ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ
Programa de Pós - Graduação em Engenharia Elétrica

AJUSTE DE FUNÇÕES DE PERTINÊNCIA UTILIZANDO ALGORITMOS GENÉTICOS

Marcos Alberto de Carvalho

ITAJUBÁ
2000

ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ
Programa de Pós-Graduação em Engenharia Elétrica

AJUSTE DE FUNÇÕES DE PERTINÊNCIA UTILIZANDO ALGORITMOS GENÉTICOS

Dissertação apresentada à Escola Federal de Engenharia de Itajubá, para obtenção do título de Mestre em Engenharia Elétrica.

Marcos Alberto de Carvalho

Itajubá

2000

Ministério da Educação
ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ

ANEXO I

PRONUNCIAMENTO DA BANCA EXAMINADORA

A Banca Examinadora, abaixo assinada, nomeada pela Portaria 131 de 15 de junho de 2000, considerando o resultado do Julgamento da Prova de Defesa Pública da Dissertação de Mestrado intitulada: "**Ajuste de funções de pertinência utilizando algoritmos genéticos**", apresenta pronunciamento no sentido de que o Coordenador dos Cursos de Pós-Graduação em Engenharia Elétrica da Escola Federal de Engenharia de Itajubá solicite ao DRA (Departamento de Registro Acadêmico) a expedição do título de Mestre em Ciências em Engenharia Elétrica, na área de Automação em Sistemas Elétricos Indústrias, satisfeitas as demais exigências regimentais, à Marcos Alberto de Carvalho.

Itajubá, 15 de junho de 2000

Prof. Wilton Ney do Amaral Pereira
1° Examinador - UNITAU

Prof. Lúcia Regina Horta Rodrigues Franco
2° Examinadora - EFEI

Prof. Luiz Eduardo Borges da Silva
3° Examinador - EFEI (Co-orientador)

Prof. Germano Lambert Torres
4° Examinador EFEI (orientador)

Ministério da Educação

ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ

ANEXO II

FOLHA DE JULGAMENTO DA BANCA EXAMINADORA

Título da Dissertação: " Ajuste de funções de pertinência utilizando algoritmos genéticos".

Autor : Marcos Alberto de Carvalho

JULGAMENTO

Examinadores	Conceito	Rubrica
1º	A +	
2º	A +	
3º	A +	
4º	A +	

Resultado Médio: Conceito A + , ou seja, Aprovado com distinção

Observações: _____

Itajubá, 15 de junho 2000

Prof. Wilton Ney do Amaral Pereira
1º Examinador- UNITAU

Prof. Lúcia Regina Horta Rodrigues Franco
2º Examinador- EFEI

Prof. Luiz Eduardo Borges da Silva
3º Examinador - EFEI (Co-orientador)

Prof. Germano Lambert Torres
4º Examinador - (orientador) EFEI

Pró-Diretoria de Pesquisa e Pós-Graduação - EFEI - Escola Federal de Engenharia de Itajubá
Av BPS, 1303 - Cx Post 50 - 37500-000 - ITAJUBÁ/MG - BRASIL - Tel.: (035)629-1118 - 629-1121;
Fax (035)629-1120 - International Dialing: Please replace (035) by (+5535)

Marcos Alberto de Carvalho

AJUSTE DE FUNÇÕES DE PERTINÊNCIA UTILIZANDO ALGORITMOS GENÉTICOS

Dissertação apresentada à Escola Federal de Engenharia de Itajubá, para obtenção do título de Mestre em Engenharia Elétrica.

Área de concentração:
Automação e Sistemas Elétricos Industriais

Orientador:
Germano Lambert Torres

Co-orientador:
Luiz Eduardo Borges da Silva

Itajubá

Junho/2000

A minha esposa Jaqueline pelo incentivo, amor e carinho

Agradecimentos

Ao professor Germano Lambert Torres pela orientação, dedicação e amizade que resultaram neste trabalho.

À todos, que direta ou indiretamente, contribuíram para a realização desse trabalho.

À Deus que nos deu a vida, inteligência, e tudo mais que temos de graça, para que possamos ser felizes.

Sumário

Resumo

O uso da Lógica Difusa para resolver problemas de controle aumentou consideravelmente durante os últimos anos. Com isto, o ensino de Controle Difuso em cursos de engenharia está se tornando uma necessidade. Assim, anteriormente, foi desenvolvido um pacote computacional para o auto-treinamento de estudantes em teoria de controle difuso. O pacote contém todas as instruções exigidas para os usuários compreenderem os princípios do controle difuso. As instruções de treinamento são apresentadas através de um exemplo prático.

Embora esta abordagem provou ser conveniente dando aos estudantes uma oportunidade para compreender uma situação da vida real, ela sofre um séria desvantagem: o tipo de aprendizagem. Os estudantes utilizam sempre o método da "tentativa-e-erro" para conseguir uma ação de controle apropriada, como a definição de regras ou ajuste das funções de pertinência. O problema deste método de aprendizagem é que os estudantes podem adquirir o conceito errôneo de que correções no sistema de controle difuso seja uma questão de suposição. O propósito deste trabalho é apresentar uma estratégia para o ajuste automático das funções de pertinência usando algoritmos genéticos.

Para tal, foi desenvolvido e implementado um algoritmo que transforma os atributos fundamentais as funções de pertinências difusas em cromossomos que são submetidos a uma evolução, cruzamento e mutação. A idéia geral é obter uma nova família de funções de pertinência que possa melhor controlar um processo.

O algoritmo proposto foi incorporado ao pacote computacional anteriormente desenvolvido.

Abstract

The use of Fuzzy Logic for solving control problems has tremendously increased over the last few years. Thus, the teaching of fuzzy control in engineering courses is becoming a necessity. In an existent work, it has been used a computational package for students' self-training on fuzzy control theory. The package contains all required instructions for the users to gain the understanding of fuzzy control principles. The training instructions are presented via a practical example.

Although this approach has proven to be convenient in giving to students an opportunity to appreciate real life like situations, it suffers a serious disadvantage: the type of learning. In fact, students often go through a "trial-and-error" method to select an appropriate control action, such as rule definitions or membership fitting. The problem of this type of learning is a tendency from students to get the erroneous concept that corrective actions are much a matter of guess. The purpose of this paper is to present a strategy for an automatic membership function fitting using genetic algorithms.

For that, an algorithm has been developed and implemented to transform the main attributes of fuzzy membership function in chromosomes, which are submitted to an evolution, crossover and mutation. The main idea is to fit a new family of membership functions, which can establish a better system control.

The proposed algorithm has been incorporated to the computational package above.

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A Teoria dos Conjuntos Difusos foi proposta por Lofti A. Zadeh em um artigo publicado em 1965 [1]. Zadeh observou que os recursos tecnológicos disponíveis até então, não eram capazes de automatizar as atividades relacionadas a problemas de natureza industrial, biológica ou química que utilizam dados inerentemente analógicos impróprios para serem manipulados em um computador digital que trabalha com dados numéricos bem definidos, ou seja, valores discretos.

Recentemente a Lógica Difusa tem sido utilizada no controle de processos industriais, equipamentos eletrônicos de entretenimento, carros, sistemas de diagnose e, até mesmo, para o controle de eletrodomésticos. Então, em função da grande aplicação, faz-se necessário o estudo da Lógica Difusa nos cursos de engenharia.

Neste trabalho é utilizado o Pacote Computacional para Ensino da Lógica Difusa [20]. Este programa computacional foi desenvolvido para o auto-treinamento de estudantes de engenharia na teoria de controle difuso. O programa contém todas instruções necessárias para os usuários compreenderem os princípios do controle difuso.

O principal objetivo do pacote é estacionar um veículo em uma garagem, partindo de qualquer posição inicial. O usuário deve inicialmente desenvolver um conjunto de regras de controle difuso e funções de pertinência que definirão a trajetória do veículo. Os processos de fuzzificação e de defuzzificação das variáveis são realizados pelo programa sem a interferência do usuário.

Com este pacote os estudantes dispõem de um recurso que exemplifica uma situação muito conhecida da vida real. Entretanto, este pacote, apresenta uma séria desvantagem: o tipo de aprendizagem. Neste caso, os estudantes, para conseguir uma

ação de controle apropriada, utilizam o método da "tentativa-e-erro" na definição das regras e parâmetros para cada uma das funções de pertinência. O problema deste método de aprendizagem é que os estudantes podem adquirir o conceito errôneo de que correções no sistema de controle difuso seja uma questão de suposição.

O objetivo deste trabalho é apresentar uma estratégia para o ajuste automático das funções de pertinência usando algoritmos genéticos.

Algoritmos Genéticos são algoritmos de otimização global, baseados nos mecanismos de seleção natural e da genética. Eles empregam uma estratégia de busca paralela e estruturada, mas aleatória, que é voltada em direção ao reforço da busca de pontos de "alta aptidão", ou seja, pontos nos quais a função a ser minimizada tem valores relativamente baixos.

Apesar de aleatórios, eles não são caminhadas aleatórias não direcionadas, pois exploram informações históricas para encontrar novos pontos de busca onde são esperados melhores desempenhos. Isto é feito através de processos iterativos, onde cada iteração é chamada de geração. A cada geração temos uma população de indivíduos (possíveis soluções para o problema) que trocam informações entre si e, ocasionalmente, sofrem mutações.

Neste trabalho então desenvolveu-se um módulo de treinamento genético para um controle previamente criado. Os algoritmos genéticos são utilizados para achar os melhores parâmetros para as funções pertinência. A concatenação destes parâmetros constituem os indivíduos da população (cromossomos) que são avaliados de acordo com o número de iterações necessárias para o veículo estacionar de acordo com o ajuste feito por cada indivíduo.

Este trabalho está organizado da seguinte maneira. O Capitulo 2 apresenta os fundamentos teóricos de Sistemas Difusos, Algoritmos Genéticos e a Integração de

Sistemas Difusos com Algoritmos Genéticos , ressaltando principalmente os aspectos que são utilizados neste trabalho.

O Capítulo 3 apresenta o problema a ser resolvido e as limitações impostas para o sistema de controle.

O Capítulo 4 descreve o módulo de treinamento genético apresentando o algoritmo passo a passo em detalhes bem como as telas de interface com o usuário.

O Capítulo 5 mostra testes realizados com controles difusos que tiveram suas funções de pertinência ajustadas através dos algoritmos genéticos.

O Capítulo 6 apresenta as conclusões e de desenvolvimentos futuros que poderiam orientar novas pesquisas que possam dar continuidade aos resultados obtidos neste trabalho.

Capítulo 2

Sistemas Difusos, Algoritmos Genéticos e Integração de Sistemas Difusos com Algoritmos Genéticos

2.1. Sistemas Difusos

2.1.1. Introdução

O conceito de conjunto difuso foi introduzido, em 1965, por Lotfi A. Zadeh. A ele é atribuído o reconhecimento como grande colaborador do Controle Moderno. No início da década de 60, Zadeh observou que os recursos tecnológicos disponíveis eram incapazes de automatizar as atividades relacionadas a problemas de natureza industrial, biológica ou química, que compreendessem situações ambíguas, não passíveis de processamento através da lógica computacional fundamentada na lógica booleana.

Procurando solucionar esses problemas o Prof. Zadeh publicou em 1965 [1] um artigo resumindo os conceitos dos Conjuntos Difusos, revolucionando o assunto com a criação dos Sistemas Difusos.

Em 1974, o Prof. Mamdani, do Queen Mary College, Universidade de Londres, após várias tentativas em controlar uma máquina a vapor com diferentes tipos de controladores, incluindo o PID, somente conseguiu fazê-lo através da aplicação do raciocínio Difuso [2].

Vários artigos sobre lógica Difusa têm sido publicados com o intuito de organizar as diversas aplicações e desenvolvimentos por áreas de concentração [3,4,5,16].

Estimulados pelo desenvolvimento e pelas enormes possibilidades práticas de aplicações que se apresentaram, os estudos sobre Sistemas Difusos e controle de processos avançam rapidamente, culminando com a criação em 1984, da Sociedade Internacional de Sistemas Difusos.

O desenvolvimento de técnicas de inteligência artificial (IA) nos últimos anos, ocupa cada vez mais posição de destaque em pesquisas na área de controle de processos industriais e, aos poucos, começam a ser implantadas em plantas industriais com enorme sucesso [2,6].

2.1.2. Lógica Difusa e Controle Difuso

O termo "Lógica Difusa" a princípio nos convida a pensar em algo confuso (nebuloso), trazendo até a um certo cetismo com relação a controle de máquinas as tornando mais capacitadas a seu trabalho.

Lógica Difusa é um modo de utilizar dados de processos inerentemente analógicos que deslocam-se através de uma faixa contínua em um computador digital que trabalha com dados numéricos bem definidos, ou seja, valores discretos.

Por exemplo, considerando um sistema de frenagem, dirigido por um microcontrolador; o microcontrolador toma as decisões baseado na temperatura do freio, velocidade, e outras variáveis do sistema.

A variável "temperatura" neste sistema pode ser dividida dentro de uma faixa de "estados": "frio", "fresco", "nominal", "morno", "quente". Contudo, a transição de um estado para o próximo é difícil de fixar; um limiar arbitrário deve ser definido para dividir, por exemplo, o "morno" do "quente", mas isso resultaria uma mudança descontínua quando o valor de entrada passase pelo limiar. O microcontrolador deveria ser capaz de detectar isso.

O caminho é criar os "estados difusos" que é, permitir a mudança gradual de um estado para o outro. A temperatura de entrada poderia ser definida usando funções intermediárias.

Desta forma, o estado da variável de entrada não mais salta abruptamente de um estado para o próximo; ao invés disso ele perde gradualmente valor em um estado enquanto vai ganhando valor no próximo estado. Até que em algum momento, o "valor verdadeiro" temperatura do freio será quase sempre em algum ponto entre duas funções consecutivas: 0.6 nominal e 0.4 morno, ou 0.7 nominal e 0.3 fresco, e assim por diante. A Figura 2.1 mostra uma representação destas funções consecutivas.

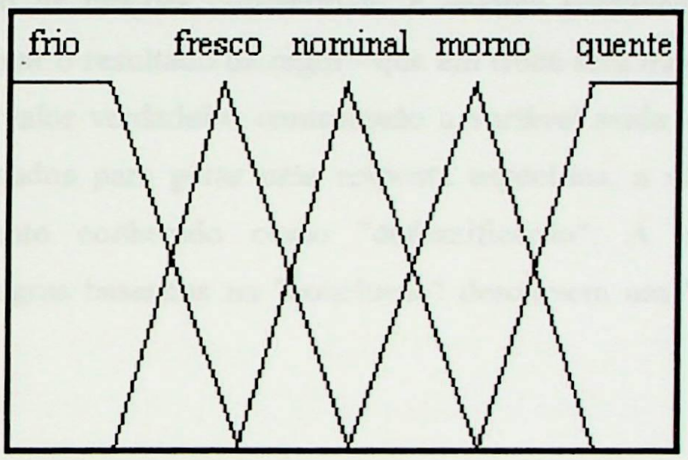


Figura 2.1 – Estados Difusos.

As variáveis de entrada em um sistema de controle difuso são em geral mapeadas dentro de conjuntos de funções consecutivas - o processo de conversão de um valor de entrada intermediário em um valor difuso é chamado de "fuzzificação". Note que um sistema de controle pode ter tipos de entradas chaveadas (on/off) junto com entradas analógicas, e tais entradas (on/off) do percurso terá sempre um valor verdadeiro igual a 1 ou 0 - mas tais entradas são realmente apenas um caso simplificado de uma variável difusa e então o sistema pode trabalhar com elas sem dificuldade.

Determinando o mapeamento das variáveis de entrada dentro das funções consecutivas e valores verdadeiros, o microcontrolador então toma decisões para que

as ações sejam efetuadas segundo as regras:

SE temperatura do freio é morna **E** velocidade é não muito rápida
ENTÃO pressão do freio é ligeiramente reduzida

- onde, neste caso, as duas variáveis de entrada são "temperatura do freio" e "velocidade". A variável de saída, "pressão do freio", é semelhantemente gerada a partir de um conjunto difuso que pode ter valores como "estático", "ligeiramente reduzido", "ligeiramente acrescido", e assim por diante.

A decisão é baseada em um conjunto de regras: todas as regras que aplicamos são invocadas, usando as funções consecutivas e valores verdadeiros obtidos das entradas, para determinar o resultado da regra - que em troca será mapeada dentro da função consecutiva e valor verdadeiro controlando a variável saída - e depois estes resultados são combinados para gerar uma resposta específica, a atual pressão do freio, um procedimento conhecido como "defuzzificação". A combinação de operações difusas e regras baseadas na "conclusão" descrevem um "sistema difuso especialista".

Tradicionais controles de sistemas são em geral baseados em modelos matemáticos que descrevem o sistema de controle usando uma ou mais equações diferenciais que definem a resposta do sistema para suas entradas; tais sistemas são freqüentemente implementados pelo chamado controlador "PID" (proporcional-integral-derivativo). Tais controladores são produtos de décadas de desenvolvimento e trabalho teórico e são altamente eficazes.

Se controladores PID e outros sistemas de controles tradicionais são tão bem desenvolvidos, por que preocupar-se com lógica difusa? Somente porque em alguns casos ela tem alguma vantagem: em muitos casos, o modelo matemático do processo pode não existir ou pode ser muito "caro" em termos de poder de processamento computacional e memória - e um sistema baseado em regras empíricas pode ser mais efetivo.

2.1.3. Controle Difuso em Detalhes

Controladores difusos são muito simples conceitualmente; eles consistem de um estágio de entrada, um estágio de processamento, e um estágio de saída. O estágio de entrada mapeia sensores ou outros tipos de entrada (dados numéricos nos quais o sistema se baseará para tomar as decisões) de maneira apropriada às funções consecutivas e valores verdadeiros; o estágio de processamento invoca cada regra adequada e gera um resultado para cada uma delas e então combina os resultados dessas regras; e finalmente o estágio de saída converte o resultado combinado no estágio anterior para dentro do controle.

A forma mais comum de funções consecutivas é a triangular (como mostrado na Figura 2.1), embora curvas trapezoidais além de outras formas também sejam usadas, mas a forma é geralmente menos importante do que o número de curvas e o local onde são postas.

O estágio de processamento é, como já discutido, baseado em uma coleção de regras lógicas na forma de declarações SE-ENTÃO, onde o SE é chamado de "antecedente" e o ENTÃO é chamado de "conseqüência". Sistemas de controle difuso típicos têm dúzias de regras.

Na prática, o conjunto de regras usualmente tem vários antecedentes que são combinados usando operadores difusos, tais como E, OU, e NÃO (apesar de novamente as definições variarem): E (em uma definição popular) simplesmente usa o peso mínimo para todos os antecedentes, enquanto OU usa os valores máximos [9]. (Existe também um operador NÃO que subtrai uma função consecutiva de 1 dando a função complementar.)

Existem vários modos diferentes para definir o resultado de uma regra, mas um dos mais comuns e simples é o chamado método de conclusão "max-min", em que a saída da função consecutiva é dada pelo valor verdadeiro gerado pela premissa.

Regras podem ser resolvidas em hardware paralelo ou em software seqüencial.

O resultado de todas as regras é "defuzzificado" para um valor conciso por um dos vários métodos; existem vários na teoria, cada qual com suas vantagens e desvantagens.

O método do centróide é muito popular, em que o "centro de massa" do resultado fornece o valor conciso; no processo de defuzzificação por média ponderada, o centróide de cada área é calculado isoladamente e o valor da saída calculado através da média desses centróides ponderada com o valor máximo da função de pertinência. No processo por média da máxima função de pertinência, é calculada a média do máximo valor da função de pertinência, sendo este o valor de saída [13].

O projeto de sistemas de controle difuso é baseado em métodos empíricos - basicamente uma aproximação metódica para tentativa-e-erro. Existem poucas regras pré-definidas no presente momento uma vez que a tecnologia é ainda nova; o processo em geral segue os seguintes passos:

- Documentam-se as especificações operacionais do sistema de entradas e saídas.
- Documentam-se os conjuntos difusos para as entradas .
- Documenta-se o conjunto de regras.
- Determina-se o método de defuzzificação.
- Executa-o através de teste para verificação do sistema, ajustando os detalhes como requerido.

2.1.4. Exemplo de um Controle Difuso

Vejamos o exemplo no qual teremos um sistema especialista para especificar a pressão a qual deve ser utilizada no freio de um veículo em função da distância do veículo a um obstáculo e a velocidade que este veículo se encontra.

1) Primeiramente definiremos as variáveis do sistema:

Velocidade do veículo;

Distância do veículo;

Pressão no freio.

