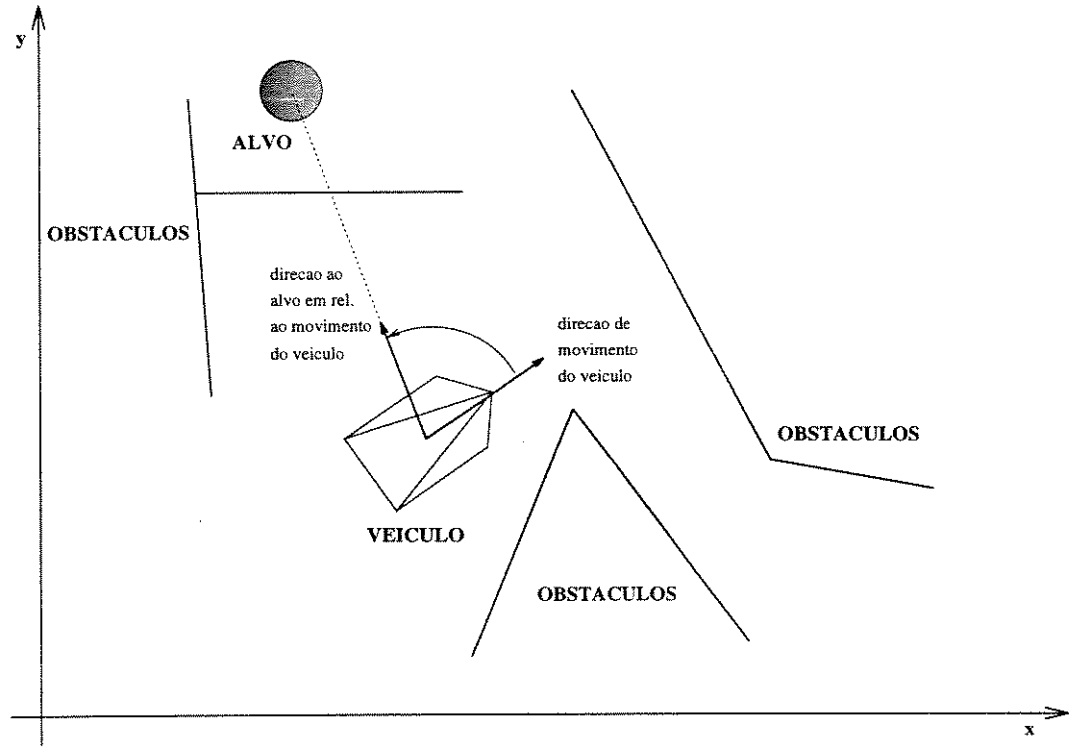


Figura 3.14: Caracterização do Veículo utilizado: sensor de alvo.

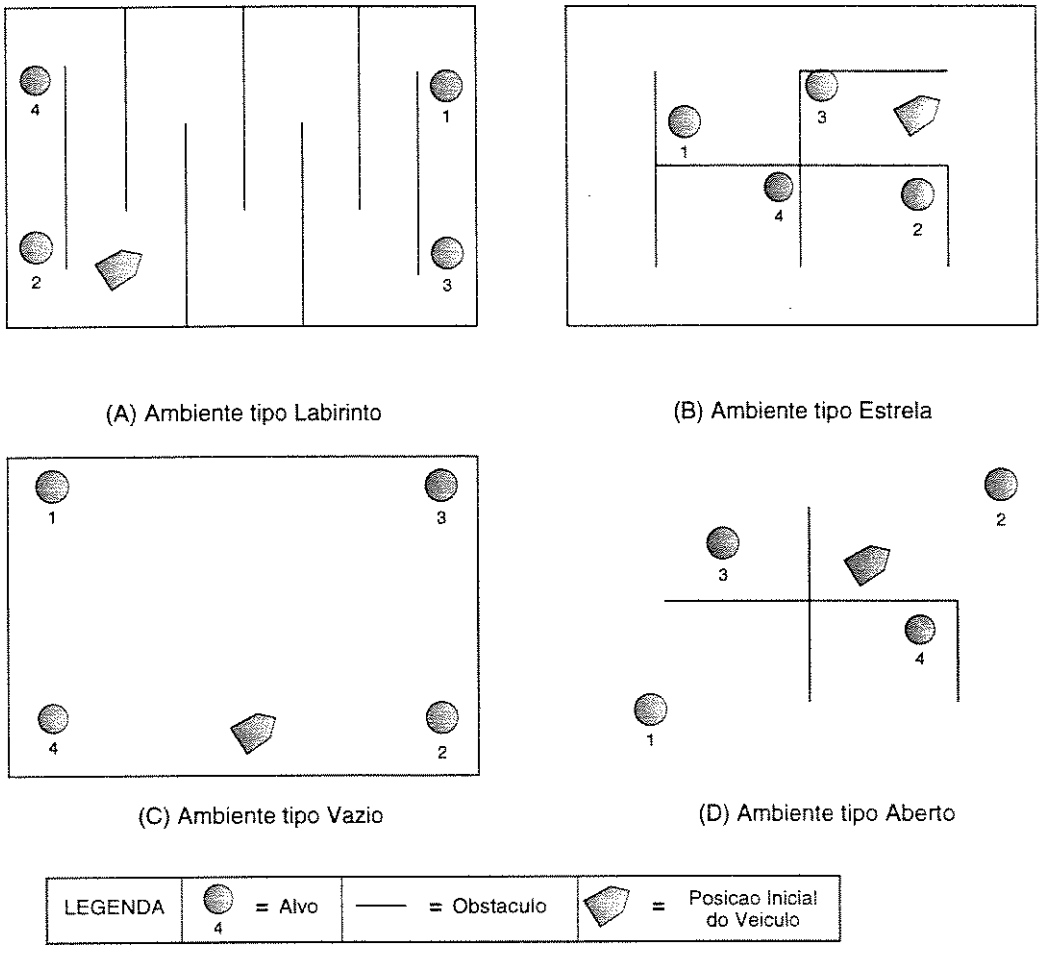


Figura 3.15: Tipos de Ambientes de Operação. Os números abaixo dos alvos indicam a ordem em que os alvos são apresentados ao veículo.

considerado negativo. U_1 possui um máximo de 5 partições. A associação de termos a cada partição é subjetiva e só terá sentido semântico após a determinação da base de regras.
A terceira e última variável, dir_{obst} possui o mesmo universo de discurso e o mesmo número máximo de partições da variável dir_{alvo}. Tomando uma discretização dos universos de discurso em 256 intervalos, obter-se-á a seguinte topologia da rede neuronebulosa:

- 256 x 3 = 768 neurônios na primeira camada (fuzzificação);
- 3 x 5 x 5 x 3 = 225 neurônios na segunda camada (matching);
- 5 x 3 x 5 = 75 neurônios na terceira camada (aggregação de antecedentes);
- 2 neurônios na quarta camada (agregação de regras: somas);
- 1 neurônio na quinta e última camada (agregação de regras, divisão).

3.4.3 Caracterização do Algoritmo Genético

O algoritmo genético empregado possui as seguintes características:

- população composta de 108 indivíduos
- cromossomo composto por dois vetores de formato misto inteiro/real. O primeiro vetor encerra as funções de pertinência das variáveis nebulosas. Cada função é representada por 2 números: um real, indicando o centro da função e um inteiro especificando o formato geométrico da função (0 para uma formato triangular e 1 para uma forma trapezoidal). As partições podem ser reconstruidas de acordo com a Fig. 3.16. O segundo vetor contém os consequentes reais de cada regra nebulosa.
- *crossover* com ponto único de corte e probabilidade fixa em 1.0, ou seja, sempre ocorre *crossover*.
- mutação dupla: na seção do cromossomo relativo às funções de pertinência a mutação ocorre por inversão (10e01) e no restante dos genes a mutação acrescenta ao gen um valor inteiro no intervalo [-4,4]. A probabilidade de mutação é de 0.00051 por gen, ou seja, 0.05 por cromossomo.
- seleção para recombinação: algoritmo de Monte Carlo.

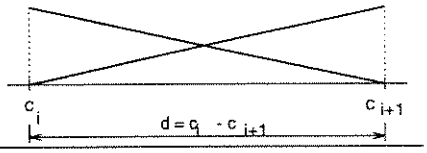
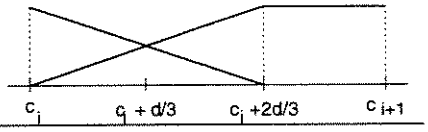
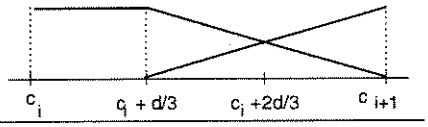
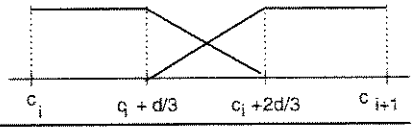
Forma da Funcao de Pertinencia (i) (centro c_i)	Forma da Funcao de Pertinencia (i+1) (centro c_{i+1})	Forma de Construcao do lado esquerdo da funcao (i+1) e do lado direito da funcao (i)
triangular	triangular	
triangular	trapezoidal	
trapezoidal	triangular	
trapezoidal	trapezoidal	

Figura 3.16: Construção das Funções de Pertinência pela Decodificação do Cromossomo.

