UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Uma Proposta Evolutiva para Controle Inteligente em Navegação Autônoma de Robôs

Renato Reder Cazangi

Prof. Dr. Fernando José Von Zuben (Orientador)

Prof. Dr. Maurício Fernandes Figueiredo (Co-orientador)

Tese apresentada à Pós-graduação da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como requisito parcial à obtenção do grau de **Mestre em Engenharia Elétrica** na área de Engenharia de Computação.

Banca examinadora:

Prof. Dr. Fernando José Von Zuben– FEEC/UNICAMP

Prof. Dr. Christiano Lyra Filho – FEEC/UNICAMP

Prof. Dr. Marconi Kolm Madrid – FEEC/UNICAMP

Profª. Drª. Roseli Aparecida Francelin Romero – ICMC/USP – São Carlos

Campinas, 28 de Maio de 2004

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

C318p

Cazangi, Renato Reder
Uma proposta evolutiva para controle inteligente em navegação autônoma de robôs / Renato Reder Cazangi. -- Campinas, SP: [s.n.], 2004.

Orientadores: Fernando José Von Zuben ; Maurício Fernandes Figueiredo.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas inteligentes de controle. 2. Computação evolutiva. 3. Robôs móveis. 4. Sistemas de veículos auto-guiados. I. Von Zuben, Fernando José. II. Figueiredo, Maurício Fernandes. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Resumo

Este trabalho apresenta um sistema autônomo evolutivo aplicado ao controle de um robô móvel em tarefas de navegação por ambientes desconhecidos. O sistema é reativo, não possui conhecimento inicial e aprende a lidar eficientemente com situações nas quais o robô tem que capturar alvos evitando colisões contra obstáculos. Para isto, ele desenvolve estratégias gerais de navegação, controlando a direção e a velocidade do robô sem qualquer auxílio externo. A abordagem evolutiva do sistema de navegação se baseia em uma versão de sistemas classificadores com aprendizado, contendo novos operadores, fluxos de controle adicionais e mecanismos específicos para o atendimento dos requisitos de navegação. Um extenso conjunto de experimentos é realizado, envolvendo: apenas simulação computacional; simulação computacional para síntese do controlador e transferência deste a um robô Khepera II; e emprego do robô Khepera II tanto na síntese do controlador quanto na atuação em ambientes reais. Os resultados obtidos apontam para a validade da proposta, indicando a eficácia e capacidade de generalização do controlador autônomo quando submetido a variadas configurações de ambiente de navegação.

Abstract

This work presents an autonomous evolutionary system applied to the control of a mobile robot when navigating in unknown environments. The system is reactive, it does not have initial knowledge and learns efficiently to deal with situations where the robot must capture targets avoiding collisions against obstacles. Toward this end, it develops general strategies, controlling the robot direction and speed without any external assistance. The evolutionary approach of the navigation system is based on a version of learning classifier systems, including new operators, additional control flows and specific mechanisms devoted to attending the navigation requirements. An extensive set of experiments is performed involving: just computer simulation; a controller matured by computer simulation, and then transferred to a Khepera II robot; and both the maturation and validation of the controller in real environments, i.e. in a Khepera II robot. The obtained results indicate the validity of the proposal, attesting the efficiency and generalization capability of the autonomous controller when navigation environments with distinct configurations are considered.

É com muito orgulho que dedico meu mestrado à minha família, especialmente aos meus pais: Vilma e João; aos meus irmãos: Gustavo e Humberto; e aos meus avós: Zenaide e Paulo (*in memorian*)

Agradecimentos

A Deus por me dar força para superar as dificuldades e, embora eu não mereça, pelo privilégio de ter colocado e continuar colocando em minha vida oportunidades e, principalmente, pessoas tão especiais. Entre elas, uma família incrível, os melhores orientadores e todos aqueles que só eu sei o quanto são e foram importantes nesta caminhada.

Ao professor Fernando Von Zuben por tamanha confiança e constante motivação, e por ser um exemplo de dedicação, competência e simplicidade.

Ao professor Maurício Figueiredo pelo apoio e companheirismo, por ter me introduzido nesta caminhada e por ter me ensinado, com tanta mestria, muito do que eu sei.

À CAPES pelo suporte financeiro, à Faculdade de Engenharia Elétrica e de Computação e ao Departamento de Engenharia de Computação e Automação Industrial por fornecer valorosos subsídios à minha pesquisa.

Aos meus pais, irmãos e avós, cada qual a seu jeito, por serem o alicerce, a força e a inspiração para que eu continue sempre. Tenho orgulho de tê-los em minha vida.

À minha Fernanda, pelo amor, carinho e compreensão, por estar ao meu lado dos piores aos melhores momentos e por me encorajar continuamente. Amo você.

Ao seu Gilberto, dona Ângela, Grasi e Bianca por serem a minha família em Maringá e por me tratarem como um verdadeiro filho. Admiro todos vocês.

Aos amigos e todas as pessoas que, de uma maneira ou de outra, me auxiliaram e acompanharam nesta caminhada.

Índice

RESUMO ..... **I**

ABSTRACT ..... **I**

AGRADECIMENTOS ..... **V**

ÍNDICE ..... **VII**

ÍNDICE DE FIGURAS ..... **XI**

ÍNDICE DE TABELAS ..... **XIII**

CAPÍTULO 1: INTRODUÇÃO **1**

1.1 INTELIGÊNCIA E AUTONOMIA 1

1.2 NAVEGAÇÃO AUTÔNOMA DE ROBÔS E APRENDIZAGEM 2

1.3 ABORDAGENS PARA SISTEMAS AUTÔNOMOS INTELIGENTES 5

1.4 PROPOSTA DE SISTEMA DE NAVEGAÇÃO 9

1.5 SIMULAÇÃO COMPUTACIONAL E IMPLEMENTAÇÃO EM ROBÔS REAIS 10

1.6 APLICAÇÕES DE NAVEGAÇÃO AUTÔNOMA 11

1.7 ORGANIZAÇÃO DO TEXTO 12

CAPÍTULO 2: SISTEMAS CLASSIFICADORES **13**

2.1 INTRODUÇÃO 13

2.2 ESTRUTURA E OPERAÇÃO 15

2.2.1 Regras, Especificidade e Energia *17*

2.2.2 Módulo de Tratamento de Regras e Mensagens *18*

2.2.3 Módulo de Atribuição de Crédito *20*

2.2.3.1 Leilão 21

2.2.3.2 Recompensa 22

2.2.3.3 Taxação 23

2.2.4 Módulo de Descoberta de Novas Regras *24*

2.3 ALGORITMO SIMPLIFICADO 26
2.4 ADEQUAÇÃO FUNCIONAL À NAVEGAÇÃO AUTÔNOMA DE ROBÔS 26
2.5 MECANISMOS ALTERNATIVOS E VARIAÇÕES DE SISTEMAS CLASSIFICADORES 27
2.5.1 Algoritmo de Bucket Brigade 28
2.5.2 Sistemas Classificadores Especiais 29
2.5.3 Sistemas Classificadores Antecipatórios 29

CAPÍTULO 3: SISTEMA DE NAVEGAÇÃO AUTÔNOMO 31
3.1 INTRODUÇÃO 31
3.2 CARACTERÍSTICAS DO ROBÔ (APÊNDICE B) 32
3.3 SIMULADOR 35
3.4 ADEQUAÇÃO FUNCIONAL DOS SISTEMAS CLASSIFICADORES À NAVEGAÇÃO AUTÔNOMA DE ROBÔS E PRINCIPAIS CONTRIBUIÇÕES 35
3.5 DESCRIÇÃO DO SISTEMA DE NAVEGAÇÃO AUTÔNOMO EVOLUTIVO 39
3.5.1 População de Regras 42
3.5.2 Módulo de Competição 45
3.5.3 Evolução 47
3.5.3.1 Colisão 48
3.5.3.2 Módulo de Avaliação da Evolução Disparada por um Evento de Colisão 49
3.5.3.3 Módulo de Reprodução da Evolução Disparada por um Evento de Colisão 50
3.5.3.4 Captura 53
3.5.3.5 Módulo de Avaliação da Evolução Disparada por um Evento de Captura 54
3.5.3.6 Módulo de Reprodução da Evolução Disparada por um Evento de Captura 55
3.5.3.7 Monotonia 55
3.5.3.8 Módulo de Avaliação da Evolução Disparada por um Evento de Monotonia 56
3.5.3.9 Módulo de Reprodução da Evolução Disparada por um Evento de Monotonia 56
3.5.3.10 Taxa de Procriação 57

CAPÍTULO 4: EXPERIMENTOS E RESULTADOS 59

4.1 INTRODUÇÃO 59

4.1.1 Interpretação dos Resultados e Informações Pertinentes 60

4.2 SIMULAÇÃO: EXPERIMENTOS E RESULTADOS 62

4.2.1 Análise Paramétrica 62

4.2.1.1 Tamanho da População 63

4.2.1.2 Quantidade de Descendentes Gerados 65

4.2.1.3 Número de Sensores 66

4.2.2 Controle de Velocidade 67

4.2.3 Taxa de Procriação 71

4.2.4 Tarefas Independentes 74

4.2.5 Tarefas Simultâneas 76

4.2.6 Consistência da Aprendizagem 77

4.2.7 Generalização 80

4.2.8 Análise das Regras 84

4.3 NAVEGAÇÃO COM ROBÔ REAL: EXPERIMENTOS E RESULTADOS 91

4.3.1 Treinamento até Maturação e Validação em Ambiente Real 91

4.3.2 Experimentos Apenas em Ambientes Reais 93

CAPÍTULO 5: CONSIDERAÇÕES FINAIS 97

5.1 SÍNTESE DO TRABALHO 97

5.2 PRINCIPAIS CONTRIBUIÇÕES 98

5.3 LIMITAÇÕES OPERACIONAIS E PROPOSTAS PONTUAIS DE EXTENSÃO 99

5.4 CONSIDERAÇÕES ADICIONAIS A RESPEITO DOS RESULTADOS OBTIDOS 100

5.5 PERSPECTIVAS FUTURAS 101

APÊNDICE A: ALGORITMOS GENÉTICOS 103

A.1 INTRODUÇÃO 103

A.2 OPERADORES DE SELEÇÃO 105
A.3 OPERADORES DE RECOMBINAÇÃO 106
A.4 OPERADORES DE MUTAÇÃO 107

APÊNDICE B: ROBÔ KHEPERA II® 109
B.1 INTRODUÇÃO 109
B.2 CARACTERÍSTICAS 109
B.3 COMUNICAÇÃO, CONTROLE E CALIBRAÇÃO 111

APÊNDICE C: AMBIENTE DE SIMULAÇÃO 115
C.1 INTRODUÇÃO 115
C.2 LINGUAGEM DE PROGRAMAÇÃO E ESTRUTURA DO CÓDIGO FONTE 115
C.3 INTERFACE E FUNCIONALIDADES DO SIMULADOR 116
C.4 FERRAMENTAS DE ANÁLISE DE DESEMPENHO 119
C.5 DESCRIÇÃO DOS COMANDOS E OPÇÕES DO PROGRAMA-SIMULADOR 120
C.5.1 Menu Arquivo (Figura C-4) 120
C.5.2 Menu Controle e Caixa de Ferramentas (Figura C-5) 121
C.5.3 Menu Modo de Execução (Figura C-6) 122
C.5.4 Menu Editar (Figura C-7) 123
C.5.5 Menu Exibir (Figura C-9) 124

REFERÊNCIAS BIBLIOGRÁFICAS 127
ÍNDICE REMISSIVO 133

Índice de Figuras

FIGURA 2-1: INTERAÇÃO DO SC COM O AMBIENTE. 16

FIGURA 2-2: DIAGRAMA GERAL DE UM SC. 17

FIGURA 2-3: EXEMPLO DA FASE DE COMPETIÇÃO ENTRE CLASSIFICADORES. AS ENERGIAS DOS CLASSIFICADORES (SEÇÃO 2.2.3.3), EMPREGADAS NA SELEÇÃO DO VENCEDOR, NÃO ESTÃO APRESENTADAS. 19

FIGURA 2-4: DIAGRAMA DO MECANISMO DE DESCOBERTA DE NOVAS REGRAS. 25

FIGURA 2-5: EXEMPLO DE SEQÜÊNCIAS DE CLASSIFICADORES ATUANTES (ÉPOCAS DE ITERAÇÕES) SEGUIDAS PELA APLICAÇÃO DE UM ALGORITMO EVOLUTIVO (A.E.) 25

FIGURA 2-6: ALGORITMO SIMPLIFICADO DO SC. 26

FIGURA 3-1: MINI-ROBÔ KHEPERA II. 33

FIGURA 3-2: DISTRIBUIÇÃO DOS SENSORES NA ESTRUTURA DO ROBÔ, CUJA FRENTE É APONTADA PELA SETA. 34

FIGURA 3-3: ALGORITMO SIMPLIFICADO DO SISTEMA CLASSIFICADOR IMPLEMENTADO NESTE TRABALHO. 40

FIGURA 3-4: DIAGRAMA QUE REPRESENTA A ESTRUTURA E FUNCIONAMENTO DO SNA. 41

FIGURA 3-5: EXEMPLO GRÁFICO DE UMA PARTE ANTECEDENTE DE ALVO (ÂNGULO DOS SENSORES × LUMINOSIDADE). 44

FIGURA 3-6: EXEMPLOS HIPOTÉTICOS DE PADRÕES INCONSISTENTES DO VETOR *RA*. 45

FIGURA 3-7: EXEMPLO DO CROMOSSOMO DE UMA REGRA CUJOS CONSEQÜENTES DETERMINAM GIRO DE −3º E AUMENTO DE VELOCIDADE. 45

FIGURA 3-8: CRUZAMENTO COM UM PONTO DE CORTE. 52

FIGURA 3-9: EXEMPLO DE COMBINAÇÃO PARA OS VETORES *RT*. 52

FIGURA 4-1: ROBÔ KHEPERA II EMPREGADO NOS EXPERIMENTOS REAIS. 59

FIGURA 4-2: ARENA USADA COMO AMBIENTE NOS EXPERIMENTOS REAIS. 60

FIGURA 4-3: AMBIENTE PADRÃO USADO NOS EXPERIMENTOS DESTA SEÇÃO. 63

FIGURA 4-4: EXPERIMENTOS EM QUE O ROBÔ APRESENTA VELOCIDADE CONSTANTE. 68

FIGURA 4-5: EXPERIMENTOS EM QUE O ROBÔ APRESENTA VELOCIDADE VARIÁVEL. 68

FIGURA 4-6: AMBIENTE COM OITO ALVOS MUITO PRÓXIMOS DO OBSTÁCULO. 70

FIGURA 4-7: AMBIENTE COM QUATRO REGIÕES DE ALVOS. 72

FIGURA 4-8: SEM TAXA DE PROCRIAÇÃO (À ESQUERDA) E COM TAXA DE PROCRIAÇÃO (À DIREITA). 73

FIGURA 4-9: AMBIENTE COM QUATRO REGIÕES DE ALVO ENTRE OBSTÁCULOS. 74

FIGURA 4-10: SEM TAXA DE PROCRIAÇÃO (À ESQUERDA) E COM TAXA DE PROCRIAÇÃO (À DIREITA). 74

FIGURA 4-11: AMBIENTE E DESEMPENHO DO ROBÔ SOMENTE EXIGINDO DESVIO DE OBSTÁCULOS. 75

FIGURA 4-12: ROBÔ NAVEGANDO EM UM AMBIENTE SEM ALVOS E SEU DESEMPENHO. 76

FIGURA 4-13: DOIS EXPERIMENTOS ENVOLVENDO SOMENTE CAPTURA DE ALVOS. 76

FIGURA 4-14: AMBIENTE COM 4 REGIÕES DE ALVOS. 77

FIGURA 4-15: DESEMPENHO DO SISTEMA EM UM EXPERIMENTO COMPLETO. 77

FIGURA 4-16: SEIS ROBÔS NAVEGANDO DE FORMA INDEPENDENTE EM AMBIENTES IGUAIS 78

FIGURA 4-17: AMBIENTE COM DUAS REGIÕES DE ALVOS. 79

FIGURA 4-18: AMBIENTE BÁSICO PARA TREINAMENTO ATÉ A MATURAÇÃO DO SNA E DADOS OBTIDOS. 81
FIGURA 4-19: AMBIENTE PARA TESTE DE GENERALIZAÇÃO COM QUATRO ALVOS. 82
FIGURA 4-20: AMBIENTE PARA TESTE DE GENERALIZAÇÃO COM TREZE ALVOS. 82
FIGURA 4-21: AMBIENTE INICIAL PARA TREINAMENTO ATÉ A MATURAÇÃO DO SNA E DADOS ASSOCIADOS AO DESEMPENHO DO ROBÔ. 83
FIGURA 4-22: AMBIENTE PARA TESTE DE GENERALIZAÇÃO COM QUATRO ALVOS. 83
FIGURA 4-23: AMBIENTE PARA TESTE DE GENERALIZAÇÃO COM OITO ALVOS. 84
FIGURA 4-24: GRÁFICOS DA VARIAÇÃO DOS TIPOS DE REGRAS EM RELAÇÃO A OBSTÁCULO (À ESQUERDA) E RELATIVO A ALVO (À DIREITA). 87
FIGURA 4-25: EXPERIMENTO SIMULADO EM AMBIENTE COM DUAS REGIÕES DE ALVOS. 87
FIGURA 4-26: GRÁFICOS DA VARIAÇÃO DOS TIPOS DE REGRAS EM RELAÇÃO A OBSTÁCULO (A), RELATIVO A ALVO (B) E ASSOCIADO A AJUSTE DE DIREÇÃO. 88
FIGURA 4-27: EXPERIMENTO CUJO AMBIENTE CONTEMPLAVA ALVOS PRÓXIMOS DOS OBSTÁCULOS E DESEMPENHO DO ROBÔ. 89
FIGURA 4-28: GRÁFICO DA VARIAÇÃO DO NÚMERO DE REGRAS SEGUNDO A POSIÇÃO DE OBSTÁCULO. 90
FIGURA 4-29: AMBIENTE REAL COM UM OBSTÁCULO ENTRE O ROBÔ E A FONTE DE LUZ. 92
FIGURA 4-30: DESEMPENHO DO SISTEMA EM AMBIENTE REAL COM UM OBSTÁCULO CENTRAL. 92
FIGURA 4-31: PERFORMANCE DO ROBÔ COM APRENDIZAGEM TOTALMENTE EM AMBIENTE REAL. 93
FIGURA 4-32: AMBIENTE COM TRÊS OBSTÁCULOS. 94
FIGURA 4-33: ATUAÇÃO DO ROBÔ EM UM AMBIENTE COMPLEXO. 96
FIGURA A-1: CICLO BÁSICO DE UM AG. 104
FIGURA A-2: EXEMPLO DE SELEÇÃO PELO MÉTODO DA ROLETA. 105
FIGURA A-3: CRUZAMENTO EM UM PONTO. 107
FIGURA B-1: MINI-ROBÔ KHEPERA II. 110
FIGURA B-2: DISPOSIÇÃO DOS SENSORES NA ESTRUTURA DO ROBÔ. 111
FIGURA B-3: ILUSTRAÇÃO DE COMO É FEITA A COMUNICAÇÃO COM O ROBÔ. 112
FIGURA B-4: CONTROLADOR EM QUE O ROBÔ É COMANDADO PELO USUÁRIO. 112
FIGURA C-1: COMPONENTES DO AMBIENTE DE SIMULAÇÃO. 117
FIGURA C-2: AMBIENTE ONDE A E B SÃO REGIÕES DE INSERÇÃO AUTOMÁTICA DE ALVOS. QUANDO O ROBÔ CAPTURA UM ALVO EM A, AUTOMATICAMENTE OUTRO APARECEM EM UMA POSIÇÃO ALEATÓRIA DENTRO DE B E ASSIM POR DIANTE. 118
FIGURA C-3: O GRÁFICO (A) É DA LEITURA DOS SENSORES, (B) DO CONTEÚDO DAS REGRAS, (C) DO DESEMPENHO DO SISTEMA E (D) DA VARIAÇÃO MÉDIA DO TIPO DAS REGRAS DA POPULAÇÃO. 120
FIGURA C-4: COMANDOS DO MENU ARQUIVO. 120
FIGURA C-5: COMANDOS DO MENU CONTROLE E CAIXA DE FERRAMENTAS. 121
FIGURA C-6: MODOS DE EXECUÇÃO. 122
FIGURA C-7: COMANDOS DO MENU EDITAR. 123
FIGURA C-8: CONFIGURAÇÕES POSSÍVEIS. 123
FIGURA C-9: COMANDOS DO MENU EXIBIR. 124

Índice de Tabelas

TABELA 2-1: VARIANTES DE SISTEMAS CLASSIFICADORES (ADAPTADO DE KOVACS (2000)). 27
TABELA 3-1: PRINCIPAIS DISTINÇÕES ENTRE O SC DE HOLLAND E O SISTEMA PROPOSTO NESTE TRABALHO. ... 36
TABELA 3-2: CARACTERÍSTICAS DOS VETORES QUE COMPÕEM UM INDIVÍDUO (REGRA) DA POPULAÇÃO. 42
TABELA 3-3: INFORMAÇÕES DO MÓDULO DE AVALIAÇÃO EM CASO DE DISPARO POR COLISÃO. 50
TABELA 3-4: INFORMAÇÕES DO MÓDULO DE AVALIAÇÃO EM CASO DE DISPARO POR CAPTURA. 55
TABELA 4-1: DESEMPENHO DO SISTEMA DE ACORDO COM O TAMANHO DA POPULAÇÃO..... 64
TABELA 4-2: RESULTADO DA SIMULAÇÃO COM 2 ROBÔS DOTADOS DE DIFERENTE NÚMERO DE SENSORES..... 66
TABELA 4-3: DADOS DOS EXPERIMENTOS COM ALVOS PRÓXIMOS DO OBSTÁCULO. 70
TABELA 4-4: RESULTADOS DO DESEMPENHO DOS SEIS ROBÔS. 79
TABELA 4-5: DADOS DE DESEMPENHO DOS SEIS ROBÔS. 80
TABELA 4-6: TIPOS EM QUE AS PARTES DAS REGRAS SÃO CLASSIFICADAS. 85
TABELA 4-7: PROPORÇÃO DE CADA TIPO DE REGRA PRESENTE NA POPULAÇÃO. 85
TABELA 4-8: PORCENTAGEM DE CADA TIPO DE REGRA PRESENTE NA POPULAÇÃO..... 86
TABELA 4-9: PROPORÇÃO DE CADA TIPO DE REGRA PRESENTE NA POPULAÇÃO. 89

Capítulo 1
Introdução

1.1 Inteligência e Autonomia

A evolução científica e tecnológica tem trazido inovações e descobertas cada vez mais freqüentes e surpreendentes. No entanto, há áreas do conhecimento que sempre foram e ainda hoje continuam sendo desafiadoras e intrigantes. É o caso da inteligência: o que a caracteriza, de que forma e quando ela opera e como reproduzi-la artificialmente?

Considerando a possibilidade de graduar a manifestação de inteligência em vários níveis e escalas, a natureza é fonte inesgotável de mecanismos, comportamentos e organismos que podem ser considerados inteligentes. Dos seres extremamente simples aos mais complexos há vestígios de inteligência, seja em indivíduos isolados ou então no comportamento coletivo de um grupo de indivíduos. São estes aspectos que atraem a atenção de pesquisadores das mais diversas áreas, como biologia, psicologia e engenharia de computação, na busca pela compreensão e reprodução artificial dos sistemas biológicos que expressam algum grau de inteligência.

Não é possível investigar a inteligência de forma isolada, dado que ela está interligada a outras propriedades presentes nos organismos biológicos. Uma das principais propriedades diretamente conectadas à existência de inteligência é a autonomia. Um agente artificial não autônomo não possui capacidade de inteligência própria. Na verdade, ele incorpora aspectos de inteligência especificados por quem o projetou. Sistemas biológicos são em geral autônomos, isto é, suas estruturas não são impostas por agentes externos, mas sim desenvolvidas e mantidas por eles próprios por meio de mecanismos como auto-organização, evolução, adaptação e aprendizagem. Considera-se, portanto, que a manifestação de algum grau de inteligência pode ser vista como um atributo de agentes autônomos (STEELS, 1995).

Dizer que um agente é autônomo implica em afirmar que ele, além de agir por si só, consegue se auto-regular gerando as próprias regras que regem sua atuação. Esta definição distingue autônomo de automático. Ser automático é ser capaz de operar em um ambiente: percebê-lo e impactá-lo visando o cumprimento de tarefas definidas. Um agente autônomo é antes de tudo automático, mas vai além disto: ele deve se autodirigir com base na sua capacidade própria de aprender e adaptar seus comportamentos. Além disso, os processos de aprendizagem e adaptação devem ocorrer enquanto o agente está operando no ambiente, e não fora dele (por exemplo, em fase de projeto).

Da mesma maneira adotada para o conceito de inteligência, é possível considerar a existência de níveis de autonomia, até pelo fato de não existir um ser totalmente autônomo. Os animais, em geral, e o ser humano, em particular, dependem de fatores externos para sobreviverem. Estes organismos dependem do ambiente em que vivem para obter alimento e oxigênio, por exemplo. Logo, quando se afirma que um agente é autônomo, deve-se ter em mente que ele detém um certo nível de autonomia e não que ele é completamente autônomo. Desta forma, é possível comparar agentes em termos de autonomia, sendo que quanto mais autônomo for o agente, menos auxílio externo ele necessita.

PFEIFER & SHEIER (1999) afirmam que agentes autônomos artificiais são ideais para se estudar os princípios da inteligência. Segundo eles, uma das motivações para o emprego destes agentes envolve a idéia de emergência. Agentes autônomos apresentam comportamentos chamados emergentes, ou seja, comportamentos que surgem pela interação do agente com o ambiente sem que tenham sido programados *a priori* pelo projetista. Por exemplo, processos de navegação autônoma envolvendo múltiplos robôs podem promover a emergência de comportamentos organizados que não são expressos por nenhum destes robôs quando isolados dos demais (CRESTANI, 2001).

1.2 Navegação Autônoma de Robôs e Aprendizagem

A pesquisa de sistemas inteligentes por intermédio de agentes autônomos é classificada como uma metodologia sintética, cuja idéia se resume em "construir para entender". A

abordagem sintética consiste em criar sistemas artificiais que reproduzam aspectos dos sistemas naturais, de modo a entender seus mecanismos internos e assim descobrir como e por que certos eventos ocorrem. O outro paradigma de abordagem, conhecido como analítico, prega a realização de experimentos em um sistema já existente (um ser humano, ou uma colônia de formigas, por exemplo) para então analisar os resultados visando desenvolver um modelo que seja capaz de prever os efeitos de experimentos futuros (PFEIFER & SHEIER, 1999).

Um dos problemas de engenharia mais complexos, desafiadores e propícios à pesquisa de sistemas autônomos inteligentes é a navegação autônoma de robôs. O problema consiste, basicamente, em desenvolver mecanismos de tomada de decisão para um ou mais robôs móveis, dispostos em um ambiente arbitrário junto ao qual devem atuar de forma autônoma, visando cumprir certas tarefas. Muito embora a navegação de robôs possa ser descrita com base nesta breve definição, existem muitos aspectos de projeto envolvidos: configurações do ambiente, modelo do robô, elenco de tarefas e critérios de desempenho. Sendo assim, o desenvolvimento de sistemas de navegação autônomos envolve desafios extremamente complexos para o trabalho de projeto (FIGUEIREDO, 1999).

A complexidade do projeto fica clara quando os aspectos nele envolvidos são esmiuçados. Primeiramente, o ambiente, além de arbitrário, pode apresentar uma conformação não-estruturada e dinâmica, sendo que o robô desconhece, *a priori*, sua topologia. O robô interage com o ambiente apenas por meio de seus sensores e atuadores, que, como é sabido, são freqüentemente de alcance limitado e sujeitos a ruído. Além de navegar pelo ambiente sem nenhum auxílio externo, o robô deve executar certas tarefas predeterminadas, possivelmente conflitantes entre si e muitas vezes de execução simultânea. As tarefas podem ser as mais variadas como: desvio de obstáculos, busca de alvos, coleta de objetos, recarga de energia, mapeamento de território, prospecção, etc.

Particularmente neste trabalho, o problema é caracterizado da seguinte forma: o ambiente, desconhecido, possui obstáculos e alvos dispostos arbitrariamente e a navegação do robô deve se dar visando realizar simultaneamente captura de alvos, um por vez, e desvio de

obstáculos, evitando colisões. Estas duas tarefas de execução concomitante são claramente conflitantes, requerendo a emergência de mecanismos de coordenação de objetivos para tomada de decisão.

Devido à grande variedade de situações com que o robô pode se defrontar ao longo da navegação, sendo algumas inéditas e não previsíveis (sobretudo em ambientes desconhecidos), o desempenho de um sistema de navegação depende de sua capacidade de aprendizagem e adaptação. Ou seja, somente aprimorando sua estratégia de navegação, por meio da incorporação de conhecimento (adquirido por experimentação) e ajuste de parâmetros, um sistema de navegação torna-se apto a guiar eficientemente o robô visando maximizar o atendimento dos objetivos de navegação. Justamente devido à importância da aprendizagem, intensos esforços de pesquisa têm sido dedicados ao aperfeiçoamento desta potencialidade em sistemas de navegação (FIGUEIREDO, 1999). Já no tocante aos mecanismos de adaptação (ajuste de parâmetros), mecanismos sofisticados e eficientes derivados da teoria de controle automático (KUO, 1991) podem ser prontamente empregados.

Um robô controlado por um sistema que possua capacidade de aprendizagem vai vivenciar suas próprias experiências, adquirindo conhecimento ao longo do tempo e por meios próprios, não pela imposição de um agente externo. Fica evidente, portanto, que um robô com capacidade de aprendizado é potencialmente mais autônomo que outro incapaz de aprender. Um benefício da aprendizagem pode ser observado na seguinte situação: supondo que o agente encontre as mesmas circunstâncias pelas quais já passou antes, depois de uma fase de aprendizado ter ocorrido, seu sistema pode reagir de forma diferente da anterior e, possivelmente, com melhores resultados. Fica explícita então a relevância dos mecanismos de aprendizagem na construção de sistemas autônomos inteligentes.

NOLFI & FLOREANO (2000) explicam que a aprendizagem em robôs baseia-se na idéia de que um sistema de controle pode se tornar apto a atuar eficientemente diante de novas situações, sustentando-se na sua capacidade de generalizar conhecimentos adquiridos anteriormente, mesmo que estes sejam diversos e incompletos. Entretanto, generalização não é a única propriedade de um sistema com aprendizagem, existem outras propriedades

importantes. Alguns atributos básicos relacionados a abordagens que contemplam aprendizagem são descritos a seguir (PFEIFER & SHEIER, 1999):

- O sistema deve ser robusto em relação a ruídos;
- Os mecanismos devem convergir rapidamente e têm que permitir a aprendizagem durante a operação do sistema (*on-line*);
- A aprendizagem deve ser incremental e continuada;
- O processo de aprendizagem precisa ser computacionalmente tratável, isto é, deve possibilitar sua execução em tempo real;
- O aprendizado deve depender apenas de informações obtidas pelo próprio robô, por meio de sua capacidade sensorial.

1.3 Abordagens para Sistemas Autônomos Inteligentes

A pesquisa em torno do desenvolvimento de sistemas autônomos inteligentes é intensa e apresenta diversas abordagens. Algumas abordagens promissoras envolvem: sistemas dinâmicos (ARROWSMITH & PLACE, 1990; BAKER & GOLLUP, 1990; JACKSON, 1991), economia comportamental (MCFARLAND & BOSSER, 1993) e esquemas (BARTLETT, 1932; ARKIN, 1993). Entretanto, a abordagem mais consolidada e grandemente utilizada, particularmente em sistemas de navegação autônomos (como são denotados os sistemas autônomos inteligentes dedicados ao problema de navegação de robôs) é a Inteligência Artificial.

A Inteligência Artificial pode ser dividida em duas partes: a IA clássica (tradicional) e a inteligência computacional. A IA clássica reúne técnicas baseadas em lógica proposicional e na manipulação algorítmica de estruturas simbólicas. Já a inteligência computacional, a qual engloba redes neurais artificiais, computação evolutiva, sistemas nebulosos e, mais recentemente, outros mecanismos de computação bio-inspirada, se caracteriza pela síntese de estruturas flexíveis para armazenagem e fluxo de informação, podendo apresentar processamento distribuído, controle descentralizado, auto-organização, não-linearidade, além de recorrer a uma grande variedade de mecanismos de inferência e busca em espaços de atributos (DE CASTRO & VON ZUBEN, 2004).

Existe um consenso em grande parte da comunidade de pesquisa em sistemas inteligentes de que as técnicas clássicas de IA falham em alguns aspectos relativos à aprendizagem e interação com o ambiente. A maioria desses aspectos tem sido bastante discutida na literatura (e.g., BROOKS, 1991; CLANCEY, 1997; FRANKLIN, 1995; HENDRIKS-JANSEN, 1996; WINOGRAD & FLORES, 1986).

Note que, neste trabalho, quando se fala em sistemas de IA clássica, refere-se a sistemas puramente simbólicos. A principal razão para os problemas encontrados na IA clássica é que os modelos de sistemas nem sempre levam suficientemente em consideração as características do mundo real. Muitos trabalhos de IA clássica têm sido dedicados a espaços abstratos, virtuais, cujos estados e operações são precisamente definidos, o que, em geral, ocorre raramente no mundo real. Outras deficiências estão relacionadas à organização hierárquica, processamento centralizado e direcionamento a tarefas bastante específicas (PFEIFER & SHEIER, 1999).

STEELS (1995) vai ainda mais longe, afirmando que sistemas construídos com base na IA clássica não são autônomos, embora sejam automáticos. A afirmação se justifica pelo fato do conhecimento presente nestes sistemas ser geralmente fornecido por especialistas e embutido de forma explícita. Neste sentido, o sistema não tem seus próprios objetivos e motivações, tem apenas os de seus criadores.

Todos estes aspectos conceituais da IA clássica se refletem na ausência ou deficiência da capacidade de aprendizagem pelos sistemas que a utilizam, o que, conseqüentemente, implica em uma redução drástica de sua autonomia. Por isso, críticas têm sido feitas a propostas de sistemas de navegação incapazes de aprender suas estratégias a partir da interação com o ambiente, visto que seus desempenhos estão limitados ao conhecimento neles implantado *a priori*. Supondo que o robô se depare com situações que não estejam previstas pelo sistema de navegação implementado, é provável que seu desempenho sofra degradações significativas e recorrentes frente à repetição do mesmo cenário. Apesar dos problemas, vale ressaltar que existem muitas aplicações bem sucedidas das técnicas de IA em outros tipos de problemas.

Propostas de sistemas de navegação baseadas em técnicas de inteligência computacional têm apresentado resultados promissores, sobretudo devido a elas possuírem características essenciais à concepção de sistemas autônomos inteligentes. Estas técnicas contemplam as seguintes propriedades:

- O emprego de mecanismos de aprendizado na aquisição de conhecimento e adaptação de comportamento a partir da interação com o ambiente;
- A capacidade de encontrar soluções factíveis, que atendam simultaneamente a múltiplos objetivos, possivelmente conflitantes, utilizando o conhecimento adquirido; e
- A habilidade para operar em condições adversas, tais como: ausência de um conjunto de informações completo para um planejamento prévio de seu comportamento, imprevisibilidade na interação com o ambiente (suposto ser de topologia desconhecida) e ruído nos sensores e atuadores (CRESTANI, 2001).