2) Em seguida definiremos os conjuntos difusos de cada variável como mostra a Figura 2.2:

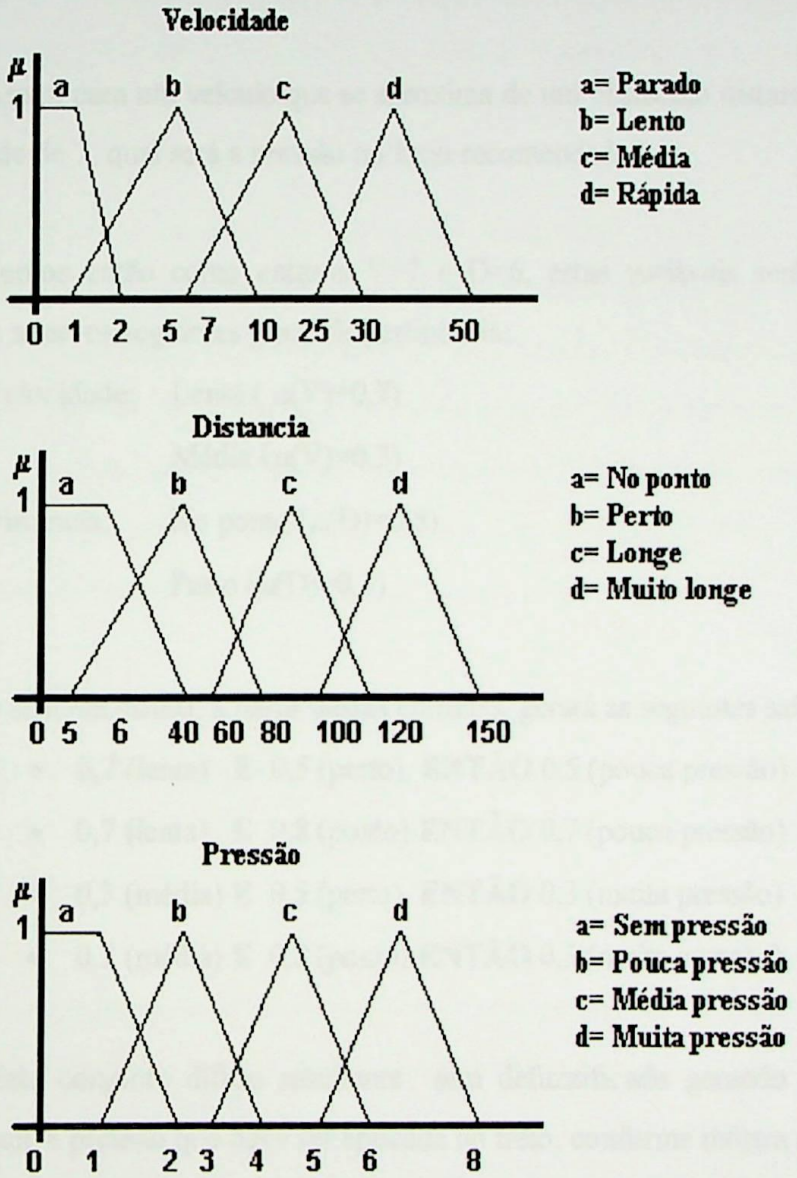


Figura 2.2 – Conjuntos difusos

Note que para uma velocidade de 1,5 temos dois graus de pertinência um para o conjunto *a* (parado) e outro para o conjunto *b* (lento). Isso é válido para o maioria dos valores das variáveis devido a interseção observada entre os conjuntos.

3) Passaremos agora a definir as regras do sistema. Vejamos algumas delas:

- **SE** Lento **E** Perto do obstáculo **ENTÃO** Pouca pressão
- **SE** Lento **E** No obstáculo **ENTÃO** Pouca pressão
- **SE** Média velocidade **E** Perto do obstáculo **ENTÃO** Muita pressão
- **SE** Média velocidade **E** No obstáculo **ENTÃO** Muita pressão

4) E como função de defuzzificação utilizaremos o método do centróide.

Assim para um veículo que se aproxima de um obstáculo distante de 6 e a uma velocidade de 7, qual será a pressão no freio recomendada?

Temos então como entrada V=7 e D=6, estas variáveis serão fuzzificadas passando a ter os seguintes graus de pertinência:

Velocidade: Lenta (μ(V)=0,7)

Média (μ(V)=0,3)

Distância: No ponto (μ(D)=0,8)

Perto (μ(D)=0,5)

O sistema difuso, a partir destas entradas, gerará as seguintes saídas:

- 0,7 (lenta) **E** 0,5 (perto) **ENTÃO** 0,5 (pouca pressão)
- 0,7 (lenta) **E** 0,8 (ponto) **ENTÃO** 0,7 (pouca pressão)
- 0,3 (média) **E** 0,5 (perto) **ENTÃO** 0,3 (muita pressão)
- 0,3 (média) **E** 0,8 (ponto) **ENTÃO** 0,3 (muita pressão)

Este conjunto difuso resultante será defuzzificado gerando a saída 4 que corresponde a pressão que deve ser aplicada no freio, conforme mostra a Figura 2.3:

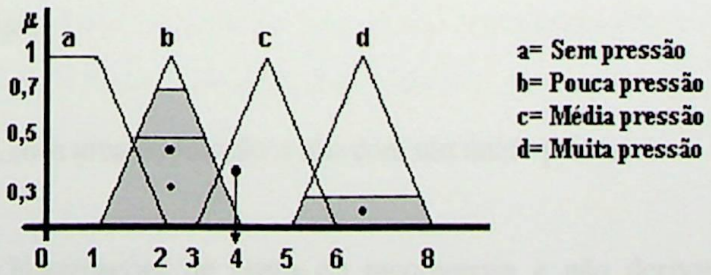


Figura 2.3 – Defuzzificação

2.2. Algoritmos Genéticos

2.2.1. Introdução

Toda tarefa de busca e otimização possui vários componentes, entre eles: o espaço de busca, onde são consideradas todas as possibilidades de solução de um determinado problema e a função de avaliação (ou função de custo), uma maneira de avaliar os membros do espaço de busca. Existem muitos métodos de busca e funções de avaliação.

As técnicas de busca e otimização tradicionais iniciam-se com um único candidato que, iterativamente, é manipulado utilizando algumas heurísticas (estáticas) diretamente associadas ao problema a ser solucionado. Geralmente, estes processos heurísticos não são algorítmicos e sua simulação em computadores pode ser muito complexa. Apesar destes métodos não serem suficientemente robustos, isto não implica que eles sejam inúteis. Na prática, eles são amplamente utilizados, com sucesso, em inúmeras aplicações [7].

Por outro lado, as técnicas de computação evolucionária operam sobre uma população de candidatos em paralelo. Assim, elas podem fazer a busca em diferentes áreas do espaço de solução, alocando um número de membros apropriado para a busca em várias regiões.

Os Algoritmos Genéticos (AGs) diferem dos métodos tradicionais de busca e otimização, principalmente em quatro aspectos [7]:

1. AGs trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros.

2. AGs trabalham com uma população e não com um único ponto.

3. AGs utilizam informações de custo ou recompensa e não derivadas ou outro conhecimento auxiliar.

4. AGs utilizam regras de transição probabilísticas e não determinísticas.

Além de ser uma estratégia de gerar-e-testar muito elegante, por serem baseados na evolução biológica, são capazes de identificar e explorar fatores ambientais e convergir para soluções ótimas, ou aproximadamente ótimas em níveis globais.

Quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes: este é o conceito básico da evolução genética biológica. A área biológica mais proximamente ligada aos Algoritmos Genéticos é a Genética Populacional.

Os pesquisadores referem-se a "algoritmos genéticos" ou a "um algoritmo genético" e não "ao algoritmo genético", pois AGs são uma classe de procedimentos com muitos passos separados, e cada um destes passos possui muitas variações possíveis.

Antes de prosseguir com a análise das características destes algoritmos, alguns conceitos básicos são necessários; estes conceitos podem ser naturalmente expostos explicando o funcionamento básico destes algoritmos.

Inicialmente, é gerada uma população formada por um conjunto aleatório de indivíduos que podem ser vistos como possíveis soluções do problema. Durante o processo evolutivo, esta população é avaliada: para cada indivíduo é dada uma nota, ou índice, refletindo sua habilidade de adaptação a determinado ambiente. Uma porcentagem dos mais adaptados são mantidos, enquanto os outros são descartados (darwinismo). Os membros mantidos pela seleção podem sofrer modificações em suas características fundamentais através de mutações e cruzamento (crossover), gerando descendentes para a próxima geração. Este processo, chamado de reprodução, é repetido até que uma solução satisfatória seja encontrada.

Embora possam parecer simplistas do ponto de vista biológico, estes algoritmos são suficientemente complexos para fornecer mecanismos de busca adaptativos poderosos e robustos.

2.2.2. Histórico

Até meados do século 19, os naturalistas acreditavam que cada espécie havia sido criada separadamente por um ser supremo ou através de geração espontânea. O trabalho do naturalista Carolus Linnaeus sobre a classificação biológica de organismos despertou o interesse pela similaridade entre certas espécies, levando a acreditar na existência de uma certa relação entre elas. Outros trabalhos influenciaram os naturalistas em direção à teoria da seleção natural, tais como os de Jean Baptiste Lamark, que sugeriu uma teoria evolucionária no "uso e desuso" de órgãos; e de Thomas Robert Malthus, que propôs que fatores ambientais tais como doenças e carência de alimentos, limitavam o crescimento de uma população.

Depois de mais de 20 anos de observações e experimentos, Charles Darwin apresentou em 1858 sua teoria de evolução através de seleção natural, simultaneamente com outro naturalista inglês Alfred Russel Wallace. No ano seguinte, Darwin publica o seu On the Origin of Species by Means of Natural Selection com a sua teoria completa, sustentada por muitas evidências colhidas durante suas viagens a bordo do Beagle.

Este trabalho influenciou muito o futuro não apenas da Biologia, Botânica e Zoologia, mas também teve grande influência sobre o pensamento religioso, filosófico, político e econômico da época. A teoria da evolução e a computação nasceram praticamente na mesma época: Charles Babbage, um dos fundadores da computação moderna e amigo pessoal de Darwin desenvolveu sua máquina analítica em 1833. Ambos provavelmente estariam surpresos e orgulhosos com a ligação entre estas duas áreas.

Por volta de 1900, o trabalho de Gregor Mendel, desenvolvido em 1865, sobre os princípios básicos de herança genética, foi redescoberto pelos cientistas e teve grande influência sobre os futuros trabalhos relacionados à evolução. A moderna teoria da evolução combina a genética e as idéias de Darwin e Wallace sobre a seleção natural, criando o princípio básico de Genética Populacional: a variabilidade entre indivíduos em uma população de organismos que se reproduzem sexualmente é produzida pela mutação e pela recombinação genética.

Este princípio foi desenvolvido durante os anos 30 e 40, por biólogos e matemáticos de importantes centros de pesquisa. Nos anos 50 e 60, muitos biólogos começaram a desenvolver simulações computacionais de sistemas genéticos. Entretanto, foi John Holland quem começou, seriamente, a desenvolver as primeiras pesquisas no tema. Holland foi gradualmente refinando suas idéias e em 1975 publicou o seu livro Adaptation in Natural and Artificial Systems [14], hoje considerado a Bíblia de Algoritmos Genéticos. Desde então, estes algoritmos vêm sendo aplicados com sucesso nos mais diversos problemas de otimização e aprendizado de máquina.

2.2.3. Características Gerais dos Algoritmos Genéticos

Algoritmos Genéticos são algoritmos de otimização global, baseados nos mecanismos de seleção natural e da genética. Eles empregam uma estratégia de busca paralela e estruturada, mas aleatória, que é voltada em direção ao reforço da busca de pontos de "alta aptidão", ou seja, pontos nos quais a função a ser minimizada (ou maximizada) tem valores relativamente baixos (ou altos).

Apesar de aleatórios, eles não são caminhadas aleatórias não direcionadas, pois exploram informações históricas para encontrar novos pontos de busca onde são esperados melhores desempenhos. Isto é feito através de processos iterativos, onde cada iteração é chamada de geração.

Durante cada iteração, os princípios de seleção e reprodução são aplicados a uma população de candidatos que pode variar, dependendo da complexidade do problema e dos recursos computacionais disponíveis. Através da seleção, se determina quais indivíduos conseguirão se reproduzir, gerando um número determinado de descendentes para a próxima geração, com uma probabilidade determinada pelo seu índice de aptidão. Em outras palavras, os indivíduos com maior adaptação relativa têm maiores chances de se reproduzir.

O ponto de partida para a utilização de Algoritmos Genéticos, como ferramenta para solução de problemas, é a representação destes problemas de maneira que os Algoritmos Genéticos possam trabalhar adequadamente sobre eles. A maioria das representações são genotípicas, utilizam vetores de tamanho finito em um alfabeto finito.

Tradicionalmente, os indivíduos são representados genotípicamente por vetores binários, onde cada elemento de um vetor denota a presença (1) ou ausência (0) de uma determinada característica. Entretanto, existem aplicações onde é mais conveniente o uso de representações por inteiros como apresentado mais adiante neste trabalho.

O princípio básico do funcionamento dos AGs é que um critério de seleção fará com que, depois de muitas gerações, o conjunto inicial de indivíduos gere indivíduos mais aptos. A maioria dos métodos de seleção são projetados para escolher preferencialmente indivíduos com maiores notas de aptidão, embora não exclusivamente, a fim de manter a diversidade da população. Um método de seleção muito utilizado é o método da Roleta, onde indivíduos de uma geração são escolhidos para fazer parte da próxima geração, através de um sorteio de roleta. A Figura 2.4 mostra a representação da roleta para uma população de 4 indivíduos.

I	Indivíduo x_i	$f_i(x)$ Aptidão (x^2)	Aptidão Relativa
1	00101	25	0,04
2	00010	4	0,01
3	10110	484	0,81
4	01001	81	0,14
Total		594	1,00

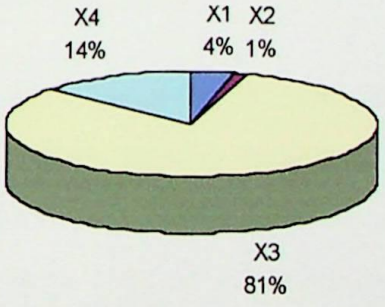


Fig. 2.4 - Indivíduos de uma população e a sua correspondente roleta de seleção.

Neste método, cada indivíduo da população é representado na roleta proporcionalmente ao seu índice de aptidão. Assim, aos indivíduos com alta aptidão é dada uma porção maior da roleta, enquanto aos de aptidão mais baixa é dada uma porção relativamente menor da roleta. Finalmente, a roleta é girada um determinado

número de vezes, dependendo do tamanho da população, e são escolhidos, como indivíduos que participarão da próxima geração, aqueles sorteados na roleta.

Um conjunto de operações é necessário para que, dada uma população, se consiga gerar populações sucessivas que (espera-se) melhorem sua aptidão com o tempo. Estes operadores são: cruzamento (crossover) e mutação. Eles são utilizados para assegurar que a nova geração seja totalmente nova, mas possui, de alguma forma, características de seus pais, ou seja, a população se diversifica e mantém características de adaptação adquiridas pelas gerações anteriores. Para prevenir que os melhores indivíduos não desapareçam da população pela manipulação dos operadores genéticos, eles podem ser automaticamente colocados na próxima geração, através da reprodução elitista.

Esse ciclo é repetido um determinado número de vezes. A seguir, é mostrado um exemplo de algoritmo genético. Durante esse processo, os melhores indivíduos, assim como alguns dados estatísticos, podem ser coletados e armazenados para avaliação.

```
Procedimento AG
{ g = 0;
inicia_população (P, g)
avaliação (P, g);
repita até (g = t)
    { g = g +1;
      seleção_dos_pais (P, g);
      recombinação (P, g);
      mutação (P, g);
      avaliação (P, g);
    }
}
```

onde:
g – geração atual;
t – número de gerações para finalizar o algoritmo;
P – população.

Estes algoritmos, apesar de serem computacionalmente muito simples, são bastante poderosos. Além disso, eles não são limitados por suposições sobre o espaço de busca, relativas a continuidade, existência de derivadas, etc.

2.2.4. Operadores Genéticos

O princípio básico dos operadores genéticos é transformar a população através de sucessivas gerações, estendendo a busca até chegar a um resultado satisfatório. Os operadores genéticos são necessários para que a população se diversifique e mantenha características de adaptação adquiridas pelas gerações anteriores.

O operador de mutação é necessário para a introdução e manutenção da diversidade genética da população, alterando arbitrariamente um ou mais componentes de uma estrutura escolhida, como é ilustrado na Figura 2.5, fornecendo assim, meios para introdução de novos elementos na população. Desta forma, a mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca será zero, além de contornar o problema de mínimos locais, pois com este mecanismo, altera-se levemente a direção da busca. O operador de mutação é aplicado aos indivíduos com uma probabilidade dada pela taxa de mutação *Pm*; geralmente se utiliza uma taxa de mutação pequena, pois é um operador genético secundário.

Antes da Mutação:	1 ⓪ 1 0 0
Depois da Mutação:	1 0 1 0 0

Figura 2.5 - Exemplo de mutação.

O cruzamento é o operador responsável pela recombinação de características dos pais durante a reprodução, permitindo que as próximas gerações herdem essas características. Ele é considerado o operador genético predominante, por isso é aplicado com probabilidade dada pela taxa de crossover *Pc*, que deve ser maior que a taxa de mutação.

Este operador pode, ainda, ser utilizado de várias maneiras; as mais utilizadas são:

Um-ponto: um ponto de cruzamento é escolhido e a partir deste ponto as informações genéticas dos pais serão trocadas. As informações anteriores a este ponto

em um dos pais são ligadas às informações posteriores à este ponto no outro pai, como é mostrado no exemplo da Figura 2.6.

Multi-pontos: é uma generalização desta idéia de troca de material genético através de pontos, onde muitos pontos de cruzamento podem ser utilizados.

Uniforme: não utiliza pontos de cruzamento, mas determina, através de um parâmetro global, qual a probabilidade de cada variável ser trocada entre os pais.

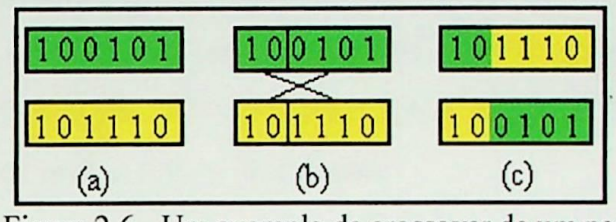


Figura 2.6 - Um exemplo de crossover de um ponto.
(a) dois indivíduos são escolhidos.
(b) um ponto (2) de crossover é escolhido.
(c) são recombinadas as características, gerando dois novos indivíduos.

2.2.5. Parâmetros Genéticos

É importante também, analisar de que maneira alguns parâmetros influem no comportamento dos Algoritmos Genéticos, para que se possa estabelecê-los conforme as necessidades do problema e dos recursos disponíveis.

Tamanho da População. O tamanho da população afeta o desempenho global e a eficiência dos AGs. Com uma população pequena o desempenho pode cair, pois deste modo a população fornece uma pequena cobertura do espaço de busca do problema. Uma grande população geralmente fornece uma cobertura representativa do domínio do problema, além de prevenir convergências prematuras para soluções locais ao invés de globais. No entanto, para se trabalhar com grandes populações, são necessários maiores recursos computacionais, ou que o algoritmo trabalhe por um período de tempo muito maior.

Taxa de Cruzamento. Quanto maior for esta taxa, mais rapidamente novas estruturas serão introduzidas na população. Mas se esta for muito alta, a maior parte

Tabela 2.1 - Comparação de características da lógica difusa com algoritmos genéticos

	Armazena Conhecimento	Aprende	Otimiza	Rápido	Sistemas Não-lineares
Sistemas difusos	✓			✓	✓
Algoritmos genéticos		✓	✓	✓	✓

Algoritmos genéticos oferecem vantagens distintas de otimização das funções de pertinência e mesmo de aprendizagem de regras difusas. Os algoritmos genéticos resultam em uma pesquisa mais global, reduzindo a chance de terminarem em um mínimo local, através de amostragem de vários conjuntos de soluções simultaneamente. A lógica difusa contribui com a função de avaliação, estágio do algoritmo genético onde o ajuste é determinado.

São várias as maneiras possíveis de se usar os algoritmos genéticos com sistemas difusos. Um tipo de sistema híbrido envolve o uso de módulos separados como parte de um sistema global. Os módulos baseados em algoritmos genéticos e na lógica difusa podem ser agrupados isoladamente ou com outros subsistemas de programas computacionais inteligentes ou convencionais que formam um sistema aplicativo.

Um outro uso é o projeto de sistemas que são principalmente de aplicações com a lógica difusa. O uso de algoritmos genéticos tem como objetivo melhorar o processo do projeto e o desempenho do sistema operacional baseado no sistema difuso. Os algoritmos genéticos podem ser usados para descobrir os melhores valores para funções de pertinência quando a seleção manual de valores é difícil ou toma muito tempo.

O procedimento geral para se usar os algoritmos genéticos com os sistemas difusos é mostrado na Figura 2.7. Por exemplo, um cromossomo pode ser definido como sendo uma concatenação dos valores de todas as funções de pertinência. Quando as funções triangulares são usadas para representar as funções de pertinência, os parâmetros são os centros e as larguras para cada conjunto difuso. De uma gama inicial de valores de parâmetros possíveis, o sistema difuso é rodado para determinar o quanto ele funciona bem. Essas informações são usadas para determinar o ajuste de

cada cromossomo e para estabelecer a nova população. O ciclo é repetido até que seja encontrado o melhor conjunto de valores para os parâmetros das funções de pertinência.

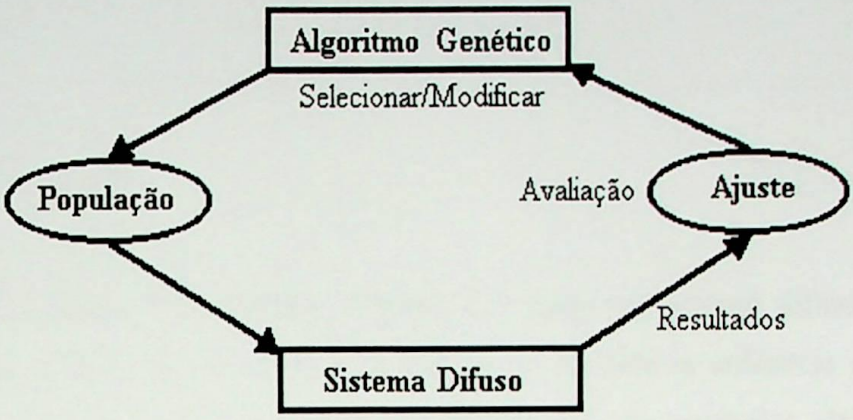


Figura 2.7 – Processo global para uso de um algoritmo genético a fim de melhorar o desempenho de um sistema difuso.

O processo acima pode ser expandido para usar os cromossomos que incluem informações sobre as condições e ações correspondentes às regras difusas. Incluí-las no tratamento genético permite ao sistema aprender ou refinar as regras difusas.

Park, Kandel e Langholz [19] relatam a bem sucedida aplicação dos modelos de raciocínio difuso aos sistemas de controle difuso, que geralmente dependem da determinação subjetiva de vários parâmetros difusos. Eles mostram que os sistemas de controle difuso podem ser melhorados se o modelo de raciocínio difuso for suplementado por um mecanismo de aprendizagem baseado em genética que gera os mais eficientes parâmetros a partir de valores iniciais aleatórios ou subjetivos.