- *gap* variável. A nova população é formada pela seleção de indivíduos a partir de uma população temporária formada pela união da população da geração anterior com a população gerada por recombinação.
- uma função de desempenho dada por:

$$f = p_1 * n^2_{\text{alvos alcançados}} + p_2 * (d_{max} - d_{final,alvo})^{-1} + p_3 * z \tag{3.3}$$

$$z = \begin{cases} dist_{percorrida} & \text{se nenhum alvo foi achado} \\ \frac{p_1}{dist_{percorrida}} & \text{se algum alvo foi achado} \end{cases} \tag{3.4}$$

onde $p_1 = 10^4$,$p_2 = 10$ e $p_3 = 10$ são pesos definidos empiricamente, $dist_{percorrida}$ é a média da distância percorrida (distância total dividido pelo número de alvos achados). Esta função de desempenho foi modificada à medida que os resultados das simulações foram sendo analisadas.

Utilizou-se também os operadores secundários *scaling* e *sharing*.

3.5 Resumo

O problema da navegação autônoma de veículos foi apresentada e suas características analisadas. Constatou-se sua complexidade e dificuldade de solução resultantes do grande número de parâmetros envolvidos e da interação entre estes.

As propostas de solução existentes foram classificadas por dois critérios básicos: filosofia de projeto e características operacionais. De acordo com o primeiro critério, existem três categorias de métodos de navegação: sistemas reativos, sistemas comportamentais e sistemas planejadores. Da grande gama de parâmetros operacionais, escolheu-se três principais: presença de memória, presença de um mecanismo de aprendizagem/adaptação e a utilização de dados locais ou globais.

Seis métodos representativos foram apresentados e analisados. Todos atendiam apenas parcialmente os requisitos de reduzida dependência de conhecimento embutido pelo projetista, facilidade de extração de conhecimento aprendido e navegação em ambientes complexos desconhecidos.

Por fim apresentou-se um método alternativo de navegação autônoma, no qual se utiliza uma rede neuronebulosa como plataforma computacional e um algoritmo genético com mecanismo de aprendizagem. Este sistema híbrido visa aproveitar as capacidades de aprendizagem não supervisionada, de tratamento de informações imprecisas, processamento paralelo e facilidade de extração de conhecimento.

O capítulo seguinte apresenta os resultados de simulações do sistema proposto. São mostradas dados sobre a fase de aprendizagem do navegador e o desempenho do navegador treinado em diversos ambientes. Ao final os resultados são analisados.

Capítulo 4

Resultados de Simulação

4.1 Introdução

Com base nos modelos expostos no capítulo anterior, implementamos um conjunto de programas visando analisar a eficácia do método proposto. À medida que obtivemos resultados, alteramos os parâmetros de simulação para verificar o comportamento resultante. Apresentaremos a seguir os resultados obtidos bem como as decisões realizadas.

Foi adotado como plataforma de controle a rede neuronebulosa proposta por Figueiredo *et al* em [11]. A aprendizagem deste controlador foi implementada através de um algoritmo genético, em substituição ao algoritmo de gradiente empregado originalmente em [11] por três motivos:

- a inexistência de derivadas conhecidas para funções nebulosas de ativação (em [11] foi proposta uma derivada nebulosa)
- a falta de conjuntos de dados do tipo (entrada, saída) que possam caracterizar um modo ótimo de navegação autônoma.
- utilização de parâmetros genéricos na otimização por algoritmo genético, como exposto no capítulo 2.

A seção seguinte descreve a variante utilizada pelo algoritmo genético e o problema a ser solucionado (ambiente, sensores, atuadores). Em seguida são apresentados os resultados

simulações, bem como a evolução sofrida das soluções a medida que se apreciava os dados colhidos das experiências [40]. Por fim apresentamos uma sinopse do trabalho realizado e as conclusões decorrentes dos resultados.

4.2 Análise de Resultados

O problema de navegação autônoma se revelou mais complexo do que inicialmente esperado, fato comprovado pela ineficácia dos parâmetros empregados na primeira experiência.

Os experimentos foram realizados usando seguinte método:

- Determinação dos parâmetros a serem utilizados durante a aprendizagem por algoritmo genético:
  - escolha de um ambiente de treinamento. A população de navegadores será avaliada de acordo com seu comportamento neste ambiente.
  - escolha de uma função de desempenho, pela qual o desempenho de cada indivíduo da população será avaliada.
- Aplicação do algoritmo genético à população de navegadores. Cada indivíduo terá 10000 unidades de tempo para navegar no ambiente de teste. Ao final deste intervalo de tempo ou em caso de colisão, lhe será alocado um valor de desempenho, calculado pela função determinada em uma etapa anterior.
- Após algumas gerações, o algoritmo é suspenso e o desempenho do melhor indivíduo é analisado em vários ambientes.

A geração na qual se suspendeu o algoritmo genético variou de experimento em experimento. Em geral, suspendeu-se o algoritmo após 4 dias de execução. O algoritmo genético foi executado em várias estações de trabalho IBM RS6K compartilhadas através de uma rede local. Dado as diferentes cargas de utilização das estações ocorreu uma variação no número de gerações completadas no intervalo de 4 dias. A tarefa exigindo o maior tempo de computação é a simulação de cada indivíduo da população em um ou mais ambientes de operação. Estas simulações ocorrem em todas as gerações. Os diferentes parâmetros

utilizados em cada experimento também contribuiram para a diferença de gerações completadas.

As figuras mostrando o comportamento dos navegadores em ambientes diversos podem ser interpretados do seguinte modo

- a seta externa ao ambiente e com flecha vazia indica o alvo atual, ou seja, o alvo atualmente buscado pelo veículo
- a seta externa ao ambiente e com flecha preenchida indica o alvo já atingido pelo veículo
- a seta dentro do ambiente e com flecha preenchida indica a posição inicial do veículo

Para fins de comparação foi utilizado uma rede neuronebulosa projetada heuristicamente. As suas funções de pertinência e o seu desempenho nos ambientes de teste utilizados podem ser vistas nas figuras 4.1 e 4.2-4.5, respectivamente.

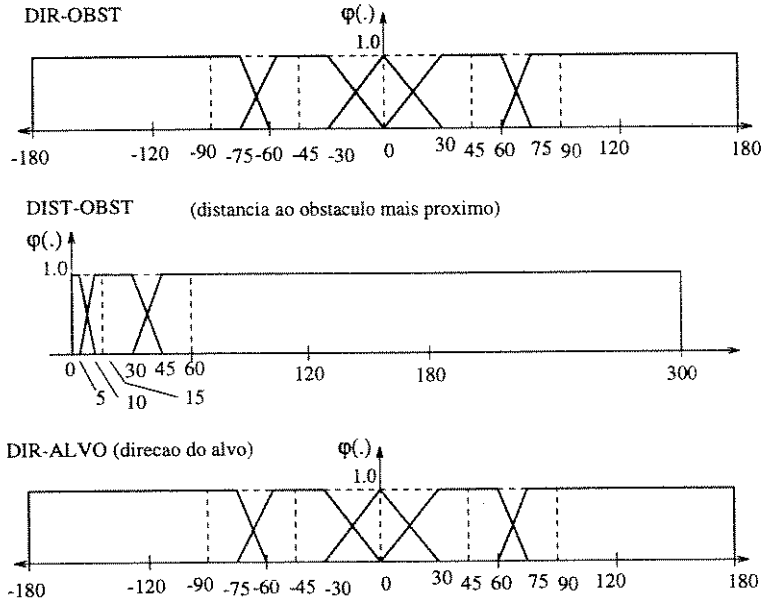


Figura 4.1: Funções de pertinência utilizados no navegador de referência.

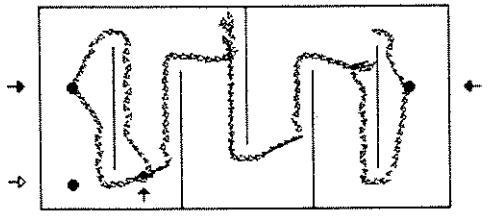


Figura 4.2: Percurso executado pelo veículo usando o navegador de referência no ambiente tipo Labirinto.

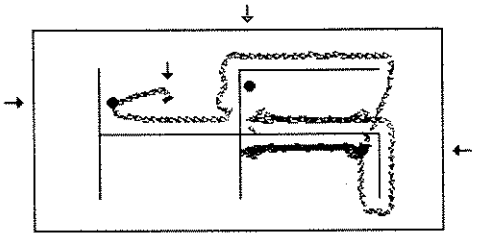


Figura 4.3: Percurso executado pelo veículo usando o navegador de referência no ambiente tipo Estrela.

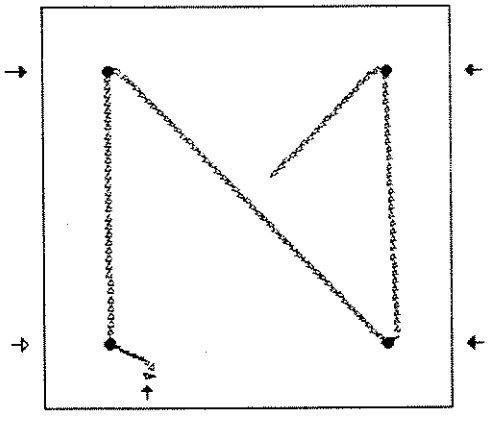


Figura 4.4: Percurso executado pelo veículo usando o navegador de referência no ambiente tipo Vazio.

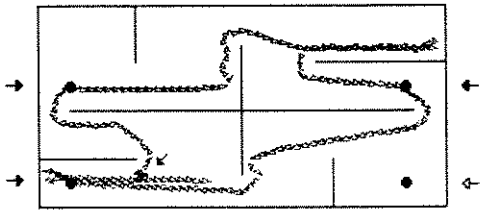


Figura 4.5: Percurso executado pelo veículo usando o navegador de referência no ambiente tipo Estrela2.

4.2.0.1 1a. Experiência

Neste primeiro experimento, utilizou-se no treinamento da rede neuronebulosa um ambiente complexo do tipo labirinto e a função objetiva bastante abrangente dada abaixo.