Dentre as técnicas de inteligência computacional, as redes neurais artificiais, cuja inspiração vem do sistema nervoso dos organismos superiores, especificamente do cérebro dos mamíferos, têm um papel muito relevante junto a sistemas de navegação autônomos. As redes neurais artificiais se mostram muito eficazes na síntese de arquiteturas de controle em navegação de robôs, por apresentarem um grande poder de representação de conhecimento e por serem excelentes modelos de aprendizagem e generalização. A seguir, são descritos alguns sistemas de navegação autônomos (SNAs) fundamentados em redes neurais artificiais. FIGUEIREDO (1997) apresenta uma proposta inovadora e eficaz com base em técnicas avançadas de redes neurais artificiais e sistemas nebulosos, capazes de realizar processamento híbrido, com etapas de natureza numérica e simbólica. A arquitetura do sistema é hierárquica e dividida em três módulos principais: dois devotados à tomada de decisão e um terceiro à coordenação. Em CRESTANI (2001), procede-se à extensão do trabalho de FIGUEIREDO (1997), mantendo a arquitetura básica e incluindo extensões como novas formas de operação interna dos módulos da rede, etapas mais elaboradas de coordenação e de aprendizado, além de controle de velocidade. Mais recentemente, HAYDU (2003) implementa um SNA baseado em uma rede neural multicamadas, com diferentes tipos de conexões sinápticas e que realiza aprendizagem por reforço.

Outra ferramenta de inteligência computacional são os sistemas imunológicos artificiais. A idéia é ter uma rede de anticorpos que reage à presença de antígenos, modificando a configuração da rede e tomando ações. Suas maiores virtudes são a capacidade de auto-organização e operação distribuída, conferindo requisitos fundamentais para a síntese de processos cognitivos baseados em comportamento dinâmico. WATANABE *et al*. (1999) investiga um SNA baseado nesta técnica, em que os anticorpos realizam reconhecimento de padrões e estão associados a comportamentos básicos do robô e os antígenos carregam os estímulos capturados pelos sensores. Aperfeiçoamentos do SNA de WATANABE *et al*. (1999) são propostos em MICHELAN (2003), incluindo a aplicação de um mecanismo evolutivo para determinação das conexões da rede imunológica.

De fato, as técnicas de computação evolutiva merecem um grande destaque em navegação autônoma, sendo a principal delas os algoritmos genéticos. Estas técnicas se inspiram basicamente na teoria da evolução das espécies e comungam uma estrutura básica: realizam reprodução, impõem variações aleatórias, promovem competição e executam seleção de indivíduos de uma dada população. Suas principais características são: grande robustez, diversidade e capacidade de busca associada ao atendimento de objetivos específicos. Alguns exemplos de trabalhos com navegação de robôs nesta área são: GREFENSTETTE (1989), WILSON *et al.* (1997) e NOLFI & FLOREANO (2000).

A grande eficiência dos métodos evolutivos viabiliza seu emprego na proposição estrutural de modelos, e não apenas em seu ajuste paramétrico, no desenvolvimento integral de sistemas de controle e na concepção e operação de robôs autônomos. Esta última utilização, conhecida como Robótica Evolucionária (NOLFI & FLOREANO, 2000), é bastante ambiciosa, embora não esteja ainda totalmente consolidada. Ela propõe que todos os módulos do robô, desde seu sistema de controle até sua carcaça, sejam obtidas por métodos evolutivos.

Um quesito muito forte em prol dos sistemas evolutivos se refere à autonomia. Ao se construir um sistema autônomo, seja robótico ou computacional, normalmente designa-se esta atividade a engenheiros. Porém, a própria evolução pode assumir o papel de projetista do sistema (DAWKINS, 1998), de forma eficiente. Além disso, agentes que possuam características evolutivas podem, em princípio, ter seu nível de autonomia bastante

incrementado. A evolução torna o agente mais independente de seu projetista e, portanto, propicia a ele potencial para ser mais autônomo que agentes cujos sistemas de controle não contemplam o evolucionismo (PFEIFER & SHEIER, 1999).

1.4 Proposta de Sistema de Navegação

Tendo em vista todas as vantagens das abordagens evolutivas, principalmente relacionadas à robustez e autonomia, este trabalho propõe e desenvolve um sistema de navegação autônomo evolutivo fundamentado na teoria dos sistemas classificadores com aprendizagem (HOLLAND, 1975).

Basicamente, os sistemas classificadores podem ser descritos como métodos e princípios para criação e atualização de regras de inferência, chamadas classificadores, que codificam eventuais ações a serem tomadas por um agente sob condições específicas do ambiente (BOOKER *et al.*, 1989). Um dos componentes fundamentais do sistema, responsável pela atualização do conteúdo das regras e conseqüente aperfeiçoamento das mesmas, são os algoritmos genéticos. A presença deles confere o caráter evolutivo aos sistemas classificadores. Por serem capazes de combinar, em um único sistema, evolução e aprendizagem de forma integrada, os sistemas classificadores podem ser considerados modelos apropriados para a síntese de sistemas complexos adaptativos (HOLLAND, 1995).

O objetivo deste trabalho é desenvolver e investigar um sistema de navegação autônomo baseado em sistemas classificadores para guiar um robô por ambientes desconhecidos e visando atender múltiplos objetivos, possivelmente conflitantes. O sistema vai ser reativo, ou seja, não possuirá conhecimentos incorporados *a priori* e utilizará apenas estímulos instantâneos capturados do ambiente. O robô é incumbido de duas tarefas: capturar alvos e desviar de obstáculos para tomar decisões. Espera-se que o sistema de navegação, por meio de interação com o ambiente, apresente a emergência destes dois comportamentos, dado que eles não são inatos, e também desenvolva os mecanismos de coordenação dos mesmos, dado que eles podem ser conflitantes. As principais inovações deste sistema de navegação consideram várias modificações nos sistemas classificadores já propostos na literatura (LANZI *et al.*, 2000), diferentes métodos de avaliação e reprodução dos classificadores

durante a etapa evolutiva, um mecanismo de equilíbrio do crescimento de populações submetidas a processos de competição e uma estrutura de controle de velocidade do robô.

1.5 Simulação Computacional e Implementação em Robôs Reais

Na etapa de projeto, a execução e a avaliação de desempenho de um sistema de controle não precisam necessariamente ser realizadas no equipamento real a ser controlado. É possível, graças aos recursos computacionais disponíveis atualmente, realizar simulações empregando modelagem matemática e processamento computacional capazes de reproduzir virtualmente condições de operação em ambientes naturais. Esta opção tem gerado muita discussão entre pesquisadores, enaltecendo as vantagens e desvantagens da simulação computacional quando comparada à implementação em robôs reais.

A simulação computacional é uma estratégia importante por dois motivos básicos: proficiência e tempo. Nem todos os pesquisadores possuem aptidão técnica e disponibilidade de recursos para construir e manipular um robô real. Além disso, a investigação de comportamentos complexos em robôs reais pode demandar testes exaustivos, dificultando ou até inviabilizando a implementação. Para se ter idéia do tempo despendido com experimentos com robôs reais em relação às simulações, NOLFI & FLOREANO (2000) citam um experimento cuja realização completa com robôs reais levou cerca de 66 horas. Caso fosse feito em simulação, o mesmo experimento seria reduzido a apenas 1 hora. Apesar dos benefícios e facilidades do uso da simulação, esta não pode substituir plenamente a experimentação em robôs reais, já que as características de operação do mundo real são extremamente difíceis de serem modeladas computacionalmente com exatidão e completude.

Segundo PFEIFER & SCHEIER (1999), à primeira vista parece melhor usar simulações devido à rapidez, custo reduzido e flexibilidade. Entretanto, sendo o objetivo final a operação em robôs reais, constata-se que simulações computacionais capazes de contemplar todos os aspectos relevantes que estão envolvidos em um experimento real são extremamente difíceis de se realizar. Logo, a conclusão é que ambas as etapas devem ser realizadas de forma complementar, sempre que possível.

Este trabalho foi concebido visando uma maior afinidade com esta linha de pensamento, realizando etapas de simulação computacional em sintonia com etapas de experimentação real. Basicamente, o sistema de navegação autônomo proposto e desenvolvido foi investigado de três maneiras:

- Puramente em simulação computacional;
- Mesclando simulação com navegação empregando um robô real; e
- Puramente em um robô real.

No segundo caso, o intuito foi maturar o sistema de controle em simulação computacional, para em seguida transferi-lo a um robô real visando validá-lo em testes em ambientes reais. O mini-robô adotado nos experimentos reais foi o Khepera II, cujas especificações estão no Apêndice B. Este robô móvel é freqüentemente empregado para fins científicos em diversos laboratórios por todo o mundo. Ele possui oito sensores que medem distância a obstáculos e luminosidade do ambiente. São controláveis, de forma bastante precisa, sua velocidade e direção de movimento.

As simulações computacionais empregadas foram realizadas em um simulador implementado pelo próprio autor deste trabalho especificamente para atender aos requisitos da pesquisa (CAZANGI, 2002). O simulador conta com ferramentas que possibilitam a construção de ambientes virtuais, a realização de diversos tipos de processos de navegação e a análise dos parâmetros envolvidos e desempenho do sistema. Maiores detalhes sobre o ambiente de simulação são apresentados no Apêndice C.

1.6 Aplicações de Navegação Autônoma

Uma das motivações mais importantes para a pesquisa de sistemas de navegação autônomos é a enorme gama de aplicações reais possíveis. Talvez os produtos mais notáveis até hoje obtidos neste domínio tenham sido os robôs *Sojourner*, *Spirit* e *Opportunity*, usados pela NASA. O primeiro robô, em 1996, e os outros dois, em 2003, foram enviados a Marte para exploração da superfície do planeta visando coleta de dados.

Seus sistemas de navegação, apesar de apenas parcialmente autônomos, são uma mostra das potencialidades da pesquisa em navegação autônoma.

Contudo, ao contrário do que se possa pensar, as aplicações de sistemas autônomos inteligentes não se restringem apenas a projetos futuristas de exploração espacial. O potencial prático destes sistemas e equipamentos abrange desde tarefas corriqueiras até mesmo situações de extrema periculosidade.

Entre as aplicações mais comuns, estão tarefas como vigilância, limpeza de superfícies, auxílio a deficientes físicos, aplicações médicas, transporte de materiais e outras. Os casos de trabalhos que envolvem situações de risco ao ser humano também são variados, como no desarmamento de minas, manutenção de material radioativo, limpeza de encanamentos de esgoto, prospecção submarina, sensoriamento remoto, auxílio em resgates e combate a incêndios.

1.7 Organização do texto

Esta dissertação está organizada da seguinte forma. No Capítulo 2, é apresentada a teoria que fundamenta este trabalho, os Sistemas Classificadores, e vários aspectos relacionados à sua aplicação. O sistema de navegação autônomo e todos os seus detalhes são descritos no Capítulo 3. Os experimentos realizados e resultados obtidos são discutidos e analisados no Capítulo 4. Finalmente, as conclusões e propostas futuras estão reunidas no Capítulo 5.

Complementarmente são anexados ao texto alguns apêndices. Os fundamentos de Algoritmos Genéticos são descritos no Apêndice A. O Apêndice B dá especificações do robô Khepera II e, por último, o Apêndice C mostra detalhes do ambiente de simulação desenvolvido.

Capítulo 2
Sistemas Classificadores

2.1 Introdução

Um sistema de tomada de decisão atuando em ambientes complexos e parcialmente desconhecidos se confronta constantemente com situações novas e circunstâncias inesperadas. Problemas do mundo real, não-estacionários e normalmente de otimização multimodal e controle multi-objetivo, são as principais fontes motivadoras para a proposição de soluções na forma de sistemas complexos adaptativos (VARGAS *et al.*, 2002). Para que o sistema opere eficientemente diante destas condições, ele deve possuir meios de reagir de forma adequada às eventuais mudanças e imprevistos. Segundo BOOKER *et al.* (1989), mesmo um sistema formulado com extrema sofisticação e detalhamento cometeria equívocos freqüentemente, desde que não contasse com uma das seguintes possibilidades: um agente externo intervindo de forma continuada e modificando a estrutura do sistema, ou o próprio sistema se atualizando e assimilando novos conhecimentos com base na experiência passada.

Ambientes realísticos, em que atuem sistemas robóticos, ecológicos, econômicos, ou imunológicos não admitem a síntese de processos de intervenção externa eficazes pelo fato de sofrerem intensas e continuadas mudanças de condições. Portanto, somente sistemas com capacidade de auto-adaptação são aptos a atuarem de forma consistente em ambientes com tais propriedades (HOLLAND, 1995).

A capacidade de auto-adaptação é vinculada diretamente à existência de mecanismos de aprendizagem, cuja operação consiste basicamente em perceber regularidades no ambiente de atuação e empregá-las na construção de um modelo ambiental genérico e dinâmico. Além de detectar as regularidades do ambiente a partir de seus componentes elementares, o sistema necessita agrupá-las em categorias significativas e associar a cada uma delas um elenco específico de ações apropriadas.

A extração e classificação de regularidades nem sempre é uma tarefa trivial, dado que muitas vezes elas são identificadas somente após longas seqüências de eventos. Por haver ruídos e dados irrelevantes envolvidos (característicos de ambientes reais), as constantes mudanças de estado do ambiente podem não apresentar uma lógica definida, implicando que as seqüências de eventos observadas no ambiente sejam bastante heterogêneas, o que dificulta ainda mais a manutenção de um modelo representativo para o sistema.

BOOKER *et al.* (1989) dividiu em quatro as principais dificuldades que um sistema com capacidade de aprendizagem precisa tratar:

- Ocorrência continuada de novidades no ambiente, freqüentemente misturadas a dados distorcidos por ruídos ou dados irrelevantes;
- Exigência de tomada de ações em tempo real;
- Objetivos do sistema são implícitos ou imprecisos, além de serem múltiplos e possivelmente conflitantes;
- Dependência de longas seqüências de ações ou eventos para obtenção de realimentação acerca da qualidade das ações e nível de atendimento dos objetivos.

Contemplando a capacidade de aprendizagem por meio de classificação das regularidades do ambiente e visando a atuação em situações que apresentam as características supracitadas, John Holland criou os Sistemas Classificadores com Aprendizagem, em meados dos anos 70 (HOLLAND, 1975; HOLLAND, 1986).

Historicamente, os Sistemas Classificadores surgiram da combinação de duas técnicas, os sistemas especialistas e os algoritmos genéticos (AGs). O grande inconveniente dos sistemas especialistas, a geração e inserção de conhecimento, que era feita manualmente pelos projetistas, foi suprimida pela incorporação dos AGs (RICHARDS, 1995). Apesar de os sistemas classificadores (SCs) possuírem características de sistemas de aprendizagem de máquina (*machine learning systems*), eles são normalmente vinculados à computação evolutiva, pois agregam um algoritmos genéticos (AGs) como componentes fundamentais. O AG, que faz parte do elenco de algoritmos para computação evolutiva, é responsável pela

busca e descoberta de novos conhecimentos e conseqüente aperfeiçoamento do sistema classificador ao longo do tempo.

A seguir, são descritas as principais características dos SCs.

2.2 Estrutura e Operação

Basicamente, um SC é um mecanismo para criação e atualização evolutiva de conhecimento representado por regras, denominadas classificadores, responsáveis pela atuação do sistema em um ambiente arbitrário. Os classificadores codificam alternativas de ações específicas que são submetidas a um processo de competição, definindo aquela que será executada. Também é possível considerar classificadores cujo papel é influir no processo de seleção de classificadores que efetivamente codificam ações. A definição do classificador que será selecionado para atuar depende das condições do ambiente naquele instante e também de um índice de avaliação de cada classificador.

Dependendo do efeito de cada ação (ou seqüência de ações) no atendimento dos objetivos (os quais podem estar implícitos), os classificadores responsáveis pelas ações são recompensados ou punidos, em termos de um fator que mede sua qualidade. Periodicamente, a população de classificadores é submetida a um processo evolutivo, realizado por meio de Algoritmos Genéticos.

O SC interage com o ambiente através de sensores e atuadores (Figura 2-1). Os sensores são responsáveis pela recepção e codificação das mensagens (em forma de estímulos) provenientes do ambiente, transferindo-as em linguagem inteligível ao sistema. Os atuadores, por sua vez, decodificam as ações tomadas pelo sistema, para que as mesmas possam ser executadas no ambiente. As conseqüências causadas pelas ações, indicadas pela realimentação, determinam a recompensa ou punição adequada aos respectivos classificadores.

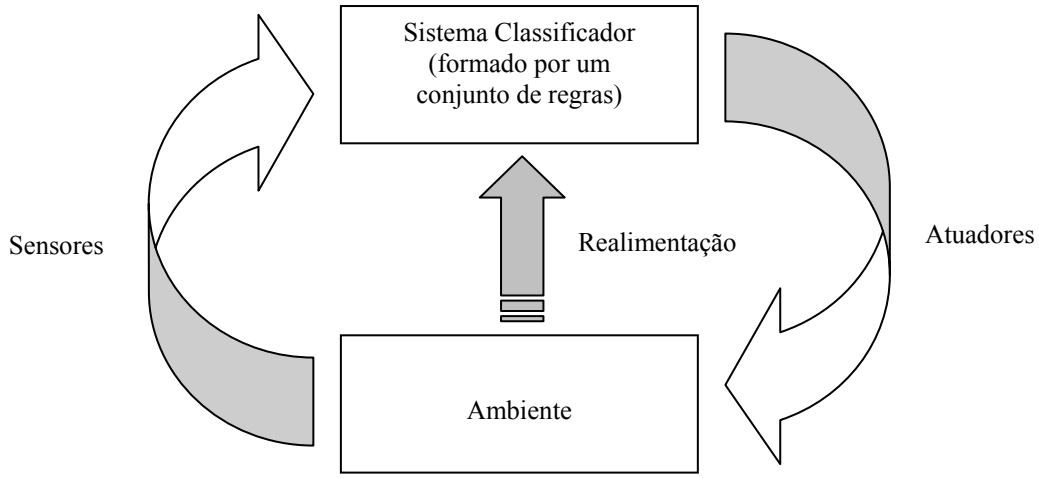


Figura 2-1: Interação do SC com o ambiente.

Embora o estudo dos SCs seja relativamente novo, existem duas variantes para formação dos classificadores: a abordagem *Pittsburgh* (DEJONG, 1988), na qual um indivíduo representa a solução para o problema, e a abordagem *Michigan*, cuja solução é dada pela população como um todo (neste caso, existe essencialmente uma única solução sendo evoluída) (MICHALEWICZ, 1996).

Este trabalho adota e descreve os fundamentos da abordagem *Michigan*, cujos componentes principais, mostrados na Figura 2-2, são:

- Módulo de tratamento de regras e mensagens;
- Módulo de atribuição de crédito;
- Módulo de descoberta de novas regras.

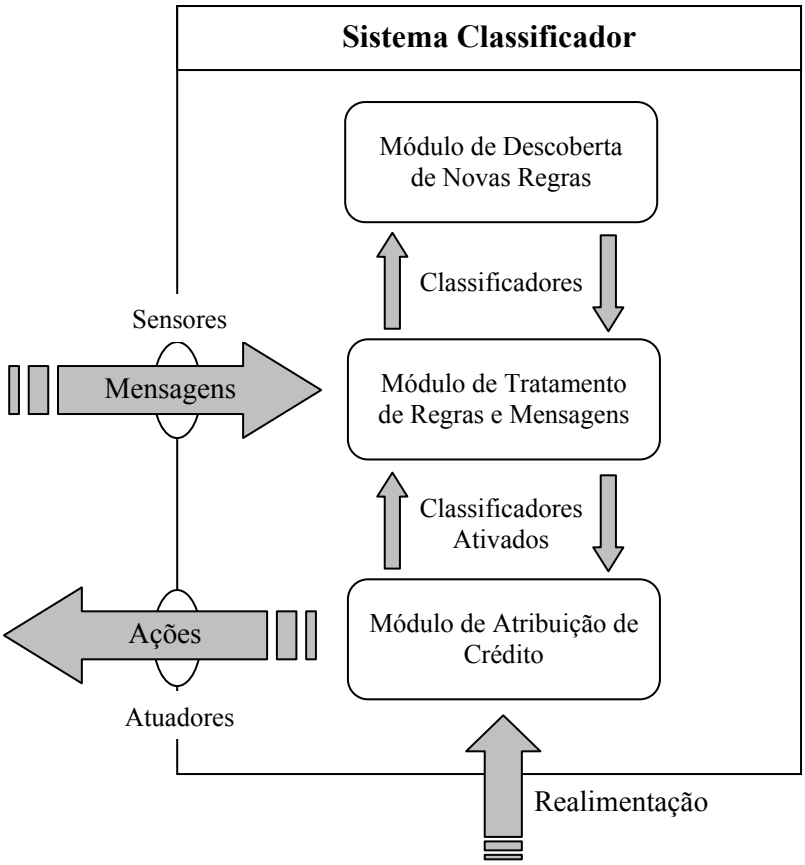


Figura 2-2: Diagrama geral de um SC.

2.2.1 Regras, Especificidade e Energia

O núcleo de um Sistema Classificador é composto por uma população de regras de inferência do tipo antecedente-conseqüente (i.e., Se-Então), denominados classificadores. A parte antecedente é composta por uma condição, a ser satisfeita, e o conseqüente é formado por uma ação. Todas as regras seguem, então, ao seguinte modelo:

```
SE <condição> - ENTÃO <ação>
```

Na proposta original, os termos `<condição>` são codificados como vetores, de tamanho fixo, do alfabeto {0, 1, #} e o termo `<ação>` é formado pelo alfabeto {0, 1}. O símbolo "#" (*don´t care*) pode valer tanto 0 quanto 1, dependendo da situação. Ele permite a existência de regras genéricas sendo que, quanto mais símbolos "#" estiverem presentes em uma regra, mais genérica ela será. O conceito usado para quantificar esta propriedade é

chamado especificidade, que é uma medida inversamente proporcional à quantidade de símbolos "#". Assim, se um classificador possui todos os caracteres de seu antecedente iguais a "#", sua especificidade é zero; caso não haja nenhum destes símbolos, a especificidade será máxima.

Nem sempre a representação das regras por codificação binária é apropriada. Em certos problemas, principalmente quando o espaço de busca é de dimensão elevada, a codificação binária pode levar a um desempenho insatisfatório. Uma alternativa é o uso de codificação com ponto flutuante (números reais) ou inteiros. MICHALEWICZ (1996) apresenta simulações computacionais comparando o desempenho de algoritmos genéticos com codificação binária e com ponto flutuante, aplicados a um problema de controle. Os resultados apresentados mostram uma clara superioridade da codificação em ponto flutuante.

Todo classificador possui uma energia específica, a qual é diretamente associada à sua utilidade dentro da população, ao longo de sua atuação. Basicamente, essa medida é determinada de acordo com o desempenho médio do classificador ao longo do tempo, levando-se em consideração a realimentação recebida para cada atuação do classificador. Isto é, quanto maior a energia de um classificador, melhor terá sido seu desempenho no sistema, e maior será sua probabilidade de atuar no ambiente e de perpetuar suas características nas futuras gerações, caso os objetivos a serem atingidos e as características do ambiente não sofram variações significativas.

2.2.2 Módulo de Tratamento de Regras e Mensagens

O processo de tratamento de mensagens é disparado quando os sensores capturam alguma mensagem do ambiente, por intermédio de estímulos sensoriais. Logo a seguir, ela é codificada de modo compreensível ao SC e inserida em um processo de competição. Nesse processo, a parte antecedente de todos os classificadores é comparada com a mensagem, tentando verificar o(s) classificador(es) cujas condições casam com ela. Aqueles que forem satisfeitos são ativados e passam a concorrer ao direito de atuar no ambiente.

É possível ainda a existência de classificadores postadores de mensagens, ou seja, cuja ação é como uma nova mensagem ambiental. Neste caso, além das mensagens do ambiente, passam a existir mensagens geradas internamente. Este tipo de classificador não será considerado neste trabalho, pois o processo de atribuição de créditos (ver seção 2.2.3) fica muito complexo pela existência de uma cascata de classificadores responsáveis pela ação e também por não se mostrarem convenientes ao problema tratado.

Se houver mais de um classificador ativado no processo de competição, é necessário definir o vencedor. Isto é feito analisando a energia de cada classificador; aqueles que possuírem mais energia têm mais chance de vencer e ter assim sua ação traduzida em uma linguagem apropriada para modificação do ambiente pelos atuadores. Caso nenhum classificador seja ativado, poderá atuar aquele com antecedente de maior semelhança com a mensagem recebida.

A competição e ativação dos classificadores, mostrada na Figura 2-3, pode ser feita de diversas formas, usando métricas (para casamento de padrões entre mensagem e antecedente das regras) e processos de seleção específicos (para determinação do classificador vencedor).

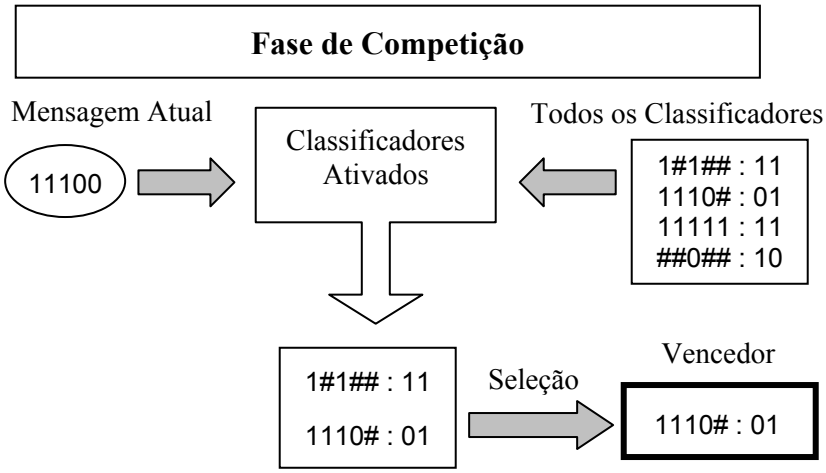


Figura 2-3: Exemplo da fase de competição entre classificadores. As energias dos classificadores (seção 2.2.3.3), empregadas na seleção do vencedor, não estão apresentadas.

2.2.3 Módulo de Atribuição de Crédito

É conhecido o papel fundamental que o problema de atribuição de crédito (MINSKY, 1963) tem dentro de qualquer sistema de aprendizagem, quando várias partes interagem visando determinar o desempenho global deste sistema. Considerando as situações em que mais de um classificador colabora, durante um certo período de tempo, para a obtenção de um objetivo desejado, é necessário avaliar qual, e de quanto, foi a contribuição de cada um deles. Esta é a definição do problema de atribuição de créditos, do qual depende boa parte do sucesso ou fracasso do sistema classificador.

A tarefa é difícil, pois, em ambientes complexos, a sinalização de que a atuação de um classificador foi positiva ou negativa raramente é explícita. Isto é, a sinalização geralmente fica clara algum tempo após a atuação do sistema. Além disso, por haver a possibilidade de vários classificadores serem ativados em paralelo e, devido à constante ocorrência de novidades, nunca o sistema pode ter absoluta certeza acerca da adequação de suas ações (BOOKER *et al.*, 1989).

Existem diversos tipos de algoritmos de atribuição de crédito para aplicação em Sistemas Classificadores, que podem, inclusive, ser personalizados de acordo com o problema. Alguns deles são mais eficientes que outros, mas todos devem garantir a aprendizagem. O algoritmo mais disseminado é conhecido como *Bucket Brigade.* Este algoritmo foi especialmente proposto para problemas em que a realimentação é espaçada e que envolvam sistemas classificadores que contemplam classificadores postadores de mensagens, ou seja, classificadores que ativam outros classificadores. Diante destas condições, o *Bucket Brigade* é capaz de identificar e recompensar (ou punir) todos os classificadores que levaram a uma certa modificação do ambiente, mesmo que esta modificação tenha sido detectada apenas após uma seqüência de classificadores atuantes. Apesar de bastante utilizado, este método apresenta dificuldades quando o sistema classificador não admite classificadores postadores de mensagens. Neste caso, uma alternativa mais simples, quando possível, é a implementação do SC segundo a abordagem de estímulo-resposta (E-R). Neste

caso, apenas um classificador é ativado para atuar no ambiente, simplificando o processo de recompensa ou punição (RICHARDS, 1995).

Os processos de perda e ganho de energia por parte dos classificadores são executados por três mecanismos inter-relacionados: leilão, recompensa e taxação. A atualização da energia dos classificadores é dependente da realimentação do ambiente e cada um dos processos será descrito a seguir. Embora os mecanismos citados não sejam utilizados no sistema implementado neste trabalho (por motivos explicados na seção 3.4), eles são descritos a seguir por fazerem parte dos sistemas classificadores com aprendizado originais.

2.2.3.1 Leilão

Quando o SC recebe mensagens do ambiente, todos os classificadores que são selecionados competem, fazendo lances em uma espécie de leilão para determinar o vencedor, que então atuará no ambiente. Cada lance é calculado em função da energia e da especificidade do respectivo classificador. A regra vencedora paga seu lance, descontando de sua energia a quantia ofertada. Os lances são calculados conforme a equação (2-1):

$$Lance_t = k_0 * \left(k_1 + k_2 * Spec^{SPow}\right) * S_t \qquad (2\text{-}1)$$

onde:

- $Lance_t$ é o lance de um classificador na iteração t.
- k_0 é uma constante positiva menor que 1 que age como um fator de risco global, influenciando na fração de energia do classificador, que será ofertada e possivelmente subtraída;
- k_1 representa um coeficiente constante, positivo e menor que 1 para a porção não específica do lance;
- k_2 representa um coeficiente constante, positivo e menor que 1 para a porção específica do lance;
- S_t representa a energia do classificador na iteração t;
- *Spec* é a especificidade do classificador (definida na equação (2-2)), associada à proporção de símbolos "#" no vetor;

- *SPow* é um parâmetro de controle da influência da especificidade no valor do lance (normalmente igual a 1).

A especificidade do classificador é definida pela seguinte equação:

$$Spec = \frac{N - total\_\#}{N} \qquad (0 \leq Spec \leq 1) \qquad (2\text{-}2)$$

onde:

- N representa o tamanho da parte antecedente do classificador;
- $total\_\#$ é a quantidade total de símbolos "#" (*don't care*) presentes na parte antecedente do classificador.

2.2.3.2 Recompensa

Dependendo do efeito da ação do classificador vitorioso (realimentação), que pode ser favorável ou prejudicial ao ambiente, aplica-se a ele o mecanismo de recompensa (ou punição) que estabelecerá sua nova energia, segundo a equação (2-3):

$$S_{t+1} = S_t + R_t - Lance_t \qquad (2\text{-}3)$$

onde:

- S_t representa a energia do classificador na iteração t;
- R_t é um valor baseado na realimentação dada pelo ambiente, caso o classificador tenha sido vencedor da competição na iteração $t-1$, sendo que:
  - $R_t = 0$ caso o classificador não tenha sido vencedor;
  - $R_t > 0$ em caso de recompensa (ação positiva);
  - $R_t < 0$ em caso de punição (ação negativa).

- $Lance_t$ representa o lance (equação 2-1) do classificador na iteração t ($Lance_t = 0$ caso o classificador não tenha sido vitorioso na competição).

2.2.3.3 Taxação

Na etapa de taxação, todos os classificadores sofrerão um decréscimo em sua energia correspondente à taxa de vida. Na equação (2-4), tem-se o cálculo desta taxa, a qual é cobrada de cada classificador a cada iteração.

$$Tax_{life} = 1 - \left(\frac{1}{2}\right)^{\frac{1}{n}} \tag{2-4}$$

onde *n* representa a meia vida do classificador (definida em número de iterações).

Uma taxa de participação no leilão, representada na equação (2-5), é cobrada de cada classificador que participou da competição.

$$Tax_{bid} = k * Lance_t \tag{2-5}$$

onde:

- *k* é uma constante aplicada sobre o lance do classificador;
- $Lance_t$ representa o lance do classificador (equação (2-1)).

Tendo sido definidos todos os mecanismos de atribuição de crédito, chega-se à fórmula geral de cálculo da nova energia do classificador, a cada iteração, representada na equação (2-6):

$$S_{t+1} = \left(1 - Tax_{life}\right) * S_t + R_t - Bid_t - Tax_{bid} \tag{2-6}$$

onde as variáveis seguem as descrições e restrições especificadas nas equações (2-1) a (2-5).

Mais detalhes acerca das motivações que conduziram à formulação das equações (2-1) a (2-6), e também acerca das propostas de valores para os parâmetros envolvidos, podem ser encontrados em RICHARDS (1995).

2.2.4 Módulo de Descoberta de Novas Regras

Os SCs precisam ser capazes de, quando necessário, substituir os classificadores (regras) de pouca utilidade presentes na população atual por novos classificadores e tentar aprimorar os demais. Este procedimento é necessário porque as populações de regras iniciais, geralmente pouco refinadas, representam uma quantidade ínfima de soluções diante do imenso espaço de busca encontrado em problemas que envolvem ambientes complexos. Mesmo as melhores regras destas populações podem ser consideradas extremamente fracas em face do potencial de aperfeiçoamento existente. Além disso, ao longo de sua atuação, o SC deve atualizar seus classificadores e criar outros novos em resposta às mudanças ambientais, as quais podem fazer com que o sistema perca desempenho.

Há diversos mecanismos passíveis de geração de novos conhecimentos em forma de regras. As novas regras podem ser geradas aleatoriamente ou mesmo por procedimentos previamente definidos. No entanto, mecanismos que independem da experiência e/ou que não privilegiam algumas propostas em detrimento de outras, evoluem muito devagar, sendo inviáveis em ambientes reais. A opção é por mecanismos que aproveitem a experiência acumulada pelo sistema, incorporada ao conteúdo das regras, para produzir regras cada vez mais refinadas (BOOKER *et al.*, 1989).

Neste contexto, o principal mecanismo empregado para descoberta de classificadores são os Algoritmos Genéticos (AG). Ao final de cada época de iterações, o AG e seus procedimentos evolutivos entram em ação em busca de uma população de classificadores cada vez mais adaptada ao ambiente, guiados pelas metas a serem alcançadas.

Os Algoritmos Genéticos evoluem uma população de indivíduos, que são potenciais soluções ao problema tratado, por meio de operadores genéticos de seleção, cruzamento e mutação. A avaliação dos indivíduos é feita por uma função de aptidão (*fitness*) definida segundo os objetivos a serem alcançados. Quando aplicados aos SC, os indivíduos do AG são os classificadores e a função de aptidão é associada aos objetivos a serem atendidos. De

forma simplificada, o diagrama de funcionamento do módulo de descoberta de novas regras é mostrado na Figura 2-4. Maiores detalhes sobre os AG são apresentados no Apêndice A.