Capítulo 3

Descrição do Problema

3.1. Introdução

O objetivo deste trabalho é demonstrar como um controle difuso pode ser otimizado através do ajuste de suas funções de pertinência utilizando algoritmos genéticos. Como citado no Capítulo 2, um exemplo de resultados obtidos nesta otimização é o apresentado por Park, Kandel e Langholz, 1994 [19], sendo que este trabalho serviu como base para este desenvolvimento.

Neste desenvolvimento foi utilizado o Pacote Computacional para Ensino da Lógica Difusa [20]. Este programa computacional foi desenvolvido para o auto-treinamento de estudantes na teoria de controle difuso. O programa contém todas instruções necessárias para os usuários compreenderem os princípios do controle difuso. Este pacote computacional tem sido utilizado em diversos cursos de controle e de lógica difusa no Brasil e no exterior.

Entretanto, o grande problema encontrado é que o processo de aprendizado é por tentativa-e-erro. O usuário fornece um conjunto de regras e funções de pertinência e, em seguida, realiza diversos testes para verificar a qualidade do controle. É sabido que este processo de aprendizado (por tentativa-e-erro) pode não trazer os resultados esperados pois diversos erros de interpretação podem ocorrer [21].

Neste trabalho, ver-se-á como os algoritmos genéticos podem fazer um ajuste automático das funções de pertinência melhorando significativamente o controle, minimizando o espaço percorrido pelo veículo até estacionar e auxiliando os estudantes no aprendizado de controle difuso.

3.2. O Pacote Computacional Original

O principal objetivo do pacote é estacionar um veículo em uma garagem, partindo de qualquer posição inicial. Para cumprir esta tarefa, o usuário deve inicialmente desenvolver um conjunto de regras de controle difuso e funções de pertinência que definirão a trajetória do veículo. Diversas janelas e rotinas numéricas estão disponíveis no programa com a finalidade de auxiliar os usuários no estabelecimento de tais regras. Os processos de fuzzificação e de defuzzificação das variáveis são realizados pelo programa sem a interferência do usuário [20].

Para representar o problema do estacionamento de um veículo o programa possui uma tela básica, que é mostrada na Figura 3.1. Nela aparecem a posição da garagem, as barreiras existentes (no caso, as paredes) e os valores das coordenadas limites. Também são apresentadas as variáveis de entrada envolvidas no problema, ou seja, *(x , y)* medidas a partir do ponto central da parte traseira do veículo e o ângulo do carro (ϕ).

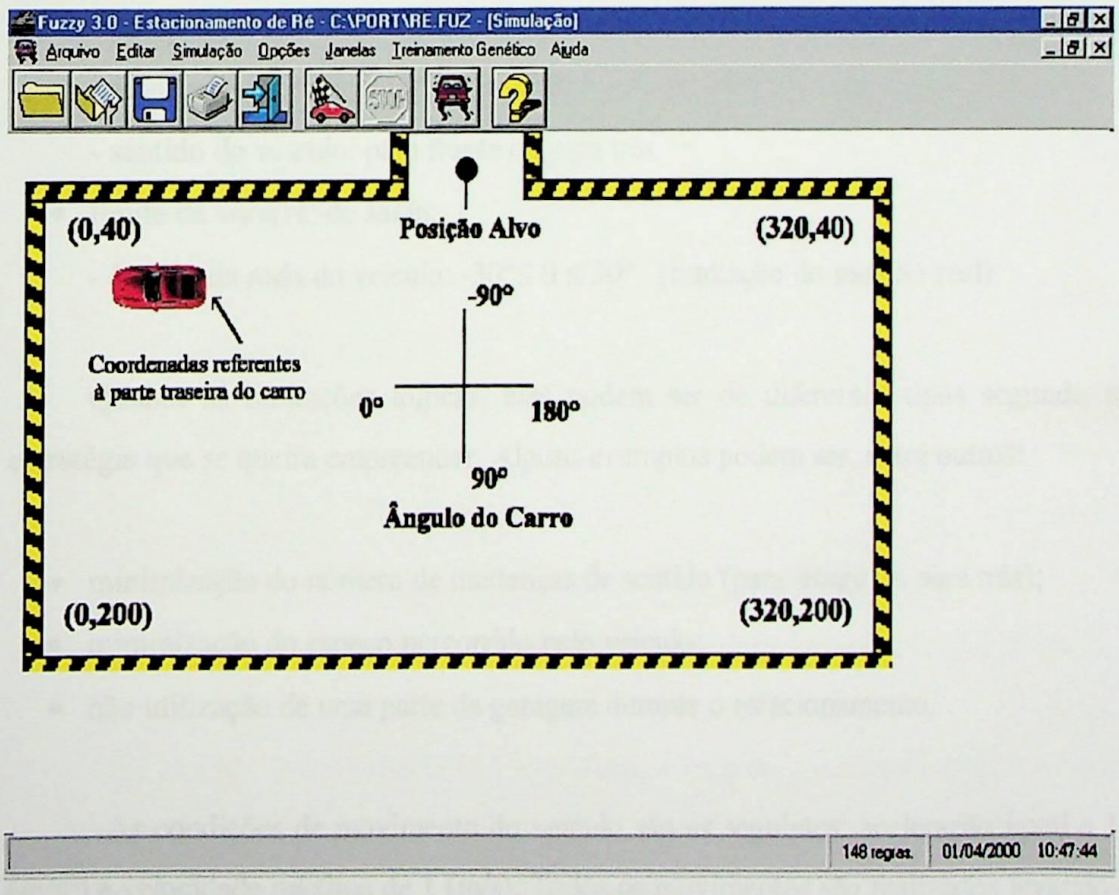


Figura 3.1 - Tela básica do programa.

A tela básica deste programa é composta também por um conjunto de menus, sendo que suas funções são indicadas pelo nome dos atributos. Ao acionar (via teclado ou via mouse) o atributo Arquivo, poder-se-á carregar em conjunto de regras e funções de pertinência anteriormente desenvolvidos, salvar um novo conjunto ou iniciar sua criação. Também é através desse atributo que existe o caminho para impressões de diversos tipos e saída do programa.

Este programa foi desenvolvido em Visual-Basic 3.0 e dispõe das várias facilidades de uso que esta linguagem e o Sistema Operacional Windows oferecem.

3.2.1. Condições Para o Estacionamento

Algumas condições são estabelecidas para o estacionamento. Elas podem ser de dois tipos: ligadas ao pacote computacional e lógicas. As ligadas ao pacote se referem as limitações físicas, são elas [20]:

- limites das variáveis de entrada:
  - posição (x, y): $0 < x < 32$ e $0 < y < 20$ (m) (limitações do estacionamento)
  - ângulo do veículo: $-90° \leq \phi \leq 270°$
  - sentido do veículo: para frente ou para trás
- limite da variável de saída:
  - ângulo da roda do veículo: $-30° \leq \theta \leq 30°$ (limitação do modelo real)

Quanto às limitações lógicas, elas podem ser de diferentes tipos segundo a estratégia que se queira empreender. Alguns exemplos podem ser, entre outros:

- minimização do número de mudanças de sentido (para frente ou para trás);
- minimização do espaço percorrido pelo veículo;
- não utilização de uma parte da garagem durante o estacionamento.

As condições de movimento do veículo são as seguintes: aceleração igual a 1 (m/s^2) e velocidade máxima de 1 (m/s). Todos os movimentos são realizados levando-se estes dois valores como referência.

Existem três possibilidades de inversão do sentido do movimento, que são [20]:

a) Choque contra a parede: quando o sistema verifica que no próximo passo da simulação, o veículo irá se chocar contra a parede;

b) Regra que força a inversão: quando como conseqüência de uma regra for utilizada a ordem inverter; ou,

c) Falta de saídas: quando o controle não utilizar nenhuma regra, ou seja, se área de saída for nula (zero).

3.3. Criação de um Controle

O usuário deste pacote computacional pode definir um novo sistema criando as funções de pertinência e as regras de controle. A Figura 3.2 mostra a janela Criar Conjuntos Difusos onde se define o número de funções de pertinência para cada variável.

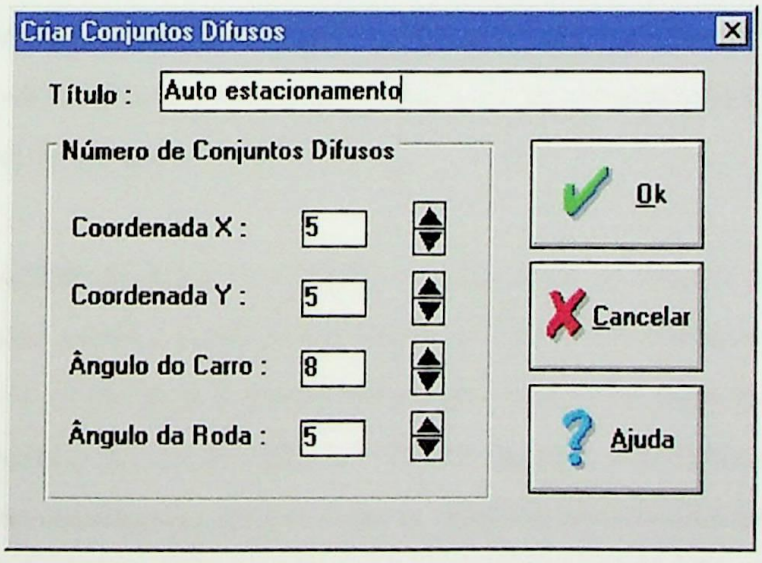


Fig. 3.2 - Janela de definição do número de funções de pertinência das variáveis.

As funções de pertinência são igualmente espaçadas na superfície de controle da variável. O usuário modifica essas funções de pertinência através da janela Editar Conjuntos Difusos. A Figura 3.3 mostra esta janela.

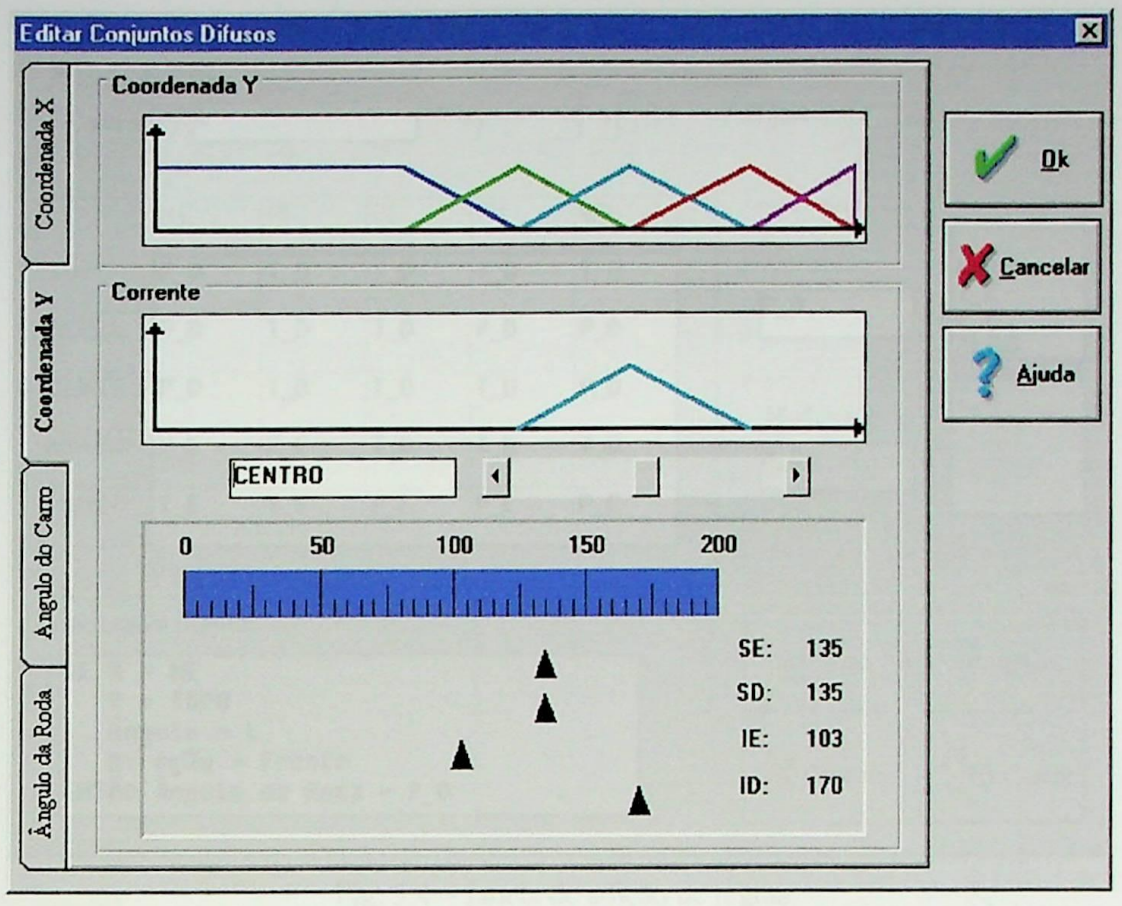


Fig. 3.3 – Janela editar conjuntos difusos.

O usuário fornece as funções de pertinência das variáveis de entrada e saída do programa. Isto pode ser feito através de arquivos anteriormente gerados ou pela edição de novas funções de pertinência. A Figura 3.3 apresenta um exemplo de edição para a variável de entrada *y*.

Para definir as regras de controle, ou seja, como as funções de pertinência serão agrupadas, existe a janela Editar Regras. A Figura 3.4 apresenta um exemplo desta janela de edição para o preenchimento dos valores de saída das regras com valores de premissa iguais a x = ME , y = TOPO, ângulo L e movimento para frente. O programa se encarrega de montar todas as possíveis combinações das funções de pertinência das variáveis de entrada, bastando ao usuário o preenchimento do valor da conclusão da regra.

Na Figura 3.4, pode-se encontrar duas regiões de interesse. A primeira onde existe a possibilidade da seleção da direção (frente ou marcha ré) e da coordenada

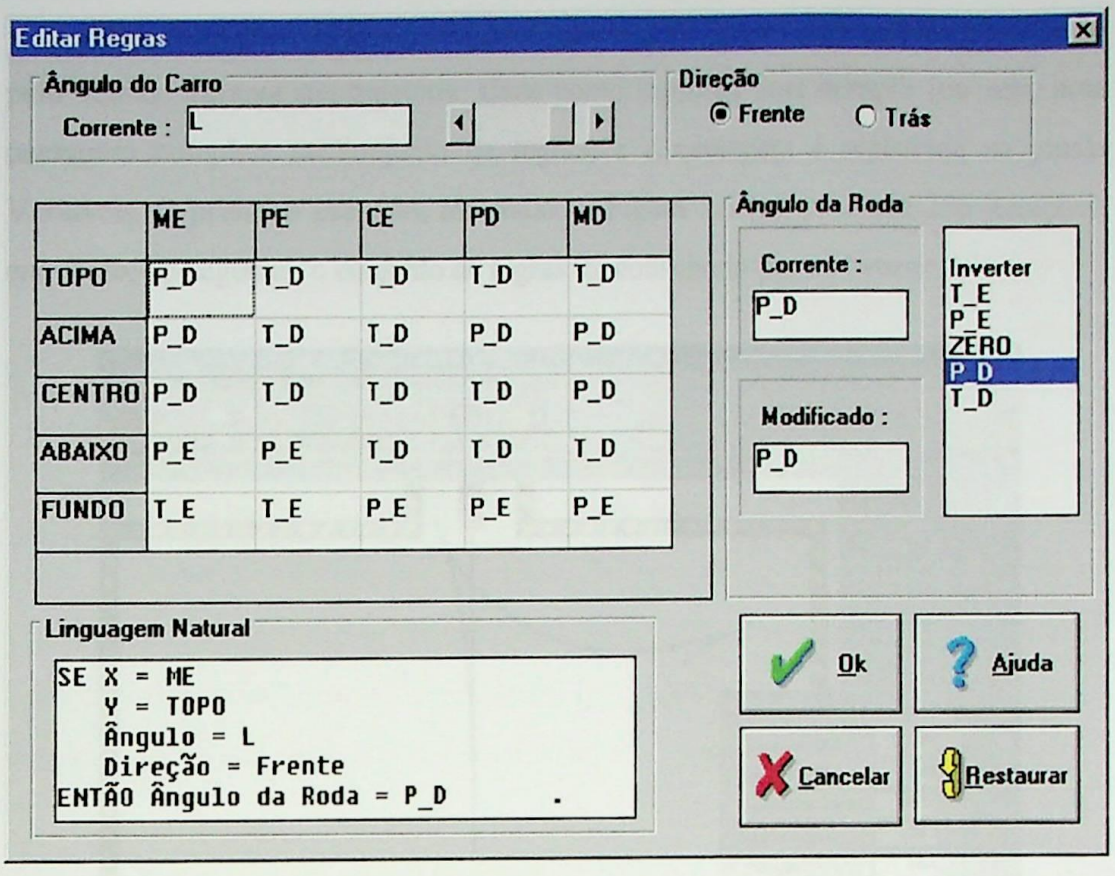


Fig. 3.4 – Janela de edição de regras.

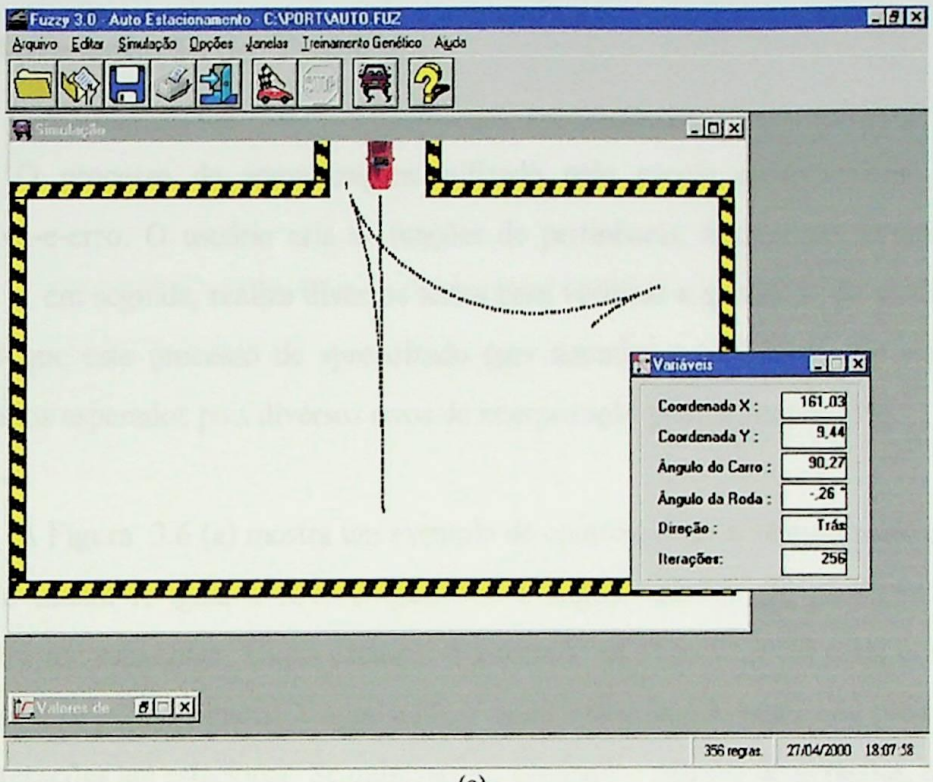
correspondente ao ângulo do carro. A segunda região de interesse contém o preenchimento da conclusão da regra. Isto pode ser feito através da seleção de um dos valores de saída (ou nenhum deles para uma regra não definida ou inverter). Por exemplo, para x = ME, y = TOPO, ângulo = L e direção = Frente foi selecionado o valor P_D como ângulo da roda, o que corresponde a regra:

SE x é ME **E** y é TOPO **E** *ângulo do carro* é L **E** Sentido do Movimento é *para frente* ENTÃO ângulo modificado é P_D.

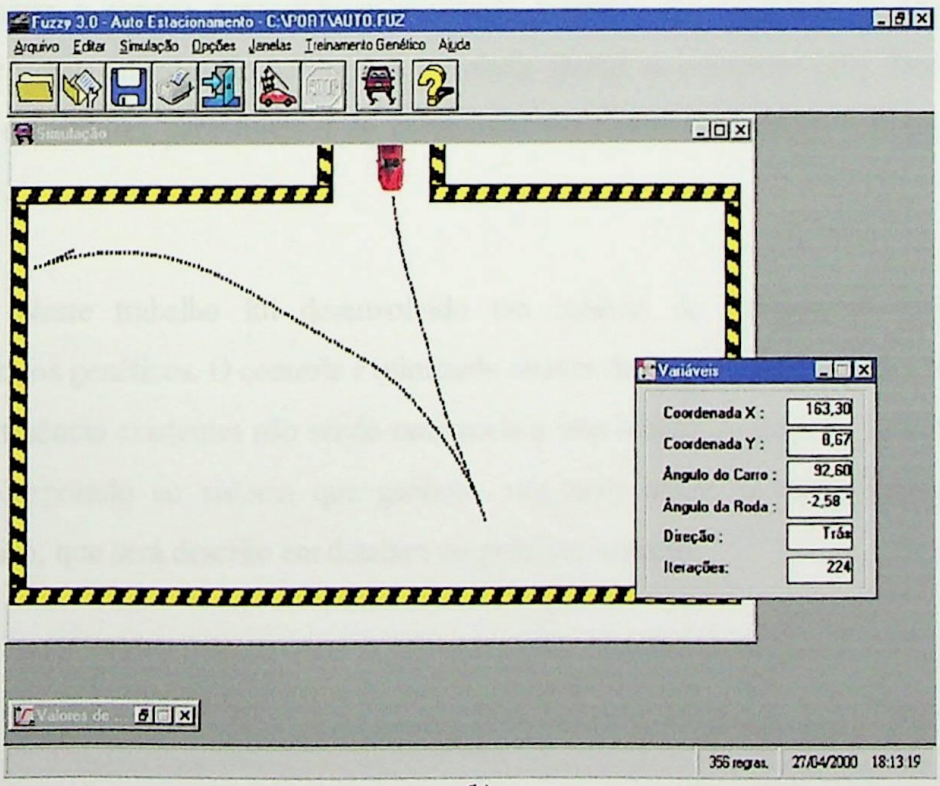
3.4. Simulações

Através da opção Posição Inicial do menu Editar o usuário pode definir uma posição inicial (Coordenada X, Y e Ângulo) para o veículo. A simulação é iniciada através da opção Iniciar do menu Simulação. A Figura 3.5 mostra o deslocamento do veículo com diferentes posições iniciais utilizando o conjunto composto por 356 regras para o deslocamento.