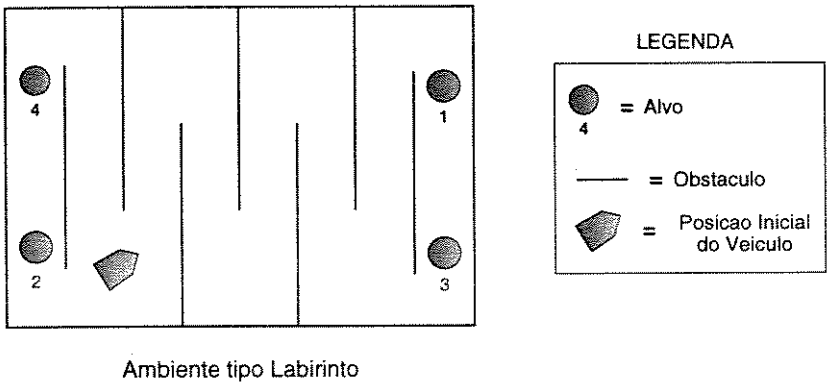


Figura 4.6: Ambiente tipo labirinto utilizado na avaliação da população de redes neuronebulosos na primeira experiência.

–

$$f = 10000 n_{\text{alvos-alcancados}} + \frac{p_1}{\left(d_{max} - d_{\text{alvo-atual,termino}}\right)} + \frac{p_2}{n_{\text{passos-dados}}} \quad (4.1)$$

– Se a avaliação terminar com colisão, $p_1 = p_2 = 20$ senão $p_1 = p_2 = 100$.

O controlador obtido não aprendeu a reconhecer nem os alvos nem os obstáculos (veja 4.9 a 4.12). É provavel que a disposição dos obstáculos tenha impedido o encontro de um alvo por parte do veículo, reduzindo a influência do alvo sobre a evolução do controlador a apenas à parcela de aproximação máxima ao alvo. A evolução do desempenho da população pode ser vista na figura 4.8.

As funções de pertinência obtidas estão apresentadas na figura 4.7. A partição assimétrica do universo de discurso da variável DIR-OBST (direção ao obstáculo mais próximo) possivelmente indica a não-aprendizagem por parte do navegador do "conceito" obstáculo à direita.

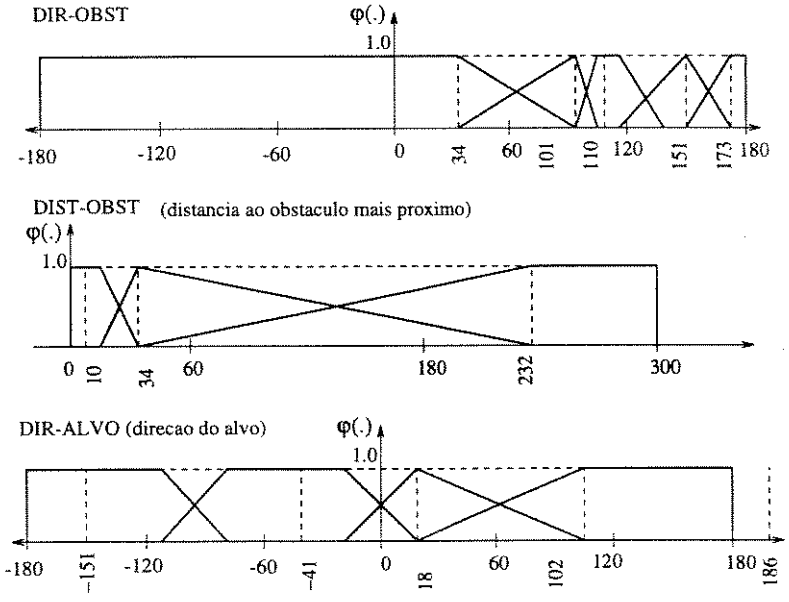


Figura 4.7: Funções de pertinência evoluidos na primeira simulação.

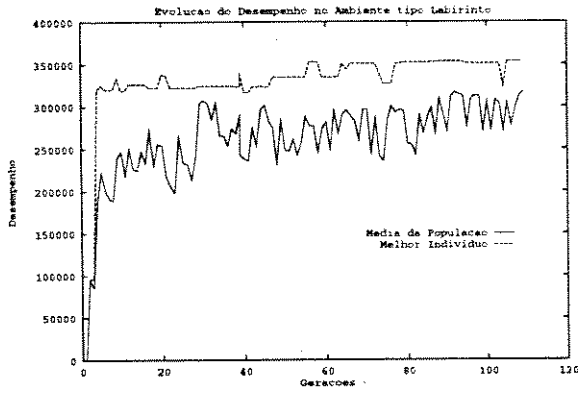


Figura 4.8: Evolução de Desempenho do Navegador avaliado em ambiente tipo Labirinto.

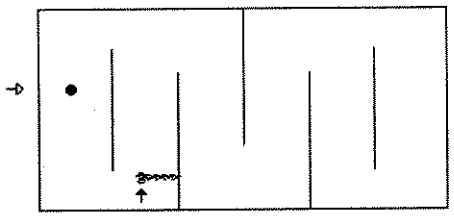


Figura 4.9: Percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador evoluido em ambiente tipo Labirinto (ver texto).

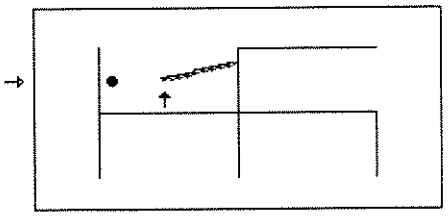


Figura 4.10: Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador evoluido no ambiente tipo Labirinto.

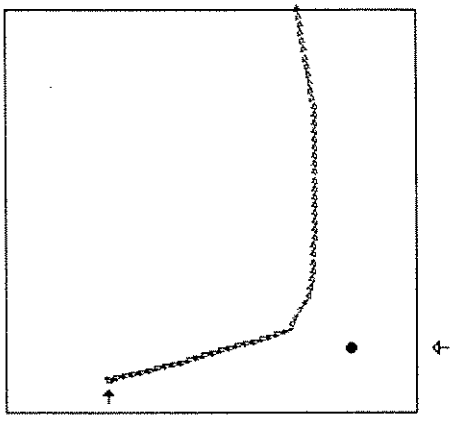


Figura 4.11: Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido em ambiente tipo Labirinto.

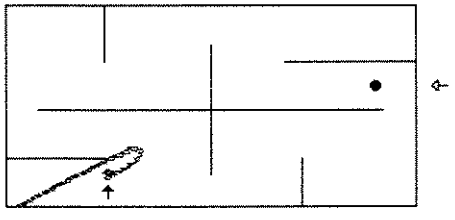


Figura 4.12: Percurso no ambiente tipo Estrela2 executado pelo veículo usando o navegador evoluido em ambiente tipo Labirinto.

4.2.0.2 2a Experiência

A possível relação entre a dificuldade do ambiente tipo labirinto e a não-aprendizagem dos "conceitos" de obstáculo e alvo levou ao uso de um ambiente mais simples, o ambiente tipo estrela (figura 4.13). Esta configuração forma um caminho periférico livre, permitindo ao veículo acesso a todo o ambiente.

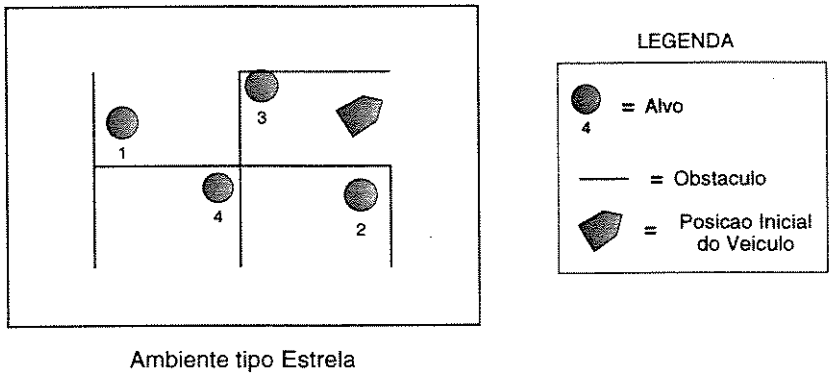


Figura 4.13: Ambiente tipo Estrela utilizado para a avaliação da população de redes neuronebulosas durante a evolução.

Os resultados foram insatisfatórios. O controlador desenvolvido após 74 gerações (figura 4.14) novamente colidia rapidamente com os obstáculos sem achar alvo algum (figuras 4.17 a 4.20). Aparentemente o ambiente continua muito complexo.

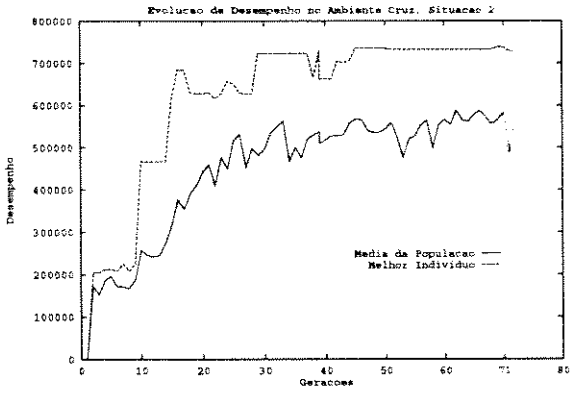


Figura 4.14: Gráfico da evolução do desempenho dos controladores ao longo das gerações.