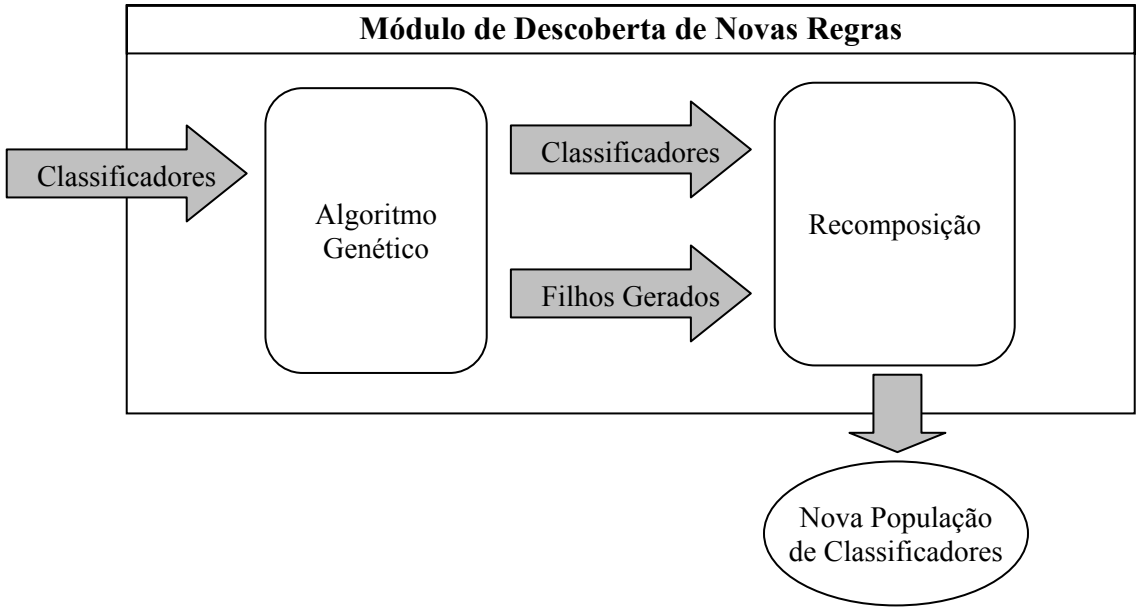


Figura 2-4: Diagrama do mecanismo de descoberta de novas regras.

Uma iteração do SC é definida como um processo que se inicia no recebimento da mensagem proveniente do ambiente e se encerra com a tomada de decisão do sistema e atuação no ambiente. Este ciclo contínuo de recebimento de mensagem e atuação do sistema somente é interrompido quando chega ao fim uma época de iterações. Portanto, uma época representa um ciclo finito de iterações, sendo que em cada iteração apenas um classificador atua. O encerramento de uma época depende de critérios de parada que podem estar relacionados à variação de desempenho do sistema, ou mesmo a um número fixo de iterações. Como já foi mencionado, o final de uma época coincide com o disparo dos procedimentos evolutivos. Quando os processos de evolução são concluídos, atualizando a população de classificadores, uma nova época de iterações é iniciada e o SC volta a atuar. Um exemplo de um ciclo de épocas e iterações, com os respectivos classificadores atuantes, é apresentado na Figura 2-5.

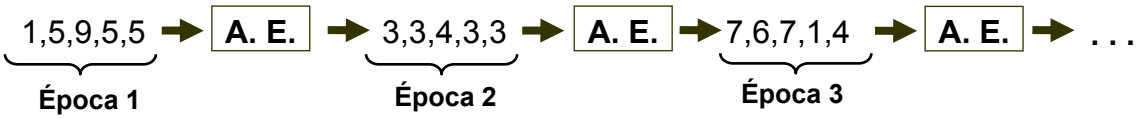


Figura 2-5: Exemplo de seqüências de classificadores atuantes (épocas de iterações) seguidas pela aplicação de um algoritmo evolutivo (A.E.)

2.3 Algoritmo Simplificado

Agora que os mecanismos dos sistemas classificadores foram detalhados separadamente, é interessante mostrar uma visão global do funcionamento do sistema de forma a facilitar sua compreensão. Por meio do algoritmo simplificado, mostrado na Figura 2-6, podem-se visualizar os passos do sistema como um todo.

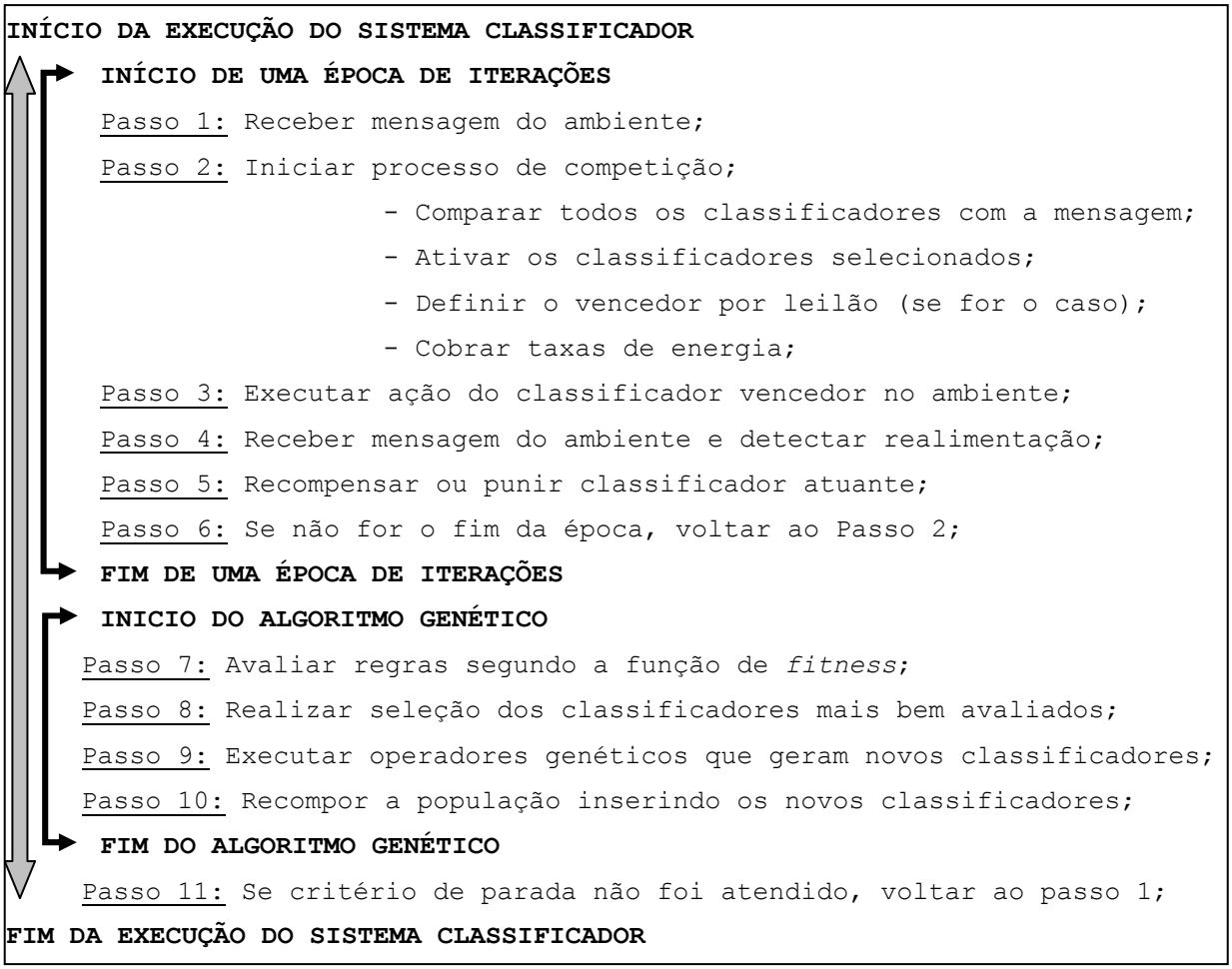


Figura 2-6: Algoritmo simplificado do SC.

2.4 Adequação Funcional à Navegação Autônoma de Robôs

Devido a certas características envolvidas no problema de navegação autônoma de robôs estudado neste trabalho, o sistema classificador implementado, embora se fundamente nos sistemas classificadores de Holland, difere em alguns aspectos relevantes do original. As

alterações funcionais foram necessárias tendo em vista a adequação do sistema e viabilização de sua aplicação ao problema. As adaptações estão relacionadas, sobretudo, ao modelo de robô utilizado, às propriedades do ambiente e às tarefas propostas. Todas as adequações e seus detalhes são apresentados no Capítulo 3.

Tabela 2-1: Variantes de sistemas classificadores (adaptado de KOVACS (2000)).

Ano	Modelo	Descrição	Autores
1975	CS	Classifier System	John Holland (HOLLAND, 1992)
1985	LCS	Learning CS	John Holland (HOLLAND, 1986)
1986	CSM	CS with Memory	Hayong Harry Zhou (ZHOU, 1985)
1989	HCS	Hierarchy CS	Lingyan Shu and Jonathan Schaeffer (SHU & SCHAEFFER,1989)
1989	SCS	Simple CS	David Goldberg (GOLDBERG, 1989)
1989	VCS	Variable CS	Lingyan Shu and Jonathan Schaeffer (SHU & SCHAEFFER,1989)
1990	PCS	Predictive CS	Piet Spiessens (SPIESSENS, 1990 in KOVACS & LANZI (1999))
1994	FCS	Fuzzy CS	Takeshi Furuhashi, Ken Nakaoka eYoshiki Uchikawa (FURUHASHI *et al*., 1994)
1994	ZCS	Zeroth-level CS	Stewart W. Wilson (WILSON, 1994)
1995	OCS	Organizational CS	Jason Wilcox (WILCOX, 1995)
1995	XCS	Special CS	Stewart W. Wilson (WILSON, 1995)
1996	ACS	Anticipatory CS	Wolfgang Stolzmann (STOLZMANN, 1996)
1999	CCS	Corporate CS	Andy Tomlinson and Larry Bull (TOMLINSON & BULL, 1999)
2001	YACS	Yet Another CS	Pierre Gérard & Olivier Sigaud (GÉRARD & SIGAUD, 2001)
2003	MACS	Modular Anticipatory CS	Pierre Gérard e Olivier Sigaud (GÉRARD & SIGAUD, 2003)

2.5 Mecanismos Alternativos e Variações de Sistemas Classificadores

O Sistema Classificador apresentado neste capítulo corresponde à versão original. A partir dela, outros pesquisadores propuseram novas versões contendo variações em seus mecanismos e estrutura. A Tabela 2-1 mostra as diferentes variantes desenvolvidas.

Embora existam mais de uma dezena de sistemas classificadores, como variantes do sistema original, dois deles têm maior destaque e merecem uma breve descrição feita a

seguir. São eles os Sistemas Classificadores Especiais (XCS) e os Sistemas Classificadores Antecipatórios (ACS). Antes, o algoritmo de *Bucket Brigade*, muito utilizado e de papel fundamental no sistema classificador de Holland, será também brevemente descrito.

2.5.1 Algoritmo de Bucket Brigade

Dado que no SC de Holland cada classificador possui uma certa energia, o algoritmo de *bucket brigade* é responsável por atualizá-la de modo a refletir a utilidade geral da regra para o sistema. Para isto, todas as regras que forem ativadas na etapa de competição (módulo de tratamento de regras e mensagens) devem fazer um lance de acordo com sua quantidade de energia, como em um leilão. Aquela que ofertar o maior lance é a vencedora e poderá atuar no ambiente, pagando, é claro, a energia ofertada. Os classificadores de atuação benéfica recebem uma recompensa, ou seja, sua energia é aumentada.

(BOOKER *et al.*, 1989) descreve a operação do algoritmo de *bucket brigade*, de modo informal, fazendo uma analogia com as operações econômicas. Segundo esta visão, cada regra tem o papel de um intermediário em uma transação financeira. Como intermediária, uma regra negocia apenas com seus fornecedores (regras que por meio de sua atuação ativaram a regra chamada intermediária) e seus consumidores (eventuais regras ativadas por causa da atuação da regra intermediária). Ao vencer um leilão, a regra deve pagar a quantidade ofertada de energia aos seus fornecedores. A seguir, ela atuará ativando outras regras (consumidores), servindo desta vez como fornecedor dos seus consumidores e recebendo o pagamento deles em forma de energia. Assim, a energia das regras funciona como um capital a ser investido em busca de lucro. O lucro se realiza quando a regra recebe mais dos seus consumidores do que pagou aos fornecedores, tendo então um ganho de energia. Uma regra tem maior chance de ter lucro se seus consumidores, nas suas transações locais, também têm, na média, obtido lucro. Os consumidores, por sua vez, terão lucro apenas se os seus consumidores derem lucro, e assim por diante. A seqüência resultante leva ao último consumidor, a regra que efetivamente cumpriu o objetivo desejado e, portanto, que receberá um pagamento de energia diretamente do ambiente. Por exemplo, se uma seqüência de regras é falha, a regra final perderá energia e, ao longo do tempo, a seqüência tenderá a ser eliminada.

2.5.2 Sistemas Classificadores Especiais

Em 1995, Stewart Wilson introduziu um novo formato de sistemas classificadores com aprendizagem, chamado sistemas classificadores especiais (XCS) (WILSON, 1995). Como citado anteriormente, o algoritmo de *Bucket Brigade* distribui energia aos classificadores segundo sua utilidade ao sistema. As grandes inovações dos XCS consistem em basear a distribuição de energia na precisão das regras, ao invés da utilidade, e em realizar o algoritmo genético com uma população de classificadores restrita (como nichos ecológicos), ou seja, participam apenas aqueles que foram satisfeitos pela mensagem de entrada e que têm a mesma ação da regra atuante (BUTZ & WILSON, 2001).

A grande deficiência dos XCS se dá pela incapacidade de aprendizagem do sistema em ambientes não-markovianos. Ambientes que violam a propriedade de Markov são aqueles em que os estímulos capturados pelos sensores de um agente fornecem apenas informações parciais sobre o ambiente, isto é, pode haver diversas situações que parecem idênticas para o agente, mas requerem ações distintas para obter sucesso (LANZI & WILSON, 2000). Além disso, o sistema classificador de Wilson encontra muitas dificuldades em ambientes cuja realimentação é demorada e esparsa. Devido às deficiências mencionadas, o XCS falha no tratamento de aspectos presentes nos ambientes de navegação autônoma de robôs. A ambigüidade de situações e a dependência de longas seqüências de ações para obtenção de realimentação do ambiente ocorrem freqüentemente, exigindo uma resposta adequada do sistema. Por esses motivos, alguns cuidados deveriam ser tomados caso o XCS fosse utilizado como sistema de controle de robôs móveis para o problema abordado neste trabalho.

2.5.3 Sistemas Classificadores Antecipatórios

Incorporando aos sistemas classificadores padrão um mecanismo de aprendizagem conhecido como controle antecipatório comportamental, Wolfgang Stolzmann criou os Sistemas Classificadores Antecipatórios (ACS), em 1997.

As regras que representam o conhecimento de um ACS possuem um componente a mais, isto é, elas são formadas por três partes: condição, ação e efeito. A idéia é, por meio de comparações entre a predição feita pela regra (efeito) e as conseqüências efetivamente causadas por sua ação no ambiente, construir classificadores capazes de antecipar corretamente as mudanças do ambiente e seus futuros estados (STOLZMANN, 1997).

Uma das principais vantagens deste sistema classificador é sua capacidade de aprendizagem latente. Isto é, mesmo que o sistema não receba por algum tempo (ou nem sequer tenha) alguma realimentação do ambiente, ele possui meios de continuar aprendendo e aperfeiçoando seus conhecimentos. Outra vantagem é a possibilidade da construção, de forma incremental, de um mapa ou representação do ambiente que pode ser usado para realização de ações com base em planejamento.

Apesar de muito promissora, a abordagem antecipatória, implementada em uma versão preliminar, não mostrou bons resultados em problemas simples de navegação autônoma testados neste trabalho. Seu desempenho abaixo do esperado se justifica em virtude de sua implementação não ter aproveitado todos os sofisticados mecanismos que fazem parte da teoria de ACS e, também, por dificuldades encontradas diante de certas situações. Segundo (BUTZ *et al.*, 1999), o ACS assume que o ambiente de atuação seja determinístico. Mais especificamente, a presença de ruídos, de mudanças irrelevantes ou aleatórias no ambiente promovem dificuldades ao mecanismo de aprendizagem. As dificuldades surgem principalmente em ambientes não-markovianos, e quando mais de um agente atua modificando o ambiente.

Contudo, muito se tem pesquisado a respeito de ACS recentemente e resultados significativos têm sido apresentados, inclusive propondo aperfeiçoamentos que efetivamente superam boa parte das deficiências constatadas originalmente. As perspectivas futuras para esta classe de sistemas classificadores são animadoras, visto que os mecanismos envolvidos são eficazes e soluções para os problemas detectados têm sido propostas (HOLLEY, 2004).

Capítulo 3
Sistema de Navegação Autônomo

3.1 Introdução

Neste capítulo, as características do robô e do sistema de navegação autônomo (SNA) proposto, assim como sua estrutura e mecanismos, são descritos detalhadamente. Boa parte do desenvolvimento do sistema foi apresentada em CAZANGI & FIGUEIREDO (2002) e CAZANGI *et al*. (2003a, 2003b), sendo sempre norteado pelos princípios colocados a seguir. Este sistema de navegação é empregado na tarefa de controle autônomo em simulação computacional e também em experimentos com o robô Khepera II.

A concepção e projeto do SNA se baseiam em sistemas classificadores e algoritmos genéticos, técnicas evolutivas que fazem parte da área de inteligência computacional. Além disso, também são empregados alguns conceitos provenientes da psicologia, relacionados ao fenômeno da aprendizagem por condicionamento operante.

A idéia fundamental é promover a autonomia do sistema, dotando-o de capacidade de aprendizagem e adaptação mediante interação com o ambiente. Desta forma, a aquisição de conhecimento supõe que o sistema autônomo seja capaz de receber, filtrar, representar e processar sinais capturados pelos sensores, de modo a desenvolver suas próprias estratégias de navegação. Além disso, o trabalho também é orientado pelo preceito de evitar a incorporação, em fase de projeto, de conhecimento abstrato. O conhecimento abstrato é definido aqui, sem formalismo, como o conhecimento resultante das capacidades sofisticadas da inteligência humana.

O SNA se caracteriza por ser reativo, pois não possui conhecimento incorporado *a priori* e utiliza apenas estímulos instantâneos capturados do ambiente pelos sensores do robô para navegar. Dado que o robô deve capturar alvos e desviar de obstáculos, o objetivo é, portanto, que o sistema de navegação, por meio de interação com o ambiente, apresente a

emergência destes dois comportamentos e, ao mesmo tempo, aprenda a coordená-los adequadamente.

A emergência e coordenação de comportamentos é resultado de processos de aprendizagem e adaptação, que, desta forma, garantem a autonomia e flexibilidade do sistema. Supondo que eventuais mudanças de cenário se apresentem durante o processo de navegação e impliquem em um comprometimento de desempenho no atendimento dos objetivos da navegação, o sistema de navegação deve disparar automaticamente uma seqüência de eventos responsáveis pela adaptação à nova condição do ambiente, visando recompor o nível de atendimento dos objetivos da navegação. Em outras palavras, deve-se promover a síntese automática de novas estratégias de navegação, compatíveis com o novo panorama.

Muitas das propriedades do SNA foram moldadas de acordo com as características e restrições do robô a ser controlado, como os sensores e atuadores. Na próxima seção, o robô adotado para experimentos práticos é apresentado em detalhes.

3.2 Características do Robô (Apêndice B)

O robô móvel utilizado foi o Khepera II, mostrado na Figura 3-1, é bastante conhecido e empregado em laboratórios por todo o mundo, em diversas linhas de pesquisa científica. O mini-robô possui 80 mm de diâmetro, 30 mm de altura e seu peso é de aproximadamente 80 gramas. Ele se movimenta por meio de duas rodas laterais independentes, cada qual controlada por um motor de passo, podendo assim girar com precisão para ambos os sentidos. Os ajustes de direção incrementais determinados pelo sistema de navegação podem variar de 15º (sentido horário) a −15º (anti-horário) a cada movimento. Este mecanismo confere ao robô a capacidade de executar manobras curtas tanto para sua direita como para sua esquerda. A margem de manobra foi estabelecida arbitrariamente na estrutura do sistema de navegação e não corresponde a restrições reais do robô, isto é, o robô Khepera II, quando independente do controlador, admite qualquer variação de direção. A velocidade do robô pode variar de 0 a 1m/s. Cada iteração do SNA corresponde a um

ciclo sensório-motor do robô, isto é, as informações capturadas pelos sensores são processadas pelo sistema, que por sua vez, determina a nova direção e velocidade do robô.

Figura 3-1: Mini-robô Khepera II.

O Khepera foi desenvolvido por um grupo de pesquisadores do *Microprocessor and Interface Laboratory* (LAMI) da Suíça, especialmente voltado à utilização em pesquisas científicas. Sua concepção obedeceu a alguns critérios como: miniaturização, modularidade e interface simples. Por ser pequeno, o ambiente de navegação pode ser construído ocupando espaços reduzidos. O robô é modular e expansível, admitindo a adição de novos componentes, como outros sensores (câmeras) e outros atuadores (garras). Os modos de comunicação com o computador são de dois tipos: por sinais de rádio ou por conexão serial via cabo.

A aquisição de informações do ambiente é feita por 8 sensores infravermelhos que capturam estímulos do ambiente. Segundo o fabricante do robô (K-Team S.A.), uma leitura completa de todos os sensores é feita a cada 20 ms, dado que cada sensor leva 2,5 ms para fazer uma medição. Maiores detalhes em relação às especificações técnicas do robô encontram-se no Apêndice B. Esses estímulos, capturados pelos sensores, correspondem a uma leitura instantânea da situação do robô no ambiente, a partir do ponto de vista do robô (proximal). Os sensores possuem duas funcionalidades: medida de luminosidade e medida de distância a obstáculos. No primeiro caso, os sensores (aqui chamados de sensores de alvo) medem a quantidade de luz infravermelha no ambiente, que é equivalente à quantidade de luz visível. Na outra funcionalidade, os sensores de obstáculo aferem a distância entre o robô e o obstáculo situado em frente ao respectivo sensor, por meio da

emissão e reflexão de raios infravermelhos. As duas funcionalidades, estão conjugadas em cada um dos 8 sensores.

Os 8 sensores são dispostos ao redor do robô, sendo 6 na frente e 2 atrás, conforme é mostrado na Figura 3-2. As medidas de distância a obstáculos são feitas por sensores infravermelhos, cujo máximo alcance é de 100 mm. Pelo fato de o robô não se movimentar para trás, somente os 6 sensores frontais são aproveitados para determinação de distância a obstáculos. No entanto, caso fossem admitidos giros completos de 180º durante a navegação, a informação de distância a obstáculos dos sensores traseiros seria muito relevante.

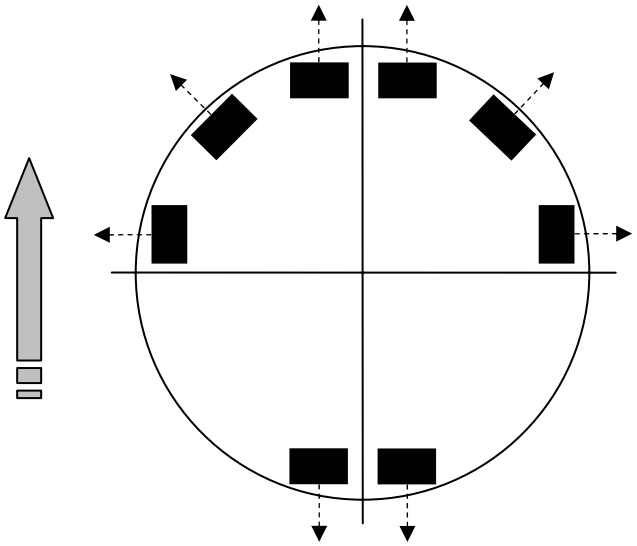


Figura 3-2: Distribuição dos sensores na estrutura do robô, cuja frente é apontada pela seta.

Em relação à medida de luminosidade, todos os sensores são empregados, inclusive os traseiros. Dado que o alvo é uma fonte luminosa, sua presença quebra a uniformidade de iluminação do ambiente, de modo que passam a existir gradientes de luminosidade. Estes gradientes servirão como referência ao sistema de navegação para estimar a direção e a distância da lâmpada, contribuindo assim para guiar o robô em busca do alvo.

Há ainda 3 sensores virtuais: um detecta colisão em obstáculos, o outro captura de alvo e um terceiro sensor, do tipo proprioceptivo (detecta estímulos internos ao sistema, ou seja, não provenientes diretamente do ambiente externo), que tem a função de detectar eventos

de monotonia (descritos na seção 3.4.3). A saída desses sensores (colisão, captura e monotonia) assume um valor binário, ou seja, 1 se o evento foi detectado, e 0 caso contrário.

Quando o robô colide, ele imediatamente gira 180º, como se fosse uma reação instintiva. Este é o único momento em que é admitido um desvio de direção acima de 15º, em módulo. Em qualquer instante, sempre há um único alvo no ambiente. No caso dele ser capturado, outro alvo é prontamente inserido em uma nova posição. Tanto os alvos como os obstáculos possuem posição fixa no ambiente.

3.3 Simulador

O simulador foi implementado de forma a possibilitar a realização de experimentos computacionais com o SNA. Neste sentido, buscou-se incorporar ao simulador as principais características de um ambiente real, para que as simulações se tornassem bastante realísticas. No entanto, como a preocupação maior se concentrou no desenvolvimento do SNA, algumas características de difícil modelagem computacional foram suprimidas, entre elas: inércia, atrito, etc.

Um modelo do robô Khepera II foi utilizado no simulador, seguindo suas propriedades reais. Além de permitir experimentos com mais de um robô ao mesmo tempo, o simulador também possui ferramentas de análise de desempenho e acompanhamento da navegação. Mais detalhes são apresentados no Apêndice C.

3.4 Adequação Funcional dos Sistemas Classificadores à Navegação Autônoma de Robôs e Principais Contribuições

Considerando as características específicas do problema de navegação tratado (relativas ao ambiente, às tarefas do robô, ao próprio robô e aos objetivos do sistema de navegação), algumas adaptações foram realizadas no sistema classificador de Holland. As adequações do SC original para aplicação no sistema de navegação desenvolvido neste trabalho, as

quais são também algumas das principais contribuições da proposta, são mostradas resumidamente na Tabela 3-1 e descritas nesta seção. A descrição completa dos componentes citados na Tabela 3-1 é feita ao longo deste capítulo.

Tabela 3-1: Principais distinções entre o SC de Holland e o sistema proposto neste trabalho.

Componentes	**SC de Holland**	**Neste Trabalho**
Partes da Regra	2	4
Codificação	Binária	Binária e inteira
Vetores de Ação	Bits padrão	Bits ponderados
***Don't Care* (#)**	Sim	Não
Energia	Sim	Não
Bucket Brigade	Presente	Parcialmente presente
Número de Iterações de uma Época	Constante	Variável
Fim da Época (Disparo do A. E.)	Todas as iterações foram executadas	Eventos de colisão, captura e monotonia
Avaliação e Reprodução	Procedimentos gerais	Procedimentos específicos a cada evento
Descendentes Gerados	Porcentagem fixa	Taxa de Procriação (1% a 10%)
Operadores Genéticos	Comuns	Específicos e Polarizados
Aprendizagem	Somente em fase de treinamento	Sempre

Primeiramente, a estrutura dos classificadores foi estabelecida com duas partes antecedentes distintas e duas partes conseqüentes também distintas: no modelo original, existe apenas uma parte antecedente e uma parte conseqüente. Como o robô possui dois conjuntos de sensores independentes e dois tipos de atuadores diferentes, as duas partes antecedentes foram associadas às duas classes de sensores (distância a obstáculos e luminosidade do alvo), e os dois atuadores (ajuste de direção e ajuste de velocidade) foram associados às duas partes conseqüentes.

Também devido ao formato dos valores fornecidos pelos sensores (números decimais inteiros), a codificação das partes antecedentes é feita por números inteiros, variando no mesmo intervalo de valores que varia a medição dos sensores. Apenas as partes conseqüentes são codificadas em binário. Contudo, os bits não têm pesos iguais, como no SC original. Embora componham um mesmo vetor, os bits são ponderados diferentemente neste trabalho. Por exemplo, o conseqüente de ajuste de direção possui 9 bits, sendo que os 5 mais significativos determinam o sinal do ajuste (se a maioria for 1, o sinal é negativo), e os 4 outros definem o valor do ajuste segundo a codificação binária usual. Ainda relacionado aos vetores binários, ao contrário do SC original, os bits *"don't care"* (#) não são utilizados pelo fato de as partes antecedentes da regra, que normalmente admitiriam bits *"don't care"*, serem de codificação inteira. Todos os detalhes referentes às regras são apresentados na seção 3.5.1.

Outro elemento do SC original ausente aqui é a variável de energia das regras. A energia de cada regra é importante quando existe a possibilidade de, no cálculo da semelhança entre antecedentes e estímulos, várias delas serem ativadas em uma mesma iteração. Neste caso, o critério de desempate é a energia. Entretanto, como neste trabalho as partes antecedentes das regras e os valores dos sensores têm codificação inteira, o grau de semelhança excursiona em um intervalo mais amplo, reduzindo muito as chances de empate. Desta forma somente o grau de semelhança entre antecedente e estímulos é suficiente para definir a regra vencedora. Sem a energia, também ficam desativados os mecanismos de manipulação da energia: leilão, recompensa e taxação. Outra razão para que a energia não seja utilizada é a falta de implementação do algoritmo de *bucket brigade* em todas as etapas do módulo de atribuição de crédito. Dado que o SC implementado neste trabalho não considera classificadores postadores de mensagens e nem energia, motivações fundamentais para o uso do *bucket brigade*, uma versão adaptada deste mecanismo de apropriação de crédito foi adotado somente no caso de eventos de monotonia, pois estes eventos são normalmente sustentados por múltiplas regras. Todos os detalhes referentes ao processo de escolha da regra vencedora são apresentados na seção 3.5.2.

O módulo de descoberta de novas regras também foi reconfigurado e estendido, tornando-se a principal parte do sistema classificador implementado. Basicamente, o algoritmo genético contido neste módulo é disparado quando uma época de iterações é encerrada. Normalmente, no SC de Holland a época é definida com um número fixo de iterações, e o critério de parada é justamente a execução de todas as iterações. No sistema de navegação implementado aqui, o número de iterações de uma época é variável e depende, única e exclusivamente, da ocorrência de certos eventos. Toda vez que um dos seguintes eventos é detectado, a época é encerrada e os processos evolutivos são disparados:

- Colisão em obstáculo;
- Captura de alvo;
- Monotonia (falta de objetividade).

Os processos evolutivos são divididos em duas etapas: avaliação e reprodução. Considerando que a ocorrência de cada um destes eventos requer tendências diferentes de convergência para o AG, é preciso que o algoritmo evolutivo seja específico para cada caso. Também são específicos os processos evolutivos para cada parte das regras (antecedentes e conseqüentes), isto é, devido aos componentes das regras serem de natureza diferente, os mecanismos que tratam de cada um deles devem ser particulares. Por isso, existem diferentes funções-objetivo e diferentes procedimentos evolutivos, em cada uma das etapas, dependendo do evento de disparo do AG e da parte das regras envolvida. Todos os detalhes referentes à avaliação das regras são apresentados nas seções 3.5.3.2, 3.5.3.5 e 3.5.3.8.

Outra diferença do SC original para o SC deste trabalho se refere ao número de descendentes produzidos por geração do algoritmo evolutivo. Originalmente, a quantidade de novas regras é igual a uma parcela fixa, e em geral não muito grande, da população total. Já nesta nova versão, a parcela de novas regras geradas, e antigas substituídas, é variável (de 1% a 10%) de acordo com um mecanismo inovador chamado taxa de procriação. Este mecanismo leva em consideração o desempenho do sistema em relação ao atendimento dos objetivos de navegação para definir se as evoluções serão mais, ou menos, abrangentes. Todos os detalhes referentes à taxa de procriação são apresentados na seção 3.5.3.10.

Um dos principais diferenciais deste trabalho, quando comparado aos demais trabalhos da literatura, está relacionado aos operadores genéticos empregados. Neste trabalho, boa parte dos operadores genéticos foram implementados de forma polarizada, ou seja, eles realizam ajustes nos cromossomos das regras em uma direção que tende a ampliar as condições para o atendimento dos objetivos de navegação. Assim, a tendência é que a população venha a ter uma convergência mais rápida do que se sofresse modificações por operadores não-polarizados, implicando em melhor desempenho e resultados. Todos os detalhes referentes aos operadores genéticos polarizados e à reprodução das regras são apresentados nas seções 3.5.3.3, 3.5.3.6 e 3.5.3.9.

De forma geral, analisando a questão da aprendizagem do sistema (realizada por meios evolutivos), pode-se dizer que no SC de Holland ela ocorre somente na fase de treinamento. Isto é, o sistema é treinado inicialmente em uma etapa específica, na qual toda sua aprendizagem ocorre, para depois atuar efetivamente. Diferentemente, no sistema de navegação deste trabalho, não há uma fase determinada somente para treinamento. Na verdade, a aprendizagem está habilitada o tempo todo, enquanto o robô navega pelo ambiente, e depende somente do disparo dos eventos já mencionados para acontecer. Note que, embora o aprendizado possa ser feito com maior ou menor intensidade (taxa de procriação), seus mecanismos nunca são desativados completamente. O sistema sofre aprendizagem e adaptação efetivamente enquanto atua sem que haja interrupção da navegação.

Finalmente, é apresentado na Figura 3-3 o algoritmo simplificado do sistema classificador implementado neste trabalho. No próprio algoritmo pode-se perceber algumas diferenças em relação ao algoritmo simplificado do SC de Holland, mostrado na Figura 2-6, mais especificamente nos passos 2, 4, 5, 6 e todos os passos do algoritmo genético.

3.5 Descrição do Sistema de Navegação Autônomo Evolutivo

O sistema de navegação é responsável pelo controle total do robô, tomando todo tipo de decisão, tanto relativa a ações imediatas quanto a estratégias de navegação e também a

aprendizagem, parte essencial do sistema. Ele é reativo, isto é, ao contrário de sistemas deliberativos, ele não utiliza informações obtidas previamente para decidir a próxima ação de controle. Assim, o sistema se baseia apenas em informações (estímulos) capturados no instante da tomada de decisão.

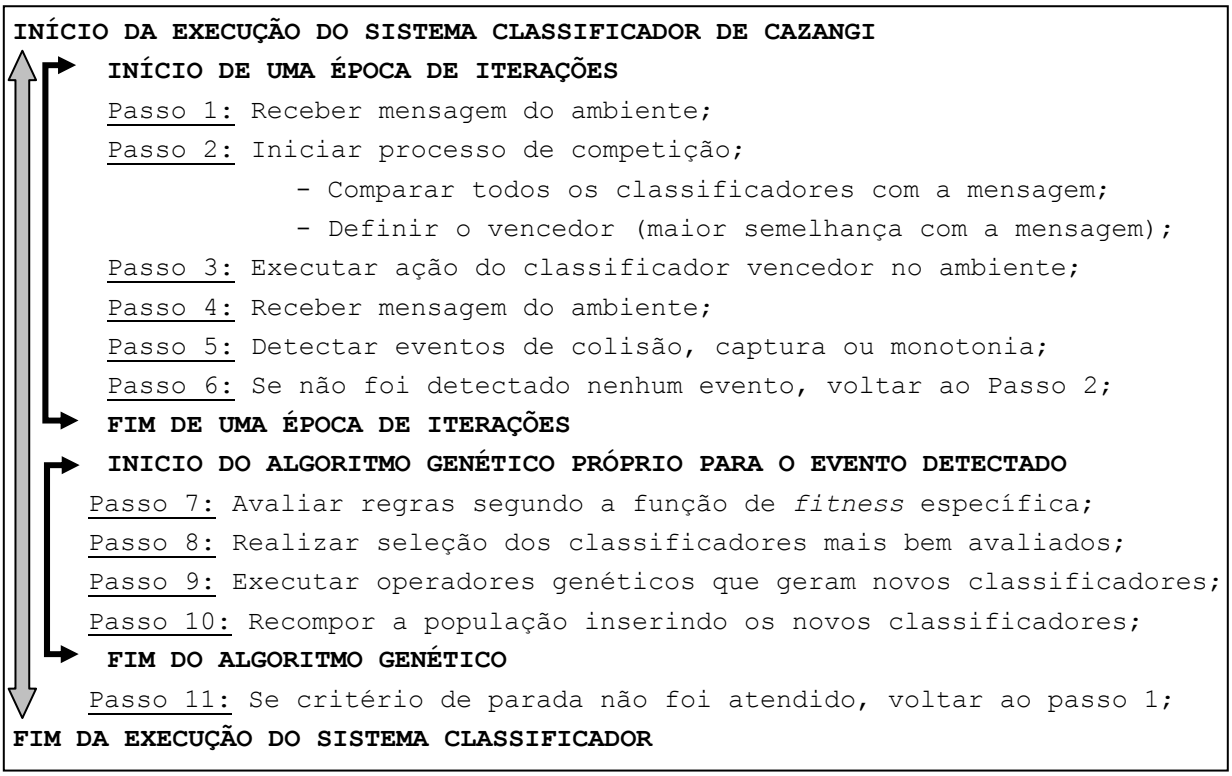


Figura 3-3: Algoritmo simplificado do sistema classificador implementado neste trabalho.