Nos exemplos de simulação da Figura 3.5, pode-se verificar o rastro deixado pelo veículo durante sua trajetória. Cada ponto significa uma iteração (ou seja, uma passagem completa no conjunto de regras) e a contagem é registrada na janela Variáveis. O primeiro exemplo, mostrado na Figura 3.5 (a), produziu 256 iterações; enquanto, no segundo, o conjunto de regras foi consultado por 224 vezes.



(a)



(b)

Figura 3.5 - Exemplos de simulação do pacote computacional.

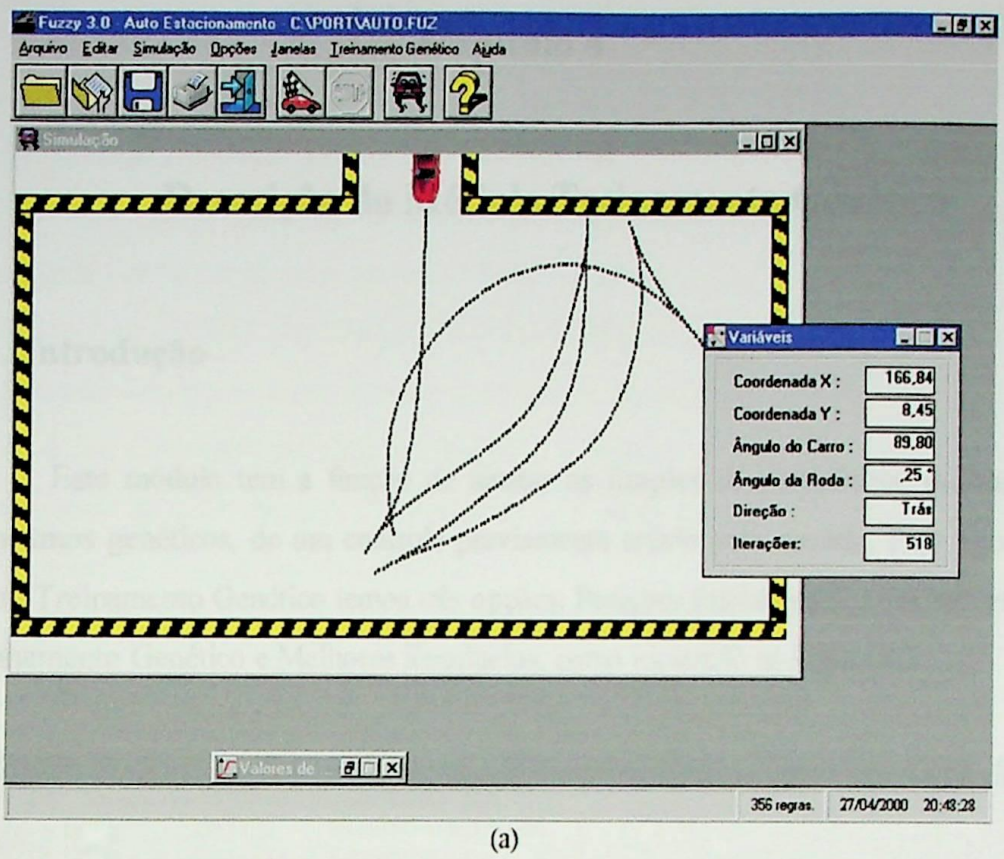
O pacote computacional dispõe de recursos, que permitem variar o tamanho do carro entre: pequeno, médio ou grande. Esta variação cria a oportunidade de verificar o comportamento do sistema de controle para um equipamento que tenha alteradas algumas de suas grandezas. O pacote computacional possui também três métodos de defuzzificação, que são: o centróide, média das áreas e média das máximas [13].

3.5. Apresentação do Problema

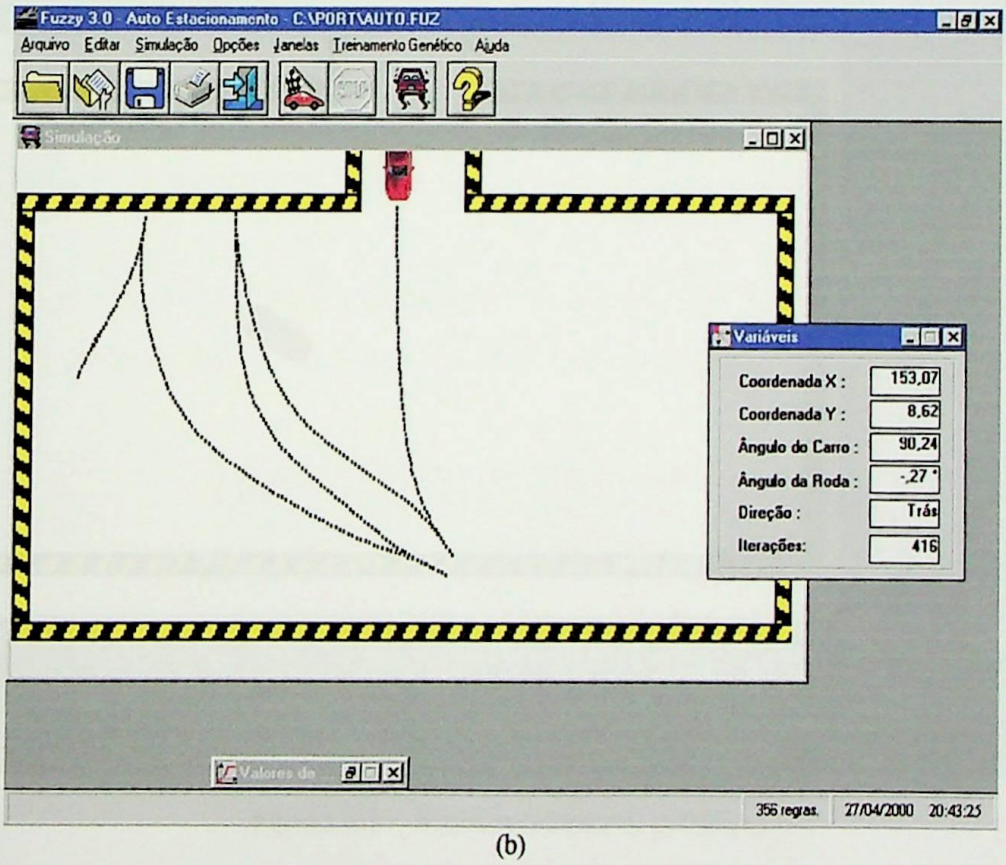
O processo de aprendizagem utilizado pelo pacote computacional é por tentativa-e-erro. O usuário cria as funções de pertinência, fornece um conjunto de regras e, em seguida, realiza diversos testes para verificar a qualidade do controle. É sabido que este processo de aprendizado (por tentativa-e-erro) pode não trazer os resultados esperados pois diversos erros de interpretação podem ocorrer [19].

A Figura 3.6 (a) mostra um exemplo de controle onde o veículo parte de uma posição inicial X igual a 154, Y igual 166 e ângulo igual a –80 produzindo 518 iterações até estacionar. Outro exemplo e mostrado na Figura 3.6 (b) onde o veículo parte de uma posição inicial X igual a 26, Y igual a 88 e ângulo igual a 62 produzindo 416 interações até estacionar. Se quiséssemos diminuir o número de iterações, ou seja, minimizar o espaço percorrido pelo veículo teríamos que modificar as regras ou inserir novas funções de pertinência ou ainda ajustar as existentes [20]. Definir os melhores valores para funções de pertinência manualmente é difícil e toma muito tempo.

Neste trabalho foi desenvolvido um módulo de treinamento utilizando algoritmos genéticos. O controle é otimizado através do ajuste automático das funções de pertinência existentes não sendo necessária a intervenção do usuário. Este módulo foi incorporado ao sistema que ganhou um novo menu, o menu Treinamento Genético, que será descrito em detalhes no próximo capítulo.



(a)



(b)

Figura 3.6 – Exemplos de controle.

Capítulo 4

Descrição do Módulo Treinamento Genético

4.1. Introdução

Este módulo tem a função de ajustar as funções de pertinência, utilizando algoritmos genéticos, de um controle previamente criado pelo usuário. Para isso no menu Treinamento Genético temos três opções: Posições Iniciais para o Treinamento, Treinamento Genético e Melhores Resultados, como mostrado na Figura 4.1

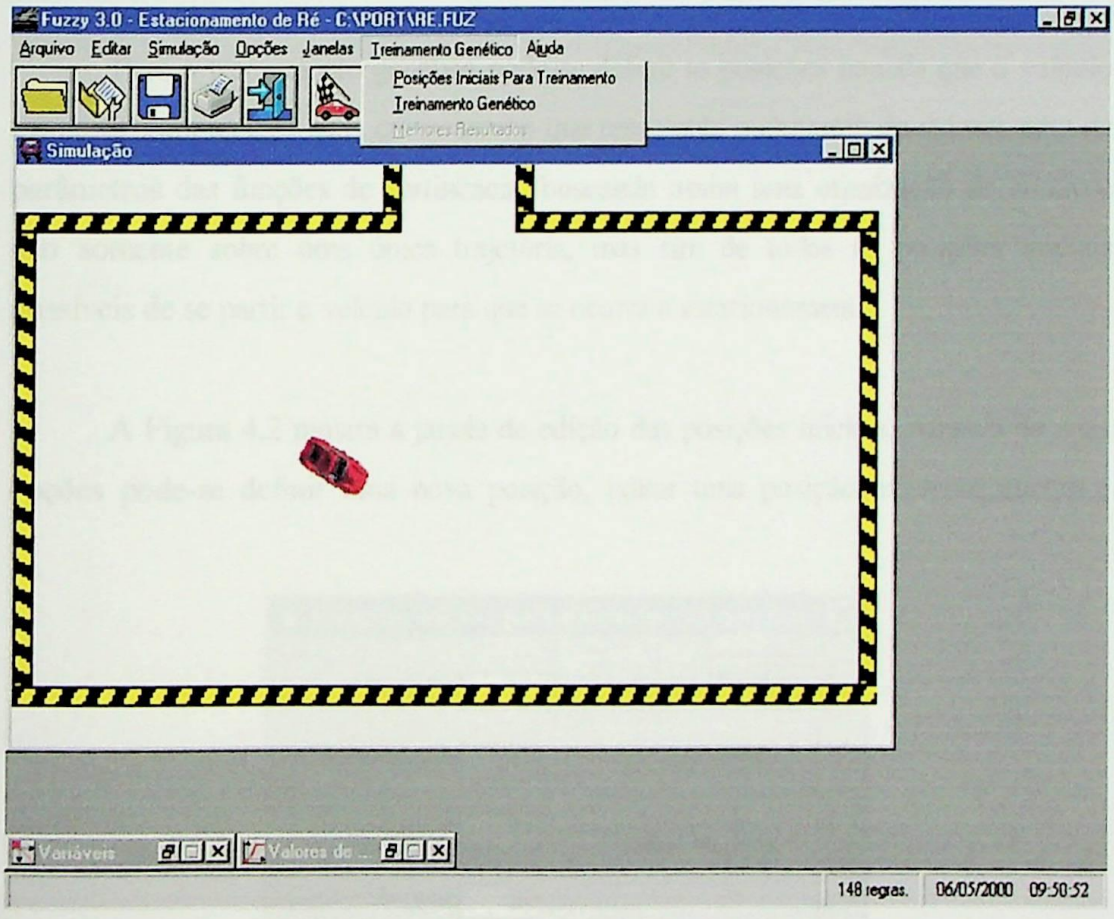


Figura 4.1 – Menu treinamento genético.

De forma geral, a integração dos algoritmos genéticos com o controle difuso foi implementada da seguinte maneira:

a) O cromossomo foi definido como sendo a concatenação dos valores de ajuste das

funções de pertinência.

b) Os parâmetros são os centros e as larguras de cada conjunto difuso. Estes parâmetros compõem os genes do cromossomo.
c) De uma gama inicial de valores de parâmetros possíveis, o sistema difuso é rodado para determinar o quanto ele funciona bem.
d) Essas informações são usadas para determinar o ajuste de cada cromossomo (adaptabilidade) e estabelecer assim uma nova população.
e) O ciclo é repetido até que se complete o número de gerações definidas pelo usuário. A cada geração é encontrado o melhor conjunto de valores para os parâmetros das funções de pertinência.

4.2. A Opção Definir Posições Iniciais

Para o treinamento genético pode-se definir as posições iniciais que o veículo irá partir para avaliar cada cromossomo que representa o conjunto de valores para os parâmetros das funções de pertinência, buscando assim uma otimização do controle não somente sobre uma única trajetória, mas sim de todas as posições iniciais possíveis de se partir o veículo para que se ocorra o estacionamento.

A Figura 4.2 mostra a janela de edição das posições iniciais. Através de suas opções pode-se definir uma nova posição, editar uma posição existente, excluir e

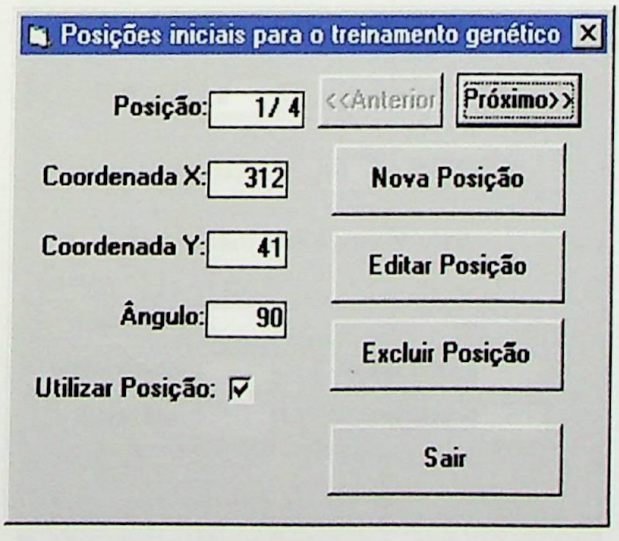


Figura 4.2 – Posições iniciais para treinamento genético

habilitar ou desabilitar uma posição para que esta seja ou não utilizada no treinamento genético. Isso é feito através da opção Utilizar Posição.

4.3. A Opção Treinamento Genético

Através desta opção o usuário faz o ajuste dos parâmetros para o treinamento genético.

Através da janela mostrada na Figura 4.3, o usuário pode além de definir os parâmetros (população, gerações, taxa de cruzamento e mutação), estabelecer o valor de ajuste para as funções de pertinência que é o quanto a função deslocará para esquerda ou direita e quanto ela se encolherá ou expandirá. Após iniciar o treinamento, pode-se acompanhar o treinamento genético a cada geração. Para cada cromossomo que constitui o conjunto de parâmetros de ajuste das funções de pertinência tem-se o número de iterações total geradas para estacionar o veículo partindo de todas as posições iniciais estabelecidas. Outra informação exibida por esta janela é a melhor geração, ou seja, em qual geração temos a melhor minimização de espaço percorrido pelo veículo até estacionar partindo de todas as posições iniciais estabelecidas.

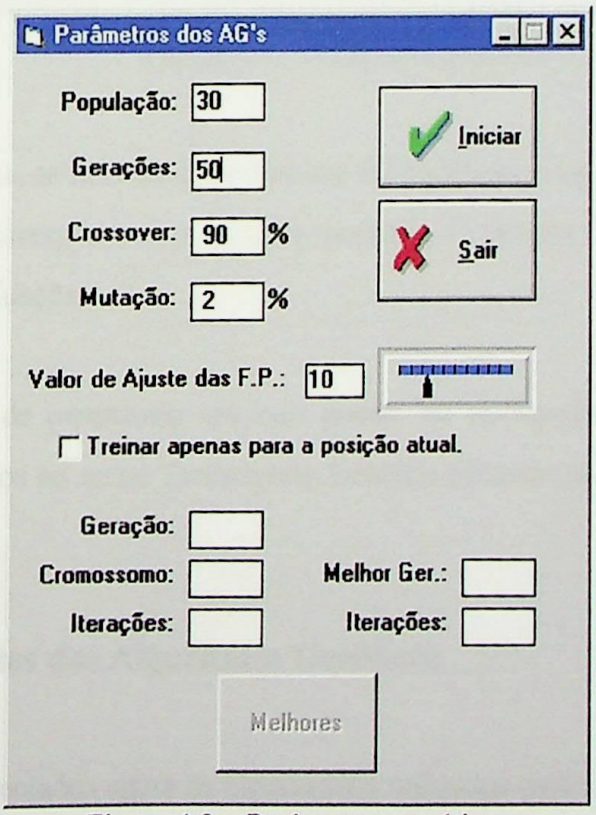


Figura 4.3 – Parâmetros genéticos

Concluídas todas as gerações, pode-se escolher o melhor resultado encontrado entre todas gerações através do botão Melhores. Esta operação permite avaliar as soluções propostas pelos algoritmos genéticos. Para isso o usuário deverá escolher uma geração e clicar no botão Ajustar como mostrado na Figura 4.4.

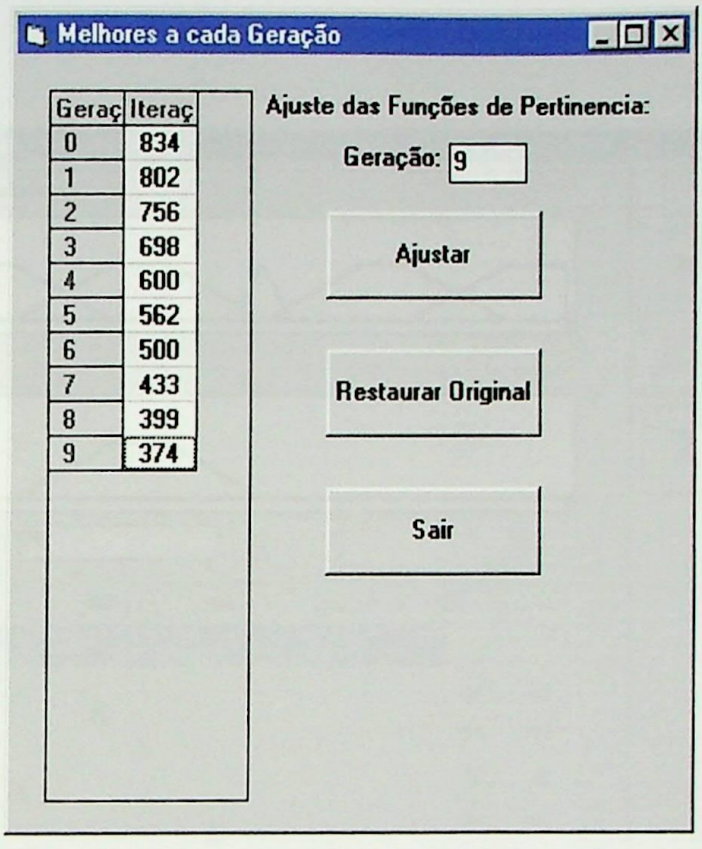


Figura 4.4 – Melhores resultados

Após o ajuste, as funções de pertinência são redefinidas segundo os parâmetros do cromossomo correspondente a geração escolhida. O sistema fará o controle com base nestas novas funções.

As funções de pertinência originais podem ser restauradas através da opção Melhores Resultados no menu Treinamento Genético clicando-se no botão Restaurar Original.

4.4. Componentes dos Algoritmos Genéticos

Serão apresentados agora os mecanismos utilizados para a geração de ótimas funções de pertinência no contexto do controlador difuso utilizado através dos

algoritmos genéticos.

Para descrever cada função de pertinência do controlador difuso implementado pelo pacote computacional, são definidos quatro parâmetros. São eles: *IE* (inferior esquerdo), *ID* (inferior direito), *SE* (superior esquerdo) e *SD* (superior direito).

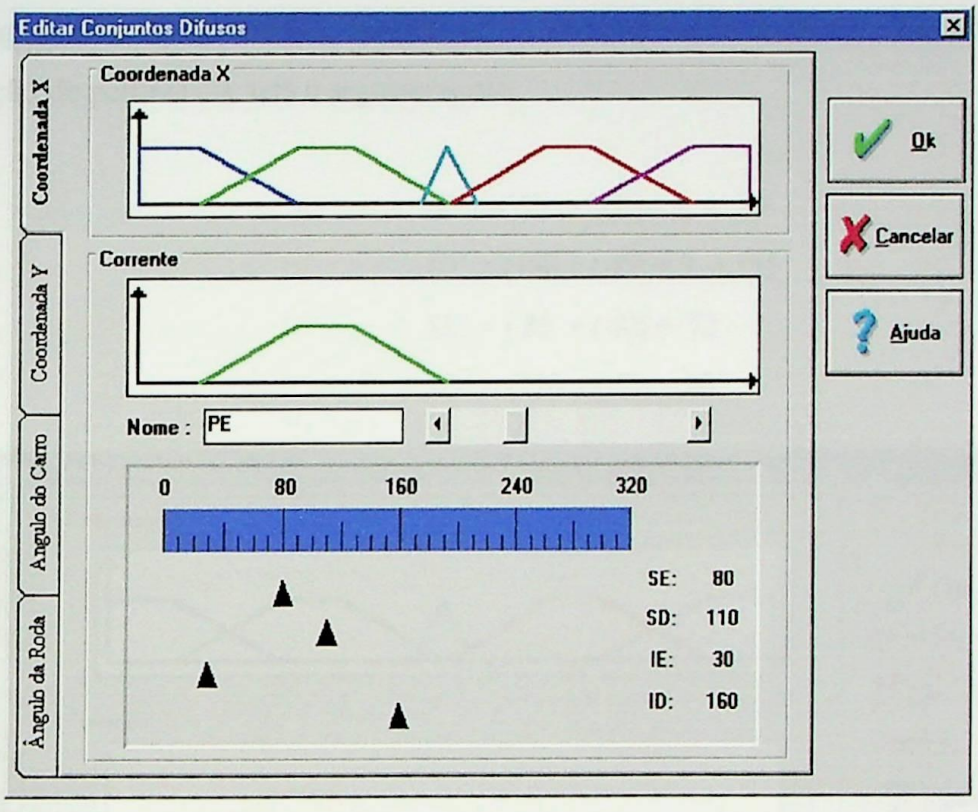


Figura 4.5 – Parâmetros das funções de pertinência.