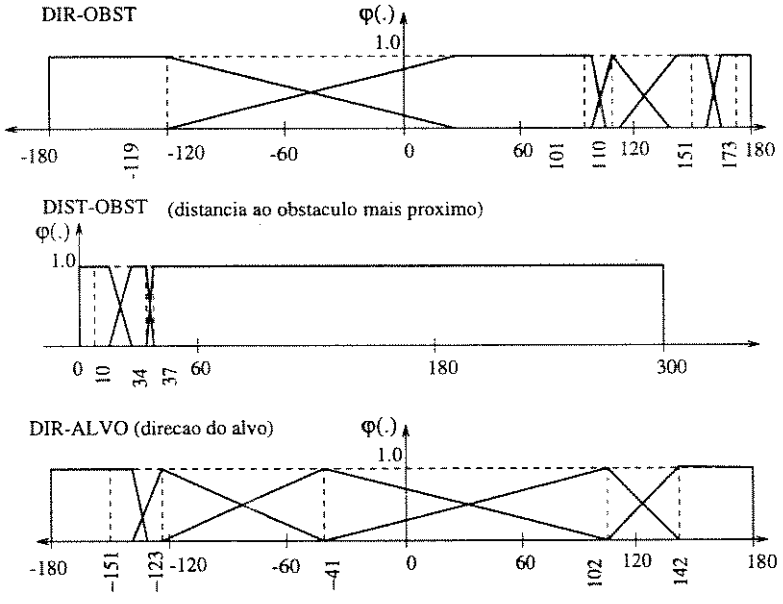


Figura 4.15: Funções de pertinência evoluidos na segunda simulação.

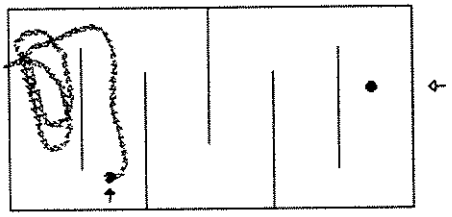


Figura 4.16: Percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador evoluido em ambiente tipo Estrela.

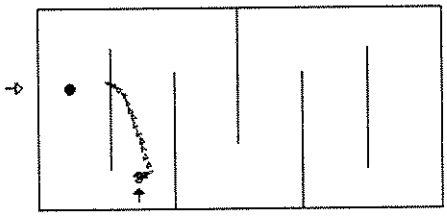


Figura 4.17: Outro percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador evoluido em ambiente tipo Estrela.

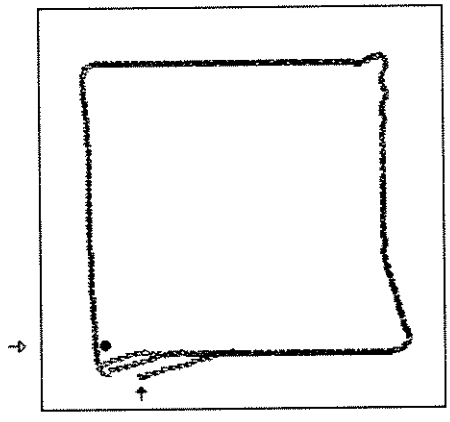


Figura 4.18: Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido em ambiente tipo Estrela.

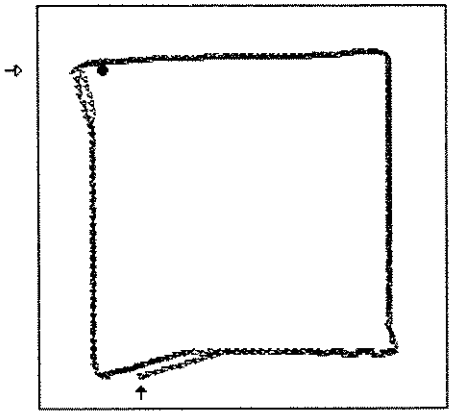


Figura 4.19: Novo percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido em ambiente tipo Estrela.

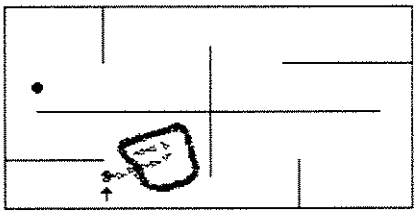


Figura 4.20: Percurso no ambiente tipo Estrela2 executado pelo veículo usando o navegador evoluido em ambiente tipo Estrela.

4.2.0.3 3a. Experiência

Passou-se a utilizar um ambiente desprovido de obstáculos, com exceção das paredes limítrofes (veja figura 4.21). Os resultados foram significativos. No ambiente no qual foi desenvolvido, o veículo evoluído após 30 gerações foi capaz de achar todos os alvos sem colidir. Entretanto, inserido em ambientes mais complexos, observou-se a colisão quase imediata (figuras 4.24 a 4.27). Ocorreu o desenvolvimento apenas do comportamento de buscar alvos. A rede aprendeu a reconhecer apenas os obstáculos limítrofes. Este resultado é esperado pois não se forneceu informação alguma sobre a existência de outro tipo de obstáculo.

Foi constatado que tanto o excesso de informação quanto a falta desta são detrimentais ao desenvolvimento de uma rede satisfatória. No primeiro caso a grande quantidade de obstáculos impediu que o navegador chegasse ao alvo. Portanto, a parcela da função de fitness correspondente ao alvo não exerceu efeito algum sobre a evolução do navegador. No segundo caso, permitiu-se a contribuição desta parcela. Entretanto, a falta de obstáculos internos levou à não-aprendizagem do conceito de obstáculos.

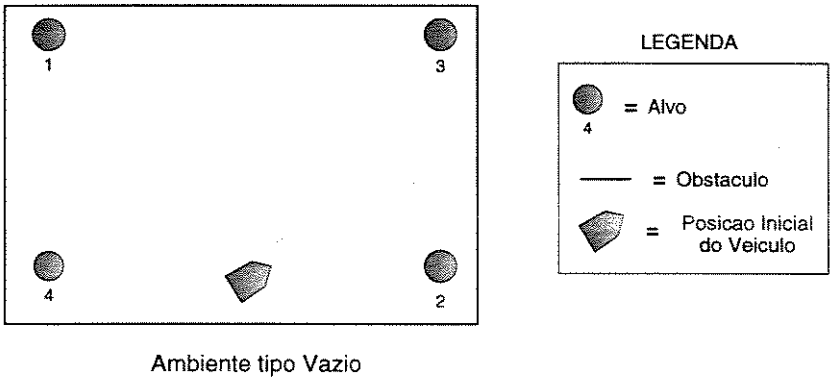


Figura 4.21: Ambiente tipo Vazio utilizado para a avaliação da população de redes neuronebulosas durante a evolução.

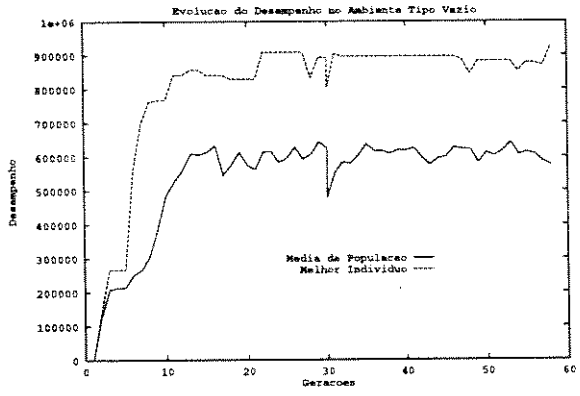


Figura 4.22: Gráfico da evolução do desempenho dos controladores ao longo das gerações na terceira simulação.

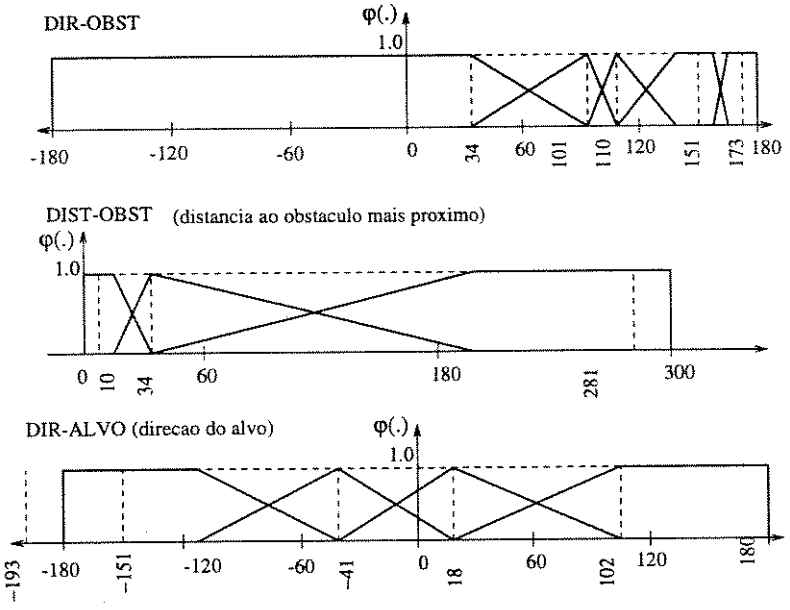


Figura 4.23: Funções de pertinência evoluidos na terceira simulação.

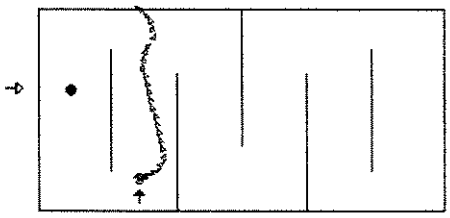


Figura 4.24: Percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador evoluido em ambiente tipo Vazio.

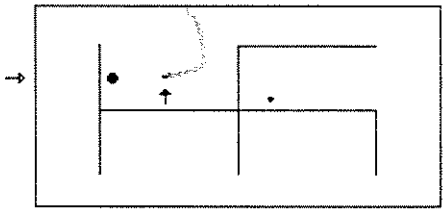


Figura 4.25: Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador evoluido no ambiente tipo Vazio.