Inicialmente, não há conhecimento significativo presente no sistema, ou seja, o robô não possui estratégias de navegação definidas e desconhece a topologia do ambiente. A assimilação de conhecimento sofisticado, emergência de comportamentos e conseqüente desenvolvimento de estratégias de navegação gerais, se dão pela ocorrência de processos de aprendizagem.

O SNA proposto, que possui inspiração nos sistemas classificadores (devidamente adaptados ao contexto da aplicação, conforme foi apresentado na seção anterior), interage com o ambiente por meio de sensores e atuadores (mecanismo de ajuste de direção e velocidade), e está organizado em quatro componentes principais: população de regras, módulo de competição, módulo de avaliação e módulo de reprodução. O funcionamento e a interação das partes do sistema são mostrados no diagrama da Figura 3-4.

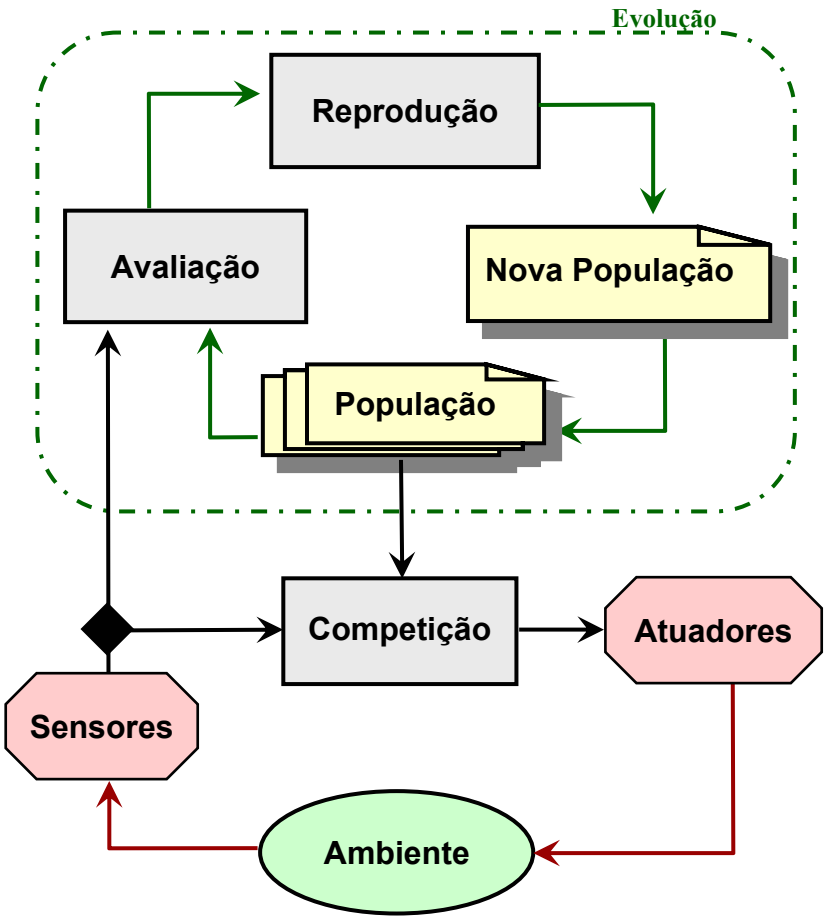


Figura 3-4: Diagrama que representa a estrutura e funcionamento do SNA.

A população de regras representa o conhecimento do sistema e evolui durante a navegação do robô, mais precisamente entre dois ciclos sensório-motores, a serem definidos em seguida. O módulo de competição recebe os estímulos, capturados pelos sensores de alvo e obstáculo, e os compara com as regras, escolhendo a mais apropriada. Essa regra vencedora, por sua vez, tem sua ação enviada para o atuador, que finalmente ajusta a velocidade e direção do robô. Esse processo se repete a cada movimento do robô, compondo um ciclo sensório-motor. O ciclo somente é interrompido quando se faz necessária uma evolução da população (aprendizagem), caracterizando o fim de uma época de iterações. Três eventos particulares disparam a aprendizagem: colisão, captura e monotonia (evento virtual que ocorre quando o robô apresenta comportamento monótono). Nestes casos, as regras são avaliadas e alteradas, por meio de operadores evolutivos, produzindo uma nova população. Concluída a atualização da população, uma nova época é iniciada e o sistema volta a atuar no ambiente.

Uma iteração do sistema, correspondente a um ciclo sensório-motor, é representada por um ciclo do diagrama da Figura 3-4 que começa pela leitura dos sensores e termina com a atuação no ambiente. Este ciclo consome, em simulação, um tempo desprezível mesmo em computadores com velocidade de processamento menores, como 300 MHz. Já quando controlando o robô Khepera II, são acrescidos os tempos de sensoriamento, atuação e transmissão de dados, que, somados, não excedem 400 ms, em geral. Os procedimentos evolutivos, quando disparados, também tomam um tempo desprezível, visto que o processamento é feito pelo processador do computador controlador. Todos estes tempos consumidos durante a atuação do robô não influem significativamente em sua navegação, considerando-se as condições específicas encontradas neste trabalho, ou seja, em que o robô não atinge velocidades elevadas. Caso o robô se movimentasse em alta velocidade, os atrasos de processamento poderiam promover efeitos indesejáveis durante a navegação.

3.5.1 População de Regras

Tal qual em um sistema classificador (SC), cada indivíduo da população é representado por uma regra de inferência do tipo se ⟨condição⟩ – então ⟨ação⟩. O tamanho da população é fixo, contando com 200 regras. Este número foi definido empiricamente por intermédio de experimentos envolvendo vários tamanhos de população, os quais são apresentados na seção 4.2.1.1. A estrutura de cada regra, ou seja, seu cromossomo, é composta por quatro vetores: *RO, RA*, *RD* e *RV*. As características de cada vetor estão na Tabela 3-2.

Tabela 3-2: Características dos vetores que compõem um indivíduo (regra) da população.

Nome	**Parte da Regra**	**Representação**	**Elementos**	
			Quantidade	**Tipo**
RO	Antecedente	Distância a Obstáculos	6	Inteiro
RA	Antecedente	Luminosidade do Alvo	8	Inteiro
RD	Conseqüente	Ajuste de Direção	9	Binário
RV	Conseqüente	Ajuste de Velocidade	5	Binário

Os vetores *RO* e *RA* compreendem a parte antecedente da regra, que representa uma possível situação do ambiente, ou seja, uma determinada leitura dos sensores do robô. Por isso, os vetores consistem, respectivamente, de 6 e 8 elementos, que são os mesmos números de sensores de proximidade e sensores de luminosidade, respectivamente. Desta forma, cada elemento dos vetores tem correspondência a um único sensor, possibilitando a realização de comparações precisas entre a parte antecedente das regras e a leitura corrente dos sensores.

A parte conseqüente da regra é composta pelos vetores *RD* e *RV*. São 9 os elementos de *RD*, sendo que seus 5 bits mais significativos representam o sinal e os 4 bits restantes, o valor absoluto do ajuste de direção proposto pela regra. No caso dos bits de sinal, se a maioria deles for 0, então o sinal é negativo, caso contrário o sinal é positivo. O sinal, na prática, indica o lado do ajuste: negativo é direita (sentido horário) e positivo é esquerda (sentido anti-horário). Note que os 4 bits de valor absoluto cobrem os 15º, que é a margem de manobra máxima do robô. Cabe uma explicação quanto à existência de 5 bits para determinação de um única variável binária (o sinal). Com mais bits evita-se que alterações esporádicas e pontuais causem mudanças drásticas na ação indicada pela regra. Isto é, a determinação do sinal não fica concentrada em um único bit, mas sim distribuída por vários bits. Assim como mencionado na seção 3.4, é importante notar que os bits do tipo "*don´t care"* (#) não são utilizados neste sistema.

Em relação ao vetor RV, a mesma idéia é empregada. Se a maioria de seus 5 bits for 0, a velocidade do robô deve ser reduzida, caso contrário, deve ser aumentada. Aqui não há variações diferentes de velocidade, simplesmente a regra sinaliza para que a velocidade seja incrementada ou decrementada, sendo sempre de um valor constante.

A construção da população inicial, no princípio da navegação, é realizada com a seleção de valores aleatórios para todos os campos das regras, em suas partes antecedente e conseqüente. É importante notar que, embora aleatórios, os valores de cada campo são escolhidos de acordo com seu tipo, em uma distribuição de probabilidade uniforme.

Entretanto, há uma exceção no caso do vetor *RA* (antecedente de alvo). Para que os padrões do vetor da regra sejam razoavelmente compatíveis aos padrões produzidos pelos sensores do robô, utiliza-se uma distribuição gaussiana. Portanto, o conteúdo dos campos no vetor *RA* é determinado por meio de uma variável aleatória gaussiana, segundo a equação (3-1).

$$RA(i) = Lum * exp\left(\frac{Ang^2}{2\Pi * NSens^{-1}}\right) \tag{3-1}$$

onde *RA(i)* é o i-ésimo elemento do vetor *RA*, *Lum* é um valor aleatório entre 0 e 500, *Ang* é calculado pela equação (3-2), e *NSens* é o número de sensores de alvo.

$$Ang = \begin{cases} min\left(\left|RndAng - AngSens(i)\right|, \left|RndAng - AngSens(i) + 2\Pi\right|\right) & se\ RndAng < \Pi; \\ min\left(\left|RndAng - AngSens(i)\right|, \left|RndAng - AngSens(i) - 2\Pi\right|\right) & caso\ contrário. \end{cases} \tag{3-2}$$

cuja variável *RndAng* é um ângulo aleatório entre 0 a 2Π e *AngSens(i)* é o ângulo do i-ésimo sensor de alvo.

A equação (3-1) é utilizada para que todas as regras tenham, em seus vetores *RA*, padrões regulares. A partir das informações de luminosidade do alvo, ângulo do sensor corrente e número de sensores, calcula-se o valor de cada elemento do vetor *RA*. Quando todos os elementos são calculados, o vetor resultante apresenta um padrão que se assemelha a uma distribuição gaussiana e que agrupa, em uma única representação, duas informações sensoriais: luminosidade e direção do alvo. Um exemplo de padrão representado pelo vetor *RA* é mostrado na Figura 3-5.

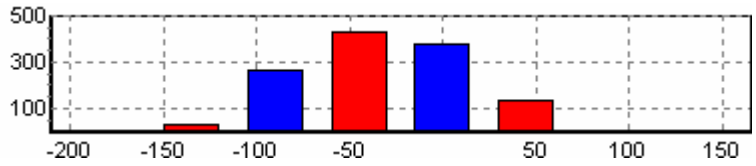


Figura 3-5: Exemplo gráfico de uma parte antecedente de alvo (ângulo dos sensores × luminosidade).

Embora o processo seja essencialmente aleatório, há algumas restrições básicas que devem ser obedecidas. Isto é, cada elemento dos vetores deve ser (sempre) dos tipos numéricos pré-estabelecidos (inteiros, reais e binários), maior ou igual a zero e menor ou igual ao respectivo limite máximo (1024 para *RO* e 500 para *RA*).

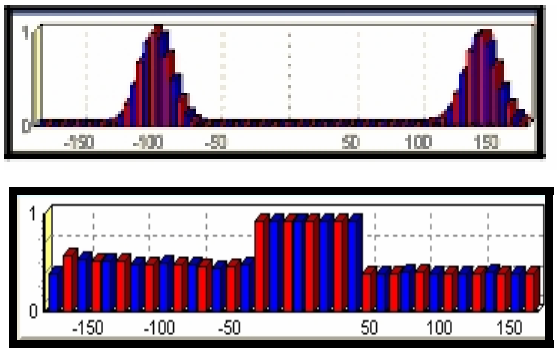


Figura 3-6: Exemplos hipotéticos de padrões inconsistentes do vetor *RA*.

Além disso, os vetores da parte antecedente não podem conter padrões inconsistentes. Por exemplo, no caso do vetor de direção de alvo, padrões que indicam mais de um alvo ou então nenhum alvo no ambiente, não podem existir na população. Na Figura 3-6, dois exemplos de padrões inconsistentes para o vetor *RA* são mostrados. A Figura 3-7 apresenta um exemplo do cromossomo de uma regra completa (e consistente) da população.

Antecedente de Obstáculo (RO) *e* *Antecedente de Alvo (RA)*

900	900	900	700	90	50		0	0	20	100	350	100	20	0

Conseqüente de Direção (RD) *e* *Conseqüente de Velocidade (RV)*

1	1	0	0	0	0	0	1	1		0	0	1	1	1

Figura 3-7: Exemplo do cromossomo de uma regra cujos conseqüentes determinam giro de –3º e aumento de velocidade.

3.5.2 Módulo de Competição

O módulo de competição tem a tarefa de definir qual regra, dentre todas da população, possui a parte antecedente que melhor corresponde à situação corrente observada no

ambiente e traduzida pelos estímulos capturados pelos sensores do robô. A esta regra é, então, dado o direito de atuar modificando o ambiente.

A cada movimento do robô, os sensores de obstáculos e de alvo capturam estímulos do ambiente na forma de vetores, *EO* e *EA*, respectivamente, os quais são transferidos para o módulo de competição. Nesta etapa, todas as regras competem para definir a vencedora, ou seja, aquela cuja parte antecedente é mais semelhante aos estímulos. Note que aqui não é usada a idéia de energia das regras. A semelhança de cada regra *r*, *S(r)*, é dada por:

$$S(r) = \frac{\|RO(r) - EO\|}{MaxO} + \frac{\|RA(r) - EA\|}{MaxA} \tag{3-3}$$

onde *MaxO* e *MaxA* representam valores adequados para normalização do primeiro e segundo termos de *S*(*r*), respectivamente. $\|x - y\|$ é a norma euclidiana da diferença entre os vetores *x* e *y*.

A equação (3-3), responsável por medir a similaridade entre regras e estímulos, é composta por dois termos normalizados. O primeiro termo indica a semelhança relativa às partes de obstáculos, e o segundo termo é relativo à semelhança das partes de alvo. A combinação dos dois termos (por meio de adição) em um único valor foi estabelecida como uma forma de impor que o sistema decidisse a regra vencedora baseado em um único critério. Eventualmente, caso houvesse duas medidas independentes (uma de cada termo), seria obrigatório o uso de um critério extra que definisse a regra vencedora, dado que muito provavelmente cada termo indicaria uma regra distinta para atuar. Portanto, da maneira como a equação foi definida, simplesmente a regra com maior *S*(*r*) é escolhida para atuar.

As partes conseqüentes da regra vencedora são enviadas aos atuadores. O conseqüente *RD* fornece uma correção de direção, e o conseqüente *RV* indica o que deve ser feito com a velocidade. Os atuadores, de posse destas informações, movimentam o robô com a nova direção e velocidade.

3.5.3 Evolução

O processo de evolução da população de regras é executado pelos módulos de avaliação e reprodução. A evolução é disparada ao final de uma época de iterações, ou seja, sempre que um dos seguintes eventos é detectado: colisão em obstáculo (os sensores de proximidade medem um valor mínimo pré-estabelecido), captura de alvo (os sensores de luminosidade detectam valores acima de um limiar pré-determinado) e monotonia. Um evento de monotonia é gerado se o robô não capturar um alvo por um período determinado, ou se o somatório dos ajustes de direção nas iterações anteriores superar um certo limiar.

Cabe aqui uma ressalva importante a respeito do evento de colisão em obstáculos. Quando uma colisão é detectada pelo sistema, na realidade não significa que o robô efetivamente chocou-se com o obstáculo. É feito um tratamento, durante a leitura dos sensores, que faz com que os eventos de colisão sejam detectados tão logo o robô se encontre a uma distância mínima predefinida do obstáculo (mas não encostado). Desta forma, não há danos físicos à estrutura do robô. Embora este evento passe uma imagem negativa, a ocorrência de colisões é essencial à evolução do sistema de navegação. Parece controverso, já que o objetivo da navegação é não colidir. Entretanto, se o robô não sofresse colisões, não aprenderia nunca a desviar de obstáculos, visto que este comportamento não é inato ao sistema e se desenvolve justamente ao longo da navegação. Em resumo, as colisões em obstáculos são negativas unicamente em relação aos objetivos do processo de navegação, mas são fundamentais à evolução do sistema de navegação autônomo.

A ocorrência de colisões e monotonias é reflexo de deficiências no sistema de navegação, provenientes de um estado primitivo do sistema, ou então por regras malformadas, inconsistentes ou incompletas presentes na população. Portanto, estes dois eventos funcionam como uma realimentação negativa do ambiente, demandando a execução de processos evolutivos que venham a corrigir e aprimorar as regras que definem os comportamentos. Uma captura de alvo é um evento oposto aos outros dois, no sentido que as regras que atuaram levando à captura foram competentes. Por isso, a captura de alvo é um evento que gera uma realimentação positiva, incentivando a realização de processos

evolutivos que disseminem as características das regras que representam o comportamento de captura pela população.

Estes três eventos que disparam a evolução da população de regras possuem, cada qual, objetivos distintos associados a processos evolutivos específicos. Isto é, os procedimentos disparados por cada um deles são diferentes e, por isso, serão detalhados nas seções de descrição do módulo de avaliação e módulo de reprodução, separadamente.

Outra distinção que se faz durante o processo evolutivo diz respeito às partes antecedentes e conseqüentes da regra, as quais são de diferentes naturezas. Devido a este fato, as partes antecedentes das regras são avaliadas e reproduzidas por processos específicos e independentes daqueles que lidam com as partes conseqüentes. Na prática, a população é dividida em três, uma de antecedentes, outra de conseqüentes de direção e uma terceira de conseqüentes de velocidade. Cada qual passa pelos respectivos processos evolutivos para, só então, serem reunidas na nova população.

Cabe destacar que o processo evolutivo completo é executado antes que o ciclo normal de atuação do robô volte a ocorrer, ou seja, no intervalo de tempo entre a última movimentação e o novo sensoriamento. Não há atrasos ou paralisações significativos da navegação do robô enquanto o SNA passa pelos processos evolutivos (considerando as características específicas deste trabalho), pois o tempo consumido pela evolução de uma geração da população é menor que o tempo gasto em cada ciclo sensório-motor (considerando os recursos computacionais utilizados).

3.5.3.1 Colisão

Se uma colisão ocorre, um processo evolutivo é realizado com o objetivo de aprimorar o comportamento de desvio de obstáculos apresentado pelo robô. Neste sentido, os módulos de avaliação e reprodução são configurados visando este intuito.

3.5.3.2 Módulo de Avaliação da Evolução Disparada por um Evento de Colisão

Logo após a detecção do evento, inicia-se a avaliação das regras atuais através de medidas do nível de atendimento dos objetivos da navegação. O primeiro passo é separar as partes antecedentes das partes conseqüentes de cada regra, formando as sub-populações.

A avaliação da parte antecedente das regras se baseia em sua similaridade com a situação instantânea na colisão. Ou seja, o valor da avaliação é igual a $S(r)$, obtido pela equação (3-3), em que EO e EA são exatamente os mesmos vetores de estímulos capturados no momento da colisão. Quanto menor for $S(r)$, mais semelhante aos estímulos é o antecedente da regra r. Logo, os antecedentes são ordenados segundo $S(r)$, e aqueles que possuírem menor valor terão, proporcionalmente, maiores probabilidades de serem selecionados para reprodução.

A parte conseqüente de ajuste de direção das regras é avaliada em termos de sua semelhança com o reflexo instintivo do robô, cujo valor depende do tipo de colisão que ocorreu. Por exemplo, se o robô colidiu com sua lateral esquerda, o reflexo instintivo assume o valor do ajuste de direção máximo para o lado direito (−15º) e vice-versa. Assim, os conseqüentes que possuírem valores mais próximos ao reflexo instintivo terão maiores chances de se reproduzirem. A avaliação é dada por $T(r)$, calculado pela equação (3-4).

$$T(r) = \begin{cases} \left| [RD(r)]_d - 15 \right|, \text{ se colisão à esquerda;} \\ \left| [RD(r)]_d + 15 \right|, \text{ caso contrário.} \end{cases} \tag{3-4}$$

onde $[RD(r)]_d$ é o ajuste de direção definido pelo conseqüente da regra r (número decimal).

No caso da colisão ter ocorrido na região frontal do robô, o melhor reflexo instintivo seria desviar o máximo para qualquer um dos lados. Assim, neste caso, determina-se o reflexo instintivo aleatoriamente havendo duas possibilidades: ou 15º ou −15º. Desta forma, espera-se que o robô não fique polarizado a desviar sempre para o mesmo lado.

A última avaliação é da parte conseqüente de ajuste de velocidade das regras. Ela é inversamente proporcional à distância *Hamming*, dada por $dH(r)$, entre o vetor $RV(r)$ e um vetor padrão que representa redução de velocidade (todos os elementos são 1). Considerando que em situações de iminente colisão a velocidade deve ser reduzida, os conseqüentes que sugerem redução de velocidade serão bem avaliados. A equação (3-5) mostra como é feito o cálculo:

$$dH(r) = \sum_{i=1}^{5} \left| RV(r)_i - PRV_i \right| \tag{3-5}$$

onde $RV(r)$ é o ajuste de velocidade indicado pelo conseqüente da regra r, $PRV = [1,1,1,1,1]$ e i representa o i-ésimo elemento do vetor.

As configurações do módulo de avaliação, mostradas resumidamente na Tabela 3-3, foram assim definidas entendendo que as colisões são provocadas pela seguinte deficiência: as regras que atuaram nas iterações próximas da colisão foram ativadas, pois suas partes antecedentes foram satisfeitas, no entanto seus conseqüentes não estavam adequados. Com os critérios definidos, busca-se a manutenção dos antecedentes relacionados a estas situações e a união com conseqüentes que garantam o desvio do obstáculo, ou seja, conseqüentes com ajuste de direção que leve a um desvio da direção em que se encontra o obstáculo e a uma redução de velocidade.

Tabela 3-3: Informações do módulo de avaliação em caso de disparo por colisão.

Parte da Regra	Critério de Avaliação (normalizado de 0 a 1)	Nº da Equação
Antecedente	Semelhança com Situação de Colisão	3-3
Conseqüente de Direção	Semelhança com Reflexo Instintivo	3-4
Conseqüente de Velocidade	Semelhança com Vetor de Redução	3-5

3.5.3.3 Módulo de Reprodução da Evolução Disparada por um Evento de Colisão

Em seguida ao módulo de avaliação, executa-se o módulo de reprodução, onde entram em ação os operadores genéticos que produzirão os novos indivíduos. Deve-se enfatizar que o

módulo de avaliação estabelece classificações diferentes para cada parte das regras e, portanto, os operadores irão agir em fases independentes: no grupo dos antecedentes e depois no grupo dos conseqüentes.

Um fator importante a ser mencionado antes do início da descrição do processo é o conceito de taxa de procriação (*TxP*). A taxa de procriação define quantos indivíduos filhos serão produzidos pelo processo evolutivo. Já nas versões originais de sistemas classificadores, o número de filhos gerados é sempre uma pequena parcela do tamanho total da população, sendo esta parcela fixa. Os filhos gerados irão substituir os indivíduos pais. Neste trabalho, o número de filhos gerados não é fixo, variando segundo *TxP*, a qual é descrita detalhadamente mais adiante, ainda neste capítulo, mas já se adianta que ela pode variar de 10% a 1%. Portanto, o número de filhos produzidos em cada geração (*Nf*) é calculado assim:

$$Nf = TamPop * TxP \tag{3-6}$$

onde *TamPop* indica o tamanho da população e *TxP* é a taxa de procriação que pode variar de 10% a 1%.

A primeira operação a ser executada é a seleção, feita pelo método da roleta, descrito no Apêndice A. São selecionados indivíduos em quantidade igual a *Nf*, levando-se em conta a pontuação feita pelo módulo de avaliação. Estes indivíduos selecionados serão os genitores (pais) que, tomados 2 a 2, formarão os filhos. A cada dois pais selecionados, dois filhos são gerados pelos operadores evolutivos. Entende-se aqui por pais e filhos os indivíduos de cada sub-população de antecedentes e conseqüentes. Os indivíduos completos, ou seja, as regras do sistema de navegação somente serão geradas após a união de cada nova parte antecedente e conseqüente produzida por processos independentes.

Para se gerar as novas partes antecedentes de obstáculo (vetores *RO*) são aplicados o cruzamento de um ponto e a mutação tradicional. O cruzamento toma um par de pais selecionados e os combina gerando dois filhos, como visto na Figura 3-8. A mutação é ativada de acordo com uma certa taxa de probabilidade e altera aleatoriamente o conteúdo

de um gene do cromossomo. Os novos conseqüentes de direção e velocidade (vetores *RD* e *RV*) são gerados pelos mesmos operadores.

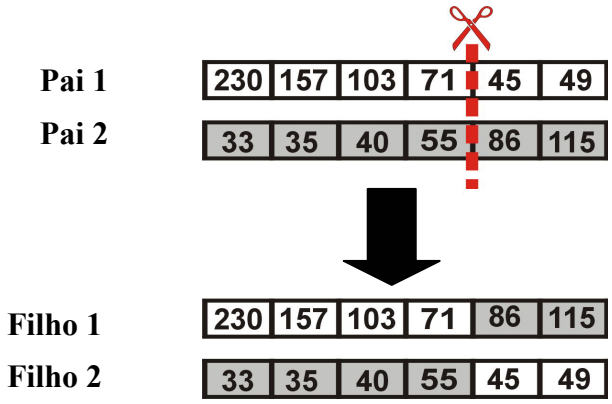


Figura 3-8: Cruzamento com um ponto de corte.

O antecedente de alvo (vetor *RA*) é gerado por um operador específico, pois a aplicação dos operadores clássicos aos indivíduos genitores poderia produzir filhos com padrões inconsistentes, como por exemplo aqueles mostrados na Figura 3-6. O operador de combinação utilizado pega o ângulo associado ao elemento de maior intensidade luminosa de cada pai e calcula o ângulo médio. Este sofre uma leve perturbação aleatória (segundo uma certa variância) e é atribuído a *RndAng* na equação (3-2), para que os novos vetores *RA* dos filhos sejam gerados pela equação (3-1). Também a variável *Lum* da equação (3-1) é obtida pela média da luminosidade máxima dos pais, acrescida de uma perturbação aleatória. Um exemplo de combinação de dois pais pelo método descrito é mostrado na Figura 3-9.

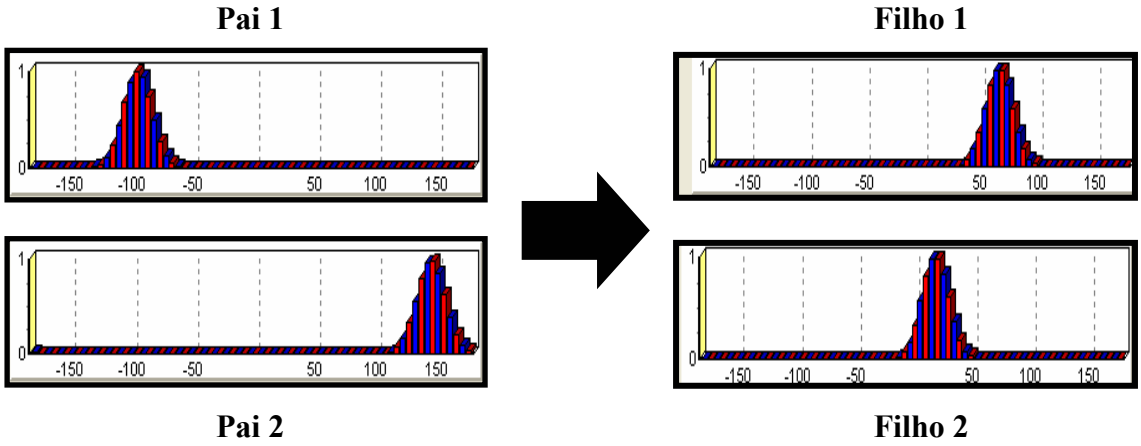


Figura 3-9: Exemplo de combinação para os vetores *RT*.

Consumados os procedimentos do módulo de reprodução, tem-se um conjunto de novas partes antecedentes e um conjunto de novas partes conseqüentes isoladas. Para que os novos indivíduos estejam completos, escolhem-se aleatoriamente os vetores de cada parte que são então re-acoplados. Este processo se repete até que todas as *Nf* novas regras estejam formadas. Uma das novas regras, escolhida aleatoriamente, dá lugar para uma regra chamada ideal. A regra ideal tem as partes antecedentes idênticas aos estímulos sensoriais da situação de colisão (*RO=EO* e *RA=EA*), a parte conseqüente de direção é igual ao reflexo instintivo (equação (3-4)), e a parte conseqüente de velocidade é feita igual ao padrão de redução de velocidade (*RV=PRV*) (equação 3-5). Desta forma espera-se que haja uma convergência mais rápida e precisa da população de regras.

Não há perda de qualidade ou divergência de conhecimento relevante pelo fato de o re-acoplamento das partes antecedentes e conseqüentes das novas regras ser feito de forma aleatória. Considerando que as regras genitoras (cuja combinação e mutação resultam nas regras descendentes) foram selecionadas pelos mesmos critérios de similaridade, logo todas as partes das novas regras têm vários aspectos em comum. Assim, por ser uma parcela restrita da população que participa do processo, e pela taxa de monotonia ser muito pequena, dificilmente qualquer combinação das partes das regras seria divergente do grupo de novas regras como um todo. E, eventualmente, as regras que divergirem do grupo são naturalmente suprimidas ao longo do processo evolutivo.

Finalmente, os procedimentos evolutivos disparados pela colisão são concluídos substituindo-se a fração de população antiga, cujos representantes são os pais, pelos novos indivíduos (os filhos). Então o robô continua a navegar normalmente pelo ambiente.

3.5.3.4 Captura

Ao contrário da colisão, o processo evolutivo disparado por uma captura de alvo visa reforçar ainda mais o comportamento de busca de alvos. Portanto, os módulos de avaliação e reprodução possuem configurações voltadas a este objetivo.

3.5.3.5 Módulo de Avaliação da Evolução Disparada por um Evento de Captura

O processo de avaliação é iniciado separando-se as partes antecedentes das partes conseqüentes de cada regra, formando as sub-populações, assim como no caso de colisão. A avaliação dos vetores antecedentes também é análoga àquela feita na colisão. Isto é, avaliam-se os antecedentes por similaridade com a situação instantânea de captura. Uma regra r é avaliada de acordo com seu $S(r)$ (equação (3-3)), cujos vetores EO e EA são definidos com os mesmos valores detectados pelos sensores no instante da captura do alvo. Conseqüentemente, os antecedentes mais semelhantes à condição do ambiente no momento da captura, cujo $S(r)$ é menor, terão maior probabilidade de serem selecionados para reprodução.

A avaliação do conseqüente de direção (vetor RD) depende do ângulo do sensor que detectou a captura (o mais próximo do alvo) e do próprio ajuste de direção da regra avaliada. A idéia é que os conseqüentes de direção cujos ajustes levariam a um maior alinhamento entre o sensor da captura e o alvo sejam mais valorizados. A avaliação de uma regra r, dada por $E(r)$, é calculada como segue:

$$E(r) = \left| [RD]_d - \alpha \right| \tag{3-7}$$

onde $[RD]_d$ é o ajuste de direção definido pelo conseqüente (número decimal) e α é o ângulo do sensor de captura que detectou o evento.

A terceira avaliação é da parte conseqüente de ajuste de velocidade das regras. O processo é idêntico ao caso de colisão, calculada inversamente proporcional à distância *Hamming* (equação (3-5)), dada por $dH(r)$, entre o vetor $RV(r)$ e um vetor padrão que representa redução de velocidade (todos os elementos são 1). Aqui também os conseqüentes que representam redução de velocidade são desejados, pois, geralmente, o ato de capturar um alvo pode estar associado à execução de tarefas adicionais envolvendo o alvo capturado.

Ao invés de corrigir deficiências na população, o processo evolutivo disparado pela captura de um alvo busca premiar os bons indivíduos que compõem o comportamento de captura,

fazendo com que outras regras da população sofram sua influência, herdando, ou se combinando com suas boas características. As configurações do módulo de avaliação são exibidas na Tabela 3-4.

Tabela 3-4: Informações do módulo de avaliação em caso de disparo por captura.

Parte da Regra	Critério de Avaliação (normalizado de 0 a 1)	Nº da Equação
Antecedente	Semelhança com Situação de Captura	3-3
Conseqüente de Direção	Semelhança com Ângulo do Sensor	3-7
Conseqüente de Velocidade	Semelhança com Vetor de Redução	3-5

3.5.3.6 Módulo de Reprodução da Evolução Disparada por um Evento de Captura

Encerrado o módulo de avaliação, inicia-se o processo de reprodução com os operadores genéticos agindo nos vetores separadamente. Os procedimentos do módulo de reprodução para captura são os mesmos que se realizam no caso de colisão, descritos na seção 3.5.3.3. Exceto que a regra ideal a ser inserida no lugar de uma das novas regras, tem as partes antecedentes idênticas aos estímulos sensoriais da situação de captura (*RO=EO* e *RA=EA*), a parte conseqüente de direção é igual ao valor máximo permitido, entre 15 e -15^{o}, mais próximo de α (equação (3-7)), e a parte conseqüente de velocidade é feita igual ao padrão de redução de velocidade (*RV=PRV*) (equação 3-5). Após a execução deste módulo, o robô segue navegando pelo ambiente.

3.5.3.7 Monotonia

É importante ressaltar que inicialmente o sistema de navegação não possui qualquer estratégia, nem de prevenção de colisão, nem de captura de alvo. Portanto, no começo da navegação, e também em situações em que os comportamentos estão malformados, o robô vagueia pelo ambiente sem objetivo aparente, acontecendo capturas de alvo casualmente. Caso não ocorram eventos de colisão nem de captura por um certo período de tempo, um evento de monotonia é detectado, e o respectivo processo evolutivo é desencadeado. O objetivo deste processo é eliminar as regras que estão contribuindo para que o robô apresente comportamento monótono e, indiretamente, estimular a busca de alvos por parte do robô.