Na Figura 4.5 estão sendo mostrados os parâmetros da função de pertinência *PE* da variável *x*. Para está função o valor de *IE* é igual a 30, *ID* igual a 160, *SE* igual a 80 e *SD* igual a 110.

Para o ajuste das funções de pertinência foram definidas para cada uma delas as seguintes equações:

$$IE = (IE + k_i) - w_i$$

$$ID = (ID + k_i) + w_i$$

$$SE = (SE + ki)$$

$$SD = (SD + k_i)$$

onde k_i e w_i são coeficientes da ajustes. O k_i faz cada função de pertinência mover-se para a direita ou para esquerda sem perder sua forma. O coeficiente w_i faz com que a função de pertinência encolha ou se expanda. Estes coeficientes assumem qualquer valor inteiro negativo ou positivo conforme o valor de ajuste definido pelo usuário na janela Parâmetros Genéticos como mostrado na Figura 4.3.

A Figura 4.6 mostra um exemplo de ajuste na função de *PE* com valores de *IE* igual a 30, *ID* igual a 160, *SE* igual a 80 e *SD* igual a 110. Sendo $k = -8$ e $w = 3$ a função de pertinência terá o seguinte ajuste:

$$IE' = (30 + (-8)) - 3 = 19$$
$$ID' = (160 + (-8)) + 3 = 155$$
$$SE' = (80 + (-8)) = 72$$
$$SD' = (110 + (-8)) = 102$$

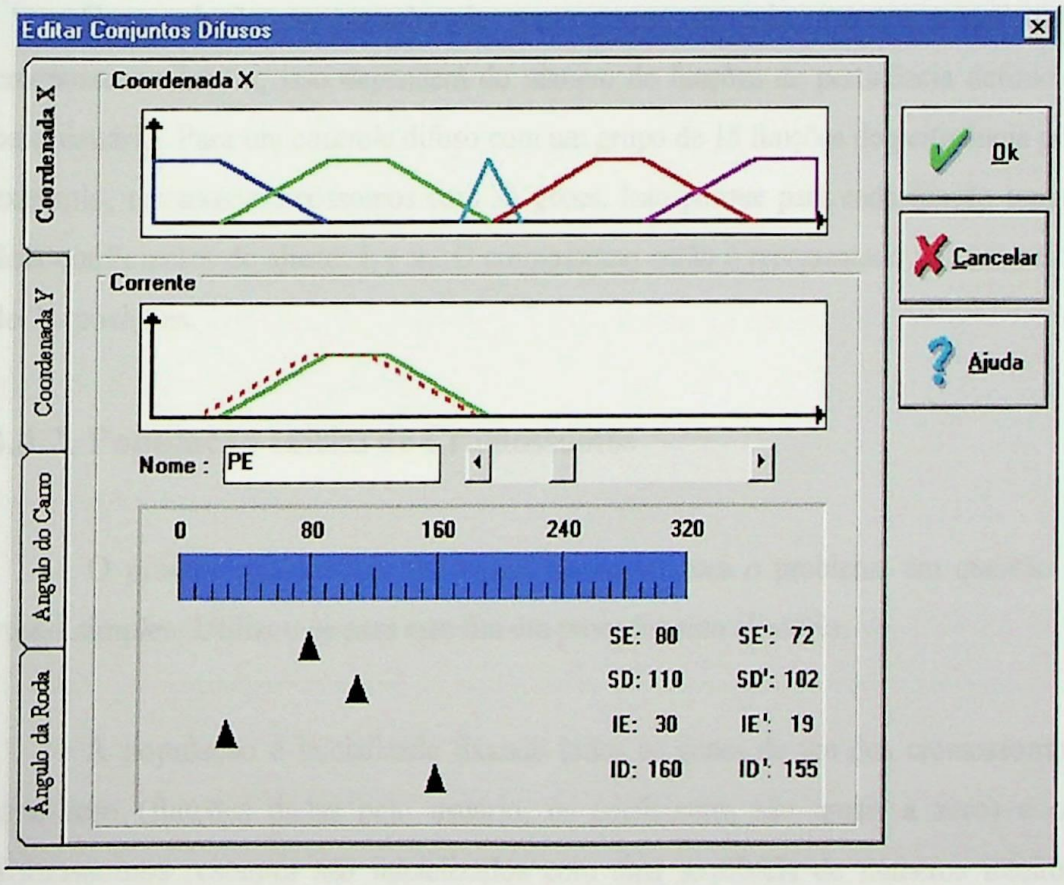


Figura 4.6 – Exemplo de ajuste de uma função de pertinência.

Os algoritmos genéticos são utilizados para achar os ótimos valores, segundo a estratégia e pontos iniciais utilizados, de k_i e w_i para as funções de pertinência.

4.4.1. Representação Genética das Soluções

Usualmente uma solução viável de um problema está associado a um cromossomo p na forma de um vetor com m posições $p=\{x_1,x_2,x_3,...,x_m\}$ onde cada componente x_i representa um gene.

Dentre os tipos de representação dos cromossomos, os mais conhecidos são: a representação binária e a representação por inteiros. A representação binária é a clássica, proposta por J. H. Holland [14].

Entretanto, neste trabalho utilizou-se a representação por inteiros uma vez que os genes de cada cromossomo são compostos pelos coeficientes de ajuste k_i e w_i que são valores inteiros.

Com relação ao tamanho do cromossomo, ou seja, quantos genes cada cromossomo irá ter, isso dependerá do número de funções de pertinência definidas pelo usuário. Para um controle difuso com um grupo de 18 funções de pertinência por exemplo, teremos cromossomos com 36 genes. Isso porque para cada função temos dois coeficientes de ajuste: k_i e w_i. O cromossomo então é representado por um vetor de 36 posições.

4.4.2. População Inicial de Cromossomos

O processo de geração da população inicial para o problema em questão é muito simples. Utilizou-se para este fim um procedimento aleatório.

A população é inicializada fixando todos os genes de um dos cromossomos para zero (funções dadas pelo usuário, os coeficientes são iguais a zero) e os cromossomos restantes são inicializados com uma seqüência de números inteiros positivos ou negativos de acordo o valor de ajuste definido pelo usuário. Por exemplo: Para um valor de ajuste igual a 10 e um número m de genes, podemos ter $p_1=\{-3, 0, 8, 2, -9,...,x_{m-1},x_m\}$ $p_2=\{-10, 1, 3, 7, 3,...,x_{m-1},x_m\}$ ... Os valores x_i podem ser qualquer inteiro no intervalo [–10, 10].

4.4.3. Função de Avaliação

Tal função tem o papel de avaliar o nível de aptidão (adaptação) de cada cromossomo gerado pelos algoritmos. Para o problema em questão o objetivo é minimizar o trajetória do veículo até estacionar. No caso, a função de avaliação é dada por:

$$FA = \frac{1}{1+I}$$

onde I é o total de iterações até estacionar com base no ajuste feito por cada cromossomo nas funções de pertinência.

De acordo com esta função, a aptidão de cada cromossomo será inversamente proporcional ao número de iterações.

4.4.4. Operadores Genéticos

Os operadores genéticos consistem no princípio da evolução natural e determinam a renovação dos cromossomos. Os operadores genéticos são necessários para que a população se diversifique e mantenha características de adaptação adquiridas pelas gerações anteriores.

4.4.4.1. Crossover

O processo de crossover ou recombinação envolve um corte aleatório que será efetuado nos cromossomos pais de quem então os genes serão trocados gerando dois descendentes. Considere por exemplo, dois cromossomos pais p_1 e p_2. Um ponto de cruzamento é escolhido aleatoriamente. As informações anteriores a este ponto em um dos pais são ligadas às informações posteriores a este ponto no outro pai.

4.4.4.2. Mutação

O processo de mutação consiste na realização de alterações de acordo com o valor de ajuste definido pelo usuário, nos valores de um ou mais genes do

cromossomo. Para um valor de ajuste igual a 10 por exemplo, a alteração de um gene selecionado será para algum valor no intervalo [-10, 10].

4.4.5. Critério de Renovação e Seleção

O critério de renovação e seleção implementado neste trabalho pelos algoritmos foi o da reprodução. A reprodução é um processo no qual são copiados cromossomos para a próxima geração de acordo com os valores da função de avaliação. Cromossomos com alto valor de aptidão contribuirão com um ou mais descendentes exatamente iguais para a próxima geração.

Considere o exemplo da Tabela 4.1 que mostra uma população $N = 4$ com o valor da função de avaliação de cada cromossomo $f_1 = f_2 = 16$, $f_3 = 48$ e $f_4 = 80$ na população corrente. Como $\Sigma f_i = 160$, a aptidão parcial de cada cromossomo é igual a 16/160 = 0,1 (10%), 16/160 = 0,10 (10%), 48/160 = 0,30 (30%) e 80/160 = 0,50 (50%) respectivamente. A partir da aptidão parcial é calculado o número de descendentes esperados de cada cromossomo na próxima geração. No exemplo, para cada cromossomo espera-se 0,1 x 4 = 0,4, 0,1 x 4 = 0,4, 0,3 x 4 = 1,2 e 0,5 x 0,4 =2 respectivamente.

O número de cromossomos efetivamente reproduzidos na próxima geração é dado pela parte inteira do número de descendentes esperados de cada cromossomo. Então para o exemplo temos uma reprodução de x_3 e duas reproduções de x_4. A seleção de mais um cromossomo para a reprodução para completar a população de 4 cromossomos, é feita através da seleção proporcional ao ajustamento.

Tabela 4.1 – Critério de seleção.

População	fi(x) Avaliação	Aptidão Parcial (%)	Descendentes Esperados	Cromossomos Reproduzidos
x_1	16	10	0,40	0
x_2	16	10	0,40	0
x_3	48	30	1,20	1
x_4	80	50	2,00	2
Soma	160	100	4	3

Este processo foi implementado através da técnica da roleta onde cromossomos que apresentam maior adaptação possuem maiores probabilidades de serem selecionados.

Sendo Σf_i=160 a roleta para este exemplo é representada pela Figura 4.7.

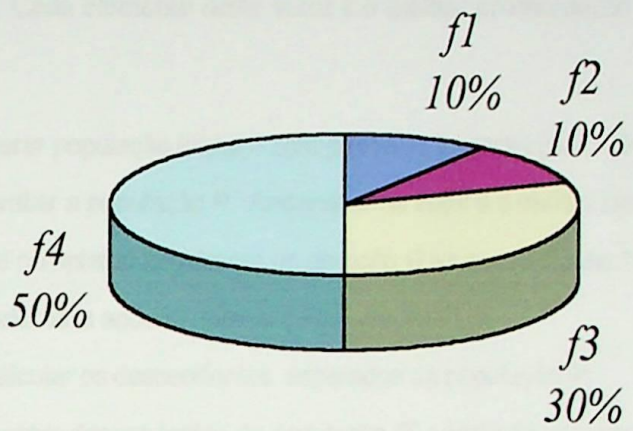


Figura 4.7 – Representação da roleta.

. Na prática a roleta é representada por um vetor v de M elementos (ordenados de $\{1,..., N\}$ e seja um índice aleatório r, r = 1,... ,M. Então, $v(r)$ corresponde a que cromossomo i foi selecionado. Por exemplo: Com M = 10, para o exemplo anterior tem-se, v = {1, 2, 3, 3, 3, 4, 4, 4, 4, 4}. Se r = 4, então o cromossomo selecionado é o de número 3.

Finalmente, a roleta é girada um determinado número de vezes, dependendo do número de cromossomos que faltam para completar a população, e são escolhidos, como cromossomos que participarão da próxima geração, aqueles sorteados na roleta.

4.4.6. Critério de Parada

O critério de parada utilizado foi o da definição do número máximo de gerações a serem produzidas. Quando o número de gerações é completado pelos algoritmos genéticos, o processo de geração de novas populações é terminado e a melhor solução é aquela dentre os indivíduos que mais se adaptam à função de avaliação.

4.5. Apresentação do Algoritmo

Seja G o número de gerações, P a população, Pc a taxa de crossover (recombinação), Pm a taxa de mutação e VA o valor de ajuste permitido para as funções de pertinência, o algoritmo abaixo apresentado gera como saída o vetor s com G posições. Cada elemento deste vetor é o melhor cromossomo de uma geração.

Passo 1. Gerar população inicial P com genes no intervalo [$-VA$, $+VA$].

Passo 2. Avaliar a população P. Armazenar no vetor s o melhor cromossomo.

Passo 3. Se completou no número de geração G vá para o Passo 13.

Passo 4. Calcular a aptidão relativa da população P.

Passo 5. Calcular os descendentes esperados da população P.

Passo 6. Sortear descendentes da população P' a partir da população P.

Passo 7. Montar o agrupamento da população P'

Passo 8. Sortear ponto de cruzamento para os cromossomos pais da população P'.

Passo 9. Efetuar os cruzamentos para a população P' segundo Pc.

Passo 10. Efetuar a mutação em cada cromossomo segundo Pm.

Passo 11. Avaliar a população P'. Armazenar no vetor s o melhor cromossomo.
Fazer $P = P'$.

Passo 12. Retornar ao Passo 3.

Passo 13. Fim

4.6. Acompanhamento do Algoritmo

Nesta seção será feito um acompanhamento passo-a-passo do algoritmo apresentado na Seção 4.5. A Figura 4.8 mostra a trajetória feita pelo veículo que parte da coordenada X igual a 40, Y igual a 183 e ângulo do carro igual a –46 gerando 558 iterações até estacionar. O algoritmo será utilizado para minimizar o número de iterações e, consequentemente, o espaço percorrido. Para este acompanhamento definimos que o algoritmo terá $G = 5$ gerações e uma população de 4 cromossomos. Taxa de crossover $Pc = 90$ % e taxa de mutação $Pm = 2$ %. A Figura 4.9 mostra as funções de pertinência que serão ajustadas.

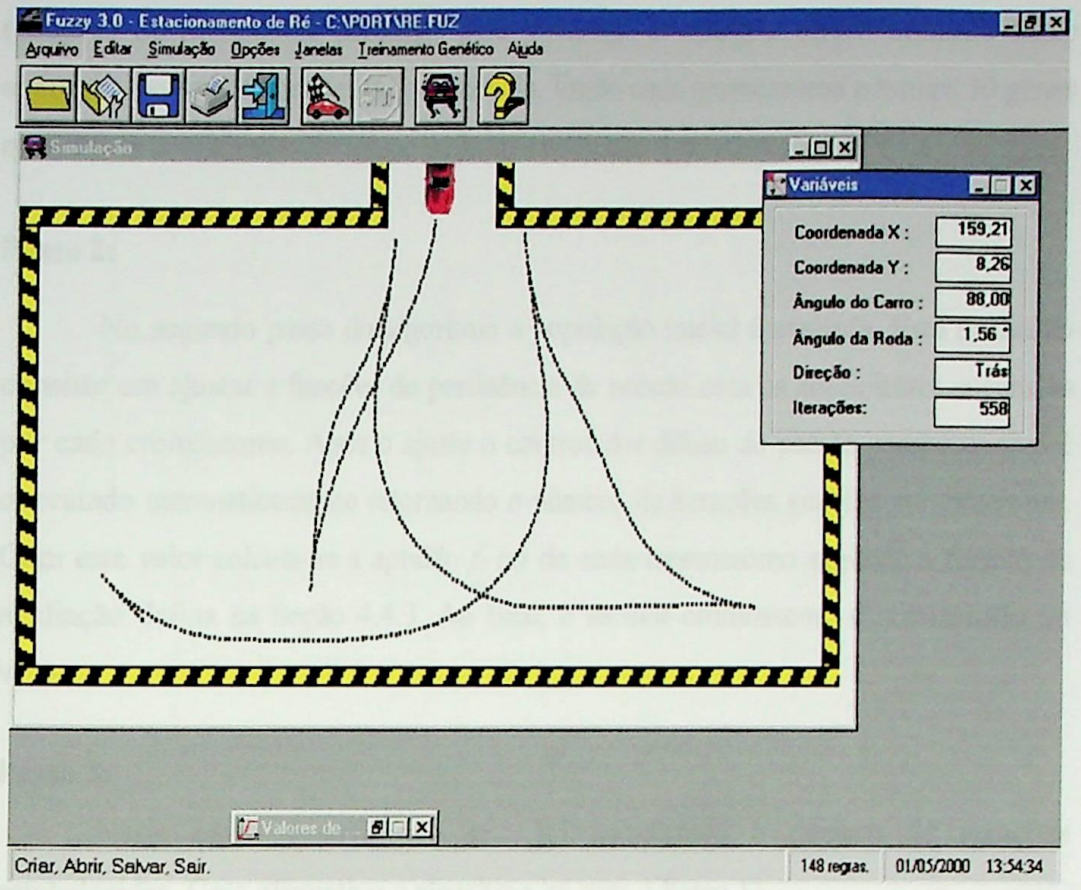


Figura 4.8 – Simulação sem treinamento genético.

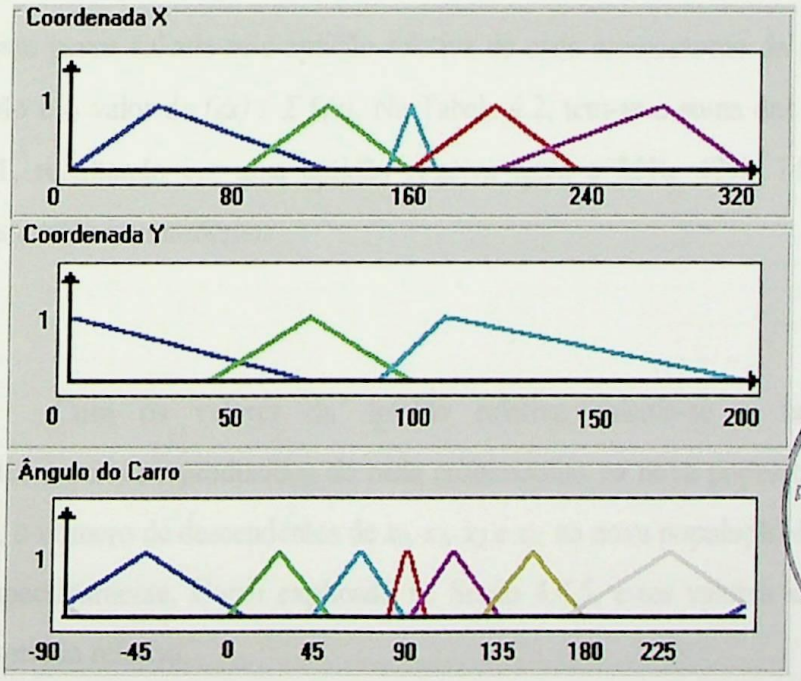


Figura 4.9 – Funções de pertinência sem ajuste.

Passo 1:

O primeiro passo do algoritmo gera a população inicial *P* com genes no intervalo [-5, +5] lembrando que um dos cromossomos tem todos os genes iguais a 0

(funções de pertinência definidas pelo usuário). O controle difuso utilizado neste exemplo, possui 15 funções de pertinência. Então cada cromossomo possuirá 30 genes que são os coeficientes de ajuste definidos na Seção 4.4.

Passo 2:

No segundo passo do algoritmo a população inicial é avaliada. Esta avaliação consiste em ajustar a funções de pertinência de acordo com os coeficientes sugeridos por cada cromossomo. Após o ajuste o controlador difuso do pacote computacional é executado automaticamente retornando o número de iterações geradas até estacionar. Com este valor calcula-se a aptidão $f_i(x)$ de cada cromossomo segundo a função de avaliação defina na Seção 4.4.3. Ao final, o melhor cromossomo é armazenado no vetor *s*.

Passo 3:

Neste passo é verificado se foi completado o número de gerações estabelecidas. Caso afirmativo o algoritmo irá para o Passo 13.

Passo 4:

Neste passo calcula-se a aptidão relativa de cada cromossomo da população. Esta aptidão é o valor de $f_i(x) / \Sigma f_i(x)$. Na Tabela 4.2, tem-se a soma das avaliações igual 711, resultando em uma aptidão relativa igual a 25%, 47%, 14% e 14% respectivas a cada cromossomo.

Passo 5:

Com os valores da aptidão relativa calcula-se o número de descendentes a serem reproduzidos de cada cromossomo na nova população *P'*. Na Tabela 4.2, o número de descendentes de x_0, x_1, x_2 e x_3 na nova população é igual a 1, 1, 0, 0 respectivamente. Como explicado na Seção 4.4.5, estes valores são a parte inteira da aptidão relativa.

Passo 6:

Através da roleta, serão sorteados os cromossomos necessários para se completar a nova população *P'*. Na Tabela 4.2 e 4.3 notamos que dois cromossomos da nova população são sorteados pela roleta. Já na tabela 4.4, 3 cromossomos são

sorteados pela roleta. Os cromossomos sorteados são reproduzidos na nova população *P'*.

Tabela 4.2 – Primeira geração.