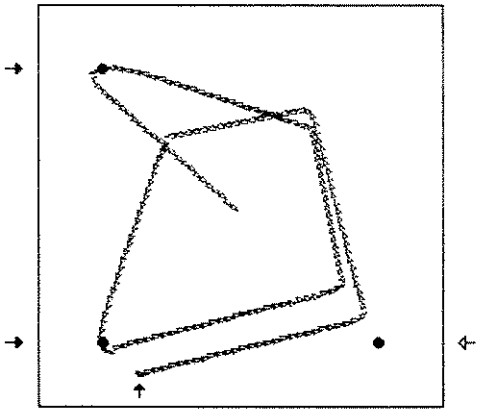


Figura 4.26: Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido em ambiente tipo Vazio.

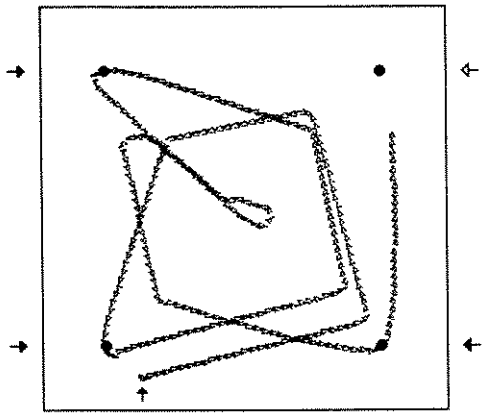


Figura 4.27: Novo percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido em ambiente tipo Vazio.

4.2.0.4 4a. Experiência

Decidiu-se utilizar a população obtida na terceira experiência como base para uma nova evolução no ambiente tipo Estrela. Esperava-se obter uma rede que mantivesse o comportamento de buscar alvos enquanto desenvolvesse o comportamento de evitar obstáculos. Após 160 gerações (contando com os da evolução anterior), obteve-se uma rede neural cujo comportamento pode ser visto nas figuras 4.30 a 4.35. O comportamento de buscar alvos foi suplantado pelo de evitar obstáculos.

A oscilação entre comportamentos suscitados pela rápida especialização evolutiva nos ambientes propícios a cada tipo de comportamento determinou dois possíveis caminhos:

- utilizar um conjunto da ambientes na avaliação da população a cada geração, buscando evitar a super-especialização;
- separar a evolução dos dois comportamentos, reunindo os controladores obtidos posteriormente em um controlador composto.

A primeira opção visa criar uma avaliação com diversidade o suficiente para

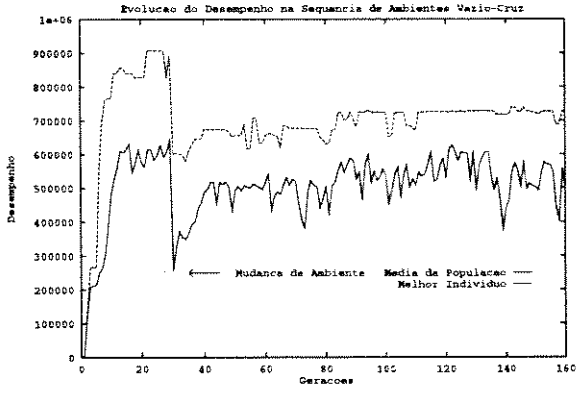


Figura 4.28: Gráfico da evolução do desempenho dos controladores ao longo das gerações na quarta simulação.

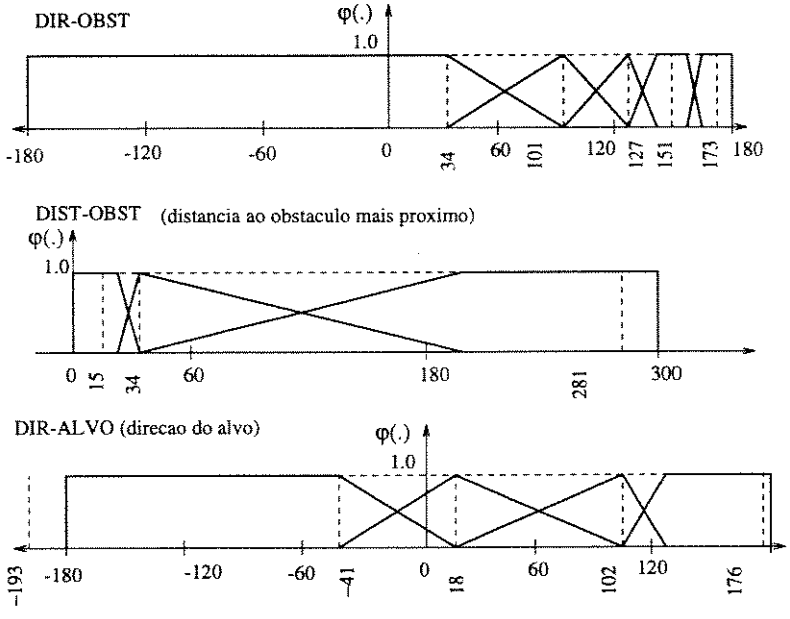


Figura 4.29: Funções de pertinência evoluidos na quarta simulação.

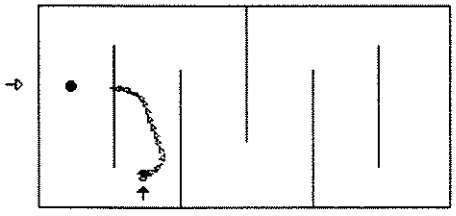


Figura 4.30: Percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador evoluido na sequencia de ambientes Vazio-Estrela.

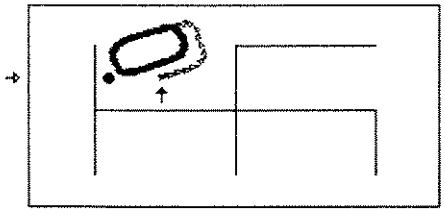


Figura 4.31: Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador evoluido na sequencia de ambientes Vazio-Estrela.

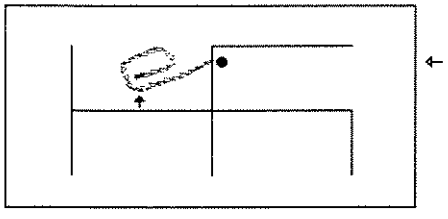


Figura 4.32: Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador evoluido na sequencia de ambientes Vazio-Estrela.

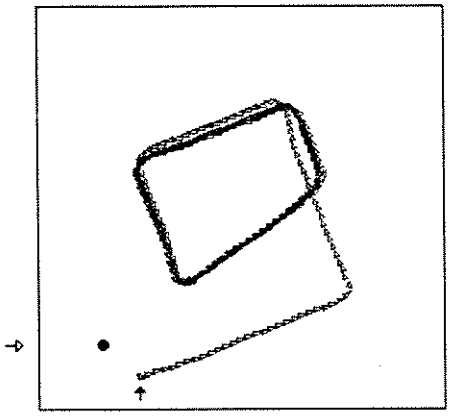


Figura 4.33: Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido na sequencia de ambientes Vazio-Estrela.

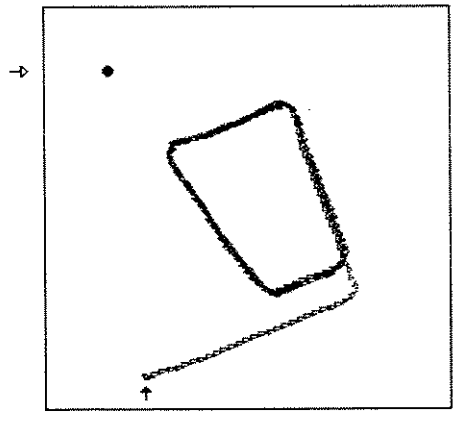


Figura 4.34: Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido na sequencia de ambientes Vazio-Estrela.

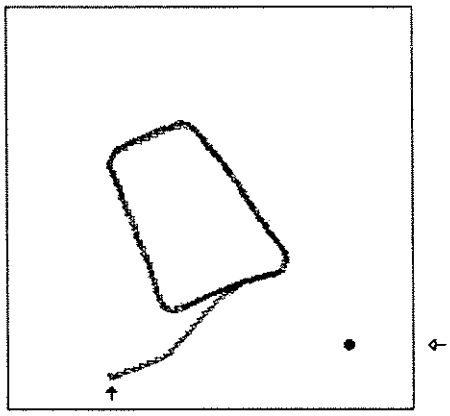


Figura 4.35: Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador evoluido na sequencia de ambientes Vazio-Estrela.

estimular o surgimento espontâneo dos múltiplos comportamentos necessários para obter um navegar satisfatório. A função de desempenho seria uma média dos desempenhos obtidos em cada ambiente.

Descartou-se esta alternativa pelos seguintes motivos:

- possível especialização da rede nos comportamentos associados aos comportamentos iniciais ou finais, em um processo semelhante à especialização das redes neurais;
- longo tempo de execução decorrente da multiplicidade de ambientes. Como medida de comparação, a execução no ambiente labirinto consumiu aproximadamente 6 dias para evoluir 109 gerações (em um IBM RS6K compartilhado). O extenso período de simulação inviabilizaria o ciclo de simulação, avaliação, modificação e nova simulação.

A segunda alternativa evita os problemas associados aos múltiplos ambientes e simplifica a função objetivo. A simulação independente também incorre na aceleração do processo evolutivo.

4.2.0.5 5a. Experiência

Duas evoluções independentes foram implementadas. Uma desenvolveria uma rede capaz de buscar alvos e a outra uma rede capaz de evitar obstáculos.