Além disso, o evento de monotonia tem certo respaldo nos sistemas biológicos. As formigas são um exemplo. Se uma formiga se isola do seu grupo, ela pode acabar perdida, vagueando a procura do formigueiro até sua morte por exaustão (BONABEAU *et al.*, 1999)

3.5.3.8 Módulo de Avaliação da Evolução Disparada por um Evento de Monotonia

O processo de avaliação, no caso de monotonia, consiste em analisar um histórico recente das regras atuantes, até um certo número de iterações anteriores à corrente, verificando quais delas provocaram, por culpa de sua ação, o aumento da distância angular do robô ao alvo. Isto é, todas as regras que agiram em seqüência dentro do período de análise e levaram à piora do alinhamento do robô com o alvo são sinalizadas. Para isto, verifica-se na seqüência, a variação da distância angular entre robô e alvo causada pela ação de uma regra em relação à iteração imediatamente anterior. Os procedimentos relacionados à monotonia não separam as partes das regras, pois estas são tratadas por inteiro.

Sob a perspectiva da análise da seqüência de regras que atuaram anteriormente ao evento de monotonia, feita pelo módulo de avaliação, identifica-se uma certa analogia com o algoritmo de *bucket brigade.* Em ambos os casos a seqüência de atuação e o impacto causado pela ação de cada regra são fatores determinantes para a avaliação delas, embora os métodos sejam diferentes.

No *bucket brigade* utiliza-se a energia como medida de desempenho, e a seqüência de regras atuantes leva à recompensa ou punição da última regra da série. Já no caso da avaliação disparada por um evento de monotonia, tem-se a medida de distância angular entre robô e alvo para qualificar a regra, sendo que todas as regras da seqüência são punidas ou recompensadas conforme esta medida.

3.5.3.9 Módulo de Reprodução da Evolução Disparada por um Evento de Monotonia

Todas as regras sinalizadas pelo módulo de avaliação são consideradas improdutivas e prejudiciais ao bom desempenho do robô. Por isso estas, após terem sido avaliadas e identificadas como causadoras de comportamento monótono, são simplesmente substituídas

completamente por novas regras aleatórias. Ou seja, o processo reprodutivo elimina os indivíduos responsáveis pela ocorrência de um evento de monotonia e insere novos indivíduos aleatórios. Isto é feito na tentativa de suprimir o comportamento monótono e proporcionar maiores chances de atuação às regras de captura. As demais regras da população são conservadas intactas durante o processo.

3.5.3.10 Taxa de Procriação

Conforme dito anteriormente, a taxa de procriação determina o número de filhos produzidos por geração. Este mecanismo é responsável por manter o equilíbrio populacional entre indivíduos representantes dos comportamentos de desvio de obstáculo e captura de alvo. Um mecanismo de equilíbrio é necessário, pois há uma tendência natural de disseminação das regras de captura devido à ocorrência consecutiva de capturas de alvos quando o sistema de navegação esta operando com desempenho satisfatório (atendendo aos objetivos eficientemente).

Da maneira como foi desenvolvido o SNA, é de se esperar que o robô, ao navegar por períodos longos, deixe de colidir em obstáculos e passe a capturar alvos com maior freqüência. Considerando que a cada captura de alvo a população de regras sofre um processo evolutivo que gera novos indivíduos, a tendência é que a população seja dominada somente por regras de comportamento de captura, excluindo e deteriorando regras e conhecimentos adquiridos em processos evolutivos anteriores. Com isso, os comportamentos, principalmente de desvio de obstáculos, cujas regras não contribuem para a captura de alvo e, assim, se tornam candidatas à eliminação ou reestruturação, acabam se deteriorando ao longo do tempo. Esta deterioração, também confirmada empiricamente nos experimentos, causa uma instabilidade de desempenho periódica ao robô. Quando é conseguido o desempenho desejado (capturas consecutivas e sem colisões), este não é mantido, pois o comportamento de desvio de obstáculos passa a se deteriorar até voltar a um nível primitivo. Conseqüentemente, os eventos de colisão voltam a ocorrer de forma cíclica.

Portanto, para amenizar este problema de interferências prejudiciais entre processos evolutivos seqüenciais, o mecanismo de taxa de procriação é implementado. Seu funcionamento consiste em reduzir o número de novas regras produzidas por geração quando da ocorrência de capturas consecutivas. Ou seja, de acordo com a quantidade de alvos que vão sendo capturados em seqüência, o número de filhos produzidos por geração é reduzido gradualmente. Desta forma, conforme o desempenho do sistema em relação ao atendimento dos objetivos de navegação, o mecanismo define se as evoluções serão mais ou menos abrangentes. O cálculo da taxa de procriação (*TxP*) é feito segundo a equação (3-8). Esta equação foi definida desta forma com base na idéia de variação do número de filhos apresentada na Tabela 3-5.

$$TxP = \begin{cases} 1/100 \ \text{ se } \ CC > 9; \\ (10 - CC)/100 \ \text{ se } \ 2 \le CC \le 9; \\ 10/100 \ \text{ se } \ CC < 2. \end{cases} \tag{3-8}$$

onde *CC* é o número de capturas consecutivas.

As capturas consecutivas são computadas quando o robô captura alvos em seqüência, sem sofrer colisão nem monotonia. Caso um evento de colisão ou de monotonia volte a ocorrer, *CC* recebe zero imediatamente e *TxP* passa a valer 10%, pois é um sinal de que o sistema ainda não está bem formado. A Tabela 3-5 apresenta a relação entre *TxP*, *CC* e o número de filhos por geração. Observa-se que, nos casos em que o robô capturou apenas um alvo ou sofreu colisão/monotonia (*CC*=0), a quantidade de filhos gerados é de 20, que é o valor máximo. O número mínimo possível de filhos produzidos por geração é 2, no caso de ocorrem mais que 8 capturas consecutivas. Portanto, o sistema nunca pára de evoluir, pois pelo menos 2 filhos são sempre produzidos pelo processo evolutivo associado à captura de alvos (caso extremo).

Tabela 3-5: Relação entre *CC*, *TxP* e número de filhos produzidos por geração com população de 200 indivíduos.

CC	1	2	3	4	5	6	7	8	>8
***TxP* (%)**	10	8	7	6	5	4	3	2	1
Nº de Filhos	20	16	14	12	10	8	6	4	2

Capítulo 4
Experimentos e Resultados

4.1 Introdução

Este capítulo é dedicado à apresentação e análise dos resultados obtidos nos testes feitos com o sistema de navegação autônomo (SNA) implementado. Os testes foram realizados no intuito de avaliar o desempenho do sistema, validar suas potencialidades e detectar limitações. Foram conduzidas duas modalidades de testes por meio de simulações e experimentos reais, além da associação delas.

As simulações computacionais se deram por meio da ferramenta de simulação de navegação desenvolvida neste trabalho, detalhada no Apêndice C. O simulador fornece os artifícios necessários à construção de ambientes virtuais, à navegação de robôs e à análise dos parâmetros envolvidos de forma prática e rápida.

Os experimentos reais contemplaram a utilização do robô Khepera II, mostrado na Figura 4-1, cujas especificações são descritas no Apêndice B. A arena onde os ambientes reais foram montados consiste em uma mesa de madeira de comprimento 60 cm e largura 40 cm, toda envolvida por paredes almofadadas de 10 cm de altura, conforme pode ser visto na Figura 4-2. Os obstáculos caracterizam-se por serem de cores claras e o alvo uma lâmpada incandescente de 40 W. A luminosidade da sala é ajustada para que não influencie nos experimentos e algumas condições básicas são admitidas: a superfície de movimento do robô é plana e os efeitos da inércia são desprezados.

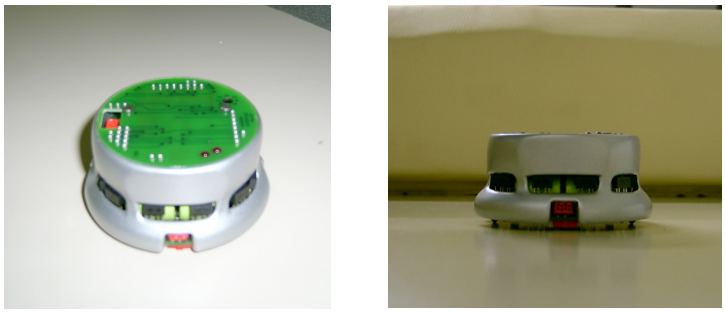


Figura 4-1: Robô Khepera II empregado nos experimentos reais.

O SNA empregado em todos os experimentos mantém as configurações descritas no Capítulo 3, exceto quando alguma alteração foi necessária para se adequar ao experimento. De qualquer forma, as exceções serão sempre informadas. A população inicial, na maioria dos experimentos possui conteúdo aleatório, a não ser em casos específicos a serem apontados. Os seguintes parâmetros evolutivos são constantes: o tamanho da população do sistema é de 200 indivíduos, a taxa de cruzamento é 100% e a taxa de mutação é igual a 0,5%. Esses parâmetros foram determinados de forma empírica, de acordo com o resultado de diversos experimentos apresentados na seção 4.2.1.

Figura 4-2: Arena usada como ambiente nos experimentos reais.

4.1.1 Interpretação dos Resultados e Informações Pertinentes

Para se interpretar os resultados, recorre-se à apresentação do desempenho do SNA nos experimentos na forma de gráficos. Os gráficos contêm três curvas coloridas, cada qual representante de um evento evolutivo. A curva vermelha está associada às colisões em obstáculos, a curva verde representa as capturas de alvos, e os eventos de monotonia são indicados pela curva azul. Cada curva é plotada em função do número acumulado de ocorrências do evento (eixo das ordenadas) pelo número acumulado de gerações do processo evolutivo (eixo das abscissas). Cada evento conta como uma geração e, portanto, o número acumulado de gerações é igual à soma dos eventos de captura, de colisão e de monotonia. Em resumo, cada ponto (x, y) do gráfico indica a quantidade de eventos sucedidos (y) até a respectiva geração (x).

Além dos gráficos, outras informações relativas aos experimentos acompanham os resultados. O número de iterações de uma navegação representa o número de ciclos sensório-motores (movimentos) do robô e, conseqüentemente, o número de decisões tomadas pelo sistema. Dado que o robô possui velocidade variável, computa-se a velocidade média e a distância total percorrida.

É importante que fique clara a distinção entre os conceitos de iterações e gerações utilizados neste trabalho. Cada movimento do robô (ciclo sensório-motor), que se inicia com a leitura dos sensores e é encerrado com a execução de uma ação motora, que representa uma iteração. As gerações são contabilizadas de acordo com a ocorrência de eventos evolutivos (colisão, captura e monotonia) que encerram uma época de iterações. A quantidade e ordem de ocorrência das gerações independe do número de iterações do robô. A única restrição existente é a impossibilidade de acontecer mais que uma geração em uma única iteração.

Durante as simulações, empregou-se um mecanismo automático de inserção de alvos. No ambiente são definidas regiões específicas, dentro das quais os alvos são inseridos em posição aleatória logo que o alvo anterior tenha sido capturado pelo robô. A motivação para a utilização de regiões específicas para inserção de alvos está associada à necessidade de se dispor os alvos em certas posições do ambiente de forma a exigir determinados comportamentos do robô como, por exemplo, capturar alvos separados por um obstáculo entre eles. Já no caso dos experimentos reais, ao alcançar um alvo (lâmpada), a navegação é pausada por alguns segundos e a lâmpada disposta em outra posição do ambiente, dando continuidade ao experimento. Sempre há apenas um alvo no ambiente, tanto nas simulações como nos experimentos reais.

Considerando a eventual situação em que o alvo (fonte luminosa) está posicionado atrás de um obstáculo, do ponto de vista do robô, parece ser impossível que os sensores detectem o alvo, pois a luminosidade é bloqueada pelo obstáculo. Entretanto, pelo fato da fonte luminosa emitir luz em todas as direções, um gradiente de luminosidade se forma na vizinhança das regiões extremas do obstáculo. Esta iluminação residual é capturada pelos

sensores, o que faz com que o sistema guie o robô até estas regiões, a partir das quais a fonte luminosa fica totalmente perceptível. Portanto, o gradiente de luminosidade, além de indicar um caminho ao alvo, também auxilia no desvio de obstáculos.

De forma simplificada os objetivos do robô durante o processo de navegação são capturar alvos continuamente e evitar colisões em obstáculos, partindo de conhecimento inicial inexistente. Traduzindo em gráficos, o que se espera é que a curva de colisões estabilize após algum tempo, e a curva de capturas cresça continuamente. Os eventos de colisão são fundamentais à evolução do sistema, pois sua ocorrência faz com que o sistema aprenda a desviar de obstáculos e assim tenda a não mais colidir. O robô não sofre danos materiais ao colidir, dado que a colisão é virtual. Isto é, o evento é disparado quando uma proximidade menor que um limiar estipulado é detectada. Os eventos de monotonia não são desejados, mas são necessários quando o robô fica por muito tempo sem colidir nem capturar alvos.

O principal intuito geral das experiências realizadas neste trabalho, apresentadas a seguir, é demonstrar a capacidade de aprendizagem do sistema de navegação autônomo em diversas formas. Para isso, os resultados são divididos em várias seções, cada qual analisando características e situações específicas do problema. As seções estão organizadas em duas partes: simulação e navegação com robô real.

4.2 Simulação: Experimentos e Resultados

4.2.1 Análise Paramétrica

Nesta seção, vários experimentos foram realizados visando investigar o processo de ajuste de alguns parâmetros do SNA. Partindo de um ambiente padrão, apresentado na Figura 4-3, e variando os parâmetros analisados, buscou-se avaliar quais valores propiciavam melhores resultados. As regiões denotadas com A e B no ambiente são áreas de inserção de alvos: quando o robô captura um alvo em A, outro é imediatamente inserido em uma posição aleatória de B e vice-versa. Todas as simulações desta seção foram encerradas quando 100 gerações foram alcançadas, independente do número de iterações.

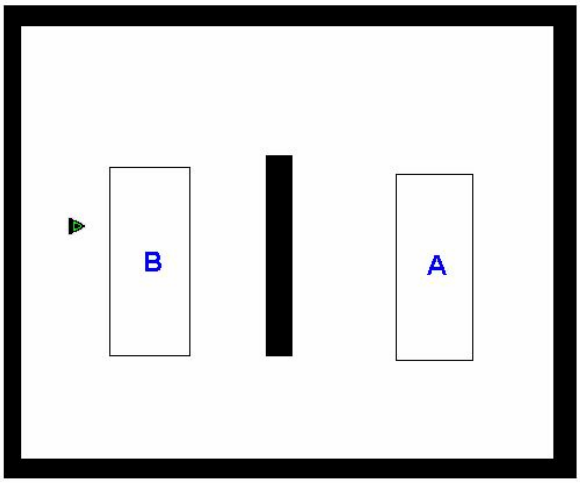


Figura 4-3: Ambiente padrão usado nos experimentos desta seção.

4.2.1.1 Tamanho da População

A população de indivíduos presente no SNA é composta de regras de inferência do tipo *se (condição) – então (ação),* conforme é descrito na seção 3.4.1 do Capítulo 3. A parte de condição corresponde às informações recebidas pelos sensores. A parte de ação possui dois componentes: um ajuste de direção (em graus) e um ajuste de velocidade (aumenta ou diminui). A população como um todo representa a solução do problema, isto é, emprega-se a abordagem Michigan (MICHALEWICZ, 1996). É o conjunto das regras que define o elenco de comportamentos do robô, e isso se dá pelos processos evolutivos.

A quantidade de regras que compõe a população é um fator determinante em um sistema evolutivo. Definindo apropriadamente o tamanho da população, promove-se uma busca mais eficaz, pois a diversidade populacional permanece em níveis capazes de conduzir a uma maior exploração do espaço de busca. É fato que o número de indivíduos está intimamente associado às características do problema, como a dimensão do espaço de busca. Simulações foram feitas variando esse valor na tentativa de descobrir qual o número mais adequado de indivíduos e também para saber a influência deste parâmetro no desempenho do sistema. Os resultados de cada teste estão na Tabela 4-1.

Tabela 4-1: Desempenho do sistema de acordo com o tamanho da população.

Tamanho da População	Colisões	Capturas	Monotonias
30	31	29	40
60	33	38	29
90	31	45	24
120	21	58	21
150	22	54	24
180	18	58	24
200	16	62	22
250	23	53	24
300	29	48	23

As informações da Tabela 4-1 indicam que um tamanho adequado da população está na faixa de 200 regras. Percebe-se que quanto menor a população, maior é o número de eventos de monotonia. Isto significa que a população entra em crise de diversidade e fica estagnada, havendo a necessidade de freqüentes disparos de eventos de monotonia, que neste contexto funcionam como um tipo de "mutação generalizada". Isto é, o processo evolutivo disparado por um evento de monotonia, diferentemente dos processos associados aos outros eventos, seleciona as regras ruins da população (todas aquelas que estão causando comportamento monótono) e as modifica de forma aleatória.

Outra constatação interessante deste experimento é que, com o aumento do tamanho da população (acima de 200), houve uma piora de desempenho. Conforme se pode ver na Tabela 4-1, com 250 e 300 regras na população, o número de colisões aumenta e o número de capturas diminui. Isto ocorre porque uma população com mais regras precisaria de mais gerações para que seu nível médio de desempenho alcançasse graus equivalentes aos que a população de 200 regras obteve ao longo de 100 gerações.

O constante acionamento do processo evolutivo de monotonia causa uma deterioração na qualidade da população, levando ao aumento do número de colisões e à redução das capturas, provocando um desempenho global irregular. Com populações muito grandes são necessários muito mais eventos evolutivos para que os comportamentos desejados predominem. Isso não fica evidente na Tabela 4-1, mas foi observado em alguns

experimentos em que se evoluía o sistema com grandes populações e o número de iterações necessário a uma composição razoável dos comportamentos foi bastante elevado.

4.2.1.2 Quantidade de Descendentes Gerados

Outro parâmetro relacionado à população é a quantidade de descendentes que é produzida em cada nova geração e, conseqüentemente, a fração da população antiga que é substituída pelas novas regras. Como o problema abordado é multi-objetivo (evitar colisões e buscar alvos) e, por estarem associadas a objetivos distintos, as regras são evoluídas em instantes diferentes e de maneira distinta, segundo a seqüência de eventos ocorridos. Quando todas as regras participam do processo evolutivo em todas as situações, a população demora muito a convergir e, por vezes, nem sequer converge. Por exemplo, as regras que sofrem aprendizagem por colisão são gradativamente distorcidas por eventuais aprendizagens de captura.

Para atenuar a dificuldade descrita, foi definido inicialmente que apenas seriam gerados filhos em número igual a um terço da população total. Assim, a idéia era que, teoricamente, um terço de todos as regras adquirissem um comportamento relativo à captura, o outro terço se dedicasse a desvio de obstáculos e ainda restaria mais um terço para efeito de diversidade populacional.

No entanto, em alguns testes mais específicos, verificou-se que do terço de genitores selecionados, apenas cerca de 20 a 30% deles realmente eram aptos a produzirem bons descendentes, todos os demais entravam no processo somente para completar o terço. Com isso, a convergência não era tão eficiente. Alguns experimentos demonstraram que cerca de 10%, em média, dos genitores selecionados eram adequados. Assim, definiu-se que, no máximo, a quantidade de filhos gerados deveria ser de 10% da população total, que então substituem os pais selecionados no início do processo evolutivo. É importante destacar que este parâmetro é controlado pela taxa de procriação (Capítulo 3, seção 3.5.3.10), e o que foi determinado nesta seção foi o valor máximo que a taxa pode assumir.

A partir deste ajuste percebeu-se que os processos evolutivos passaram a ocorrer de forma mais ordenada e não tão casuais como antes. É claro que, com menos regras sendo geradas por vez, a convergência pode demorar mais. No entanto devido ao tamanho da população este efeito é atenuado em qualquer caso.

4.2.1.3 Número de Sensores

Em um robô, as informações do ambiente são obtidas por meio de sensores e, é claro, quanto mais dados se deseja, maior o número e a variedade de sensores que devem ser utilizados. Quando o robô é simulado computacionalmente, incrementar a quantidade de sensores é algo relativamente simples. Entretanto, em robôs reais esta iniciativa se torna custosa e, por vezes, inviável.

Nesta seção, analisa-se o efeito da variação do número de sensores do robô por meio de simulação. Para isto, um experimento coletivo foi realizado: dois robôs foram colocados no mesmo ambiente para disputarem os alvos. Um deles possuía 7 sensores (semelhante ao robô Khepera II) e o outro 37. Os dados da experiência, com 100 e 1200 gerações, estão mostrados na Tabela 4-2.

Tabela 4-2: Resultado da simulação com 2 robôs dotados de diferente número de sensores.

Gerações	Sensores	Colisões	Capturas	Monotonias
100	7	16	33	5
100	37	16	28	4
1200	7	118	351	88
1200	37	78	536	46

Analisando a Tabela 4-2 fica evidente que, em curto prazo, ambos os robôs apresentam desempenho semelhante. Todavia, com um número bem maior de gerações após as 100 iniciais, o robô que dispõe de 37 sensores chega a resultados melhores em todos os quesitos. Apesar de confirmar que um número maior de sensores garante melhores resultados, confirma-se também que o SNA, mesmo com poucos sensores, é capaz de atuar de forma satisfatória.

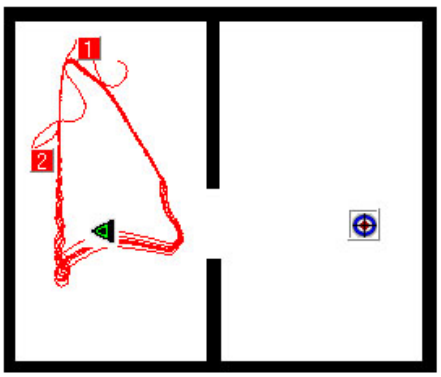
4.2.2 Controle de Velocidade

Esta seção tem como objetivo conferir a importância do controle de velocidade no robô e os benefícios advindos de sua implementação. Reconhecidamente um veículo que possua velocidade constante é muito mais restrito, em termos de manobras, quando comparado a outro capaz de controlar sua velocidade. Certas situações, principalmente em ambientes com áreas de navegação estreitas, com passagens apertadas ou obstáculos muito próximos dos alvos, exigem manobras cautelosas. Justamente estas situações são averiguadas nos experimentos apresentados a seguir.

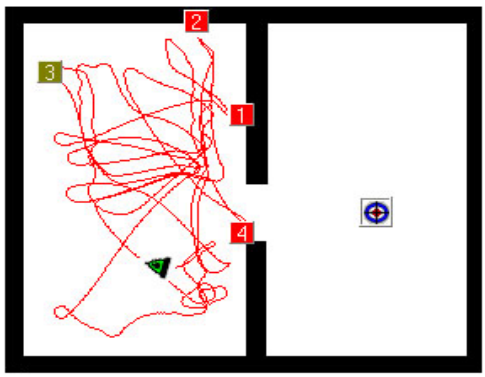
Os quatro próximos experimentos são constituídos por ambientes iguais: o robô inicia sua navegação no compartimento esquerdo e deve capturar um alvo posicionado no compartimento direito, considerando que há entre os dois compartimentos uma passagem estreita. O experimento era encerrado quando o robô capturasse o alvo ou caso excedesse 2600 iterações (cerca de 5 minutos de navegação). Nos dois primeiros, exibidos na Figura 4-4, o robô foi configurado com velocidade constante (1 u.v.), enquanto nos outros dois ele foi equipado com o controle de velocidade (de 0,2 u.v. a 3 u.v.).

No caso da simulação, 1 unidade de velocidade corresponde a 1 pixel por iteração, que é igual a aproximadamente 0,014 centímetros. Na Figura 4-4 pode-se ter uma noção não muito precisa da velocidade do robô pela sua trajetória. Cada ponto da trajetória é 1 pixel/iteração e portanto 1 u.v. Já no caso dos experimentos reais (seção 4.3), 1 u.v. corresponde a 0,2 cm por iteração.

Pode-se observar na Figura 4-4 que, com velocidade constante, o robô não conseguiu atravessar a passagem estreita e capturar o alvo em 5 minutos. No experimento da esquerda houve duas colisões (marcos 1 e 2), e no da direita ocorreram 3 colisões (marcos 1,2 e 4) e um evento de monotonia (marco 3).



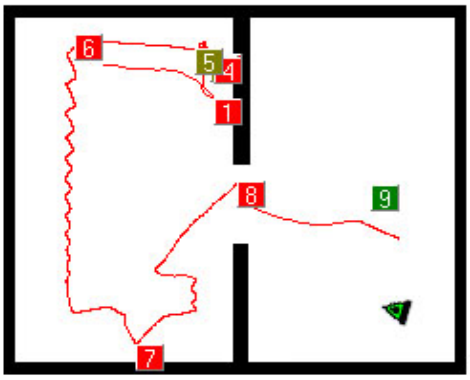
Colisões	Capturas	Monotonias	Tempo	Iterações
2	-	-	5 min	2658



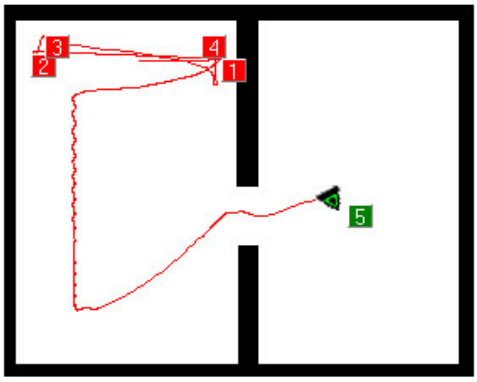
Colisões	Capturas	Monotonias	Tempo	Iterações
3	-	1	5 min	2664

Figura 4-4: Experimentos em que o robô apresenta velocidade constante.

Fica evidente, nos experimentos da Figura 4-5, a vantagem de se controlar a velocidade. Com velocidade variável, o robô foi capaz de transpor a passagem estreita e capturar o alvo. Isto foi possível porque, ao detectar informações de proximidade de obstáculos em ambos os lados (situação de estreitamento), o SNA associou-as ao conhecimento obtido anteriormente e então passou a executar ações de redução de velocidade, de forma a atravessar a abertura com segurança, se movimentando com manobras lentas e cautelosas



Colisões	Capturas	Monotonias	Tempo	Iterações
6	1	2	4:34	2431



Colisões	Capturas	Monotonias	Tempo	Iterações
4	1	-	3:41	1969

Figura 4-5: Experimentos em que o robô apresenta velocidade variável.

.

Como já foi detalhado na seção 3.5.1 do Capítulo 3, o controle da velocidade é feito pelas próprias regras, que indicam se o robô deve reduzi-la ou aumentá-la. O aperfeiçoamento deste controle se dá pelo processo evolutivo causado pelos eventos de colisão e captura que ajustam as regras adequadamente.

É conveniente destacar as diferenças de comportamentos dos robôs, representadas pelas trajetórias distintas. Mesmo em ambientes idênticos o comportamento do robô não é sempre igual, como visto nas Figuras 4-4 e 4-5. Isto ocorre pelo fato de o SNA ser inicializado em cada caso com regras aleatórias, implicando na ausência de conhecimento. O conhecimento vai sendo adquirido por meio dos processos evolutivos disparados em função da ocorrência dos eventos numerados. Portanto, o sistema aprende de maneiras distintas, segundo a ordem e características dos eventos evolutivos, sendo isto explicitado pelos comportamentos e trajetórias dos robôs.

Além disso, pode-se detectar outro detalhe interessante nestes experimentos: o número de colisões necessárias à formação do comportamento de desvio de obstáculos é maior quando a velocidade é variável. Isto é, no período inicial da simulação o robô com controle de velocidade colidiu mais vezes, pois além de aprender a desviar, ele precisou aprender a controlar adequadamente sua velocidade. Até ajustar o controlador, ocorreram mais eventos de colisão, em geral, se comparado à ausência de variação de velocidade. Com velocidade constante não há necessidade de aprendizagem de ajuste de velocidade, apenas aprendizagem de desvio. Conseqüentemente com menos eventos evolutivos o robô consegue evitar obstáculos adequadamente.

Os próximos experimentos contemplam o ambiente da Figura 4-6. Note que os alvos estão muito próximos do obstáculo, o que dificulta muito suas capturas. A navegação é encerrada quando os oito alvos são capturados. Embora a Figura 4-6 mostre os oito alvos posicionados, há somente um alvo ativo (em destaque) durante a navegação. Quando um é capturado, este é removido, e prontamente o próximo é ativado.

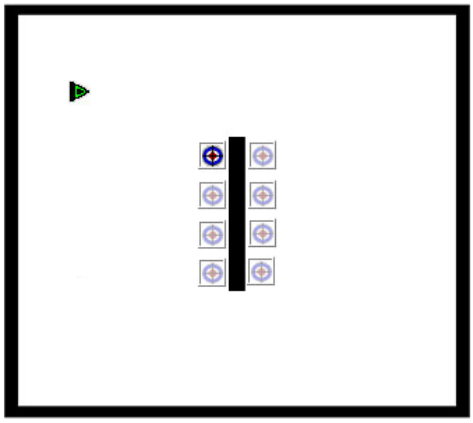


Figura 4-6: Ambiente com oito alvos muito próximos do obstáculo.

Dado o ambiente da Figura 4-6, quatro experimentos semelhantes foram realizados, cada qual empregando velocidades diferentes. No primeiro, o robô tinha velocidade mínima (0,2 u.v.), no segundo velocidade 1,5 u.v., no terceiro tinha velocidade máxima (3 u.v.) e o último contava com velocidade variável. O resultado dos experimentos está exposto na Tabela 4-3.

Tabela 4-3: Dados dos experimentos com alvos próximos do obstáculo.

Velocidade (u.v.)	Colisões	Capturas	Monotonias	Distância Percorrida (u.d.)	Iterações
0,2	6	8	4	4141,1	20707
1,5	11	8	5	5646,0	3762
3	28	8	2	10755,0	3584
variável	8	8	4	4142,3	9437

Analisando a Tabela 4-3, nota-se que quanto mais alta a velocidade fixada, maior o número de colisões e também a distância percorrida. Considerando que esta distância serve de medida de consumo (de combustível ou baterias, por exemplo), conclui-se que os robôs com maior velocidade foram menos econômicos. Os robôs que percorreram as menores distâncias totais neste experimento foram o de velocidade constante 0,2 u.v. e de velocidade variável. Ambos também foram os que cumpriram a tarefa de capturar os 8 alvos com melhor desempenho, ou seja, menor número de colisões. Contudo, o robô equipado com velocidade variável foi ainda melhor, levando-se em conta que ele precisou de praticamente metade das iterações que o robô de velocidade 0,2 u.v. necessitou. Isto se justifica porque a

possibilidade de variação de velocidade, além de permitir manobras lentas, também admite a aceleração em trechos menos conturbados, reduzindo o tempo de navegação.

Todos os três tipos de processos evolutivos, no tocante ao ajuste de velocidade, tendem a compor regras com ações de redução de velocidade. Desta forma, as regras com ação de aumento de velocidade somente existirão por exclusão, ou seja, regras que participaram poucas vezes ou mesmo não participaram de processos evolutivos. Como não ocorre aprendizagem em situações de distanciamento de obstáculos, as regras com padrões sensoriais de obstáculos distantes não participam de processos evolutivos e, mais uma vez por exclusão, acabam associadas a ações de aumento de velocidade. Portanto, é assim que o robô consegue acelerar em situações de ambiente livre.

Estes experimentos salientam uma dificuldade séria do SNA: ambigüidade sensorial (*perceptual aliasing*). Esta limitação é percebida nos experimentos pelo fato das colisões não estabilizarem, isto é, o sistema mesmo aprendendo a desviar não é capaz de identificar as situações de proximidade de obstáculos porque há a presença forte de estímulos de proximidade de alvo. Isto ocorre basicamente por haver confusão ou incapacidade de interpretação das informações lidas pelos sensores. Ou seja, os estímulos captados pelos sensores são ambíguos ao sistema, podendo estar associados a situações distintas e até mesmo contrárias, levando a tomada de decisões inconsistentes. No caso do ambiente cujos alvos estavam encostados no obstáculo, o robô recebia estímulos fortes de proximidade de alvo ao mesmo tempo em que também detectava proximidade do obstáculo. Nesta situação, o SNA enfrenta uma ambigüidade, pois não fica claro qual decisão tomar, se de captura ou de desvio.

4.2.3 Taxa de Procriação

A taxa de procriação é um mecanismo que regula o número de descentes produzidos a cada geração do processo evolutivo, assim como descrito no Capítulo 3. Ela é calculada em função do número de capturas consecutivas feitas pelo robô durante a navegação e pode variar de 1 a 10% do total de regras da população, segundo a equação (3-8).

Sabe-se que o objetivo do SNA é alcançar um desempenho em que a curva de colisões se mantenha estável, e a curva de capturas cresça continuamente. Entretanto, o crescimento contínuo da curva de capturas significa a ocorrência consecutiva de capturas de alvo, que, por conseqüência, significa uma constante geração de novas regras na população. Estas novas regras em excesso acabam por substituir ou deteriorar aquelas que têm maior afinidade ao comportamento de desvio de obstáculos, levando o robô a voltar a colidir. Visando manter o equilíbrio populacional e evitar estes problemas, a taxa de procriação foi implementada. E, para ilustrar o funcionamento do mecanismo, alguns experimentos foram realizados.

Primeiro foi utilizado o ambiente da Figura 4-7, com quatro regiões de alvos. O ambiente foi assim construído para exigir que o robô percorresse todos os corredores obrigatoriamente. Em um experimento empregou-se um robô sem o mecanismo de taxa de procriação, e no outro o robô incorporava o mecanismo. Ambos foram encerrados com 50 mil iterações.

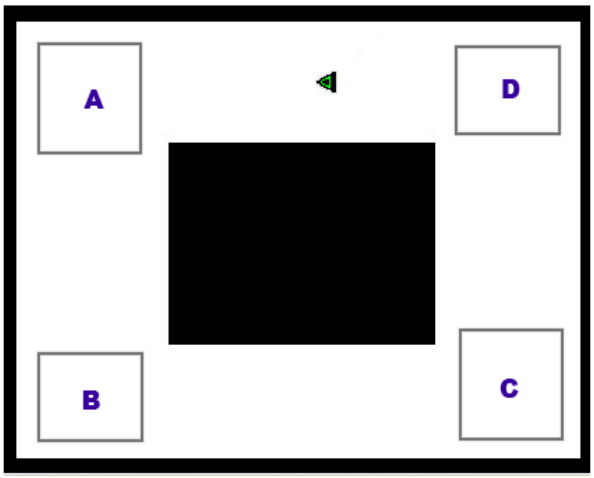
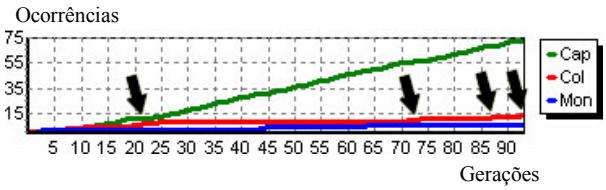


Figura 4-7: Ambiente com quatro regiões de alvos.

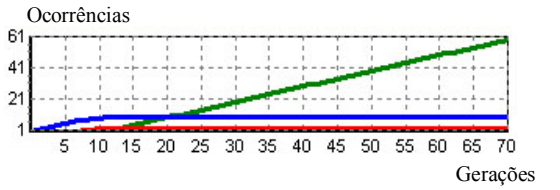
O desempenho de ambos os robôs é mostrado na Figura 4-8. Observando o gráfico à esquerda da Figura 4-8, relativo ao robô sem taxa de procriação, identificam-se os pontos de desestabilização do comportamento de desvio de obstáculos pelas setas pretas. Percebe-se que, após algumas colisões iniciais, o comportamento se estabilizou e o robô passou a capturar alvos. Por volta da geração 20, as constantes capturas levaram o robô a voltar a

colidir, recompondo o comportamento de desvio que havia sido deteriorado. Depois da geração 25 até a 70 o robô voltou a desempenhar bem o seu papel. No entanto, as cerca de 40 capturas consecutivas levaram novamente à deterioração do comportamento de desvio (geração 70). O mesmo processo se repete ainda duas vezes, por volta das gerações 86 e 92.