População Inicial	Iterações	Avaliação	Aptidão Parcial (%)	Descendentes Esperados	Seleção	
					Reprodução	Roleta
x_0	557	179	25	1,00	1	0
x_1	300	332	47	1,88	1	0
x_2	999	100	14	0,56	0	1
x_3	999	100	14	0,56	0	1
Soma	2855	711	100	4	2	2
Média	713.75	177,75	25			
Max	999	332	47			

População P'	Emparelhamento	Ponto de Cruzamento	Genes selecionados para mutação
x_0	1	6	1 – 12 – 18
x_1	0	6	10 – 13
x_2	3	20	12
x_3	2	20	19

Passo 7:

Neste é feito o agrupamento para o crossover onde os cromossomos pais são emparelhados através de sorteio. Na Tabela 4.2 o emparelhamento ocorre da seguinte forma: x_0 com x_1, x_2 com x_3.

Tabela 4.3 – Segunda geração.

Nova População	Iterações	Avaliação	Aptidão Parcial (%)	Descendentes Esperados	Seleção	
					Reprodução	Roleta
x_0	999	100	14	0,56	0	0
x_1	303	329	46	1,84	1	1
x_2	551	181	26	1,04	1	1
x_3	999	100	14	0,56	0	0
Soma	2852	710	100	4	2	2
Média	713	177.5				
Max	999	329				

População P'	Emparelhamento	Ponto de Cruzamento	Genes selecionados para mutação
x_1	3	4	-
x_2	2	2	-
x_1	1	2	-
x_2	0	4	-

Passo 8:

Neste passo um ponto de cruzamento é escolhido aleatoriamente para cada par de cromossomos emparelhados como mostrado nas Tabelas 4.2, 4.3, 4.4 e 4.5.

Passo 9:

Neste passo é efetuado o cruzamento entre os cromossomos pais segundo a probabilidade de cruzamento estabelecida pelo usuário em *Pc*. De acordo com o ponto de cruzamento definido no passo anterior os genes anteriores a este ponto em um dos pais são ligados aos genes posteriores a este ponto no outro pai gerando assim dois novos descendentes.

Passo 10:

A mutação consiste em selecionar cada gene e alterá-lo para um valor do intervalo [-5, +5] segundo a probabilidade de mutação estabelecida em *Pm*. Nas Tabelas 4.2, 4.3, 4.4 e 4.5 verifica-se uma baixa ocorrência de mutação em função da baixa probabilidade estabelecida neste exemplo (*Pm=2%*). Este valor evita que o processo se torne essencialmente aleatório.

Passo 11:

Neste passo a nova população *P'* é avaliada. A população *P* passa a ser igual a *P'*.

Tabela 4.4 – Terceira geração.

Nova População	Iterações	Avaliação	Aptidão Parcial (%)	Descendentes Esperados	Seleção	
					Reprodução	Roleta
x_0	551	181	19	0,76	0	0
x_1	299	333	36	1,44	1	1
x_2	489	204	22	0,88	0	2
x_3	466	214	23	0,92	0	0
Soma	1805	932	100	4	1	3
Média	451	233				
Max	489	333				

População P'	Emparelhamento	Ponto de Cruzamento	Genes selecionados para mutação.
x_1	3	23	-
x_2	2	28	2 – 27
x_1	1	28	-
x_2	0	23	-

Passo 12:

Retornar ao Passo 3.

Passo 13:

Neste passo o algoritmo se encerra. Na Tabela 4.5 é mostrado a avaliação da última geração e o número de iterações geradas. Os melhores cromossomos de cada geração são listados para que o usuário escolha a geração com o menor número de iterações.

Tabela 4.5 – Última geração.

Nova População	Iterações	Avaliação	Aptidão Parcial (%)	Descendentes Esperados	Seleção	
					Reprodução	Roleta
x_0	545	183	22	0,88	0	1
x_1	999	100	12	0,48	0	0
x_2	489	204	25	1,00	1	0
x_3	299	333	41	1,64	1	1
Soma	2332	820	100	4	2	2
Média	583	205				
Max	999	333				

População P'	Emparelhamento	Ponto de Cruzamento	Genes selecionados para mutação.	Iterações	Avaliação da última geração
x_2	2	22	-	550	181
x_3	3	10	1 – 22	303	329
x_3	0	22	16	337	265
x_0	1	10	-	299	333
Soma				1489	1108
Média				372	277
Max				550	333

Após a escolha do melhor cromossomo entre todas as gerações será feito o ajuste das funções de pertinência do controlador difuso utilizado neste exemplo. Na Figura 4.10 são mostrados os melhores resultados. Na Figura 4.11 temos a simulação feita após o ajuste. A Figura 4.12 mostra as funções de pertinência após o ajuste feito nas funções e implicação das variáveis x, y e ângulo do carro.

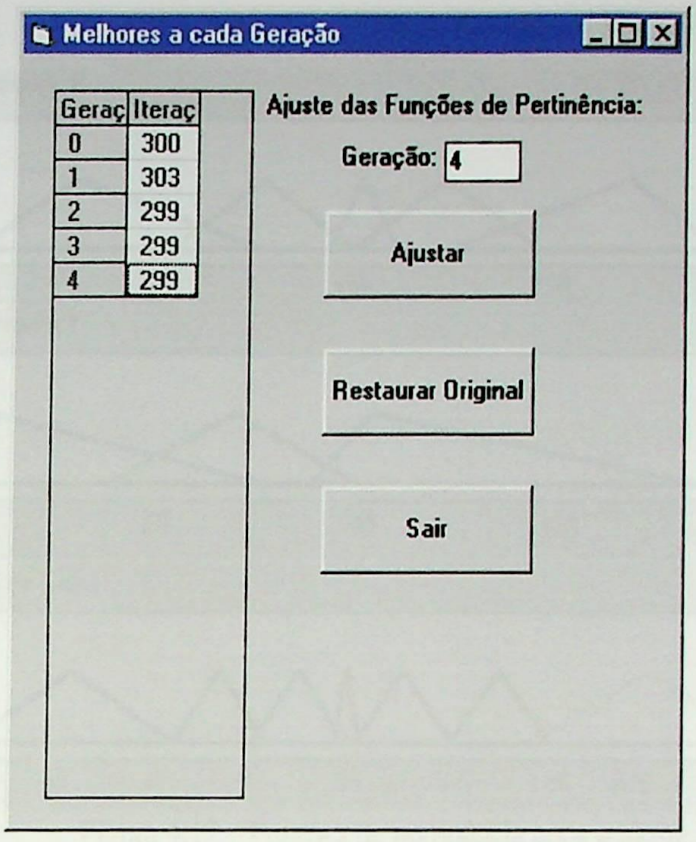


Figura 4.10 – Melhores resultados do treinamento genético.

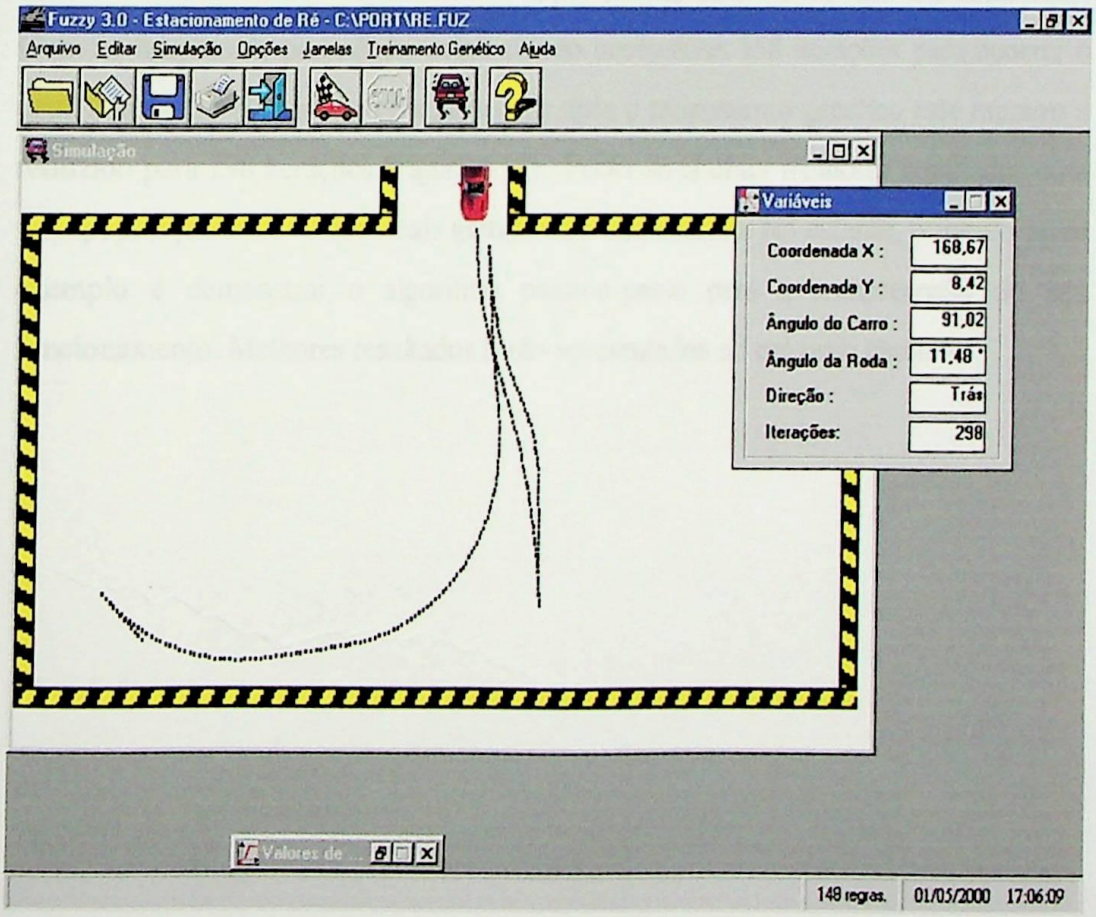


Figura 4.11 – Simulação após treinamento genético.

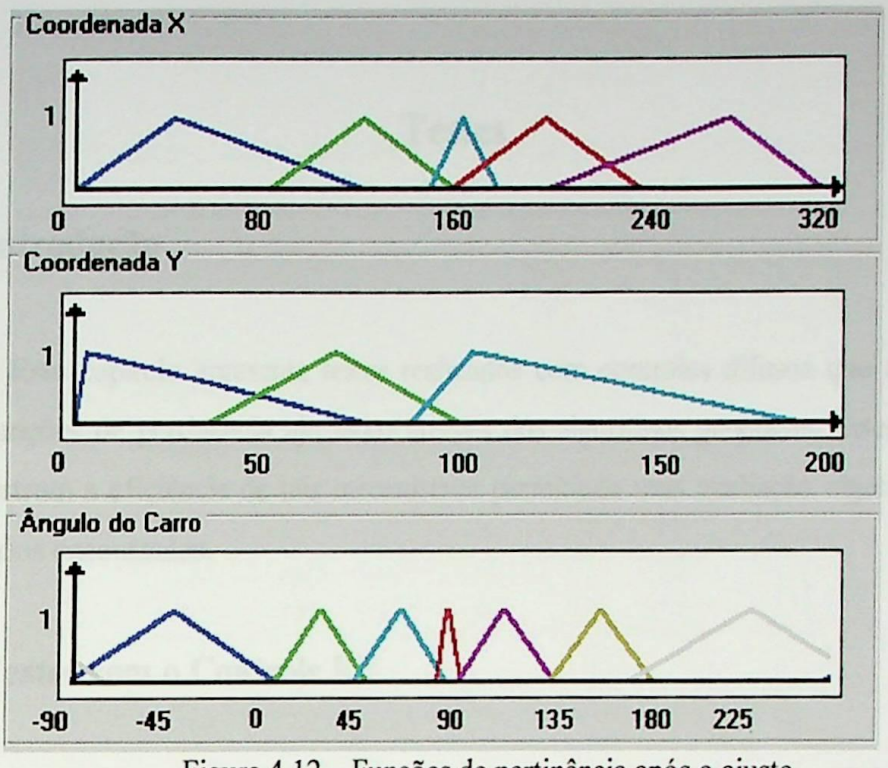


Figura 4.12 – Funções de pertinência após o ajuste.

Nota-se uma sensível redução na trajetória do veículo até estacionar. Com as funções de pertinência originais são necessárias 558 iterações para ocorrer o estacionamento (Figura 4.8) enquanto que após o treinamento genético este número é reduzido para 298 iterações (Figura 4.11). Poder-se-ia obter melhores resultados com uma população maior e com mais gerações no treinamento. No entanto, o intuito neste exemplo é demonstrar o algoritmo passo-a-passo para a compreensão do seu funcionamento. Melhores resultados serão apresentados no próximo capítulo.

Capítulo 5

Testes

5.1. Introdução

Este capítulo apresenta testes realizados com controles difusos que tiveram suas funções de pertinência ajustadas através dos algoritmos genéticos. Estes testes demonstram a eficiência de tais mecanismos permitindo uma avaliação objetiva dos resultados encontrados.

5.2. Testes com o Controle RE

Esta seção apresenta dois exemplos de ajustes nas funções de pertinência do controle RE (controle difuso previamente criado) gerando dois novos controles: O GEN1 e GEN2. Estes controles possuem 148 regras e 15 funções de pertinência para as variáveis de entrada x, y e ângulo da roda. As funções de pertinência originais são mostradas na Figura 5.1.

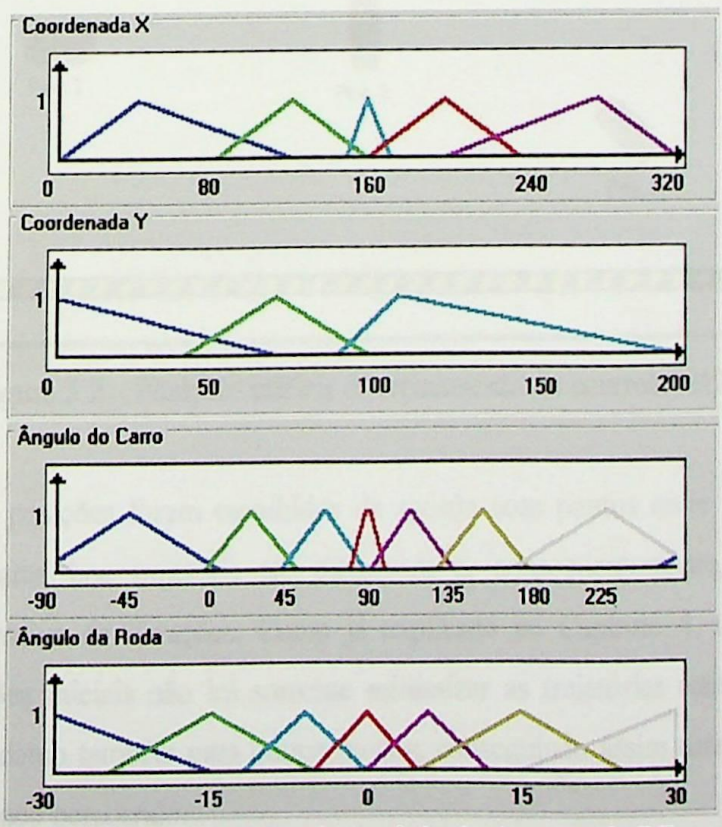


Figura 5.1 – Funções de pertinência do controle RE.

5.2.1. O controle GEN1

O treinamento deste controle foi feito a partir de três posições iniciais conforme mostra a Tabela 5.1. Nesta tabela temos também o número de iterações geradas pelo veículo até estacionar utilizando as funções de pertinência originais.

Tabela 5.1 – Posições iniciais para o controle GEN1.

Posição	X	Y	Ângulo do Carro	Iterações sem treinamento
1	25	120	180	330
2	160	130	-90	888
3	275	160	-40	655

A Figura 5.2 mostra o veículo em cada uma das posições iniciais.

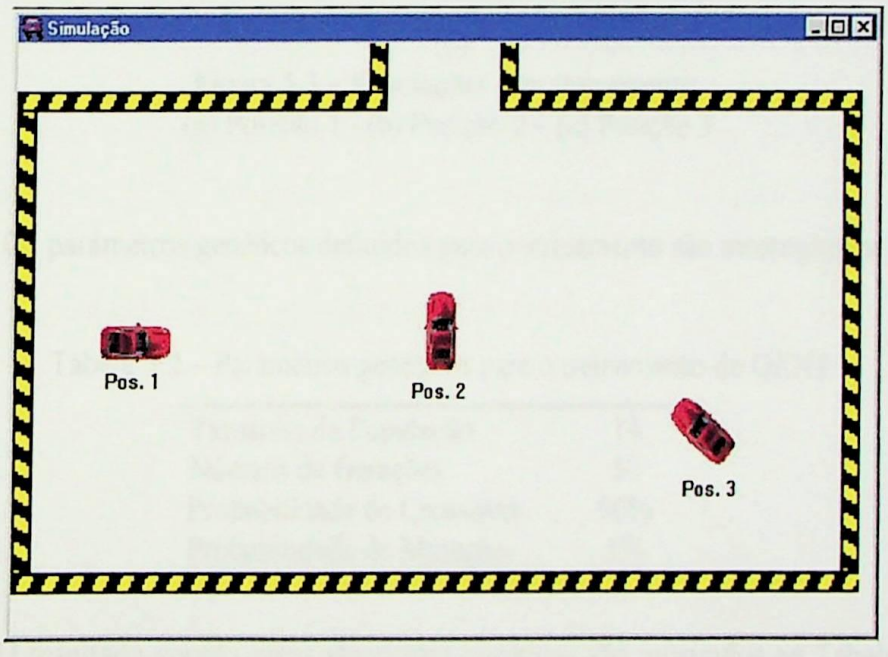
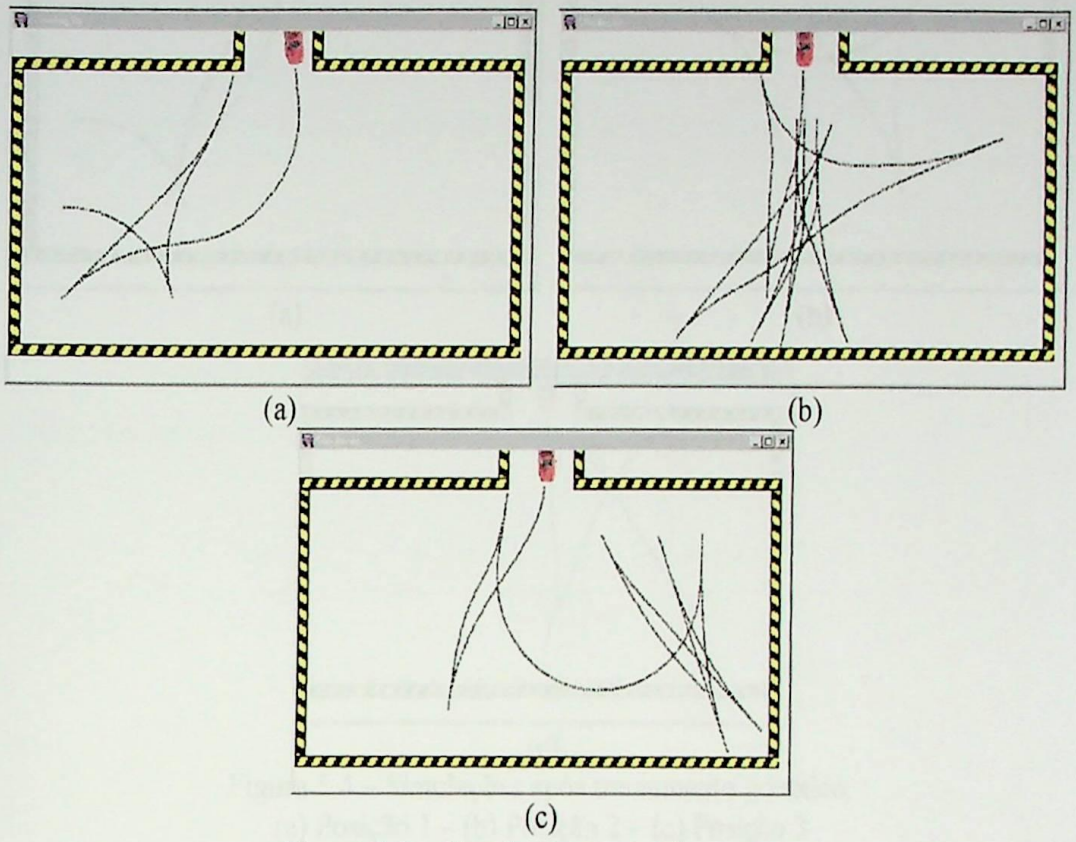


Figura 5.2 – Posições iniciais de treinamento do controle GEN1.

Estas posições foram escolhidas de acordo com pontos onde o veículo não desenvolve uma boa trajetória até estacionar e, consequentemente, gerando um número excessivo de iterações. Como já explicado no Capítulo 4, a definição de várias posições iniciais não irá somente minimizar as trajetórias referentes a estes pontos, mas como também para outros pontos, conseguindo assim uma minimização global de espaço percorrido.

A Figura 5.3 mostra as trajetórias referentes a cada posição inicial.



(a) (b)

(c)

Figura 5.3 – Simulações sem treinamento.
(a) Posição 1 - (b) Posição 2 – (c) Posição 3

Os parâmetros genéticos definidos para o treinamento são mostrados na Tabela 5.2.

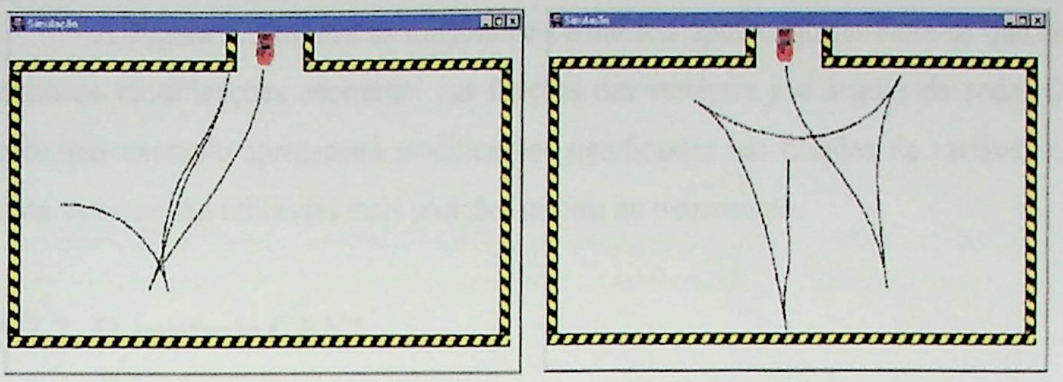
Tabela 5.2 – Parâmetros genéticos para o treinamento de GEN1.