O algoritmo genético usado na evolução do rastreador de alvos sofreu várias alterações. A função objetivo agora é formada apenas pela parcela referente aos alvos (eq. 4.2). A rede neuronebulosa foi simplificada, sendo estimulada apenas pela variável de entrada indicando a direção do alvo. Assim, no máximo cinco regras são aprendidas pela rede. Utilizou-se o ambiente tipo Vazio.

$$f = 10000 n_{\text{alvos-alcancados}} \tag{4.2}$$

Modificações semelhantes foram feitas no algoritmo genético do navegador que evita obstáculos. A variável relacionada com o alvo foi descartada, restando apenas *direção* e *distância ao obstáculo mais próximo*. Foi utilizada o ambiente tipo Labirinto.

Estas modificações levaram ao desenvolvimento de 2838 e 190 gerações de redes neuronebulosos aplicados à busca de alvos e à não-colisão, respectivamente, no mesmo espaço de tempo em que se desenvolveu 30 gerações da rede completa no ambiente tipo Labirinto. Os gráficos da evolução de desempenho durante o desenvolvimento do rastreador de alvos e do navegador que evita obstáculos estão nas figuras 4.36 e 4.37.

A especialização destas redes é claramente evidenciada pelo fraco desempenho das mesmas em ambientes mais complexos (figuras 4.40-4.45 e 4.46-4.50, respectivamente).

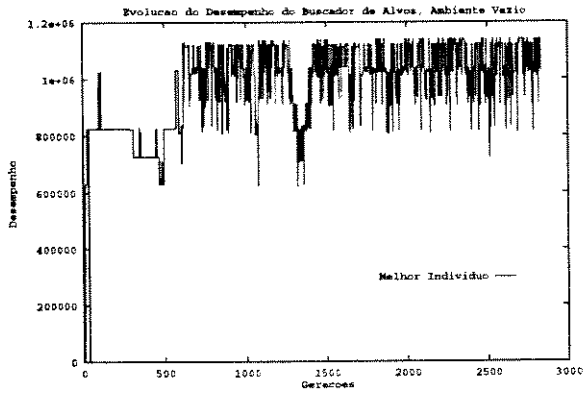


Figura 4.36: Evolução do desempenho do rastreador de alvos.

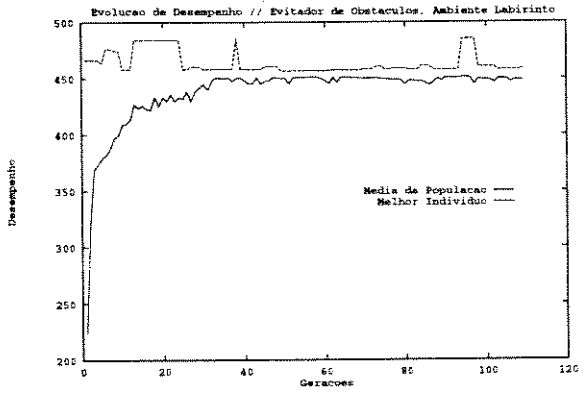


Figura 4.37: Evolução do desempenho do navegador que evita obstáculos.

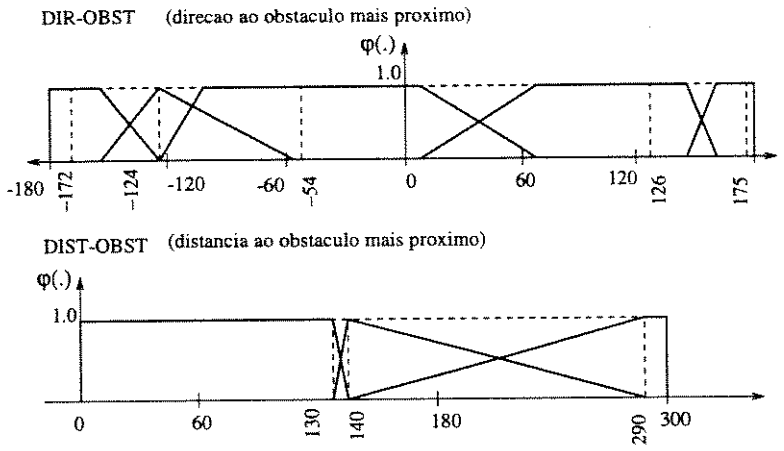


Figura 4.38: Funções de pertinência obtidas pela evolução do navegador que evita obstáculos.

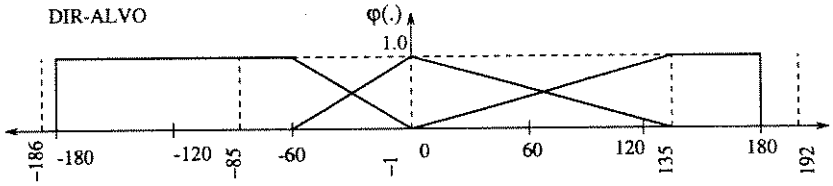


Figura 4.39: Funções de pertinência obtidas pela evolução do rastreador de alvos.

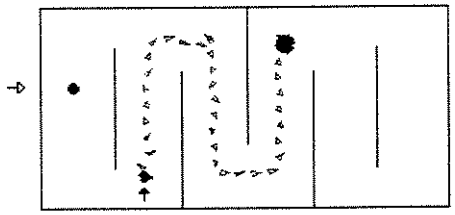


Figura 4.40: Percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador que evita obstáculos gerado pelo algoritmo genético. Observe a predominância do comportamento de evitar obstáculos.

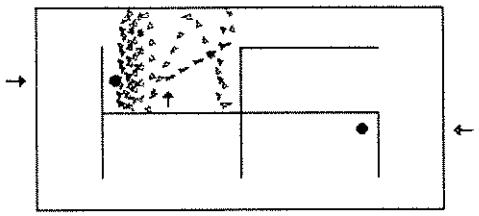


Figura 4.41: Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador que evita obstáculos gerado pelo algoritmo genético.

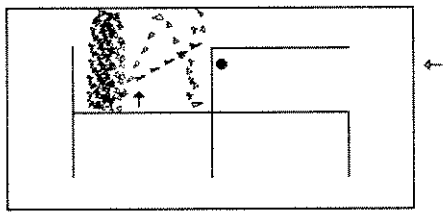


Figura 4.42: Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador que evita obstáculos gerado pelo algoritmo genético a procura de outro alvo.

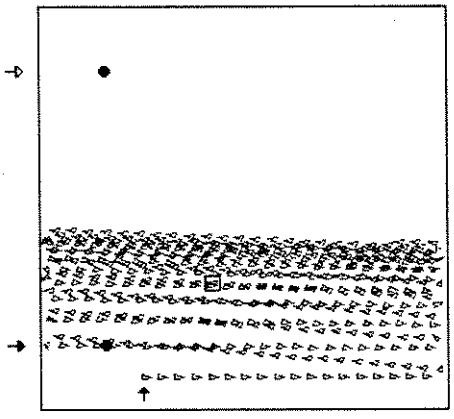


Figura 4.43: Percurso executado no ambiente tipo Vazio pelo veículo usando o navegador que evita obstáculos gerado pelo algoritmo genético a procura do primeiro alvo (canto superior esquerdo). Observe a predominância do comportamento de evitar obstáculos.

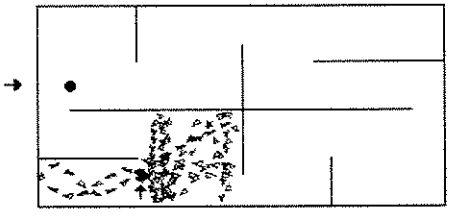


Figura 4.44: Percurso executado no ambiente tipo Estrela2 pelo veículo usando o navegador que evita obstáculos gerado pelo algoritmo genético a procura do primeiro alvo.

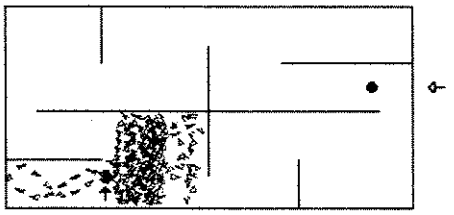


Figura 4.45: Percurso executado no ambiente tipo Estrela2 pelo veículo usando o navegador que evita obstáculos gerado pelo algoritmo genético a procura de outro alvo (canto superior direito).

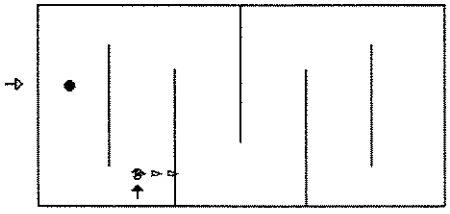


Figura 4.46: Percurso executado no ambiente tipo Labirinto pelo veículo usando o navegador "rastreador de alvos" gerado pelo algoritmo genético a procura do primeiro alvo (lateral esquerda). Observe a colisão decorrente da especialização.

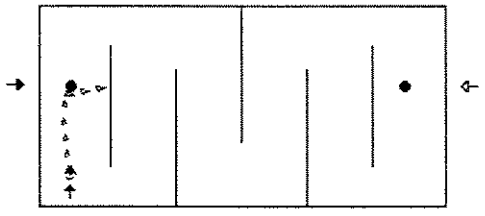


Figura 4.47: Percurso executado pelo veículo usando o navegador "rastreador de alvos" gerado pelo algoritmo genético a procura do segundo alvo (canto superior direito) no ambiente tipo Labirinto. Ocorre nova colisão.

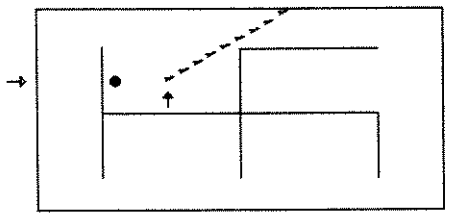


Figura 4.48: Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador "rastreador de alvos" gerado pelo algoritmo genético a procura do primeiro alvo (quadrante esquerdo superior).