Diante dessa análise, fica claro que as capturas consecutivas causam a degradação do comportamento de desvio. Além disso, também se detecta que este problema é cíclico, se repetindo sempre e cada vez com um período menor. Comparado ao desempenho do robô com taxa de procriação (à direita), constata-se que este problema não ocorre, e que a quantidade de colisões é reduzida drasticamente. O gráfico indica que, depois de duas colisões iniciais, a curva se estabiliza e o robô não colide mais, ao passo que realiza capturas de alvos de forma consecutiva sem acarretar qualquer influência negativa no outro objetivo. Portanto fica assim demonstrada a eficiência da taxa de procriação na interrupção da interferência entre objetivos.



Colisões	Capturas	Monotonias	Gerações	Iterações
14	73	6	93	50000



Colisões	Capturas	Monotonias	Gerações	Iterações
2	59	9	70	50000

Figura 4-8: Sem taxa de procriação (à esquerda) e com taxa de procriação (à direita).

Para reforçar a verificação da taxa de procriação, conduziu-se o experimento a seguir. O ambiente também possui quatro regiões de alvos, mas apresenta topologia diferente, como pode ser visto na Figura 4-9. Esta simulação teve o dobro da duração da experiência anterior (Figura 4-8), mais especificamente, 100 mil iterações.

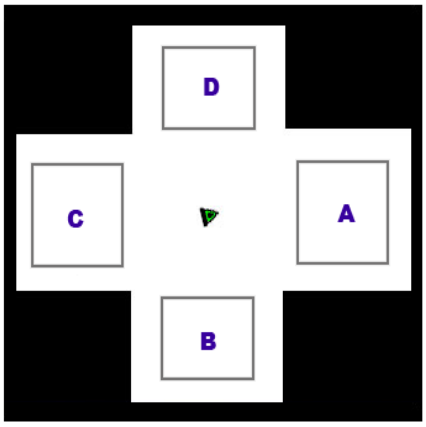
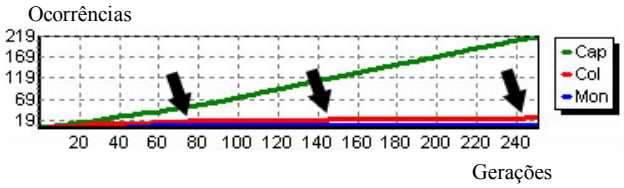
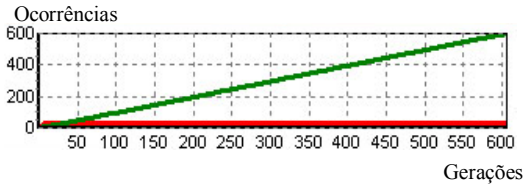


Figura 4-9: Ambiente com quatro regiões de alvo entre obstáculos.

Os gráficos dos experimentos, vistos na Figura 4-10, são análogos ao caso anterior. Percebe-se que o robô sem taxa de procriação (gráfico à esquerda) sofreu três fases de aprendizado por colisão, delimitadas por volta das gerações 76, 148 e 245. Já o robô que dispunha da taxa de procriação obteve um desempenho muito bom e completamente estável, colidindo 8 vezes apenas em uma única fase de aprendizado (contra 26 colisões do outro robô) e capturando 598 alvos (contra 217 do outro robô). Fica mais uma vez evidenciada a importância deste mecanismo e a melhora que ele promove junto ao desempenho do SNA.





Colisões	Capturas	Monotonias	Gerações	Iterações	Colisões	Capturas	Monotonias	Gerações	Iterações
26	217	8	251	100000	8	598	0	606	100000

Figura 4-10: Sem taxa de procriação (à esquerda) e com taxa de procriação (à direita).

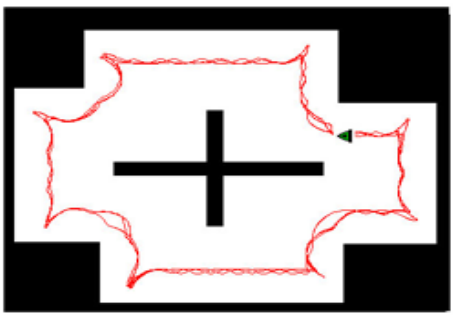
4.2.4 Tarefas Independentes

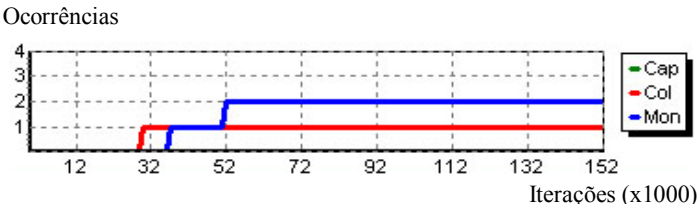
Como já foi dito anteriormente, as tarefas do SNA são capturar alvos e desviar de obstáculos de forma simultânea. Entretanto, antes de verificar o problema completo, é

conveniente dividi-lo em sub-partes. Isto é, será analisada nesta seção a capacidade do sistema em aprender e executar os comportamentos de forma independente e separada.

Embora esta seção contemple experimentos mais simples que as seções anteriores, a ordem das seções foi assim definida, pois os resultados dos experimentos anteriores foram incorporados ao SNA a partir daqui. Os parâmetros ajustados na seção 4.2.1, o mecanismo de controle de velocidade avaliado na seção 4.2.2 e a taxa de procriação validada na seção 4.2.3 passam a fazer parte do sistema de navegação de forma definitiva nos próximos experimentos.

Os dois primeiros experimentos desta seção somente exigiram do sistema a tarefa de desvio de obstáculos. Portanto não havia alvos no ambiente (as regiões de alvos foram desativadas). Os experimentos são independentes, todos foram encerrados quando o robô estabilizou seu comportamento de desvio (curva vermelha estável). As Figuras 4-11 e 4-12 mostram o ambiente e o desempenho obtido no experimento. Os traços em vermelho no ambiente representam a rota do robô.

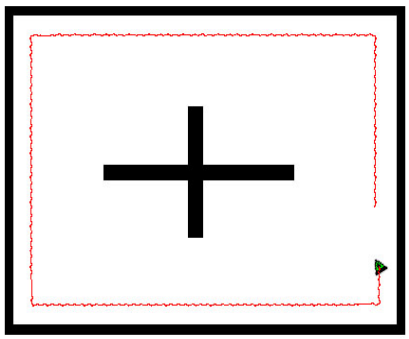


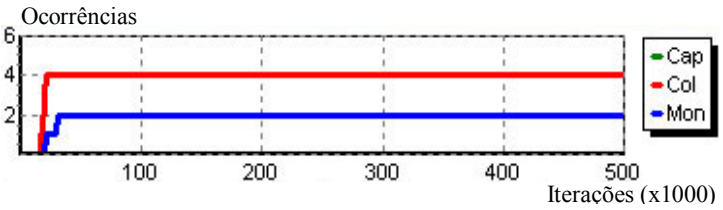


Colisões	Monotonias	Gerações	Iterações
1	2	3	15200

Figura 4-11: Ambiente e desempenho do robô somente exigindo desvio de obstáculos.

Analisando esses experimentos, verifica-se a grande eficiência de aprendizagem do SNA para o comportamento de desvio de obstáculos, dado que foram necessárias apenas poucas colisões (1 e 4) para que o robô passasse a evitar os obstáculos adequadamente, mesmo partindo de um conjunto aleatório de regras. Nos gráficos, observa-se um período inicial de aprendizagem (disparo do processo evolutivo) e, logo em seguida, a curva de colisões estabilizada.

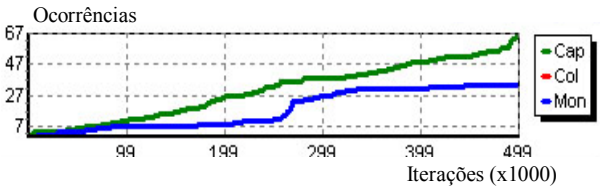




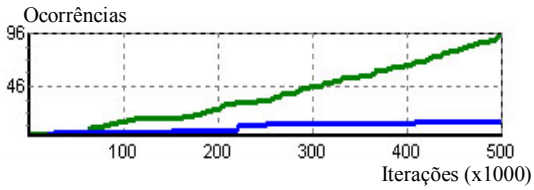
Colisões	Monotonias	Gerações	Iterações
4	2	6	50000

Figura 4-12: Robô navegando em um ambiente sem alvos e seu desempenho.

A seguir, dois experimentos que exigiram somente captura de alvos foram executados. O ambiente era composto simplesmente por uma única grande região de inserção automática de alvos sem nenhum obstáculo. O desempenho do sistema está demonstrado na Figura 4-13. Analisando-a, verifica-se que a tarefa de capturar alvos foi cumprida, apesar da ocorrência de alguns eventos de monotonia. Como nos experimentos anteriores, a população inicial de regras foi gerada aleatoriamente.



Capturas	Monotonias	Gerações	Iterações
65	34	99	49990



Capturas	Monotonias	Gerações	Iterações
94	14	108	50000

Figura 4-13: Dois experimentos envolvendo somente captura de alvos.

4.2.5 Tarefas Simultâneas

Tendo comprovado a eficiência do SNA para cada comportamento separadamente, foi então testado o sistema diante do problema completo, ou seja, executando as duas tarefas simultaneamente. O ambiente utilizado no último experimento desta seção está mostrado na Figura 4-14. As regiões de A a D são áreas de inserção automática de alvos. Tão logo o robô captura um alvo em A, aparece outro sucessivamente em B, C e D, recomeçando em A, e assim por diante.

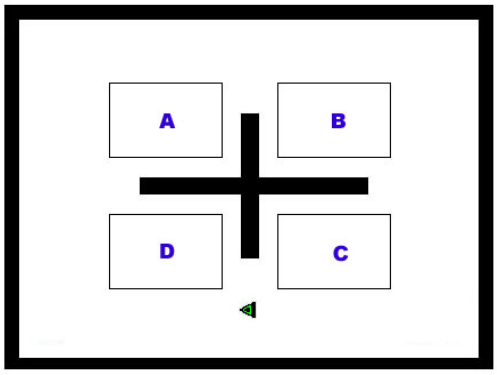
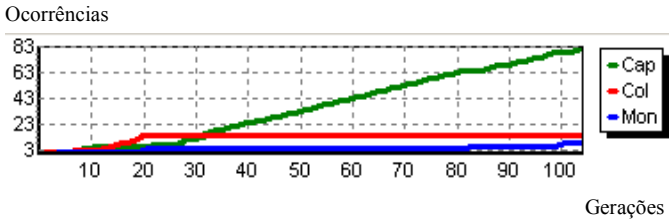


Figura 4-14: Ambiente com 4 regiões de alvos.

A Figura 4-15 apresenta o resultado do experimento, encerrado após 50 mil iterações. Assim como nos experimentos de tarefas independentes, neste também o robô cumpriu seus objetivos satisfatoriamente. Este gráfico está em função do número de gerações para mostrar que o SNA com poucas gerações foi capaz de compor adequadamente seus comportamentos. Observa-se que, após 20 gerações do sistema evolutivo, a curva de colisões estabilizou, indicando que o sistema passou a desviar efetivamente dos obstáculos. Com 30 gerações, o SNA começou a esboçar ambos os comportamentos e, daí em diante, passou a coordená-los apropriadamente, levando o robô a não mais colidir e a capturar alvos consecutivamente.



Colisões	Capturas	Monotonias	Gerações	Iterações
15	81	8	104	50000

Figura 4-15: Desempenho do sistema em um experimento completo.

4.2.6 Consistência da Aprendizagem

Nesta seção, busca-se avaliar a consistência da aprendizagem do SNA, isto é, em vários experimentos iguais verificar a regularidade da evolução do sistema. Neste sentido, dois

tipos de experimentos foram executados: um envolvendo somente desvio de obstáculos e outro completo.

Um experimento com 6 robôs em um único ambiente (fechado e com uma cruz no centro) foi realizado. Entretanto, nenhum resultado conclusivo foi obtido devido ao fato de o SNA dos robôs não ser preparado para navegar de forma coletiva. Também por conta de vários robôs navegarem por um ambiente reduzido, colisões entre eles ocorriam freqüentemente, impedindo que os sistemas de navegação evoluíssem naturalmente. Caso o ambiente fosse maior, ou houvesse menos robôs navegando, talvez eles convergissem para comportamentos mais bem definidos.

A seguir outro experimento foi feito, no qual foram preparados 6 ambientes iguais (sem alvos) e foi disposto 1 robô com ausência de conhecimento inicial para navegar em cada ambiente, em posições iniciais aleatórias, conforme mostra a Figura 4-16. A simulação durou 5000 iterações, e a trajetória de cada robô é mostrada pelos traços contínuos.

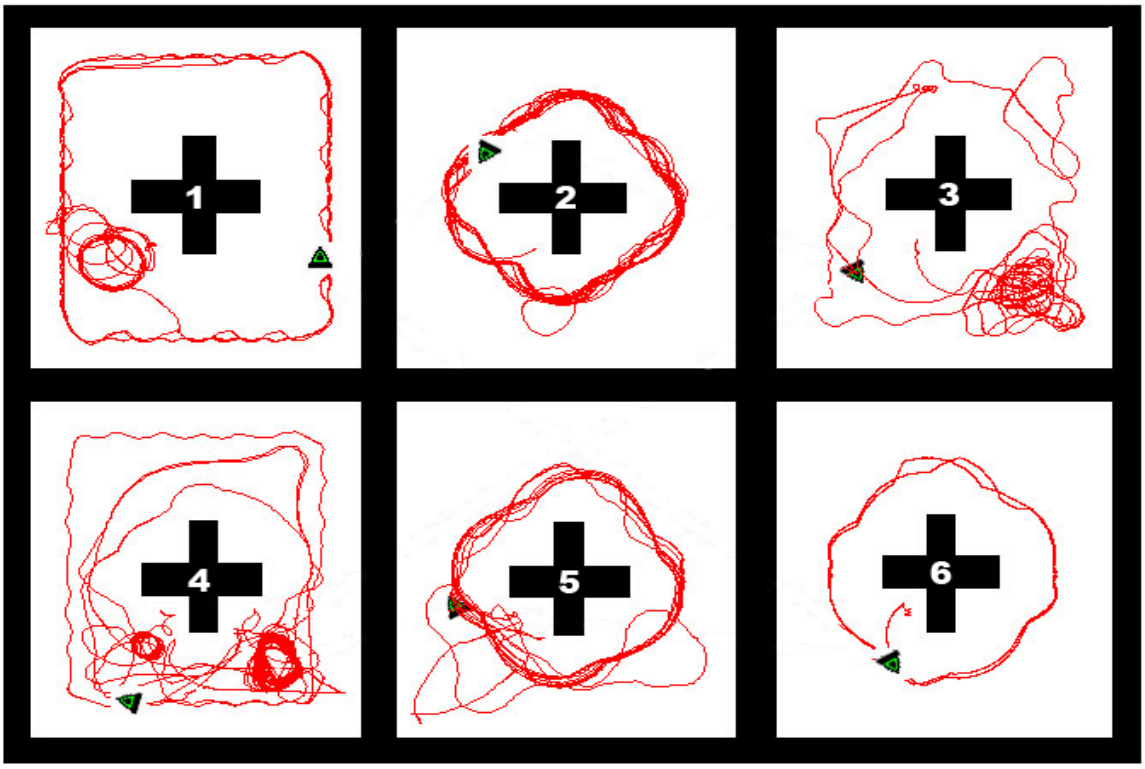


Figura 4-16: Seis robôs navegando de forma independente em ambientes iguais .

O resultado do experimento está disposto na Tabela 4-4. Observa-se que, na média dos seis robôs, ocorreram 4,33 colisões e 2,16 eventos de monotonia. A distância média percorrida foi de 4515,8 u.d. com uma velocidade média de 0,9 u.v. É interessante notar os diferentes tipos de comportamentos que os robôs apresentam para uma mesma tarefa em um mesmo ambiente. Os robôs 2, 5 e 6 adotaram condutas semelhantes, navegando ao redor do obstáculo central. Ao contrário, o robô 1 se portou de forma a acompanhar as paredes externas do ambiente e, finalmente, os robôs 3 e 4 não definiram um comportamento fixo, apresentando trajetórias relativamente aleatórias.

Tabela 4-4: Resultados do desempenho dos seis robôs.

Número do Robô	Colisões	Monotonias	Distância Percorrida (u.d.)	Velocidade Média (u.v.)
1	3	2	4383	0,87
2	1	1	7105	1,42
3	3	4	3793	0,76
4	8	4	6075	1,21
5	6	1	4709	0,94
6	5	1	1030	0,20
Média	4,33	2,16	4515,8	0,90

O experimento a seguir averigua a consistência da aprendizagem do SNA no ambiente da Figura 4-17, com duas regiões de alvos. O experimento foi completo, ou seja, envolveu tanto desvio de obstáculos quanto captura de alvos. No entanto, neste caso, os seis robôs foram testados separadamente com ausência de conhecimento inicial e com duração de 5 mil iterações.

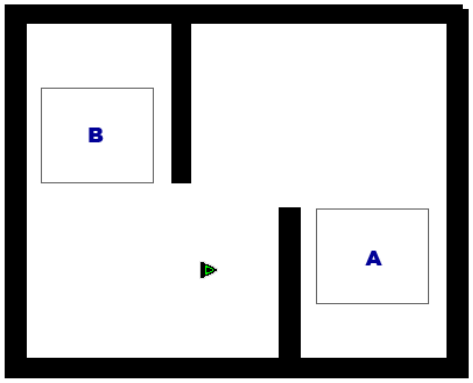


Figura 4-17: Ambiente com duas regiões de alvos.

Em média, os robôs sofreram 6,33 colisões e 4,33 eventos de monotonia, conforme pode ser visto na Tabela 4-5. Também foram capturados 15,66 alvos, em média, a distância média percorrida foi de 6418 u.d. e a velocidade média de 0,42 u.v. Comparando a velocidade média deste experimento com a do anterior, verifica-se, que quando a tarefa de captura de alvos está envolvida, a velocidade se reduz bastante. Neste caso, praticamente pela metade.

Tabela 4-5: Dados de desempenho dos seis robôs.

Número do Robô	Colisões	Capturas	Monotonias	Distância Percorrida (u.d.)	Velocidade Média (u.v.)
1	3	17	2	6014	0,40
2	8	26	4	9668	0,64
3	6	12	5	6441	0,43
4	7	13	4	4681	0,31
5	5	12	8	6099	0,40
6	9	14	3	5605	0,37
Média	6,33	15,66	4,33	6418	0,42

Os experimentos mostram certa discrepância entre os dados de cada robô, em termos de eventos de colisão, captura e monotonia, o que é aceitável, considerando que todos os robôs iniciaram sua navegação com um conjunto distinto e aleatório de regras. Mais importante que a uniformidade dos dados, é a regularidade de desempenho. Isto é, em todos os experimentos, sem exceção, os robôs alcançaram seus objetivos e apresentaram um desempenho desejado. É certo que cada qual à sua maneira e ao seu tempo, mas todos convergiram satisfatoriamente.

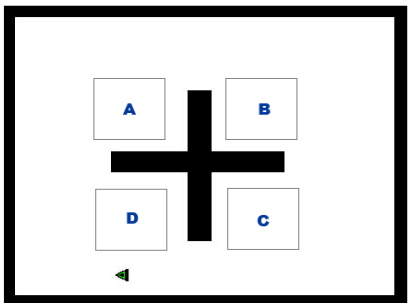
4.2.7 Generalização

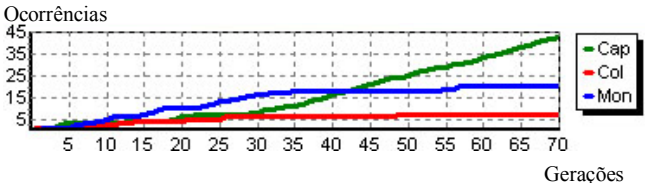
A capacidade de generalização de um sistema de navegação é fundamental para que ele consiga atuar eficientemente diante de situações inesperadas, com as quais ele nunca lidou antes. Basicamente, a generalização se dá pela associação de conhecimentos adquiridos anteriormente a situações novas e distintas daquelas que propiciaram o aprendizado. No caso específico do SNA implementado neste trabalho, isto ocorre por meio da aplicação de

regras que, apesar de terem sido aprendidas em outras condições, apresentam certa semelhança junto às situações diferentes que o robô está confrontando.

Com o intuito de examinar a capacidade de generalização do SNA, vários experimentos foram executados. Os experimentos seguem um roteiro simples: em um ambiente inicial e simples, o robô sem conhecimento inicial navega até que os objetivos de navegação sejam atendidos. Então o robô é transferido para ambientes mais complexos, diferentes do primeiro, onde ele deve capturar os alvos dispostos. O SNA é o mesmo obtido a partir do processo de treinamento realizado apenas junto ao ambiente inicial.

O ambiente inicial e simples possui quatro regiões de alvos, como exibido na Figura 4-18. O experimento envolve 50 mil iterações, e o desempenho do robô, mostrado no gráfico, indica que o robô capturou 43 alvos, colidiu 7 vezes e sofreu 20 eventos de monotonia.





Colisões	Capturas	Monotonias	Gerações	Iterações
7	43	20	70	50000

Figura 4-18: Ambiente básico para treinamento até a maturação do SNA e dados obtidos.

A seguir, o robô, com o SNA nas mesmas condições em que terminou o experimento anterior, foi colocado no ambiente da Figura 4-19. Como se pode ver, o robô cumpriu a tarefa proposta, capturando os 4 alvos (marcos verdes 1, 3, 4 e 5) , ocorrendo apenas um evento de monotonia e não colidindo.

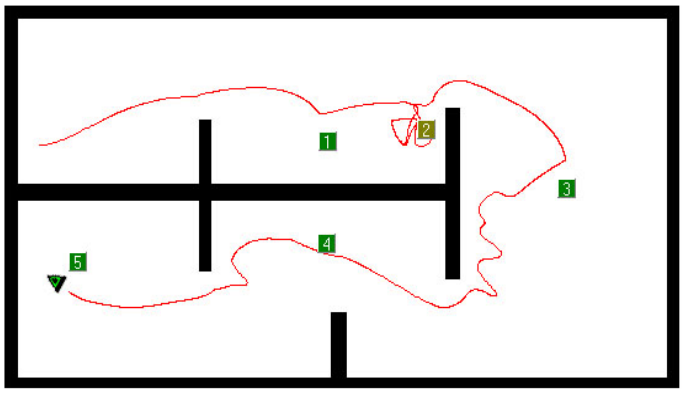


Figura 4-19: Ambiente para teste de generalização com quatro alvos.

Tomando ainda o robô com o SNA maturado no primeiro experimento desta seção, este foi colocado no ambiente da Figura 4-20, que contém 13 alvos. Mais uma vez o robô alcançou todos os alvos sem colidir, mostrando assim que foi capaz de atuar bem em ambientes cuja topologia é diferente daquela utilizada durante a fase inicial de treinamento.

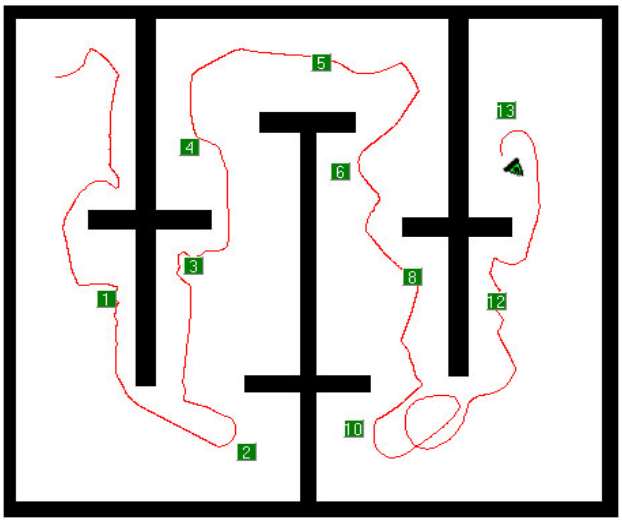
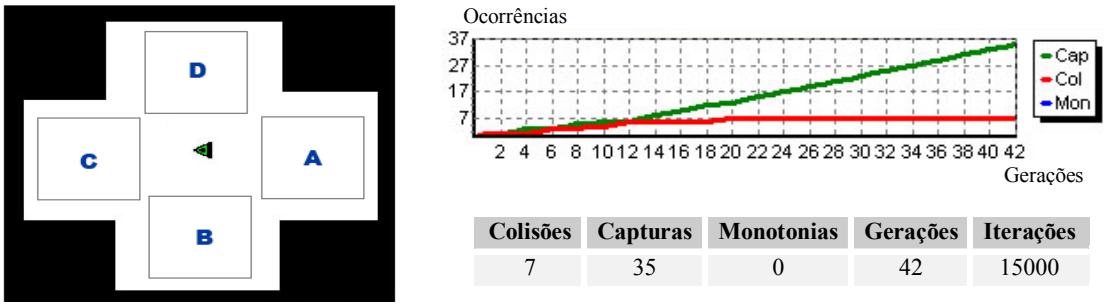


Figura 4-20: Ambiente para teste de generalização com treze alvos.

A mesma seqüência anterior é repetida para os próximos experimentos, mudando os ambientes. Pode-se observar o ambiente inicial de treinamento para maturação, com quatro regiões de alvos, e o desempenho obtido pelo robô na Figura 4-21. Em 15 mil iterações, o robô capturou 35 alvos e colidiu 7 vezes em obstáculos.



Colisões	Capturas	Monotonias	Gerações	Iterações
7	35	0	42	15000

Figura 4-21: Ambiente inicial para treinamento até a maturação do SNA e dados associados ao desempenho do robô.

Tendo sido treinado no ambiente anterior, o robô foi disposto no ambiente da Figura 4-22 para que capturasse quatro alvos. Mesmo sendo o ambiente diferente do anterior, onde o SNA se adaptou, o robô cumpriu as tarefas sem problemas.

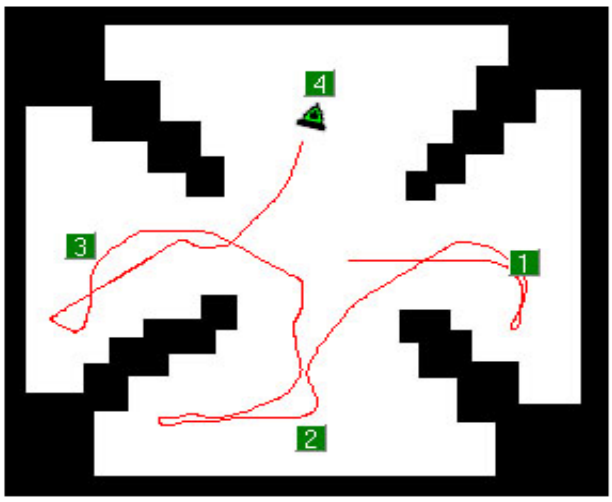


Figura 4-22: Ambiente para teste de generalização com quatro alvos.

Também colocado em um ambiente diferente do anterior, apresentado na Figura 4-23, com 8 alvos o robô cumpriu plenamente os seus objetivos. Ele capturou todos os alvos sem colidir uma única vez.

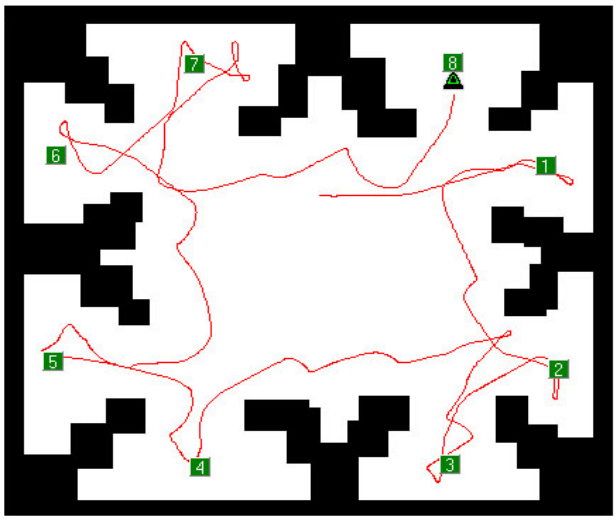


Figura 4-23: Ambiente para teste de generalização com oito alvos.

A seqüência de experimentos desta seção demonstrou que o SNA apresenta capacidade de generalização. A idéia foi evoluir o sistema até se tornar maduro em um ambiente simples e então transferi-lo para outro ambiente diferente, desafiando-o a navegar pelo ambiente desconhecido sem colidir e capturando os alvos pré-dispostos. Em todos os casos, o robô cumpriu a tarefa de forma competente. Embora as situações encontradas nos ambientes onde não houve treinamento tenham sido novas para o sistema, ele conseguiu associá-las de forma coerente aos conhecimentos obtidos anteriormente, de modo a tomar ações adequadas para as situações.

4.2.8 Análise das Regras

Esta seção tem por objetivo investigar especificamente a base de regras e sua evolução em alguns experimentos. As regras são a representação concreta dos comportamentos exibidos pelo robô durante sua navegação. Portanto, o enfoque aqui é analisar o SNA em um nível mais baixo, ou seja, analisar a composição das regras do sistema.

Existe uma variação significativa dentro do conteúdo de regras, a ponto de ser possível afirmar que não há regras iguais no conjunto todo. Para viabilizar a investigação delas de acordo com sua composição, foram definidas, classes específicas de cada parte da regra que identificam um certo padrão. As classificações são feitas conforme os componentes de cada parte da regra e dizem respeito à posição do obstáculo, alvo, ao ajuste de direção e ao ajuste

de velocidade propostos pela regra. A Tabela 4-6 mostra as possíveis classes de cada parte da regra.

Tabela 4-6: Tipos em que as partes das regras são classificadas.

Parte da Regra	Obstáculo	Alvo	Direção	Velocidade
Tipos	Longe	Longe	Virar à esquerda	Aumentar
	Próximo e à esquerda	Próximo e à esquerda	Seguir reto	Diminuir
	Próximo e à frente	Próximo e à frente	Virar à direita	-
	Próximo e à direita	Próximo e à direita	-	-
	-	Próximo e atrás	-	-

A primeira análise foi feita considerando a base de regras obtida durante o experimento da Figura 4-8 (à direita) cujo ambiente foi mostrado na Figura 4-7. Observando a Tabela 4-7, em que os dados relativos às regras são mostrados, pode-se verificar a porcentagem de cada tipo presente na população. O principal destaque foi que dentre todas as regras, 50,5% delas possuíam padrões de obstáculo próximo e à esquerda do robô. O fato de que a maioria das regras possuísse seu antecedente de desvio indicando obstáculo à esquerda se deu por influência da topologia do ambiente. Nele existia um obstáculo central, e as regiões de inserção de alvos exigiam que o robô navegasse no sentido anti-horário, conseqüentemente as colisões ocorriam mais freqüentemente junto ao lado direito do robô, produzindo regras semelhantes a esta situação.

Tabela 4-7: Proporção de cada tipo de regra presente na população.

Obstáculo	%	Alvo	%	Direção	%	Velocidade	%
Longe	24	Longe	41,5	Virar à esquerda	40,5	Aumentar	51,5
Próximo e à esquerda	50,5	Próximo e à esquerda	6,5	Seguir reto	22,5	Diminuir	48,5
Próximo e à frente	15,5	Próximo e à frente	29	Virar à direita	37	-	-
Próximo e à direita	10	Próximo e à direita	14	-	-	-	-
-	-	Próximo e atrás	9	-	-	-	-

A seguir foram avaliadas as regras do experimento da Figura 4-15, cujo ambiente foi mostrado na Figura 4-14. Conforme os dados da Tabela 4-8, verifica-se que a maioria das regras (63,5%), ao final do experimento, eram compostas por padrões de obstáculo distante.

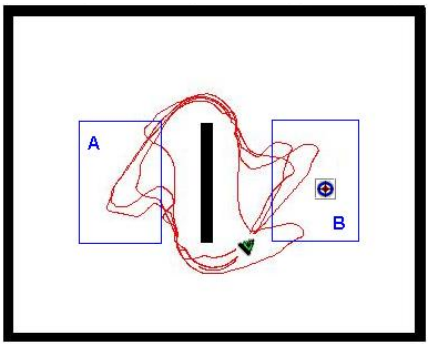
Tabela 4-8: Porcentagem de cada tipo de regra presente na população.

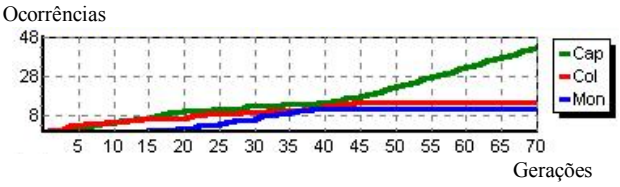
Obstáculo	%	Alvo	%	Direção	%	Velocidade	%
Longe	63,5	Longe	34,5	Virar à esquerda	41	Aumentar	45,5
Próximo e à esquerda	19	Próximo e à esquerda	16	Seguir reto	27	Diminuir	54,5
Próximo e à frente	12,5	Próximo e à frente	26	Virar à direita	32	-	-
Próximo e à direita	5	Próximo e à direita	15	-	-	-	-
-	-	Próximo e atrás	8,5	-		-	-

Conclusões interessantes podem ser tiradas ao acompanhar a Figura 4-24, em que são mostrados os gráficos do histórico de porcentagens dos antecedentes de obstáculo e de alvo durante as gerações do processo evolutivo, respectivamente à esquerda e à direita. No gráfico da esquerda, fica claro o aumento do número de regras com padrões de obstáculo longe e a redução de regras com padrões de obstáculo próximo. Isto ocorreu porque, inicialmente, quando aconteceram várias colisões, foram geradas regras relacionadas a situações de proximidade de obstáculo. Tão logo o robô começou a capturar alvos, a porcentagem de regras com padrão de obstáculo longe (curva verde) começou a aumentar. Em geral, as regras associadas ao comportamento de captura possuem padrões de obstáculo distante, pois, geralmente a situações de captura de alvo coincidem com situações de obstáculo longe. Justamente por isso este tipo de regra domina a população quando muitas capturas ocorrem. Além disso, nota-se no gráfico da direita que, quanto mais eficiente o robô se torna na tarefa de capturar alvos, mais regras com padrão de alvo à frente são geradas. A curva laranja mostra que este tipo de regra ocupava 6% da população no início da navegação e, ao final, já era 26% do total. Ou seja, o robô tende a capturar alvos de forma frontal à medida que se torna mais apto nesta tarefa.

Figura 4-24: Gráficos da variação dos tipos de regras em relação a obstáculo (à esquerda) e relativo a alvo (à direita).

Considerando a população de regras do experimento da Figura 4-25, cujo ambiente e desempenho são mostrados na mesma figura, algumas análises foram feitas. Neste ambiente, logo que o robô captura um alvo na região A, outro alvo é disposto em uma posição aleatória da região B, e assim alternadamente.





Colisões	Capturas	Monotonias	Gerações	Iterações
15	43	12	70	50000

Figura 4-25: Experimento simulado em ambiente com duas regiões de alvos.