Tamanho da População	14
Número de Gerações	30
Probabilidade de Crossover	90%
Probabilidade de Mutação	1%

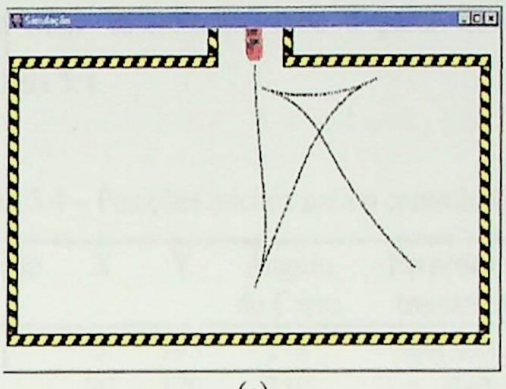
O resultado gerado pelos algoritmos genéticos são mostrados na Tabela 5.3 e Figura 5.4.

Tabela 5.3 – Iterações após o treinamento genético de GEN1.

Posição	Iterações sem treinamento	Iterações com treinamento
1	330	280
2	888	384
3	655	277
Total	1873	941
Média	624,33	331.67



(a) (b)



(c)

Figura 5.4 – Simulações após treinamento genético.
(a) Posição 1 – (b) Posição 2 – (c) Posição 3

Como mostrado na Tabela 5.3, obteve-se uma redução de 932 iterações (49,75%) para o veículo estacionar partindo-se das posições inicias fixadas para o treinamento. Na Seção 5.4 serão apresentados os resultados de simulações feitas partindo de posições iniciais não utilizadas no treinamento.

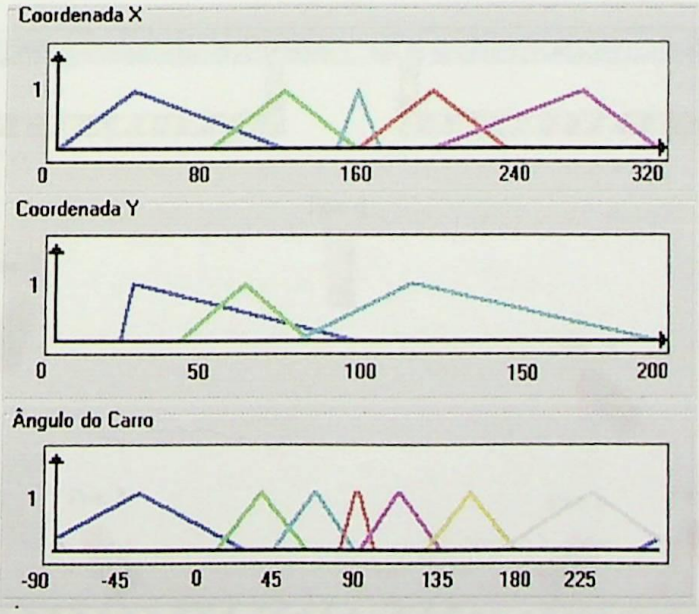


Figura 5.5 – Funções de pertinência do controle GEN1.

A Figura 5.5 mostra as funções de pertinência após o ajuste. Nota-se que as maiores modificações ocorreram nas funções das variéveis y e ângulo da roda. O próximo exemplo apresentará modificações significantes nas funções da variável x, uma vez que são utilizadas mais posições iniciais no treinamento.

5.2.2. O controle GEN2

O treinamento deste controle foi feito a partir de sete posições iniciais conforme mostra a Tabela 5.4.

Tabela 5.4 – Posições iniciais para o controle GEN2.

Posição	X	Y	Ângulo do Carro	Iterações sem treinamento
1	5	195	212	Sem solução
2	30	120	250	449
3	80	195	-50	328
4	160	100	-90	873
5	240	194	220	320
6	275	140	-38	746
7	313	194	-56	496

A Figura 5.6 mostra o veículo em cada uma das posições iniciais.

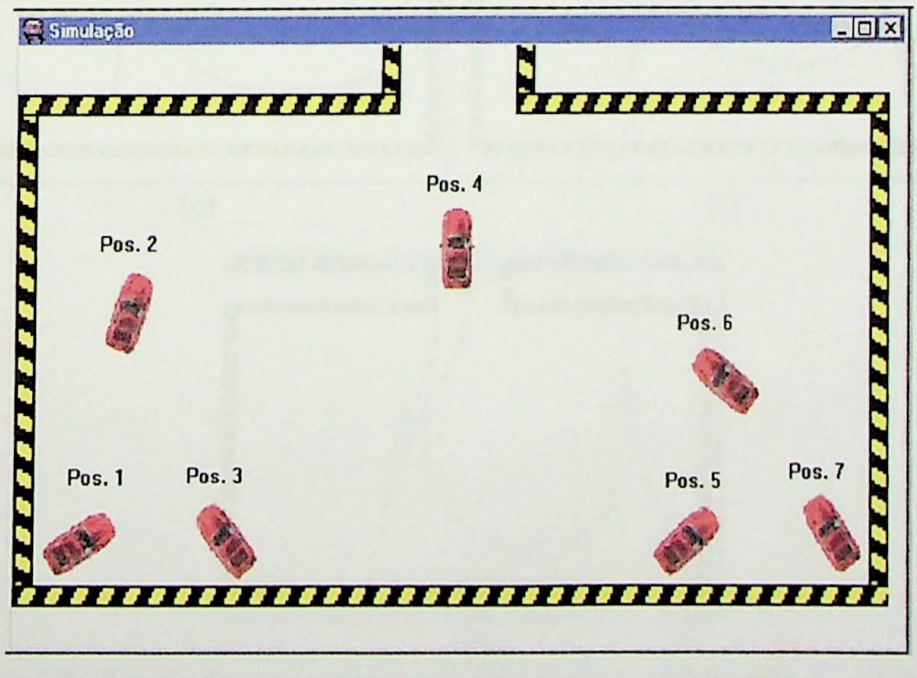


Figura 5.6 – Posições iniciais do treinamento de GEN2.

A Figura 5.7 mostra as trajetórias referentes a cada posição inicial. A Figura 5.7 (a) mostra uma posição sem solução considerada pelo algoritmo genético como uma posição de 1000 iterações.

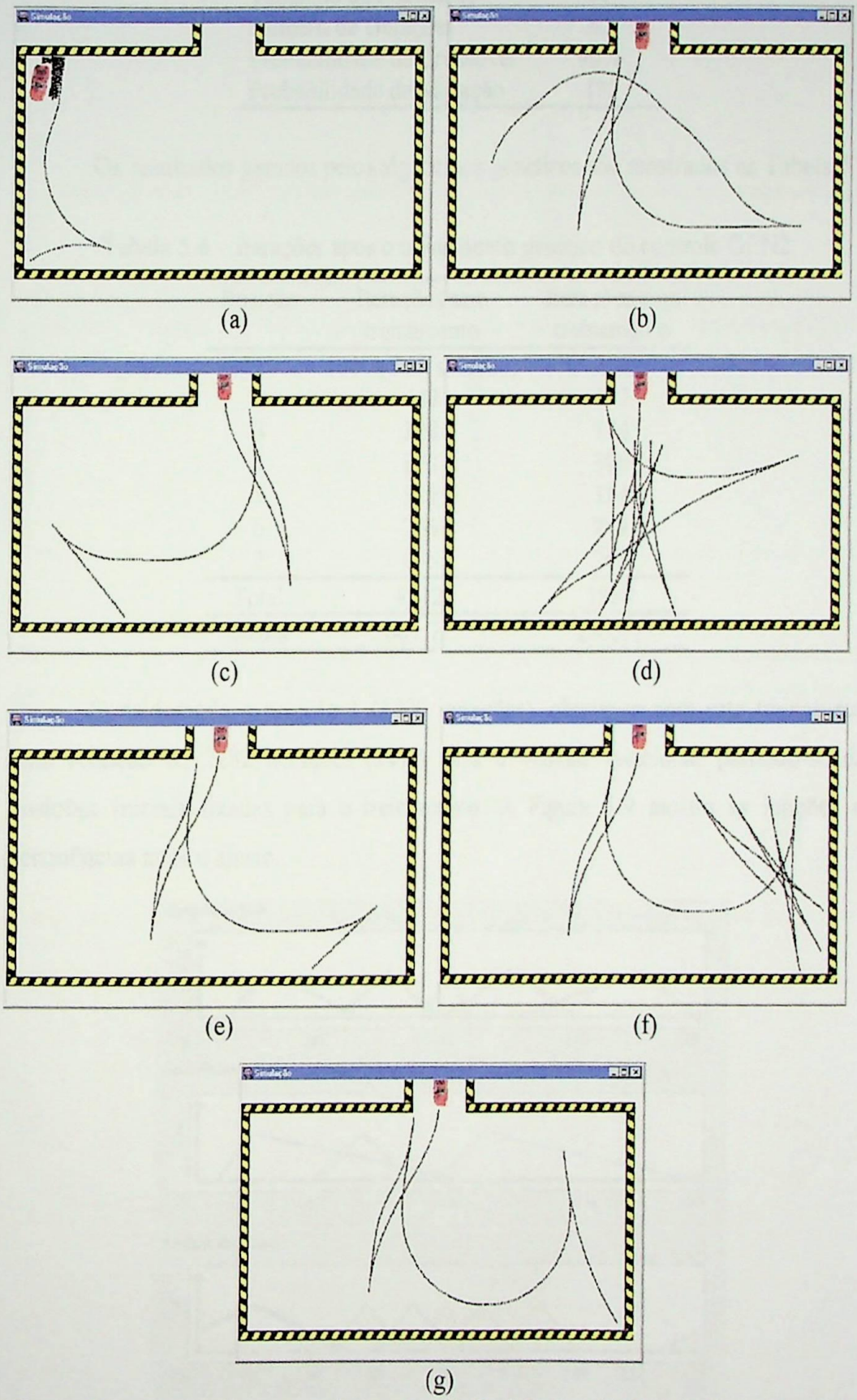


Figura 5.7 - Simulações sem treinamento. (a) Posição 1 – (b) Posição 2 – (c) Posição 3 – (d) Posição 4 – (e) Posição 5 – (f) Posição 6 – (g) Posição 7

Os parâmetros genéticos definidos para o treinamento são mostrados na Tabela 5.5.

Tabela 5.5 – Parâmetros Genéticos para o controle GEN2.

Tamanho da População	30
Número de Gerações	80
Probabilidade de Crossover	90%
Probabilidade de Mutação	1%

Os resultados gerados pelos algoritmos genéticos são mostrados na Tabela 5.6

Tabela 5.6 – Iterações após o treinamento genético do controle GEN2.

Posição	Iterações sem treinamento	Iterações com treinamento
1	1000 (Sem solução)	252
2	449	197
3	328	174
4	873	396
5	320	314
6	746	249
7	496	348
Total	4212	1930
Média	601,71	253,71

Considerando a posição 1 (1000 iterações), obteve-se com este treinamento uma redução de 2282 iterações (54%) para o veículo estacionar partindo-se das posições iniciais fixadas para o treinamento. A Figura 5.9 mostra as funções de pertinências após o ajuste.

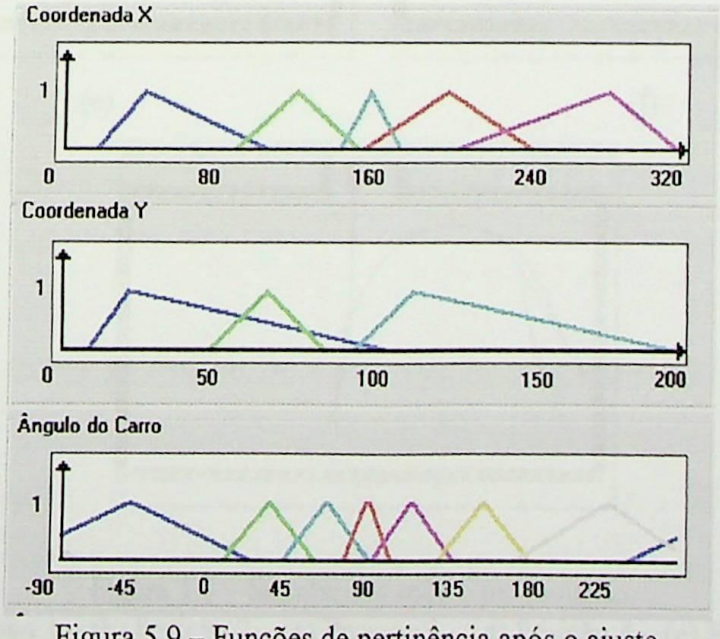
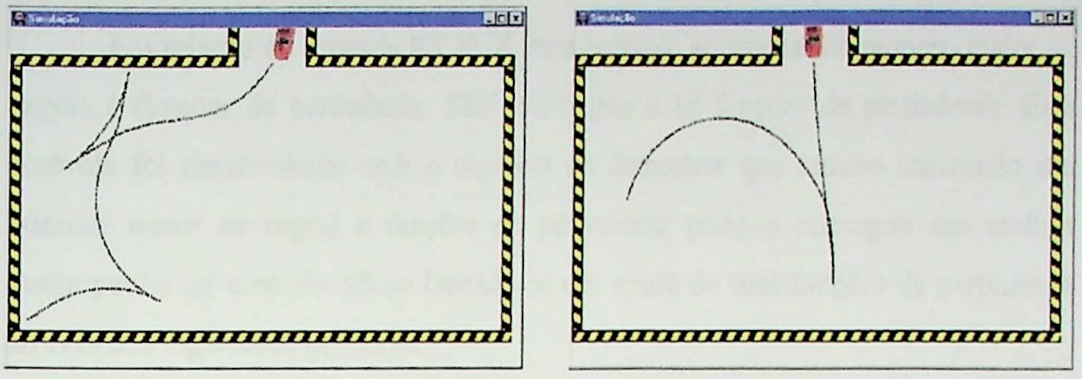
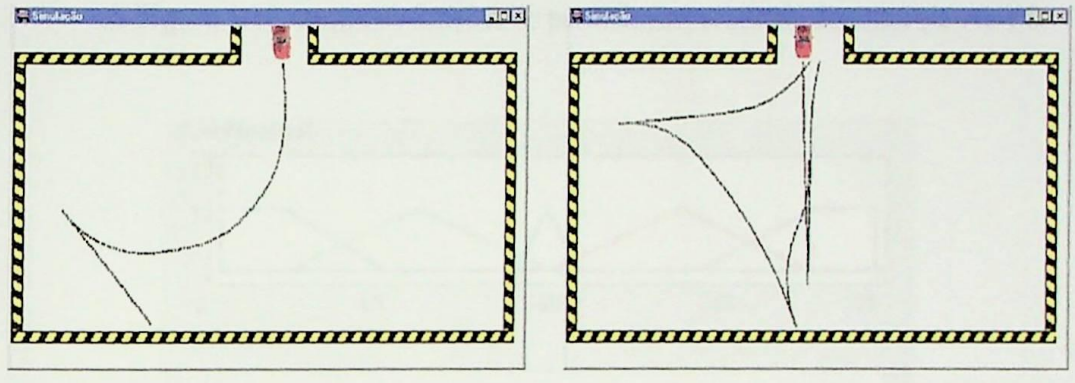


Figura 5.9 – Funções de pertinência após o ajuste.

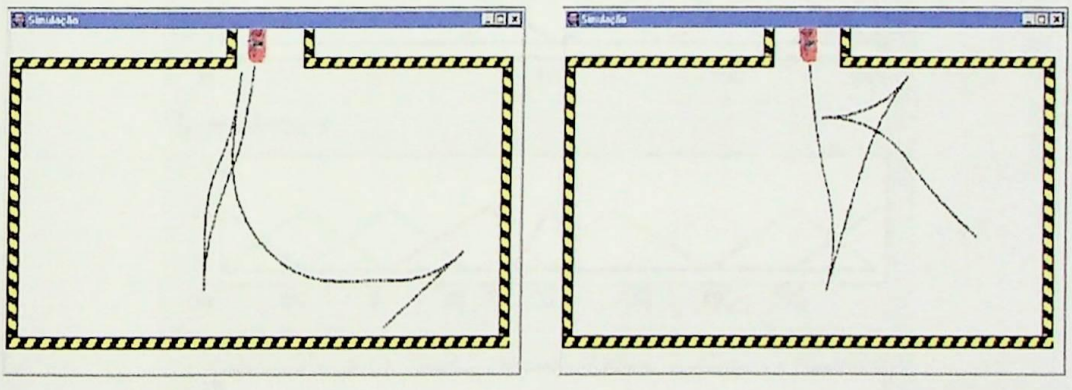
A Figura 5.8 mostra as trajetórias feitas pelo veículo após o treinamento genético.



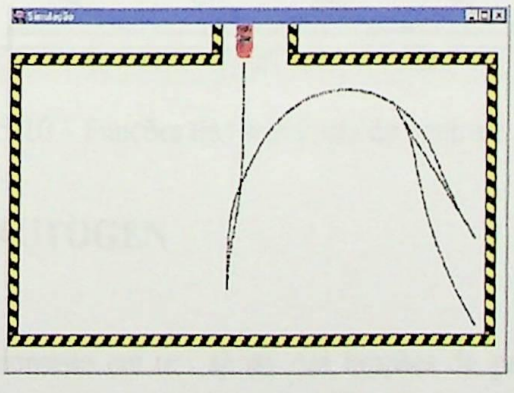
(a) (b)



(c) (d)



(e) (f)



(g)

Figura 5.8 – Simulações após o treinamento.
(a) Posição 1 – (b) Posição 2 – (c) Posição 3 – (d) Posição 4 – (e) Posição 5
(f) Posição 6 – (g) Posição 7

5.3. Testes com o controle AUTO

Em relação ao controle RE.FUZ, este controle apresenta um número maior de regras e funções de pertinência. São 362 regras e 18 funções de pertinência. Este controle foi desenvolvido com o objetivo de demostrar que mesmo utilizando um número maior de regras e funções de pertinência pode-se conseguir um melhor desempenho do controle difuso fazendo-se um ajuste de suas funções de pertinência através dos algoritmos genéticos.

A Figura 5.10 mostra as funções de pertinências originais do controle AUTO.

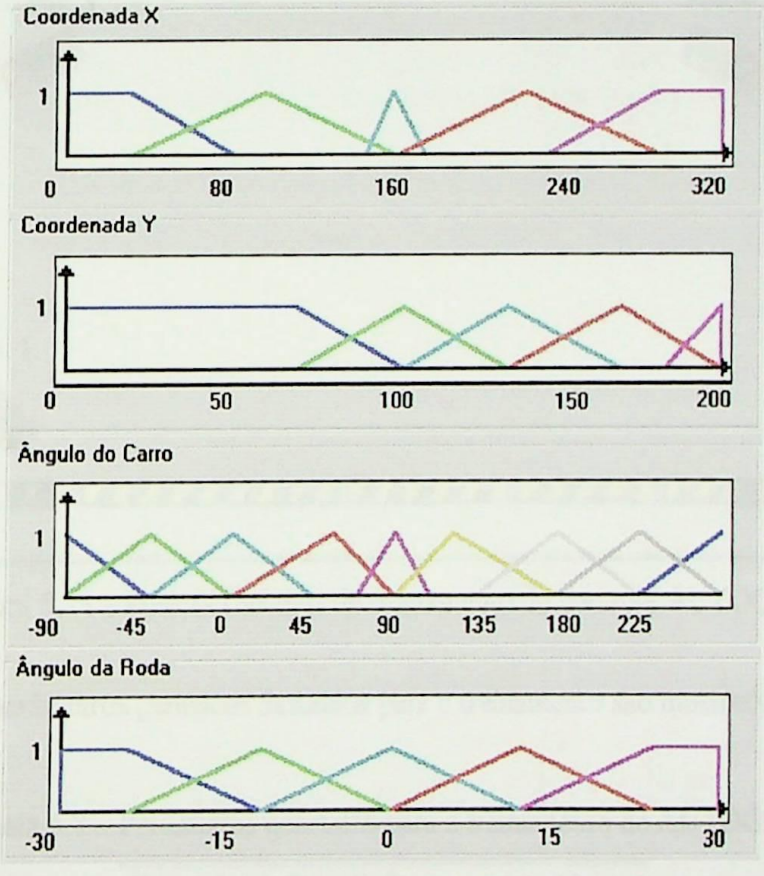


Figura 5.10 – Funções de pertinência do controle AUTO.

5.3.1. O controle AUTOGEN

Este controle consiste em um ajuste das funções de pertinência do controle AUTO. As posições iniciais utilizadas no treinamento são mostradas na Tabela 5.7.

Tabela 5.7 – Posições iniciais para o controle AUTOGEN.

Posição	X	Y	Ângulo do Carro	Iterações sem treinamento
1	44	191	-46	406
2	291	188	222	423
3	60	75	26	410
4	260	75	154	407

A Figura 5.11 mostra o veículo em cada uma das posições iniciais.