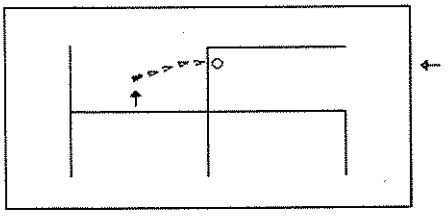


Figura 4.49: Percurso executado no ambiente tipo Estrela pelo veículo usando o navegador "rastreador de alvos" gerado pelo algoritmo genético a procura de outro alvo (canto superior direito).

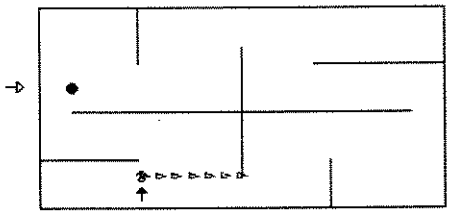


Figura 4.50: Percurso executado no ambiente tipo Estrela2 pelo veículo usando o navegador "rastreador de alvos" gerado pelo algoritmo genético.

O controlador composto foi obtido agregando as saídas das duas redes neuro-nebulosas (veja figura 4.51) através de um neurônio motor. Inicialmente, utilizou-se os pesos 0.38 e 0.62 para as sinapses ligando o neurônio motor à rede rastreadora de alvos e à rede evitador de obsta'culos, respectivamente(eq. 4.3). Depois, observou-se que o comportamento de evitar obstáculos se torna mais relevante quando o agente se encontra nas proximidades de um obstáculo. Desta forma, introduziu-se uma fusão das respostas de cada rede neuronebulosa é dependente da distância entre o robô e obstáculo mais próximo: quanto mais próximo do obstáculo, maior a contribuição da rede responsável por evitar colisões ao resultado final (eq. 4.4).

$$\theta_{mov} = 0.62\theta_{c,obst}(1 - d_{obst}) + 0.38\theta_{c,alvo} \tag{4.3}$$

$$\theta_{mov} = \theta_{c,obst}\left(1 - \frac{1}{1 + e^{(6.2-d_{obst})}}\right) + \theta_{c,alvo}\left(\frac{1}{1 + e^{(6.2-d_{obst})}}\right) \tag{4.4}$$

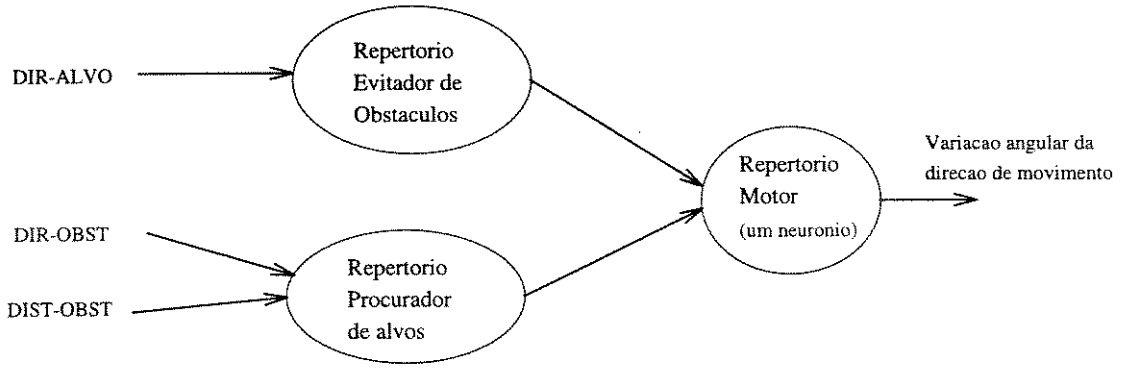


Figura 4.51: Arquitetura do navegador proposto reformulado como um sistema comportamental tipo Neuronal Group Selection.

O controlador resultante apresentou um resultado satisfatório (veja figuras 4.52 a 4.55) e comparável ao controlador neuronebuloso padrão em diversos ambientes. A exceção ocorre no ambiente tipo Estrela, no qual o veículo é incapaz de escapar de obstáculos em formato de U. O controlador neuronebuloso padrão foi projetado heuristicamente e

serve como padrão de comparação para as redes desenvolvidas pelo método proposto. Este controlador também não consegue sair de obstáculos em U.

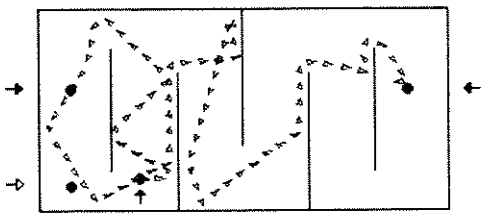


Figura 4.52: Percurso no ambiente tipo Labirinto executado pelo veículo usando o navegador composto.

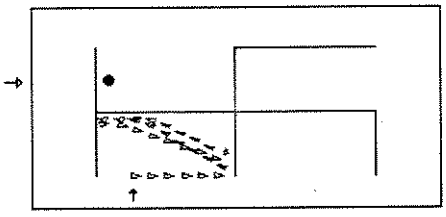


Figura 4.53: Percurso no ambiente tipo Estrela executado pelo veículo usando o navegador composto.

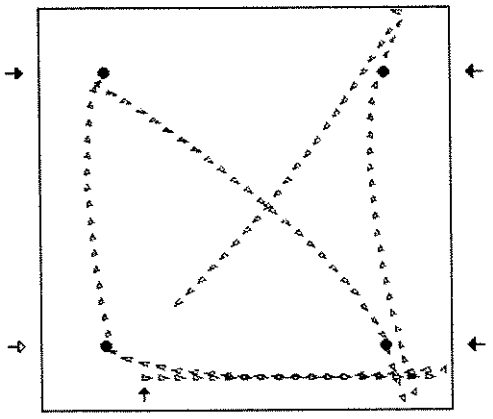


Figura 4.54: Percurso no ambiente tipo Vazio executado pelo veículo usando o navegador composto.

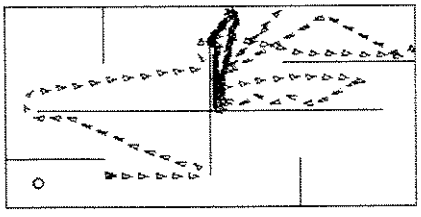


Figura 4.55: Percurso no ambiente tipo Estrela2 executado pelo veículo usando o navegador composto.

4.3 Resumo

Os resultados dos experimentos realizados indicam claramente a complexidade do problema de navegação autônoma. O navegador proposto não foi capaz de aprender simultâneamente a evitar obstáculos e achar alvos. Entretanto a evolução separada destes dois comportamentos foi bem-sucedida, indicando que o algoritmo de aprendizagem pode ser aplicada com êxito a esta espécie de problema.

A união dos dois controladores evoluídos em separado resultou em um navegador flexível, capaz de operar em diversos ambientes, falhando apenas em ambientes tipo " U". Neste caso o veículo se encontra limitado por três obstáculos, estando o alvo atrás do obstáculo central. A falha em ambientes com obstáculo em " U" não deve ser atribuído ao método de controle, mas sim ao conjunto de sensores utilizado. O controlador de Li [34], o qual possui apenas alguns sensores a mais em relação ao navegador proposto, é capaz de contornar tais situações.

Foi notada também a influência do ambiente de treinamento sobre a evolução do comportamento exibido pelo navegador. Um ambiente com poucos obstáculos leva a formação de navegadores capazes de achar alvos, porém um ambiente com muitos obstáculos nem sempre gera navegadores capazes de evitar obstáculos. Verificou-se também que o uso de uma população evoluída para uma tarefa como a população inicial para a evolução de outro comportamento leva a substituição de comportamentos e não à fusão dos mesmos. Assim, propõe-se o uso de dois ou mais ambientes de simulação propícios a formação de cada comportamento desejado na composição da função de desempenho da população. O desempenho de um indivíduo será então uma média dos seus desempenhos em cada ambiente.

A função de desempenho também poderá ser alterada para fornecer mais informações (*hints*, dicas) sobre o comportamento desejado, como por exemplo uma parcela negativa penalizando a aproximação a obstáculos. Entretanto, esta alternativa pode limitar em demasia a gama de comportamentos possíveis de serem evoluídos e portanto a diversidade de ambientes de operação. A parcela acima mencionada, por exemplo, não permitirá a navegação em ambientes com corredores estreitos, por exemplo.

Todas as propostas acima, bem como a divisão de comportamentos realizada

nas experiências, representam o acréscimo de informação sobre o problema no sistema.

A questão fundamental é, portanto, quanta informação *a priori* deseja-se embutir no navegador. Quanto mais informação embutida no sistema, melhor será o desempenho mas também menos flexível ele será, pois toda e qualquer informação representa uma suposição sobre o comportamento desejado e portanto uma limitação deste.

Deste modo, dado a configuração adotada para o veículo, os resultados das simulações comprovam a eficácia do método de navegação autônoma proposto.

Capítulo 5

Conclusão

Foi proposto um método de navegação autônoma baseado em paradigmas de computação flexível: redes neurais artificiais, sistemas nebulosos e algoritmos genéticos. A simbiose destes paradigmas resultou em um sistema com várias propriedades interessantes, as quais possibilitam ao sistema atender as duas exigências básicas típicas do controle autônomo: localidade de operação e referência local e aquisição dinâmica de conhecimento. As propriedades relevantes dos sistema desenvolvido são:

- capacidade de aprendizagem não supervisionada, oferecida pelo algoritmo genético;
- extração fácil de conhecimento adquirido, devido ao mapeamento entre a estrutura e pesos da rede neuronebulosa e a base de regras nebulosas equivalente;
- uso de informações locais na sua operação.