Notando-se os gráficos da Figura 4-26, relativos ao experimento da Figura 4-25, percebe-se como as regras evoluíram. No gráfico (a), mais uma vez a curva verde (obstáculo longe) cresceu conforme o robô passou a capturar alvos. Outro fato a se notar, foi o incremento inicial das regras de obstáculo próximo e à frente (curva laranja). Devido à topologia do ambiente, com um obstáculo entre o robô e os alvos, inicialmente houve colisões frontais causando este incremento. Entretanto, a partir do momento que o robô aprendeu a não mais colidir, a curva decresceu até se estabilizar. Este decrescimento ocorre porque este tipo de regra acaba sendo substituído por outros tipos que são gerados com maior intensidade.

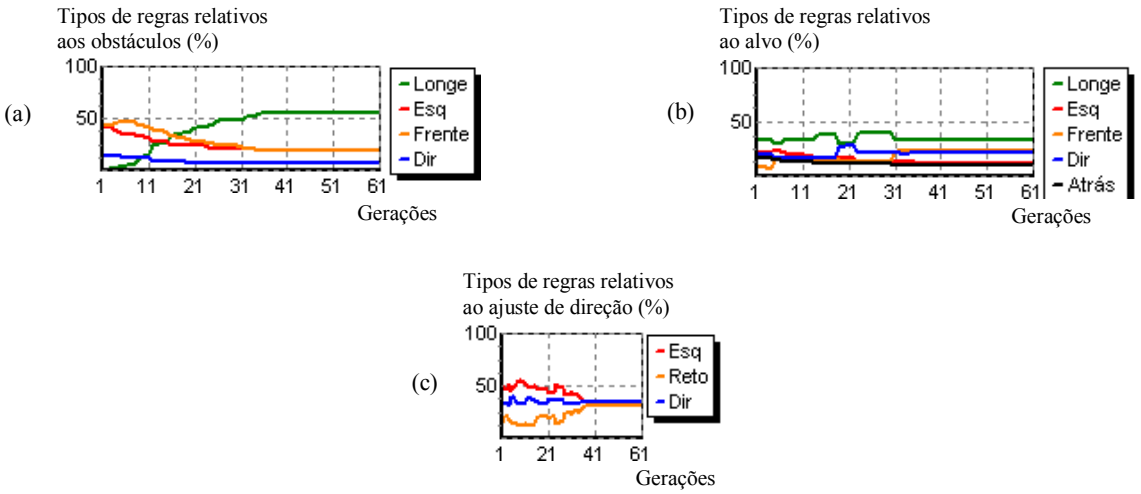
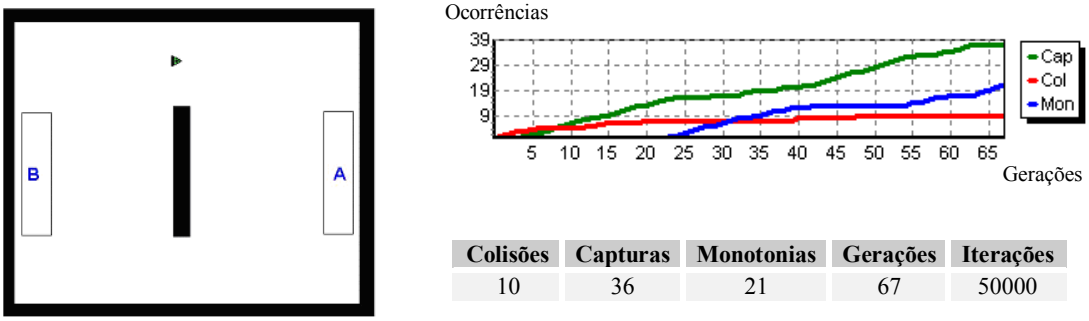


Figura 4-26: Gráficos da variação dos tipos de regras em relação a obstáculo (a), relativo a alvo (b) e associado a ajuste de direção.

No gráfico (b) outro fato se repete: há um grande crescimento das regras com padrão de alvo à frente (curva laranja) à medida que o robô passa a capturar alvos mais eficientemente, partindo de 8% até 23,5%. Por último, no gráfico (c), é interessante notar pelas curvas que os três tipos de regras, em termos de ajuste de direção, ocuparam partes aproximadamente iguais, por volta de 33% da população total. Mesmo iniciando em quantidades diferentes, próximo da geração 35 estas regras se estabilizaram em um patamar de equilíbrio.

O último experimento desta seção contou com um ambiente onde as áreas de inserção de alvos foram dispostas muito próximas dos obstáculos, segundo pode ser visto na Figura 4-27. O intuito foi mostrar como as regras evoluem de forma diferente nestas condições. O gráfico do experimento mostra que, embora os comportamentos de captura e desvio tiveram um desenvolvimento equivalente aos demais experimentos, os eventos de monotonia ocorreram de forma intensa, principalmente ao final da navegação. A ocorrência mais freqüente de eventos de monotonia se deu pelo fato dos alvos estarem dispostos muito próximos dos obstáculos, acarretando assim uma maior dificuldade e demora para a captura.



Colisões	Capturas	Monotonias	Gerações	Iterações
10	36	21	67	50000

Figura 4-27: Experimento cujo ambiente contemplava alvos próximos dos obstáculos e desempenho do robô.

Os dados das regras foram colocados na Tabela 4-9. Examinando-os, nota-se que em relação a obstáculo e alvo, as regras se constituíram de forma diferente dos experimentos anteriores. Provando a influência que tem a topologia do ambiente na evolução das regras, neste caso as regras com padrões de obstáculo próximo e também de alvo próximo tiveram uma participação maior na população.

Tabela 4-9: Proporção de cada tipo de regra presente na população.

Obstáculo	%	Alvo	%	Direção	%	Velocidade	%
Longe	28,5	Longe	40,5	Virar à esquerda	48,5	Aumentar	49
Próximo e à esquerda	20,5	Próximo e à esquerda	13,5	Seguir reto	19,5	Diminuir	51
Próximo e à frente	31,5	Próximo e à frente	12	Virar à direita	32	-	-
Próximo e à direita	19,5	Próximo e à direita	17,5	-	-	-	-
-	-	Próximo e atrás	16,5	-	-	-	-

Acompanhando o gráfico da evolução das regras, em termos de padrões de obstáculo, exibido na Figura 4-28, fica claro que as regras com padrão de obstáculo próximo e à frente (curva laranja) dominaram a população a maior parte do experimento. Devido ao fato de as capturas de alvo acontecerem em situações em que o obstáculo estava sempre bem próximo do robô, as regras geradas pelo processo evolutivo de captura contemplavam este tipo de padrão de obstáculo. Na parte final do gráfico, observa-se que as regras com padrão de obstáculo longe começam a tomar o lugar das regras com padrão de obstáculo próximo e à

frente (curvas verde e laranja, respectivamente). Os responsáveis por isso foram os eventos de monotonia ocorridos freqüentemente ao final do experimento. Os processos evolutivos de monotonia, cumprindo seu papel de introduzir diversidade à população, fizeram aumentar o número de regras do tipo minoritário (curva verde), conseqüentemente reduzindo a quantidade de regras do tipo que era maioria (curva laranja).

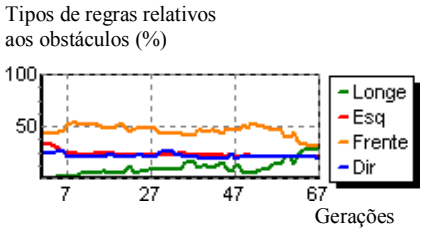


Figura 4-28: Gráfico da variação do número de regras segundo a posição de obstáculo.

A análise do conteúdo das regras realizada nesta seção propiciou algumas observações gerais relevantes. Primeiramente, os gráficos mostram claramente que a variação do número de regras de um certo tipo influencia diretamente a proporção dos outros tipos correlatos na população. Ou seja, as regras substituem umas às outras ao longo das gerações e seus tipos são determinados de acordo com a freqüência, ordem e situações dos eventos evolutivos. Controlar estas variações e manter o equilíbrio na população são justamente as funções do mecanismo de taxa de procriação (Capítulo 3, seção 3.5.3.10). Percebe-se também que a partir do momento em que o robô passa a cumprir adequadamente suas tarefas, as variações das proporções de tipos de regras diminuem bastante, e as curvas estabilizam. Com relação aos ajustes de direção das regras, os três tipos, em geral, tendem a assumir proporções aproximadamente iguais na população. O mesmo ocorre com as regras em questão de ajuste de velocidade: em geral elas se dividem em metade para redução e metade para aumento de velocidade.

4.3 Navegação com Robô Real: Experimentos e Resultados

4.3.1 Treinamento até Maturação e Validação em Ambiente Real

A simulação computacional de sistemas de navegação autônomos traz facilidades como economia de tempo e equipamentos. Entretanto, por mais completo que seja o simulador, é praticamente impossível reproduzir nele todos os pormenores de um ambiente real. Neste sentindo, é importante realizar testes para se validar o sistema de navegação tanto em simulação como de forma prática. Com este objetivo, dois experimentos interligados foram conduzidos, sendo que no primeiro houve o treinamento do SNA até sua maturação por meio de simulação e sua validação no ambiente real, a bordo do robô Khepera II. É importante destacar que mesmo quando encerrada a navegação no ambiente de treinamento, o SNA continua a evoluir normalmente, mesmo no ambiente de validação.

O simulador foi implementado de forma que o modelo do robô em simulação fosse equivalente ao robô real em termos de suas características como número e posição de sensores, capacidade de manobras, etc.

O primeiro experimento considerado foi aquele mostrado na Figura 4-24, realizado em modo de simulação. As regras do sistema foram inicializadas aleatoriamente. O experimento durou cerca de 10 minutos, contabilizando 50 mil iterações. Durante este período, o robô colidiu 15 vezes até estabilizar este comportamento, sofreu 12 eventos de monotonia e capturou 43 alvos, sendo as 26 últimas vezes consecutivas. O desempenho do SNA, apresentado no gráfico da Figura 4-25 mostra que depois de 45 gerações do sistema evolutivo o robô passou a executar ambas as tarefas adequadamente e, ao final do período, já demonstrava uma conduta satisfatória, capturando alvos consecutivamente e não incorrendo em colisões.

Tendo havido, durante a simulação, o treinamento do SNA até sua maturação, este foi transferido para o robô Khepera II, e um novo experimento foi executado em um ambiente real, mostrado na Figura 4-29, de topologia análoga ao ambiente virtual. Quando o robô

alcança a lâmpada, esta é colocada no lado oposto do ambiente, e assim por diante. É interessante enfatizar que, apesar do treinamento do sistema ter sido feito em simulação, ao navegar pelo ambiente real o SNA continua evoluindo e se adaptando normalmente, de acordo com a ocorrência dos eventos evolutivos.

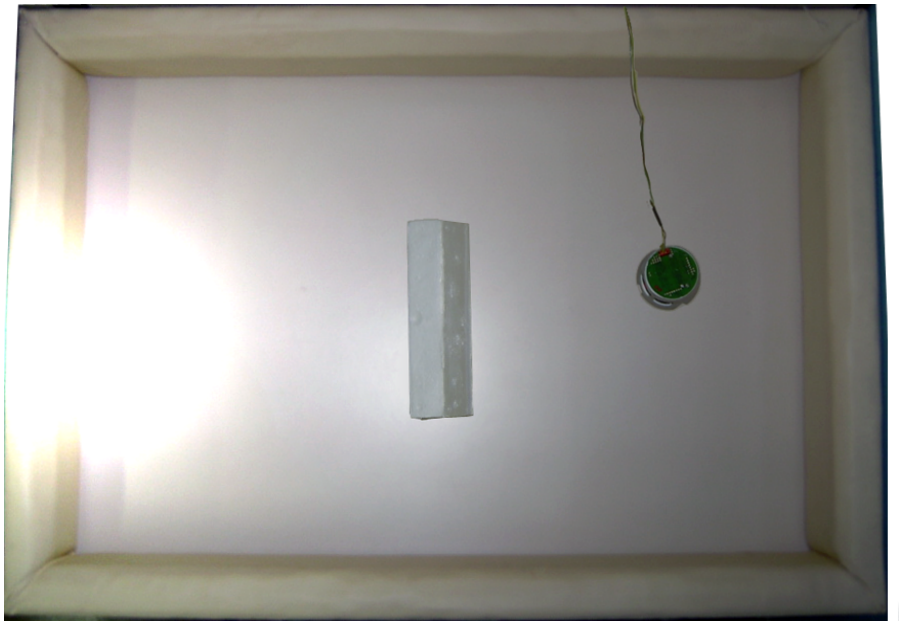
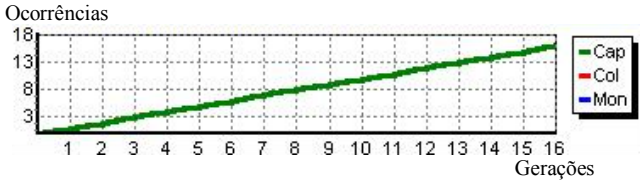


Figura 4-29: Ambiente real com um obstáculo entre o robô e a fonte de luz.

O desempenho obtido neste experimento está sintetizado na Figura 4-30. O teste durou 58 minutos e foi encerrado quando 5000 iterações foram executadas. Neste intervalo, o robô atingiu a fonte luminosa 16 vezes sem que colidisse e nem sequer sofresse eventos de monotonia.



Colisões	Capturas	Monotonias	Gerações	Tempo	Iterações
-	16	-	16	58 min	5000

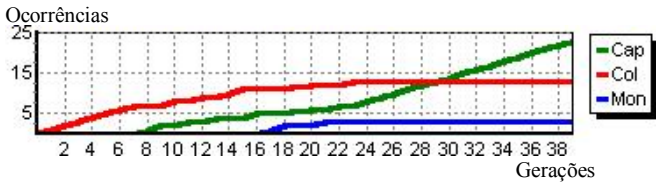
Figura 4-30: Desempenho do sistema em ambiente real com um obstáculo central.

Diante destes resultados, pode-se afirmar que os comportamentos que emergiram por meio da simulação foram consistentes e eficazes também na situação real. O SNA mostrou-se robusto, pois foi capaz de cumprir seus objetivos, tratando adequadamente certos fatores do ambiente real ausentes em seu treinamento virtual, como imprecisão na leitura dos sensores. Além de validar o sistema em situações reais, os dois experimentos anteriores demonstraram também que o simulador é suficientemente adequado ao treinamento, ou maturação, do sistema de navegação em ambientes que obedeçam a uma dinâmica não muito complexa. Ficou igualmente claro o dispendioso tempo gasto em experimentos reais quando comparado à simulação.

4.3.2 Experimentos Apenas em Ambientes Reais

Embora a maioria dos experimentos tenha sido feita por meio de simulação, não significa que o sistema é incapaz de se desenvolver e evoluir diretamente no ambiente real, ou seja, fora do simulador. Para demonstrar a evolução do sistema diretamente no ambiente real, partindo da ausência de conhecimento inicial, e comparar paralelamente à evolução por simulação, implementou-se o experimento a seguir. O ambiente utilizado foi o mesmo apresentado na Figura 4-29.

A performance do Khepera, exibida na Figura 4-31, ratifica a eficiência do SNA desenvolvido, desta feita sempre em situações reais. Observa-se que ocorreram 13 colisões, 3 eventos de monotonia e 23 capturas durante todo o experimento, cuja duração foi de 74 minutos (5000 passos do robô).



Colisões	Capturas	Monotonias	Gerações	Tempo	Iterações
13	23	3	39	74 min	5000

Figura 4-31: Performance do robô com aprendizagem totalmente em ambiente real.

Tanto este experimento quanto o primeiro da seção 4.3.1 foram feitos em ambientes de mesma topologia, sendo um em modo de simulação e outro em modo real. O número de colisões necessárias para que o comportamento de desvio de obstáculos se estabilizasse foi próximo em ambos os experimentos, 15 e 13 respectivamente. A quantidade de capturas foi bem maior na simulação pelo fato do experimento ter sido mais longo em termos de iterações. Houve grande diferença em relação aos comportamentos monótonos: no experimento real, ocorreram muito poucos eventos de monotonia. Isto se explica por existir mais variação nos estímulos sensoriais no ambiente real, também motivado pelo fato dos sensores serem a todo tempo acometidos de ruídos, implicando em uma maior diversidade na ativação das regras.

O último experimento é uma continuação do terceiro e tem dois objetivos: complementar a validação do SNA em ambientes reais e examinar a capacidade de generalização do sistema em situações reais. Para isto, foi preparado um ambiente um pouco mais complexo, contendo 3 obstáculos, conforme mostra a Figura 4-32, isto é, um ambiente diferente daquele onde o robô havia sido treinado. Inicialmente, o sistema de navegação evoluído no experimento anterior foi transferido e colocado para controlar o robô.

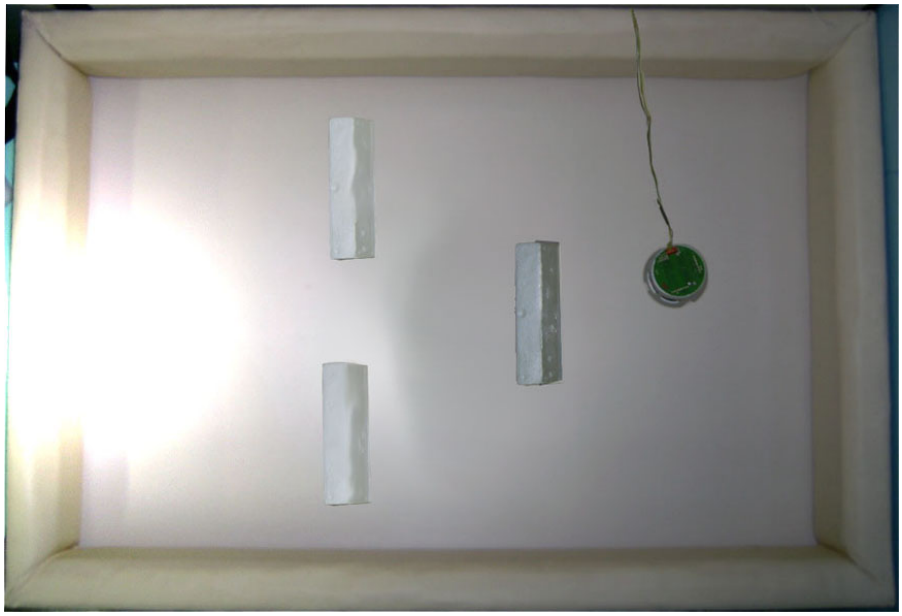


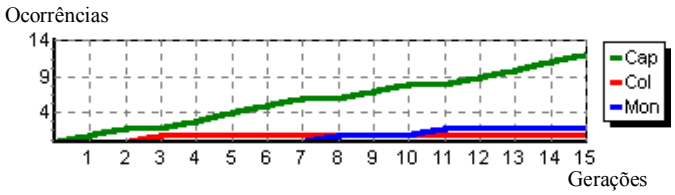
Figura 4-32: Ambiente com três obstáculos.

A experiência durou 87 minutos e foi concluída ao completar 5000 iterações. Segundo a Figura 4-33, o robô capturou 12 alvos, sofreu 2 eventos de monotonia e colidiu uma única vez. A única colisão ocorreu na quina de um obstáculo próximo à lâmpada, possivelmente influenciada pela forte luminosidade, que provoca leituras praticamente aleatórias nos sensores. Quando incide sobre o robô uma luminosidade muito forte, ele se torna incapaz de detectar obstáculos e acaba colidindo.

Com relação à ocorrência de dois eventos de monotonia, a explicação é a seguinte. As duas vezes que o robô se portou de forma monótona aconteceram justamente pelo motivo contrário do caso do evento de colisão em obstáculo: a ausência de luminosidade. Logo que o robô alcançou a fonte luminosa, a mesma foi colocada do lado oposto do ambiente, fazendo com que a luminosidade ficasse muito reduzida no local onde se encontrava o robô, levando-o a um comportamento monótono pela insuficiência de estímulos.

Os eventos de colisão e monotonia que ocorreram neste experimento foram causados por circunstâncias casuais do ambiente, e não exatamente por deficiências do sistema. Independentemente disto, o mais importante é notar que o SNA foi capaz de generalizar o conhecimento produzido em um ambiente mais simples e atuar efetivamente em um ambiente diferente daquele onde ele havia se habituado anteriormente.

A duração deste experimento foi relativamente longa, prolongando-se por 1 hora e 27 minutos. Durante o período do experimento, ocorreram apenas 15 gerações. Isto se deve à complexidade do ambiente, cuja topologia continha passagens estreitas e assim exigia que o robô navegasse em velocidade mínima (0,2 cm/iteração). Navegando em baixa velocidade, o tempo gasto para chegar até o alvo foi bem maior que nos outros casos. Entretanto, isto garantiu uma navegação mais segura e, mais uma vez, justificou a importância do mecanismo de controle de velocidade, especialmente no caso de experimentos em ambientes reais.



Colisões	Capturas	Monotonias	Gerações	Tempo	Iterações
1	12	2	15	87 min	5000

Figura 4-33: Atuação do robô em um ambiente complexo.

Capítulo 5

Considerações Finais

5.1 Síntese do Trabalho

A navegação de robôs é um problema muito oportuno à pesquisa de sistemas inteligentes e suas propriedades, sendo algumas delas a aprendizagem, a adaptação e a autonomia. Estas propriedades interdependentes, encontradas em abundância nos sistemas biológicos, motivaram sua reprodução artificial em um sistema de controle para navegação de robôs. Este sistema, chamado sistema de navegação autônomo (SNA), foi desenvolvido ao longo deste trabalho tendo como objetivo guiar, sem qualquer auxílio externo, um robô móvel por ambientes arbitrários e inicialmente desconhecidos, cumprindo simultaneamente duas tarefas: evitar colisões em obstáculos e capturar alvos dispostos arbitrariamente no ambiente de navegação.

O SNA possui capacidade de aprendizagem e adaptação que possibilitam que o robô, a partir da ausência de conhecimento inicial, adquira novos conhecimentos, proponha e coordene comportamentos possivelmente conflitantes. O atendimento dos objetivos de navegação deve ser buscado mesmo perante situações inesperadas. Estas habilidades do SNA advêm da teoria que o fundamenta e inspira: os Sistemas Classificadores (SC) de Holland, cujas propriedades de aprendizagem e adaptação são inerentes e sustentadas por mecanismos evolutivos. Diversas adequações junto aos SC originais foram incorporadas ao SNA e contemplam algumas das principais contribuições do trabalho (seção 3.4).

Além de evolutivo, o SNA é reativo. Isto é, o sistema atua com base em informações instantâneas capturadas pelos sensores e não com base em dados incorporados previamente. Já os processos evolutivos são disparados por eventos de colisão, captura e monotonia que ocorrem durante a navegação do robô. Cada um destes três eventos dispara processos evolutivos específicos responsáveis pela emergência dos comportamentos não-inatos de desvio de obstáculos e captura de alvos.

O SNA e seus diversos aspectos foram testados e validados por um extenso conjunto de experimentos de navegação (Capítulo 4). Os experimentos visaram demonstrar as virtudes do sistema e detectar deficiências. Embora uma grande diversidade de experimentos tenha sido realizada, eles podem ser agrupados em 3 classes:

- Puramente em simulação computacional;
- Mesclando simulação com navegação empregando um robô real; e
- Puramente em um robô real.

As simulações computacionais foram realizadas por meio de um simulador especificamente desenvolvido para este trabalho, o qual é apresentado no Apêndice C. O robô Khepera II, empregado nos experimentos reais e cujo modelo matemático foi usado no simulador, possui tamanho reduzido e 8 sensores infra-vermelhos que medem distância a obstáculos e luminosidade do alvo, conforme descrito no Apêndice B.

5.2 Principais Contribuições

As principais contribuições deste trabalho estão associadas às modificações feitas no SC original para que este pudesse ser adequadamente aplicado ao problema tratado e produzisse bons resultados.

Entre os aperfeiçoamentos estão: a estrutura e a codificação das regras, procedimentos específicos de avaliação e reprodução, incluindo operadores genéticos dedicados e um mecanismo reprodutivo que emprega taxa de procriação.

Outra contribuição é a síntese de um mecanismo de controle de velocidade embutido na própria estrutura do SC.

A partir da implementação do SNA com todas as contribuições já citadas, a realização de um elenco de experimentos possibilitou a validação da proposta e uma análise de desempenho.

5.3 Limitações Operacionais e Propostas Pontuais de Extensão

Uma das deficiências mais marcantes percebidas durante os experimentos foi a dificuldade do SNA em coordenar os comportamentos de desvio de obstáculos e captura de alvos quando o robô foi exposto a situações de navegação em que alvo e obstáculos estavam muito próximos. Esta dificuldade está diretamente relacionada à natureza dos sensores, visto que a fonte luminosa associada ao alvo acaba interferindo na detecção sensorial de obstáculos. Como o robô Khepera II admite a adição de módulos sensoriais mais elaborados, como câmeras digitais, a adição de procedimentos básicos de tratamento de imagens permitiria contornar a questão da interferência sensorial, à custa de uma carga de processamento computacional adicional.

Uma limitação operacional da abordagem está vinculada ao tamanho fixo da população. Variações no tamanho da população seriam bem-vindas, visto que para o atendimento dos objetivos de navegação podem ser exigidas do SNA habilidades distintas e em grau variado, em cada ambiente de navegação. Processos evolutivos que controlam automaticamente o tamanho da população podem ser incorporados, embora não de forma imediata, devido às peculiaridades de concepção do SNA. Esta também é uma forma de realizar aprendizagem latente, o que não ocorre no sistema atual.

Adaptações e extensões pontuais podem ampliar o poder de atuação de operadores genéticos. No caso de mutação, em lugar de simplesmente substituir o alelo atual por um outro qualquer, pode-se definir uma vizinhança em torno do valor atual do alelo, implementando assim um processo mais assemelhado a uma busca local. Alguns operadores de mutação poderiam admitir auto-ajuste, no sentido de sofrerem variação paramétrica em resposta a estatísticas de desempenho referentes ao grau de sucesso de sua aplicação.

Várias outras propostas para o operador de recombinação poderiam ser adotadas, particularmente para as partes antecedentes de alvo. Dentre elas, se destaca o *crossover* aritmético, muito empregado quando a codificação envolve variáveis contínuas.

5.4 Considerações Adicionais a Respeito dos Resultados Obtidos

No Capítulo 4, os resultados, em geral, mostraram que o SNA desenvolvido neste trabalho obteve sucesso em relação aos objetivos propostos. Vários aspectos corroboram esta afirmação.

Os mecanismos de taxa de procriação e controle de velocidade cumpriram com êxito seus propósitos. Os experimentos mostraram que a taxa de procriação efetivamente conseguiu manter um equilíbrio entre objetivos conflitantes, e que o mecanismo de controle de velocidade, embora simples, favoreceu bastante o desempenho do robô.

A análise de mais baixo nível, feita na seção 4.2.8, mostrou o comportamento dos diversos tipos de regras dentro da população. Ficou clara a influência da topologia do ambiente e da seqüência de eventos ocorridos na constituição das regras. Também foi possível notar a variação da intensidade da aprendizagem que, em geral, foi maior no início do processo de navegação e, à medida que os objetivos eram atendidos de forma mais eficiente, a aprendizagem ocorria de forma menos abrangente. A principal conclusão que se pode tirar a partir desses resultados, é que as regras efetivamente se adaptam e se auto-organizam diante de modificações no ambiente e na qualidade de atendimento dos objetivos.

Uma das maiores virtudes do SNA implementado é sua eficiência e consistência de aprendizagem. Na maioria absoluta dos experimentos o sistema atendeu aos objetivos de navegação. Isto comprova a precisão e o acerto no desenvolvimento dos mecanismos que compõem o sistema.

Comparando-se o desempenho do sistema obtido em simulação e em ambientes reais, foi notória a robustez do SNA. Embora não tenham sido idênticos os comportamentos do robô em ambos os casos, eles foram equivalentes e cumpriram os objetivos. As diferenças de comportamento se devem ao fato do simulador não representar tão fielmente as características dos ambientes reais. Entretanto, mesmo assim, isto só vem reforçar a robustez do sistema de navegação, que, independente de estar atuando em ambientes e

situações distintas, mostrou uma boa capacidade de adaptação. Os experimentos em que o sistema foi treinado até sua maturação, por meio de simulação, e imediatamente depois colocado para atuar em ambiente real evidenciaram a adaptabilidade do sistema. Além disso, o desempenho do SNA obtido exclusivamente nos experimentos práticos validou efetivamente o SNA em ambientes reais.

A capacidade de generalização, demonstrada pelo sistema, está diretamente ligada à sua capacidade de adaptação. Estas propriedades foram apresentadas em vários experimentos, em que o sistema, a partir de uma mesma base de conhecimento (regras), foi capaz de generalizá-la e atuar com eficiência em situações diferentes daquelas para as quais ele estava inicialmente preparado.

5.5 Perspectivas Futuras

No tocante a paradigmas de navegação, uma proposta para trabalhos futuros envolve a hibridização do SNA desenvolvido neste trabalho com a teoria de Sistemas Imunológicos Artificiais (SIA) (DE CASTRO & TIMMIS, 2002). Os SIA apresentam propriedades muito interessantes de aprendizagem, adaptação e auto-organização que, quando acopladas aos Sistemas Classificadores, têm demonstrado perspectivas muito promissoras (VARGAS *et al.*, 2003a; 2003b).

Outra idéia promissora está relacionada aos Sistemas Classificadores Antecipatórios e suas propriedades preditivas. Basicamente, trata-se da possibilidade de dispor de uma estimativa a priori do efeito que uma determinada ação pode causar no ambiente de navegação.

Como o elenco de experimentos realizados e descritos ao longo do texto não foi exaustivo, planeja-se a realização de uma variedade maior de experimentos, especialmente no tocante aos experimentos reais, visando basicamente expor o SNA a novas situações e abrindo espaço para a navegação envolvendo múltiplos robôs, com cada um deles representando um obstáculo móvel para os demais e contando com a possibilidade de interação dos robôs, mais especificamente troca de mensagens.

Apêndice A
Algoritmos Genéticos

A.1 Introdução

Os algoritmos genéticos (AG) se inspiram nos processos presentes na teoria da evolução natural das espécies, proposta por DARWIN (1859) e na genética. Eles foram introduzidos por Holland (HOLLAND, 1975) com o objetivo de formalizar matematicamente e explicar rigorosamente processos de adaptação em sistemas naturais e desenvolver sistemas artificiais (simulados em computador) que retenham os mecanismos originais encontrados em sistemas naturais.

Os AG são compostos, basicamente, por uma população de indivíduos, por uma função de avaliação e por operadores genéticos. Um indivíduo da população é representado por um único cromossomo, contendo a codificação (genótipo) de um candidato à solução do problema (fenótipo). Um cromossomo é constituído por um conjunto de elementos, conhecidos como genes, geralmente implementados na forma de um vetor, sendo que os possíveis valores que um certo gene pode assumir são denominados alelos.

Pode-se formalizar um algoritmo genético da seguinte maneira: durante a iteração *t*, um algoritmo genético mantém uma população de soluções potenciais (cromossomos, vetores), $P(t) = \{ x_1^t, ..., x_n^t \}$. Cada solução x_i^t é avaliada e produz uma medida de sua adaptação, ou *fitness*. Uma nova população (iteração $t + 1$) é então formada, privilegiando a participação dos indivíduos mais adaptados. Alguns membros da nova população passam por alterações por meio de cruzamento e mutação para formar novas soluções potenciais. Este processo se repete até que um número pré-determinado de iterações seja atingido, ou até que o nível de adaptação esperado seja alcançado (IYODA, 2000). Resumidamente, o ciclo básico de execução de um AG é mostrado na Figura A-1.

Os componentes básicos de um algoritmo genético para um problema específico são os que seguem:

- Uma representação genética para as soluções candidatas (processo de codificação);
- Uma maneira de criar uma população inicial de soluções candidatas;
- Uma função de avaliação, que classifica as soluções segundo sua adaptação ao ambiente (ou seja, sua capacidade de atingir os objetivos desejados);
- Operadores genéticos;
- Valores para os diversos parâmetros usados pelo algoritmo genético (tamanho da população, probabilidades de aplicação dos operadores genéticos, etc.).

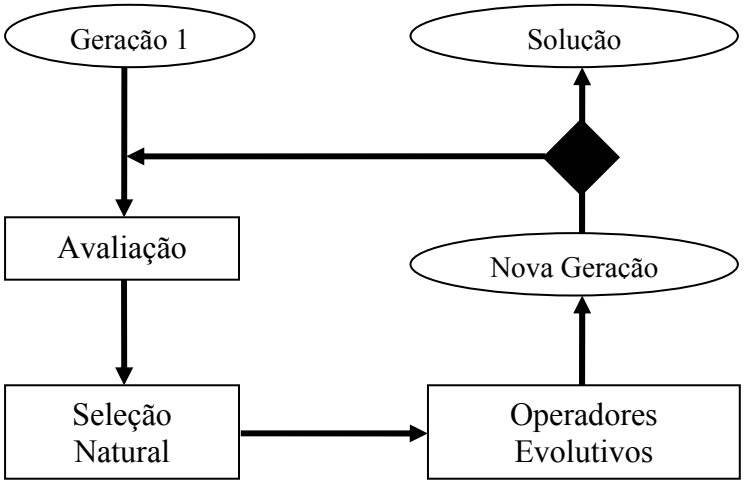


Figura A-1: Ciclo básico de um AG.

É claro que o AG utilizado em um sistema classificador sofre algumas adaptações em relação ao AG original, as quais devem ser destacadas. Primeiramente, o ciclo principal do AG é executado uma única vez a cada época de iterações do SC. Portanto, os indivíduos da população inicial do AG são exatamente os classificadores do SC, inclusive com a mesma codificação. A função de avaliação, responsável por guiar as soluções pelo espaço de busca, usa, geralmente, como critério principal a energia dos classificadores. Desta forma, os indivíduos de maior energia têm maior probabilidade de perpetuarem seus genes na próxima geração. Os operadores genéticos são executados de forma usual (HOLLAND, 1992).

Vários operadores genéticos podem ser empregados nos AG. Há alternativas diversificadas de operadores de seleção, cruzamento e mutação. Cada qual possui vantagens e desvantagens e devem ser escolhidos de acordo com as características do problema em questão.

A.2 Operadores de Seleção

O propósito da seleção é eleger os pais, ou genitores, que geram descendentes e preservam suas características genéticas para a próxima geração. Diversos esquemas de seleção já foram criados, como por exemplo, elitista, bi-classista ou por torneio. No entanto, este trabalho atém-se ao método da roleta (*roulette wheel*) que é utilizado no algoritmo genético clássico.

No método da roleta, os pais são escolhidos aleatoriamente dentro da população, por meio de um procedimento que favorece os indivíduos melhor adaptados, isto é, cada indivíduo é associado a uma probabilidade de ser selecionado, probabilidade esta representada pelo *fitness* do elemento (calculado pela função aptidão). Selecionar indivíduos segundo o valor da função aptidão implica que elementos com mais alto valor têm maior probabilidade de contribuírem com descendentes e características genéticas próprias para a geração seguinte. Em termos mais práticos, segundo o referido mecanismo, os indivíduos que receberem maior valor da função aptidão possuirão um arco maior no espaço da roleta, tendo assim maior possibilidade de serem escolhidos (Figura A-2). Um estágio de seleção seria, então, análogo a um giro dessa roleta, que selecionaria o indivíduo correspondente ao arco do círculo apontado no final do processo, ao cessar o movimento.