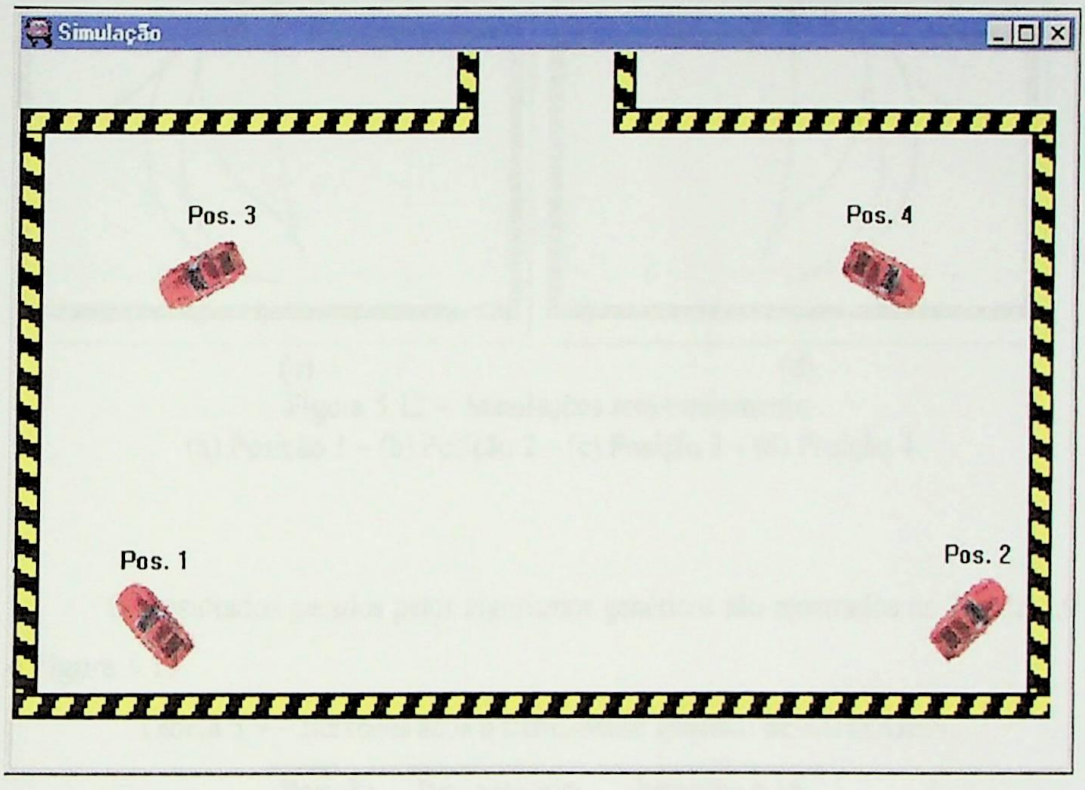


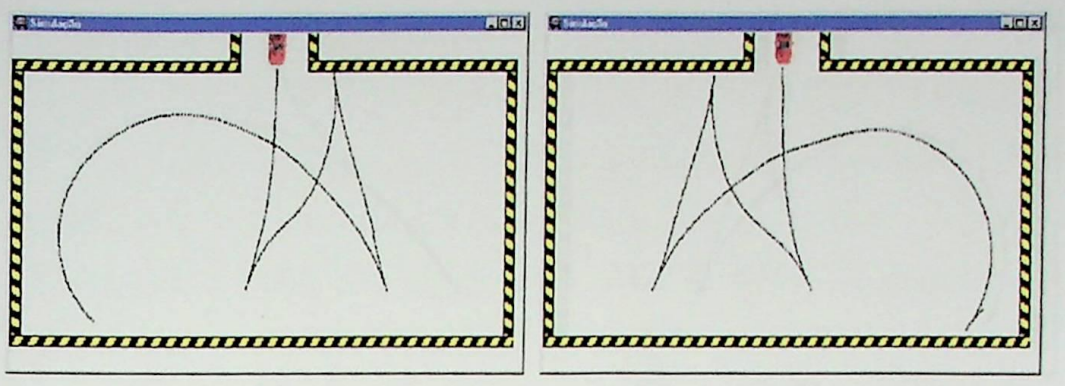
Figura 5.11 – Posições iniciais de treinamento do controle AUTOGEN.

Os parâmetros genéticos definidos para o treinamento são mostrados na Tabela 5.8.

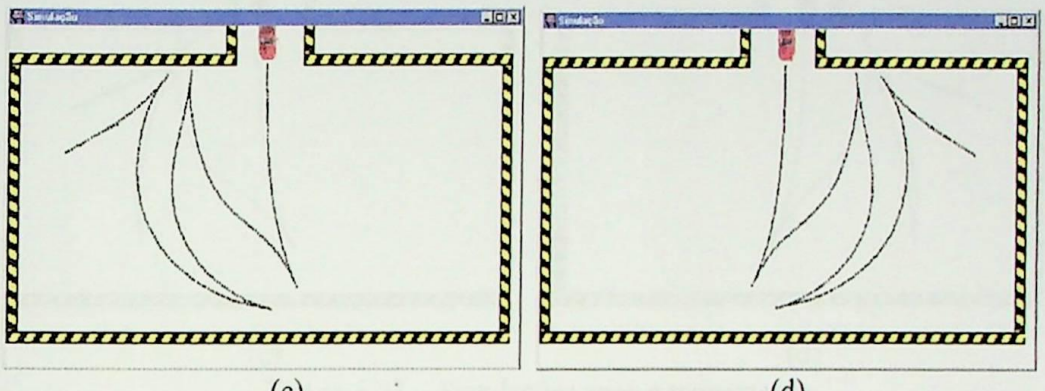
Tabela 5.8 – Parâmetros genéticos para o treinamento de AUTOGEN.

Tamanho da População	30
Número de Gerações	100
Probabilidade de Crossover	90%
Probabilidade de Mutação	1%

A Figura 5.12 mostra as trajetórias referentes a cada posição inicial.



(a) (b)



(c) (d)

Figura 5.12 – Simulações sem treinamento.
(a) Posição 1 – (b) Posição 2 – (c) Posição 3 – (d) Posição 4

Os resultados gerados pelos algoritmos genéticos são mostrados na Tabela 5.9 e Figura 5.13.

Tabela 5.9 – Iterações após o treinamento genético de AUTOGEN.

Posição	Iterações sem treinamento	Iterações com treinamento
1	406	310
2	423	376
3	410	366
4	407	364
Total	1646	1416
Média	411,50	354,00

A Tabela 5.9 mostra uma redução de 230 iterações (14%) para o veículo estacionar partindo das posições iniciais estabelecidas. Neste exemplo, conseguiu-se uma melhora menos significativa que nos exemplos anteriores uma vez que o controle original já apresenta um bom desempenho.

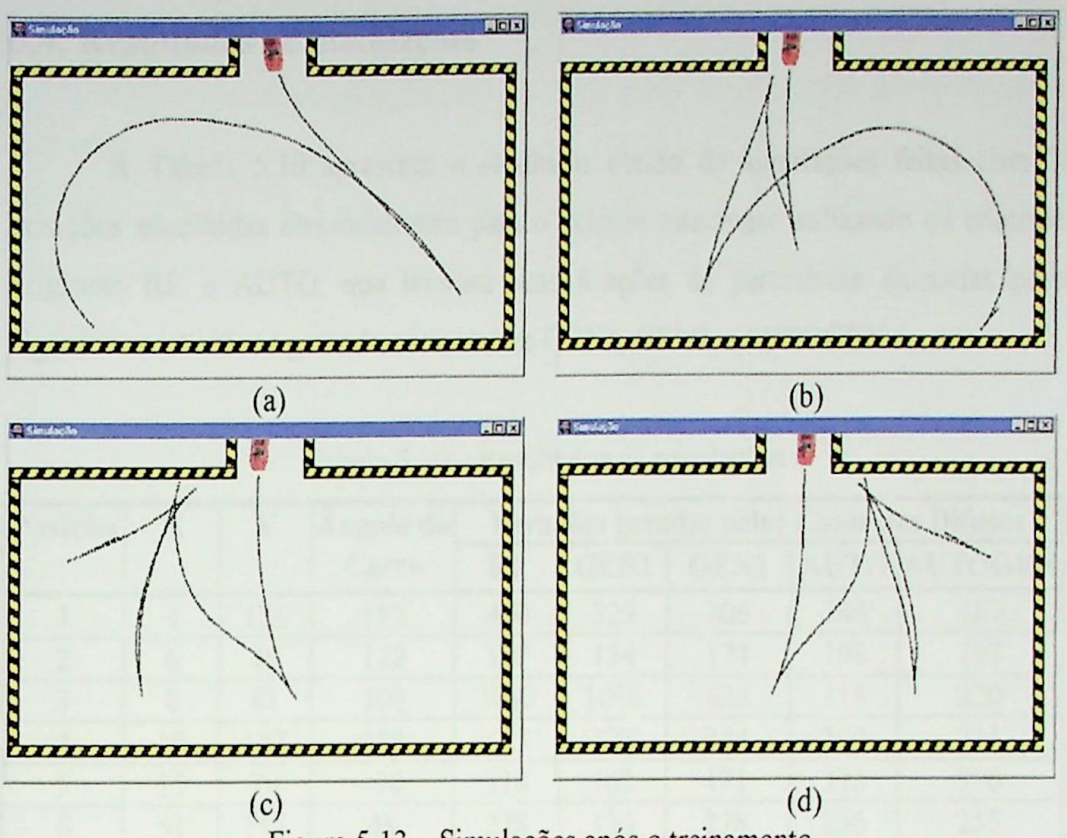


Figura 5.13 – Simulações após o treinamento.
(a) Posição 1 – (b) Posição 2 – (c) Posição 3 – (d) Posição 4

A Figura 5.14 mostra as funções de pertinência do controle AUTOGEN após o ajuste feito pelos algoritmos genéticos.

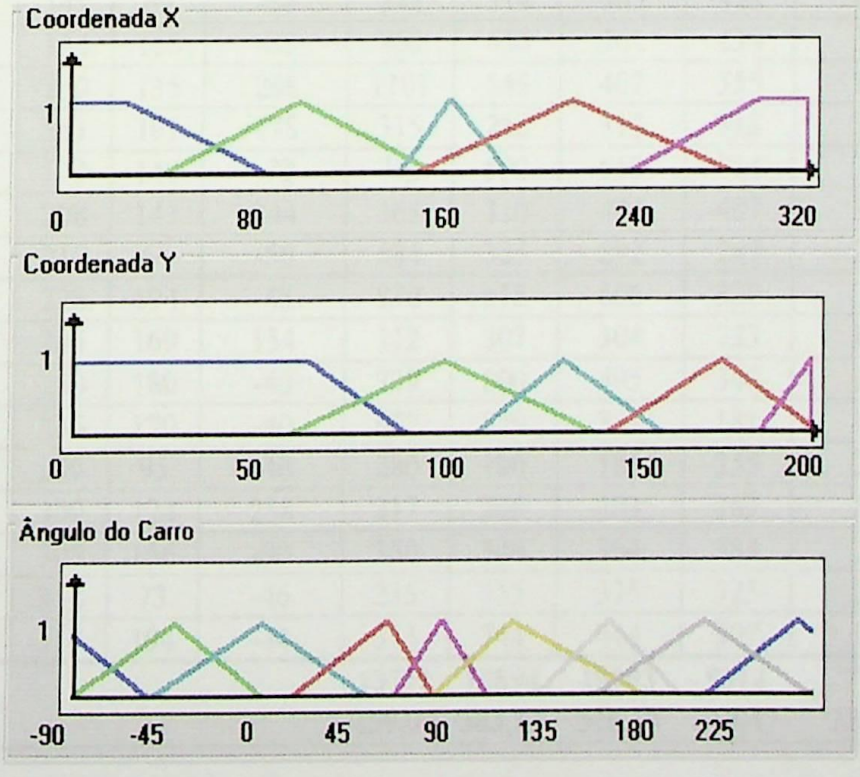


Figura 5.14 - Funções de pertinência após o ajuste.

5.4. Resultados de Simulações

A Tabela 5.10 apresenta o resultado obtido de simulações feitas com 30 posições escolhidas aleatoriamente para o veículo estacionar utilizando os controles originais RE e AUTO, que tiveram suas funções de pertinência ajustadas pelos algoritmos genéticos gerando os controles GEN1, GEN2 e AUTOGEN.

Tabela 5.10 – Resultados de simulações.

Posição	X	Y	Ângulo do Carro	Iterações geradas pelos Controles Difusos				
				RE	GEN1	GEN2	AUTO	AUTOGEN
1	1	126	182	450	329	306	184	187
2	6	46	132	167	154	174	198	197
3	8	41	190	1000	1000	626	315	220
4	10	187	228	453	328	344	200	211
5	15	70	-90	318	162	171	373	396
6	51	112	48	278	130	278	256	255
7	51	112	54	280	132	289	264	266
8	70	95	-40	275	261	268	229	237
9	74	69	190	164	164	163	312	226
10	76	193	232	605	363	410	336	358
11	88	46	44	283	305	294	339	317
12	115	120	0	182	280	158	283	190
13	120	90	45	182	156	159	285	283
14	131	140	-72	457	292	429	315	311
15	141	69	-28	342	314	305	428	420
16	154	166	-80	863	436	361	438	352
17	160	135	268	1101	545	407	355	329
18	161	191	178	315	286	312	512	314
19	173	140	-72	762	590	669	364	358
20	208	143	244	363	310	444	407	308
21	217	66	-50	684	325	292	247	240
22	228	194	-48	830	655	606	330	204
23	246	169	154	312	307	304	223	117
24	250	180	-40	739	800	495	306	176
25	265	170	-40	672	329	312	181	170
26	290	95	-40	280	190	181	355	203
27	300	124	258	317	306	303	363	262
28	305	156	-90	350	346	364	384	231
29	314	73	-46	235	355	335	325	233
30	314	194	-44	513	744	444	207	191
Total				**13772**	**10894**	**10203**	**9314**	**7762**
Média				**459,07**	**363,13**	**340,10**	**310,47**	**258,73**

Os resultados demonstram uma redução média do número de iterações para o veículo atingir a posição final de 21% para o controle GEN1, 26% para o controle GEN2 e 17% para o controle AUTOGEN. Estes valores representam uma redução global na trajetória do veículo partindo-se de posições não utilizadas no treinamento genético.

Observa-se nestes exemplos que a porcentagem de redução global dos controles GEN1 e GEN2 são bem menores que as obtidas pelas posições utilizadas no treinamento (50% para GEN1 e 54% para GEN2), enquanto que a redução global do controle AUTOGEN é maior que a obtida pelas posições utilizadas no treinamento (14%). Estes valores estão relacionados as posições iniciais utilizadas nas simulações e eles variam quando se utilizam posições iniciais diferentes. Mas verificou-se que sempre há uma redução do número de iterações [25].

Outra observação é que em determinadas posições o número de iterações é maior que as geradas pelos controles originais (sem treinamento). Na posição 29 da Tabela 5.10 por exemplo, o controle RE gera 235 iterações para estacionar o veículo enquanto que os controles GEN1 e GEN2 geram 355 iterações. Este aumento é em conseqüência das modificações nas funções da pertinência que fazem com que o veículo desenvolva uma trajetória diferente para atingir a posição final.

Capítulo 6

Conclusão e Desenvolvimentos Futuros

O sistemas difusos são uma alternativa conveniente e eficaz na solução de problemas onde os estados difusos são bem definidos. Contudo, o projeto de um sistemas difusos pode se tornar difícil para sistemas grandes e complexos quando a qualidade do controle depender de métodos de "tentativa-e-erro" para definir as melhores regras e os melhores parâmetros das funções de pertinência para o problema a ser resolvido.

O principal objetivo do Pacote Computacional para Ensino da Lógica Difusa utilizado neste trabalho, é a de proporcionar aos estudantes o aprendizado desta lógica. A escolha do estacionamento de um veículo se justifica pelo fato destes estudantes não necessitarem de um conhecimento prévio (pelo menos, em termos matemáticos) sobre o assunto para realizar o controle.

O módulo de treinamento genético desenvolvido neste trabalho acrescentou a este programa uma técnica automática para o ajuste dos parâmetros das funções de pertinência. Esta técnica mostra que o desempenho de um controle difuso pode ser melhorado através dos algoritmos genéticos substituindo o método da "tentativa-e-erro", antes utilizado pelos estudantes para este fim, onde não se conseguiam bons resultados.

Os algoritmos genéticos ofereceram vantagens distintas de otimização das funções de pertinência resultando em uma pesquisa global, reduzindo as chances de terminarem em um mínimo local, pois utiliza vários conjuntos de soluções simultaneamente. A lógica difusa contribuiu com a função de avaliação, estágio do algoritmo genético onde o ajuste é determinado.

Outra vantagem apresentada pelos algoritmos genéticos é que eles não perturbam o processo, ou seja, as ações de controle que já eram bem desenvolvidas

pelo controlador difuso antes do treinamento genético não sofrem alterações que as tornem insatisfatórias após o treinamento genético.

Este trabalho é também um exemplo de integração de sistemas difusos com algoritmos genéticos, constituindo assim um sistema híbrido. Nesta integração mostrou-se que quando se tem o domínio do problema, este pode ser explorado pelos algoritmos genéticos conduzindo a um desempenho melhor do controlador difuso.

Os resultados obtidos foram bons e consistentes e comprovam os benefícios visualizados e descritos acima.

Como desenvolvimento futuros e complementares a este pacote computacional pode-se sugerir:

a) incluir nos cromossomos informações sobre as condições e ações correspondentes às regras difusas. Incluí-las no tratamento genético permitiria ao sistema aprender ou refinar as regras difusas;
b) fazer a inserção e remoção de funções de pertinência através dos algoritmos genéticos;
c) a inclusão de uma lógica que determine se a melhor saída para o veículo é andar para frente ou para trás (atualmente, ele sai sempre para frente);
d) o desenvolvimento de barreiras na área do estacionamento, gerando dificuldades ao sistema de controle tendo novos parâmetros definidos pelos algoritmos genéticos;
e) investigar formas diferenciadas de implementação dos algoritmos genéticos, como por exemplo em máquinas paralelas, visando maior rapidez na execução destes.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] L. A. Zadeh - "Fuzzy Sets", *Information and Control*, Vol.8, pp.338-353, 1965.

[2] E. H. Mamdani, "Application for fuzzy algorithms for the control of a dynamic plant," *Proc. IEEE.* , vol. 121, pp. 1585-1588, 1974.

[3] T. Terano, K. Asai & M. Sugeno - *Fuzzy Systems Theory and Its Applications*, New York: Academic Press, 1992.

[4] P.P. Bissione, V. Badami, K.H. Chiang, P.S. Khedkar, K.W. Marcelle & M.J. Schutten - "Industrial Applications of Fuzzy Logic at General Electric", *Proc. of the IEEE*, pp.450-465, March 1995.

[5] J.A. Momoh, X.W. Ma & K. Tomsovic - "Overview and Literature Survey of Fuzzy Set Theory in Power Systems", *IEEE Trans. on Power Systems*, Vol. 10, No.3, pp.1676-1690, Aug. 1995.

[6] D. Niebur, G. Lambert-Torres, A.P. Alves da Silva, *et al.* - "Artificial Neural Network for Power Systems", *Électra*, No. 159, pp. 76-101, April 1995.

[7] David E. Coldberg, *Genetic algorithms in search, optimization and machine learning*, Addison Wesley, 1989.

[8] L.R. Medsker - *Hybrid Intelligent Systems*, Boston: Kulwer Academic Pub., 1995.

[9] A. Kaufmann – *Introduction to the Theory of Fuzzy Subsets*, Academic Presss, 1987.

[10] D. Dubois & H. Prade - *Fuzzy Sets and Systems: Theory and Applications*, New York: Academic Press, 1980.

[11] A. Kaufmann & M.M. Gupta - *Introduction to Fuzzy Arithmetic: Theory and Applications*, New York: Van Nostrand Reinhold, 1985.

[12] L. Davis, *Applying adaptive algorithms to epistactic domains*, in: Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI'85), Los Angeles, CA (1985), 162-164.

[13] A. Kandel & G. Langholz, *Fuzzy Control Systems*, CRC Press, 1993

[14] J. H. Holland, *Adaptation in natural and artificial systems* (The University of Michigam Press, Ann Arbor, 1975), reprinted by MIT Press, 1992.

[15] P. G. Rabelo & L. S. Ochi, *Um novo algoritmo genético híbrido para o problema do caixeiro viajante com grupamentos*, XXVIII SBPO e VIII CLAIO, Rio de Janeiro, 1996, 116-1165.

[16] M. Sugeno - *Industrial Applications of Fuzzy Control*, Amsterdam: North Holland, 1985.

[17] C. L. Karr, "Genetic algorithms for fuzzy controllers", *AI Expert*, vol 6, no. 2, pp. 26-33, 1991.

[18] C. L. Karr, "Applying genetics to fuzzy logic", *AI Expert*, vol 6, no. 3, pp. 38-43, 1991.

[19] D. Park & A. Kandel, - "Genetic-Based New Fuzzy Reasoning Models with Aplication to Fuzzy Control", *IEEE Trans. On Systems*, vol. 24, no. 1, pp. 39-47, January 1994.

[20] S. G. de Brito, *Pacote Computacional para o Ensino da Lógica Difusa*, M.Sc. Thesis, Escola Federal de Engenharia de Itajubá , 1998

[21] B. Valiquette, G. Lambert-Torres & D. Mukhedkar - "An Expert System Based Diagnosis and Advisor Tool for Teaching Power System Operation Emergency Control Strategies", *IEEE Trans. on Power Systems*, Vol.6, No.3, pp.1315-1322, Aug. 1991.

[22] A. Laprade & G. Lambert-Torres - "Controlando um Veículo Utilizando a Teoria dos Conjuntos Nebulosos", Anais I Cong. Bras. Automação Inteligente, pp.165-174, Rio Claro, Set. 1993.

[23] C. L. Karr & D. A. Stanley, "Fuzzy logic and genetic algorithms in time-varying control problems", in *Proc. NAFIPS-91*, 1991, pp. 285-290.

[24] D. L. Meredith, K. K. Kumar, and C. L. Karr, "The use of genetic algorithms in the design of fuzzy logic controllers", in *Proc. WNN-AIND 91*, 1991, pp. 695-702.

[25] M. A. Carvalho, G. Lambert-Torres, L. E. Borges da Silva & J. O. P. Pinto, "Fitting Fuzzy Membership Functions using Genetic Algorithms", in *2000 IEEE International Conference on Systems, Man, and Cybernetics*, Oct. 8-11 2000, Nashville, USA.