O navegador desenvolvido com o método proposto foi treinado e avaliado em diversos ambientes simulados, contendo disposições diferentes de obstáculos e alvos. As limitações apresentadas pelos navegadores desenvolvidos podem ser atribuídas à quantidade utilizada de conhecimento *a priori* sobre o problema de controle. No caso da navegação autônoma, este conhecimento está embutido na escolha dos sensores e atuadores empregados no veículo a ser controlado [41].

Nos testes, decidiu-se utilizar sensores os mais simples e em menor número possível e, que permita a navegação. A factibilidade do conjunto de sensores escolhido foi comprovada pelo desenvolvimento heurístico de um navegador.

Os resultados das simulações revelam que o método proposto é capaz de desenvolver navegadores tão bons quanto o navegador heurístico utilizado como comparação. Uma vez que as limitações apresentadas pelo método também ocorrem no navegador de comparação, conclui-se que as limitações são devidas ao conjunto de sensores escolhidos e não devido a deficiências do método de navegação autônoma proposto.

O desenvolvimento futuro desta pesquisa poderá seguir dois caminhos distintos: continuar a avaliação da forma atual do método proposto, utilizando mais conhecimento *a priori* e refinando os múltiplos parâmetros do método e/ou desenvolver mais o método, agregando de modo simbiótico, novos paradigmas.

Na primeira linha de atuação, propõe-se:

- ajustar a função de desempenho do mecanismo de aprendizagem genético para melhor descrever os comportamentos desejados.
- aumentar as populações utilizadas, fornecendo mais informações para serem exploradas pelo mecanismo de aprendizagem.
- utilizar funções de desempenho baseados na avaliação contínua no tempo do desempenho da população, evitando o uso de informações globais da população, como a média de desempenho utilizada no operador de seleção.
- empregar mais sensores, como um sensor de distâncias em múltiplas direções, para prover mais informações sobre o ambiente e suscitar a emergência de comportamentos mais flexíveis, como o de poder evitar obstáculos na forma de " U".

Já visando a evolução do próprio método, sugere-se a utilização de um conjunto maior de sensores acoplados a redes neurais auto-organizadas (como as de Kohonen) responsáveis pela identificação de padrões de estímulos relevantes. Assim mapeia-se de modo mais eficaz as ações relevantes a cada situação de operação. Esta proposta visa o desenvolvimento de um mecanismo de adaptação ao ambiente após a aprendizagem genética.

BIBLIOGRAFIA

[1] N. Almássy and E. Vinkhuyzen. Evolution of adaptive behaviour in dynamic environments. In *Proceedings of the Fifth International Symposium on Robotics and Manufacturing (ISRAM'94)*, Albuquerque, NM, August 1994. TSI Press.

[2] J. Antonisse. A new interpretation of schema notation that overturns the binary encoding constraint. In *Proceedings of the ICGA'89*, pages 86–91, 1989.

[3] A. Brindle and J. Sampson. Genetic algorithms as adaptive search mechanisms for function optimization. Unpublished manuscript, University of Alberta, Dept. of Comp.Science, Edmonton., 1981.

[4] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[5] R. Brooks. *The Behaviour Language; User's Guide.* MIT AI Lab, MIT, 1990. Memo 1127.

[6] R. Brooks. Intelligence without reason. In *Proceedings of the IJCAI-91*, pages 569–595. Morgan Kaufmann, San Mateo CA, USA, 1991.

[7] Gerald M. Edelman. *Neural Darwinism: A Theory of Neuronal Group Selection.* Basic Books, New York, 1987.

[8] G.M. Edelman, Jr. G.N. Reeke, and O. Sporns. Synthetic neural modelling: The 'darwin' series of recognition automata. *Proceedings of the IEEE*, 78(9):1498–1530, 1990.

[9] G.M. Edelman, G.N. Reeke Jr., W.E. Gall, G. Tononi, D. Williams, and O. Sporns. Synthetic neural modeling applied to a real-world artifact. In *Proc. Natl. Acad. Sci. USA*, pages 7267–7271. Natl. Acad. Sci. USA, August 1992. Vol.89, Neurobiology.

[10] M. Figueiredo. Modelling neurobiological interaction using fuzzy operators. not published, July 1995.

[11] M. Figueiredo, F.Gomide, and W.Pedrycz. A fuzzy neural network: Structure and learning. In *Proc. of 5th IFSA World Congress*, Seoul, Korea, 1994. IFSA.

[12] D.B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, January 1994.

[13] E. Gat, R. Desai, R. Ivlev, J. Loch, and D. Miller. Behaviour control for robotic exploration of planetary surfaces. *IEEE Transactions on Robotics and Automation*, 10(4):490–503, August 1994.

[14] David E. Goldberg. *Genetical Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[15] F. Gomide and A. Rocha. Neurofuzzy components based on thresholds. In *Proc. of IFAC'92.* IFAC, 1992. Sisica, Spain.

[16] D.O. Hebb. *The Organization of Behaviour.* Wiley, New York, 1949.

[17] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Nat. Bureau Standards J. Res.*, (49):409–436, 1952.

[18] J.H. Holland. *Adaptation in Natural and Artificial Systems.* PhD thesis, University of Michigan, Ann Arbor, 1975.

[19] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. National Academy of Sciences*, (79):2554–2558, 1982.

[20] K Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[21] Don R. Hush and Bill G. Horne. Progress in supervised neural networks. *IEEE Signal Processing Magazine*, pages 8–39, January 1993.

[22] Y. Ichikawa and T. Sawa. Neural network application for direct feedback controllers. *IEEE Transactions on Neural Networks*, 3(2):224–231, March 1992.

[23] Abraham Kandel. *Fuzzy Mathematical Techniques with Applications.* Addison-Wesley, Reading, MA, USA, 1986.

[24] A. Kaufmann. *Introduction to the Theory of Fuzzy Subsets.* Academic Press, New York, 1975.

[25] T. Kohonen, M. Makisara, and T. Saramaki. Phonotonic maps-insightful representations of phonological feature for speech recognition. In *Proc. 7th IEEE Internat'l Conf. on Patt. Recognition*, Piscataway NJ, USA, August 1984. IEEE, IEEE.

[26] T. Kohonen, E. Reuhkala, K. Maekisara, and L. Vainio. Associative recall of images. *Biological Cybernetics*, 22:159–168, 1976.

[27] Tuevo Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, 1984.

[28] Tuevo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78:1464–1480, 1990.

[29] Bart Kosko. *Neural Networks and Fuzzy Systems.* Prentice-Hall, 1992.

[30] J. Koza. *Genetic Programming: on the Programming of Computers by Means of Natural Selection.* ISBN 0-262-11170-5. MIT Press, 2nd edition, 1992.

[31] Ben J.A. Kroese and P. Patrick van der Smagt. An introduction to neural networks. Internet ftp file, Universitu of Amsterdam, Dept. Of Computer Science, Holland, January 1993.

[32] C.C. Lee. Fuzzy logic in control systems: Fuzzy logic controller, parts i and ii. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):404–435, March/April 1990.

[33] D. Leitch and P. Probert. Genetic algorithms for the development of fuzzy controllers for autonomous guided vehicles. In *Proceedings of 2nd European Conf. on Intelligent Techniques and Soft Computing*, pages 211–216, 1994. Aachen, BRD.

[34] W. Li. Fuzzy logic-based 'perception-action' behaviour control of a mobile robot in uncertain environments. In *Proceedings of Third IEEE International Conference on Fuzzy Systems*, pages 1626–1631, Piscataway NJ, USA, July 1994. IEEE, IEEE. Volume 3.

[35] M.J. Mataric. Behaviour-based control: Main properties and implications. 1994.

[36] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–113, 1943.

[37] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* The MIT Press, 1969.

[38] M. Mizumoto. Note on the arithmetic rule by zadeh for fuzzy conditional inference. *Cybern.Syst.*, 12:247–306, 1981.

[39] M. Oliveira, M. Figueiredo, F. Gomide, and L. Romero. Autonomous control using soft computing paradigms. In *Anais do Primeiro CBRN*, Itajuba, MG, Brasil, October 1994. EFEI.

[40] M. Oliveira, M. Figueiredo, F. Gomide, and L. Romero. Neurofuzzy navigation and neuronal group selection. In *Proc. of the Sixth IFSA World Congress*, Sao Paulo, SP, Brasil, July 1995. IFSA.

[41] Rolf Pfeifer. Informal talk on autonomous agents. Personal Communication, June 1995.

[42] C.W. Reynolds. The difficulty of roving eyes. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 262–267, Piscataway NJ, USA, July 1994. IEEE. Volume 1.

[43] A.F. Rocha. *Neural Nets.* Springer-Verlag, 1992.

[44] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Pyschological Review*, 65:386–408, 1958.

[45] D.E. Rumelhart and J.L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition (vols. 1 and 2).* The MIT Press, 1986.

[46] D.E. Rumelhart and D. Zipser. Feature discovery by competitive learning. *Cognitive Science*, (9):75–112, 1985.

[47] T.J. Sejnowski and C.R. Rosenberg. Nettalk: A parallel network that learns to read aloud. Technical Report JHU/EECS-86/01, The John Hpkins University Electrical Engineering and Computer Science Department, 1986.

[48] S.L.Hung and H.Adeli. A parallel genetic/neural network learning algorithm for mimd shared memory machines. *IEEE Transactions on Neural Networks*, 5(6):900–909, November 1994.

[49] P.F.M.J. Verschure, B.J.A. Kröse, and Rolf Pfeifer. *Robotics and Autonomous Systems*, volume 9, chapter Distributed Adaptive Control: The Self-organization of Structured Behaviour, pages 181–196. Elsevier Science Publishers B.V., 1992.

[50] Paul Wallich. Silicon babies. *Scientific American*, pages 90–91, December 1991. Douglas B. Lenat.

[51] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.