Indivíduos	*Fitness*	Graus	% do Total
A	6,0	180	50,0
B	3,0	90	25,0
C	1,5	45	12,5
D	1,5	45	12,5
Total	12	360	100,0

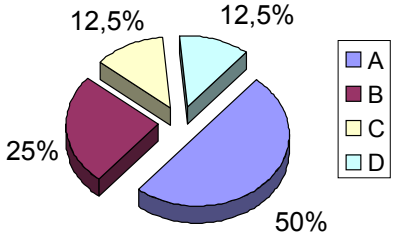


Figura A-2: Exemplo de seleção pelo método da roleta.

É importante observar que a seleção de indivíduos por esse método pode fazer com que o melhor indivíduo da população seja perdido, já que a probabilidade dele ser escolhido para compor a próxima geração não é 100%. Uma alternativa é simplesmente manter sempre o melhor indivíduo da geração atual na geração seguinte, estratégia esta conhecida como elitismo (FOGEL, 1994); (MICHALEWICZ, 1996).

São citados a seguir alguns outros possíveis mecanismos de seleção:

- Seleção por diversidade: são selecionados os indivíduos mais diversos da população;
- Seleção bi-classista: são selecionados P_1% melhores indivíduos e os $P_{2\%}$ piores indivíduos;
- Seleção aleatória: são selecionados aleatoriamente N indivíduos da população, após terem sido selecionados P% indivíduos melhores.

A.3 Operadores de Recombinação

A recombinação ou *crossover* (cruzamento) é o operador responsável pela combinação de características dos pais durante o processo de reprodução (recombinação genética), permitindo que as próximas gerações herdem essas características combinadas. A idéia intuitiva por trás do operador de cruzamento é a troca de informação entre diferentes soluções candidatas.

Algumas variações do operador de recombinação são:

- Um ponto: para a aplicação desse operador, são selecionados dois indivíduos (pais), pelo método da roleta por exemplo, e a partir de seus cromossomos são gerados dois novos elementos (filhos). Para gerar os filhos, seleciona-se um ponto de corte aleatoriamente nos cromossomos-pais, de modo que os segmentos a partir do ponto de corte sejam trocados.

O exemplo mostrado na Figura A-3 supõe que se tem os seguintes pais:

Pai 1: 010110100 Pai 2: 110010010

Se aleatoriamente o ponto "3" fosse selecionado então os filhos seriam os seguintes:

Filho 1: 010 010010 Filho 2: 110 110100

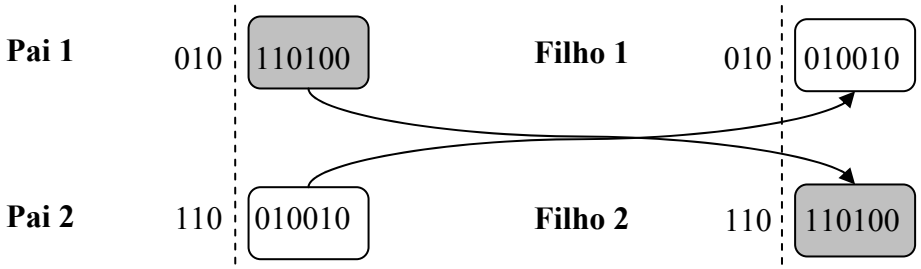


Figura A-3: Cruzamento em um ponto.

- Multiponto: é uma generalização desta idéia de troca de material genético por meio de pontos, onde muitos pontos de corte podem ser utilizados;
- Uniforme: não utiliza pontos de cruzamento, mas determina uma probabilidade de troca cada variável (gene) entre os pais.

A.4 Operadores de Mutação

O operador de mutação é necessário para a introdução e manutenção da diversidade genética da população, alterando arbitrariamente um ou mais genes de um cromossomo escolhido. Desta forma, a mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca será zero. O operador de mutação é aplicado aos indivíduos com uma probabilidade dada por uma taxa específica

A operação de mutação é relativamente simples. No caso de codificação binária, a mutação consiste em substituir com certa probabilidade (taxa de mutação) o valor de um gene. Basicamente, o gene selecionado é invertido, isto é, se o gene selecionado da i-ésima posição vale 1, passará a valer 0 (codificação binária). Em um exemplo, considerando um

indivíduo com o cromossomo 1**0**01101, uma mutação no segundo gene geraria o seguinte cromossomo modificado: 1**1**01101.

Apêndice B
Robô Khepera II®

B.1 Introdução

As pesquisas com robôs enfrentavam, há alguns anos, limitações de tecnologia que impossibilitavam os experimentos práticos para a avaliação e validação dos sistemas de controle de desenvolvidos. Estas limitações envolviam a exigência de que o pesquisador que trabalhasse na área conhecesse profundamente a eletrônica para conseguir operar os recursos disponíveis.

Na década de 90, um grupo de pesquisa suíço do *Microprocessor and Interface Laboratory* (LAMI) começou a desenvolver um modelo de robô que poderia ser operado de forma mais simples por qualquer profissional com um mínimo de experiência em computação. A intenção era que este robô servisse como padrão em pesquisas de navegação.

O projeto obteve sucesso e foi fundada a empresa *K-Team S.A.* que começou a fabricar e vender o mini-robô Khepera. A empresa vem aumentando sua variedade de robôs e, atualmente, o robô Khepera já está em sua segunda versão. O Khepera II é um mini-robô dotado de vários recursos que são suficientes para o desenvolvimento de diversos tipos de sistemas de controle para navegação. A seguir são descritas as principais especificações técnicas do robô Khepera II empregado neste trabalho.

B.2 Características

O Khepera II é um mini-robô que é mostrado na Figura B-1. Ele mede 70 mm de diâmetro e 30 mm de altura, e pesa aproximadamente 80 g. Um robô com este tamanho reduzido permite a realização de experimentos de forma mais pratica, dado que mesmo em ambientes com pouco espaço disponível o experimento é possível. O robô pode ser alimentado por cabo ou por suas baterias internas, as quais possuem autonomia de aproximadamente 1 hora.

Figura B-1: Mini-robô Khepera II.

O robô é sustentado por duas rodas, responsáveis pela sua movimentação, e por dois pinos de contato deslizantes que proporcionam equilíbrio. As rodas possuem motores elétricos independentes e, pela execução de velocidades diferentes nas rodas, se obtêm ajustes de direção (manobras). A velocidade máxima que o robô alcança é de 1 m/s, já a mínima pode chegar a 0,08 mm/s no modo de deslocamento por posição.

O robô possui, em sua estrutura básica, 8 sensores que incorporam emissores e receptores de luz infra-vermelha. Os sensores medem luminosidade do ambiente (usando somente os receptores) e distância a obstáculos. O alcance dos sensores, em relação a obstáculos, é de no máximo 10 cm. O tempo de medição de cada sensor é de 2,5 ms e, a cada 20 ms uma medição completa é feita. A saída de cada medição é um valor analógico convertido para um número de 10 bits. Alguns fatores externos, como a presença de lâmpadas incandescentes, causam interferência nas medições dos sensores. Estes e outros detalhes mais específicos a respeito dos sensores podem ser encontrados em (http://www.k-team.com/download/khepera/documentation/Kh2IRAN.pdf)

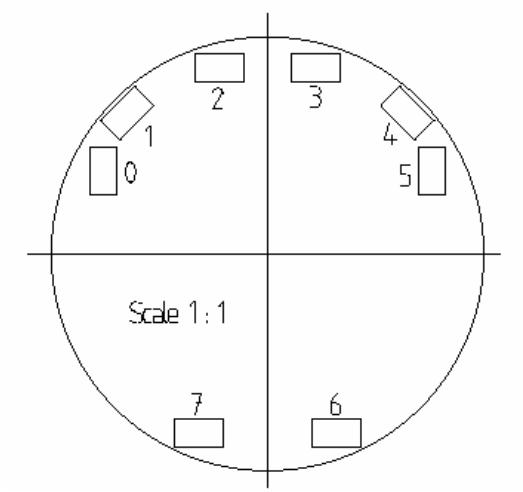


Figura B-2: Disposição dos sensores na estrutura do robô.

Quando controlado pelo sistema de navegação, as medições de distância a obstáculos dos dois sensores traseiros são ignoradas. A velocidade mínima é limitada a 0,2 cm/s e a máxima é limitada a 5 cm/s. Os incrementos ou decrementos de velocidade são feitos sempre de 0,2 cm/s.

B.3 Comunicação, Controle e Calibração

Há duas formas de se controlar o robô: com sistema embarcado ou por comunicação externa. No modo embarcado, o sistema deve ser implementado em uma linguagem específica e deve obedecer a todas as restrições do processador interno do robô. Já no caso da comunicação externa (adotada neste trabalho), o sistema de navegação é processado em um computador que é conectado ao robô por intermédio da porta serial, segundo o protocolo RS-232. Da porta serial para o robô, a conexão pode ser feita por cabo ou por rádio-transmissão. Neste trabalho optou-se pelo uso do cabo, pois assim não existe a limitação da autonomia da bateria interna do robô. A Figura B-3 ilustra o esquema de comunicação adotado.

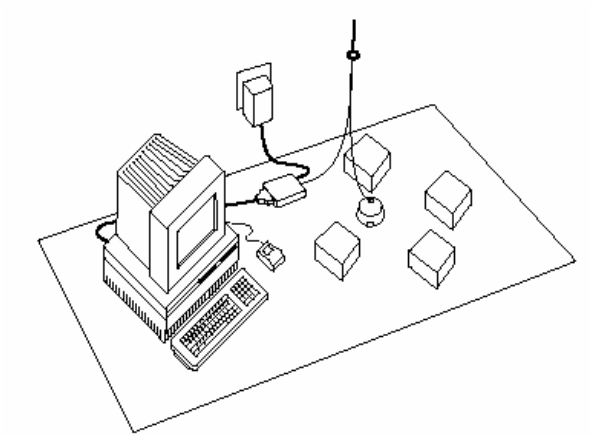


Figura B-3: Ilustração de como é feita a comunicação com o robô.

O módulo de comunicação em *software* foi inicialmente desenvolvido de forma independente do sistema de navegação. Foi implementado um controlador, mostrado na Figura B-4, em que o usuário pode comandar e utilizar praticamente todos os mecanismos do robô e nele foi feito o aperfeiçoamento do módulo de comunicação.

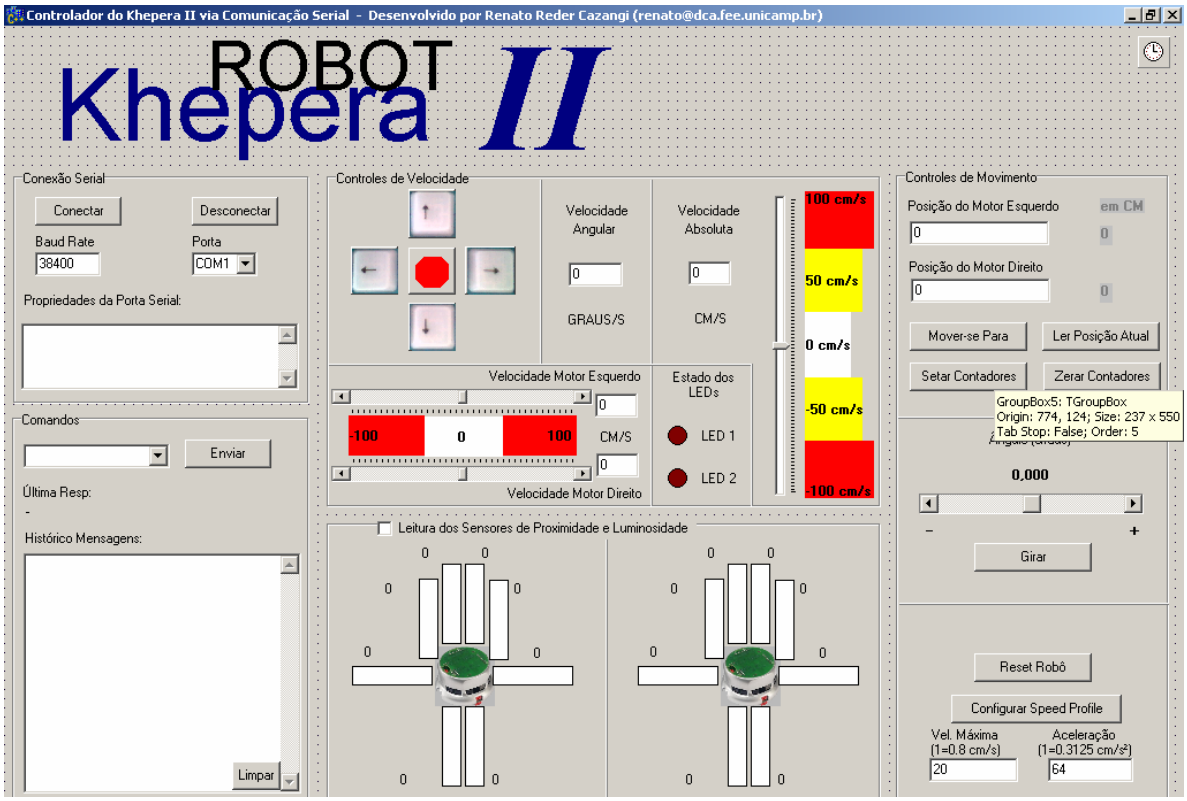


Figura B-4: Controlador em que o robô é comandado pelo usuário.

Por meio do controlador mostrado na Figura B-4, foi possível testar e compreender melhor o funcionamento do robô, além de detectar erros e dificuldades de controle. Um dos problemas foi conseguir fazer o robô girar com exatidão segundo o ângulo desejado, dado que somente é possível ajustar a velocidade ou deslocamento de cada roda. Por meio de um gabarito graduado com vários ângulos, conseguiu-se calibrar e modelar uma função de ajuste de direção que, dado um ângulo de entrada, dava como saída os deslocamentos precisos de cada roda para realizar aquele giro. Outras calibrações e modelagens foram feitas com o controlador, relacionadas à velocidade, deslocamento e aquisição sensorial.

Após a realização de todos estes testes e calibrações é que o módulo de comunicação serial foi incorporado ao simulador, permitindo que o sistema de navegação controlasse efetivamente o robô. Algumas dificuldades relativas ao tempo de resposta, imprecisão e ambigüidade sensorial ou perda de controle do robô permaneceram sem solução. Entretanto, foi possível realizar os experimentos reais de maneira satisfatória.

Apêndice C
Ambiente de Simulação

C.1 Introdução

A simulação computacional é uma etapa importante na concepção de sistemas de controle, inclusive para o problema de navegação de robôs móveis. Levando-se em conta avaliações de custo e tempo de implantação de ambientes reais de navegação, a simulação surge como uma alternativa mais rápida e econômica na obtenção e análise de alguns resultados experimentais. Entretanto, é evidente que ambientes reais, seus fenômenos e sua dinâmica sofrem simplificações e restrições quando implementados em ambientes simulados computacionalmente. Logo, nem sempre os resultados obtidos em ambientes virtuais são suficientes para validar um sistema, requerendo validações em ambientes reais.

Neste trabalho, utilizaram-se as duas modalidades de experimentos: simulação e real. Para a simulação foi desenvolvido um ambiente de simulação computacional que permite a construção de ambientes virtuais variados, a realização de processos completos de navegação e possibilita a análise do desempenho apresentado pelo sistema diante de diferentes situações. O simulador também foi preparado para assumir uma segunda função, a de controlador do robô Khepera II em ambientes reais. A seguir, são descritos detalhes técnicos de implementação e uso do simulador

C.2 Linguagem de Programação e Estrutura do Código Fonte

O simulador foi programado usando orientação a objetos (OO) e implementado no ambiente de desenvolvimento *Borland C++ Builder 6*, que emprega a linguagem de programação C++. O ambiente de desenvolvimento é visual e é voltado para a plataforma Windows. A linguagem C++ é muito utilizada e reconhecida por suas propriedades associadas ao paradigma de OO, como classes, hereditariedade, encapsulação, polimorfismo e persistência.

Justamente aproveitando o que a linguagem tem de melhor, o simulador foi organizado em diversas classes e sub-classes. Cada módulo do programa (principal, ferramentas, comunicação, *threads*, sensoriamento, navegação, atualização, aprendizagem e análise) foi colocado em uma classe. Existem ainda as classes de objetos do sistema: robô, obstáculo e alvo. Cada uma das classes contém seus próprios métodos e atributos. Além disso, todas interagem entre sim constantemente, compondo uma relação complexa entre classes e objetos.

A dinâmica do sistema é baseada em *threads* paralelas: uma para o ciclo de navegação do robô, outra para gráficos de análise de desempenho e uma terceira responsável pela comunicação com a porta serial (utilizada quando controlando o robô Khepera II).

C.3 Interface e Funcionalidades do Simulador

O simulador desenvolvido apresenta uma interface bastante amigável, cujos componentes são apresentados na Figura C-1. Existem ferramentas para a inserção e edição de objetos no ambiente (obstáculos, alvos e robôs) e para o controle da simulação (*play, pause e stop*). Todos os principais comandos do simulador são facilmente acessados por botões, menus e teclas de atalho. Diversas informações sobre o processo de navegação são mostradas e atualizadas em tempo real.

O ambiente de navegação é bidimensional e permite simular situações caracterizadas por diferentes topologias e níveis de complexidade. Obstáculos são representados por retângulos de cor preta e podem ser inseridos com o tamanho e posição desejados. Por medida de simplicidade, optou-se por restringir a geometria dos obstáculos a formas retangulares. Alvos e robôs também podem ser dispostos no ambiente em qualquer posição que não coincida com um objeto já existente. Os alvos são representados por pequenos círculos em vermelho e azul, e o robô é representado por uma figura triangular verde. O modelo do robô reproduz as especificações básicas do robô Khepera II apresentadas no

Apêndice B, como as características dos sensores e atuadores. Embora se baseie no Khepera II, é possível determinar, no momento de inserção do robô virtual, o número de sensores livremente e também a velocidade do robô, além de quantos robôs estarão presentes na simulação.

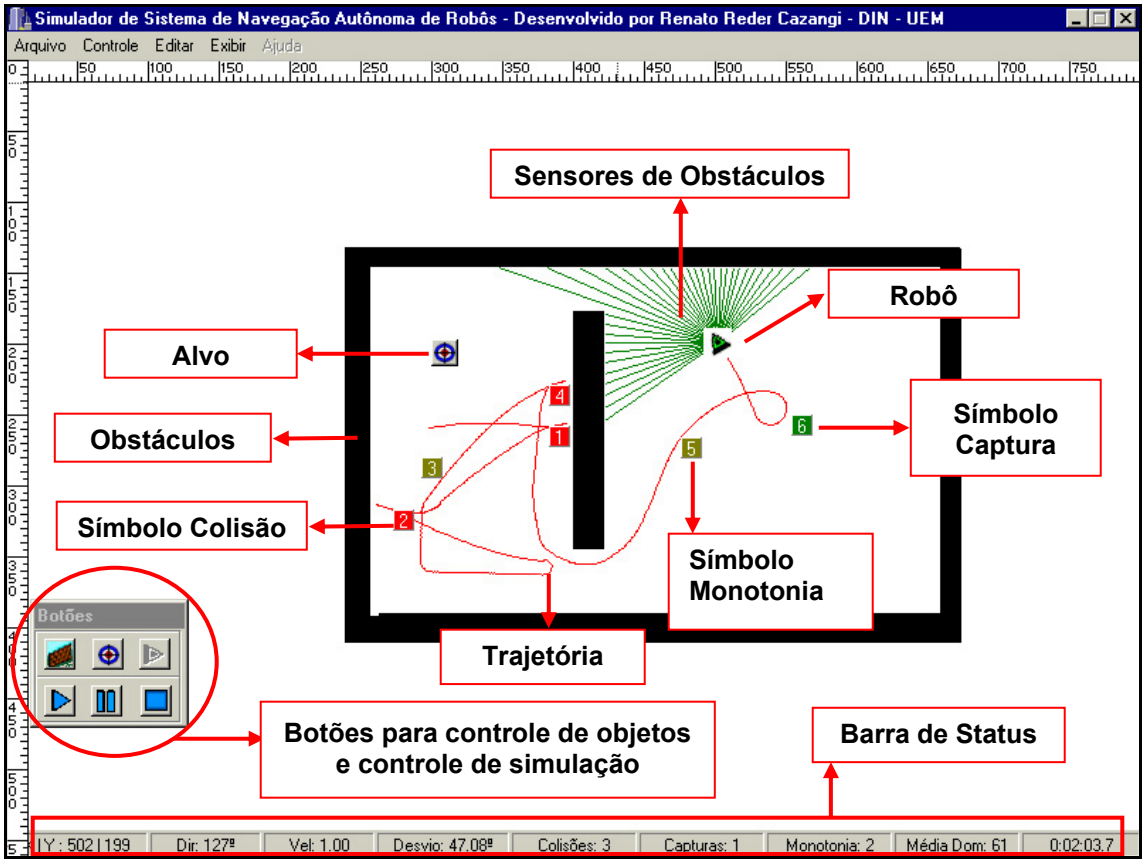


Figura C-1: Componentes do ambiente de simulação.

Existem três modos de execução do simulador possíveis: simulação normal, simulação avançada e controlar Khepera. Somente um modo pode ser ativado ao mesmo tempo. Na simulação normal o robô navega pelo ambiente virtual em tempo real e todo o processo pode ser visualizado, permitindo o acompanhamento de cada movimento do robô e seus comportamentos instantâneos. No modo de simulação avançada, a visualização da navegação do robô é desativada e o processo é executado de forma contínua e acelerada, possibilitando a realização de experimentos mais longos em menor tempo. Neste modo, somente o desempenho do sistema pode ser acompanhado em tempo real. Já no modo de

controle do Khepera, a simulação e visualização da navegação é desativada e somente a funcionalidade de controle e análise de desempenho são utilizadas. Neste caso, o módulo de comunicação com a porta serial é ativado e as entradas e saídas do sistema de navegação é passam a receber e emitir sinais para o robô real, ao invés do virtual.

Durante a simulação normal, em que é possível visualizar todos os movimentos de navegação do robô, estão disponíveis ferramentas visuais que auxiliam no acompanhamento do processo. Os sensores e suas respectivas medições podem ser mostrados ao redor do robô como traços verdes. Existem marcos numerados que identificam onde e em que ordem ocorreram os eventos de colisão (vermelho), captura (verde) e monotonia (amarelo). Há também, tracejado em vermelho, o rastro do robô que mostra sua trajetória no ambiente. Por meio desses instrumentos visuais é possível ter uma idéia bastante clara e precisa da evolução do processo de aquisição do conhecimento.

Durante as simulações, existe um mecanismo automático de inserção de alvos para automatizá-las. No ambiente são definidas regiões específicas, dentro das quais os alvos são inseridos em posição aleatória logo que o alvo anterior tenha sido capturado pelo robô. A motivação para a utilização de regiões específicas para inserção de alvos está associada à necessidade de se dispor os alvos em certas posições do ambiente de forma automática e que exija determinados comportamentos do robô como, por exemplo, capturar alvos separados por um obstáculo entre eles. A Figura C-2 mostra as regiões automática de inserção de alvos rotuladas como A e B em azul.

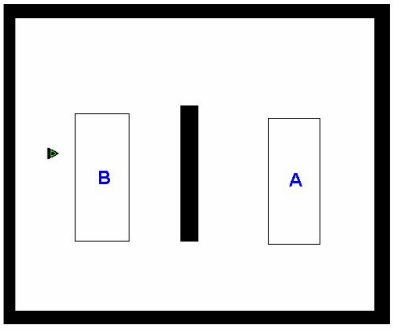


Figura C-2: Ambiente onde A e B são regiões de inserção automática de alvos. Quando o robô captura um alvo em A, automaticamente outro aparecem em uma posição aleatória dentro de B e assim por diante.

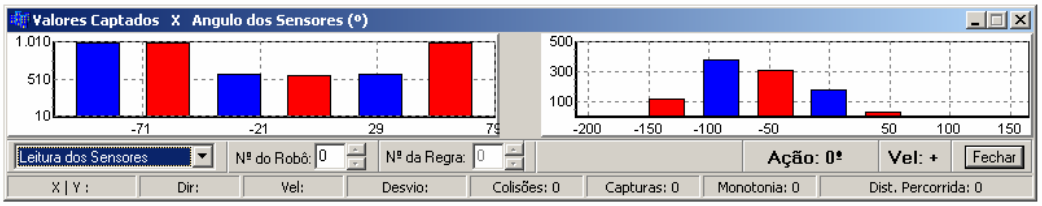
O simulador também possui funcionalidades de armazenamento de dados. Os ambientes de navegação virtuais e o conhecimento do sistema de navegação (base de regras) podem ser salvos em arquivos ".dat" em qualquer momento. Estes mesmos arquivos também ficam disponíveis para serem carregados no simulador quando desejado. Além disso, a navegação completa pode ser gravada em arquivo para ser assistir *a posteriori*

C.4 Ferramentas de Análise de Desempenho

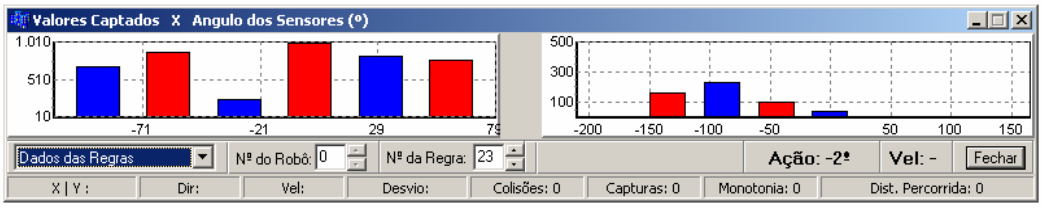
Além das funcionalidades mencionadas, relativas à construção do ambiente, ainda há mecanismos de análise do desempenho do sistema durante a navegação, os quais podem ser acompanhados em tempo real. Tais ferramentas referem-se à exibição de gráficos e dados que proporcionam uma interpretação visual e estatística de diversas informações sobre o sistema. Este módulo do simulador é rico em detalhes e fundamental para a avaliação e acompanhamento do desempenho e comportamento do robô e do sistema de navegação. A Figura C-3 mostra os diversos gráficos relacionados à navegação do robô. Os gráficos são coloridos, rotulados, interativos e atualizados em tempo real.

O gráfico (a) da Figura C-3 mostra a leitura instantânea das medições dos sensores do robô. Já o gráfico (b) permite visualizar o conteúdo de cada regra que compõem a população do sistema de navegação, simplesmente selecionado o número da regra. Também é possível exibir somente o conteúdo da regra atuante naquele instante. O gráfico (c) é o mais importante e apresenta o desempenho do sistema em termos de ocorrências acumuladas de eventos de colisão, captura e monotonia em relação ao número de gerações produzidas. O último gráfico, (d), mostra, em média, a variação dos tipos de cada parte das regras que compõem a população durante a navegação. Se houver mais de um robô navegando simultaneamente, todos os gráficos podem ser visualizados especificamente para cada robô. Todos estes gráficos podem ser melhor compreendidos no Capítulo 4, em que os resultados e gráficos obtidos são analisados detalhadamente.

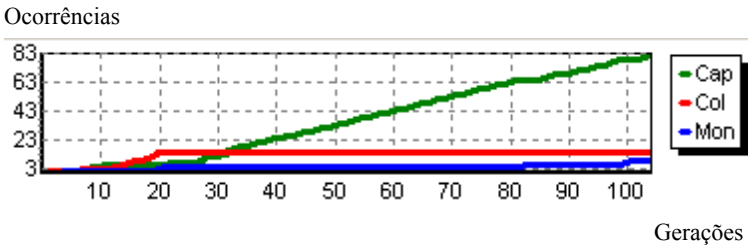
(a)



(b)



(c)



(d)

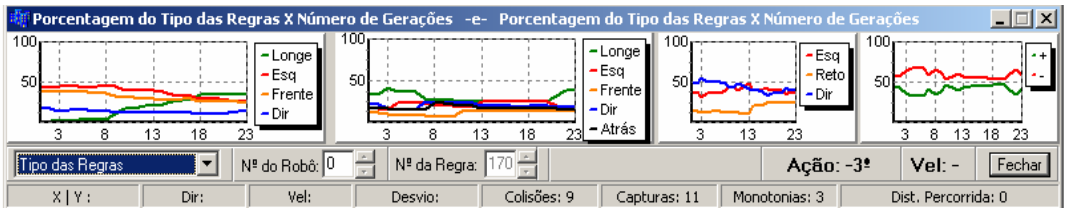


Figura C-3: O gráfico (a) é da leitura dos sensores, (b) do conteúdo das regras, (c) do desempenho do sistema e (d) da variação média do tipo das regras da população.

C.5 Descrição dos Comandos e Opções do Programa-Simulador

C.5.1 Menu Arquivo (Figura C-4)

Figura C-4: Comandos do menu arquivo.

- Através deste menu é possível **salvar** e **abrir**:
- **ambientes**: qualquer ambiente construído no simulador contendo alvos, obstáculos e robô pode ser salvo em um arquivo e depois recarregado no simulador (extensão .AMB);
- **conhecimentos do robô**: todos os conhecimentos adquiridos pelo sistema de navegação durante uma simulação, ou seja, as regras que representam o conhecimento, podem ser salvas em arquivos e depois recarregadas em outra simulação (extensão .CON);
- **simulações:** toda a simulação realizada, ou seja, a navegação do robô pelo ambiente, sua trajetória, etc. pode ser salva em arquivos e depois recarregada, permitindo ao usuário assisti-la novamente. (extensão .SIM).
- Pode-se **reiniciar o ambiente** atual, ou seja, apagar tudo e começar uma nova simulação.
- Também é possível **sair** do simulador.

C.5.2 Menu Controle e Caixa de Ferramentas (Figura C-5)

Figura C-5: Comandos do menu controle e caixa de ferramentas.

Ambos permitem controlar o andamento da simulação, os comandos estão a seguir.

- **Iniciar simulação (Play):** é necessário haver pelo menos um robô e um alvo no ambiente.
- **Pausar simulação (Pause):** paralisa a simulação temporariamente.
- **Parar simulação (Stop):** encerra a simulação.

- **Inserir obstáculo**: faz a inserção de um obstáculo de tamanho e posição desejados através do mouse. Não pode coincidir com posição de alvo ou robô. Pode ser feito tanto antes como durante a simulação normal.
- **Inserir alvo**: faz a inserção manual de um alvo na posição desejada através do mouse. Não pode coincidir com obstáculos. Pode ser feito tanto antes como durante a simulação normal.
- **Inserir robô**: insere o robô na posição indicada através do mouse. Deve haver pelo menos um alvo ou então a inserção de alvos deve ser estabelecida como automática (menu editar, opções). Somente pode ser feito antes como do início da simulação.

C.5.3 Menu Modo de Execução (Figura C-6)

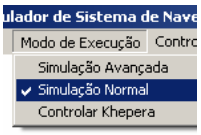


Figura C-6: Modos de execução.

- **Simulação avançada**: esse modo de simulação destina-se a simulações mais longas em que não se está preocupado em visualizar os movimentos do robô, mas sim apenas no desempenho do sistema após um elevado número de iterações. Durante uma simulação avançada não é possível acompanhar a navegação do robô. Entretanto, é possível visualizar os gráficos de desempenho do robô. Assim, esse modo propicia simulações rápidas e longas. Os obstáculos, alvos (ou regiões automáticas) e robôs devem ser dispostos antes do início da simulação.
- **Simulação normal**: se estiver assinalado significa que a simulação será realizada de modo comum, ou seja, com visualização gráfica da navegação do robô iteração a iteração com possibilidade de acompanhamento de todos os dados e gráficos. Nesse modo, a simulação é bem mais lenta que no modo avançado. Há possibilidade de uso da inserção automática de alvos (menu editar, opções).
- **Controlar Khepera**: neste modo o simulador não realiza simulação, mas sim passa a controlar o próprio robô Khepera II, recebendo dele informações sensoriais e

enviando ações a serem executadas. A comunicação é feita por meio da porta serial, onde deve estar conectado o rádio-transmissor ou o próprio cabo ligado ao robô. As configurações de *software* relativas à comunicação serial podem ser feitas no menu editar, opções. Todos os gráficos estão disponíveis neste modo.

C.5.4 Menu Editar (Figura C-7)

Figura C-7: Comandos do menu editar.

- **Ambiente:** até a versão atual do simulador só é possível editar a posição dos obstáculos. Para isso é necessário dar um duplo clique em cima do obstáculo desejado que ficará de uma cor diferente. Então é só arrastá-lo até a nova posição escolhida. Por fim, é preciso dar um novo duplo clique sobre o obstáculo para fixá-lo.
- **Opções**: ao selecionar abrirá uma janela com diversas configurações, mostradas na Figura A-8, que podem ser alteradas.
-

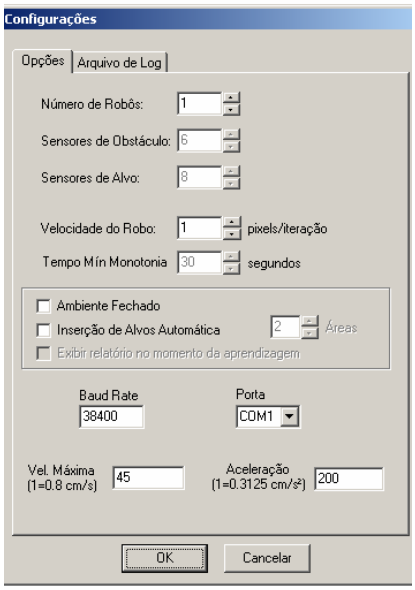


Figura C-8: Configurações possíveis.

- **Número de robô**: define o número de robôs que navegarão simultaneamente no mesmo ambiente;
- **Número de sensores**: define a quantidade de sensores de obstáculo e alvo que estarão dispostos ao redor do robô uniformemente. ;
- **Velocidade do robô**: estabelece a velocidade do robô em píxels/iteração no ambiente para simulação, ou seja, quantas unidades o veículo andará por iteração. Este valor é apenas inicial já que o sistema controla a velocidade do robô.
- **Tempo mínimo de monotonia**: determina o menor tempo possível para que ocorra um evento de monotonia.
- **Ambiente fechado**: se selecionado, coloca no ambiente obstáculos em todos os 4 lados.
- **Inserção de alvos automática**: é um mecanismo que permite ao usuário definir áreas retangulares de concentração de alvos para que o simulador coloque novos alvos de maneira automática e aleatória (dentro das áreas). Um novo alvo é inserido em uma área diferente toda vez que o último alvo é capturado. Isso evita que os alvos do ambiente acabem e o robô fique sem ter o que capturar. Na simulação avançada (menu modo de simulação) esse mecanismo é essencial.
- As demais configurações são relacionadas à comunicação com a porta serial.

C.5.5 Menu Exibir (Figura C-9)

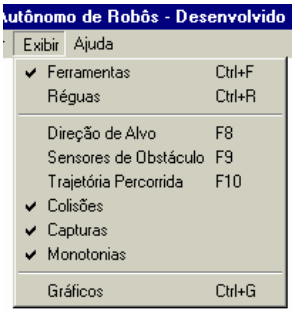


Figura C-9: Comandos do menu exibir.

- **Ferramentas**: exibe ou não a caixa de ferramentas.
- **Réguas**: exibe ou não as réguas.

- **Direção de alvo**: exibe ou não um traço azul do robô em direção ao alvo ativo.
- **Sensores de obstáculos**: exibe ou não uma representação gráfica da direção e distância captada pelos sensores.
- **Trajetória percorrida**: exibe ou não a trajetória já percorrida pelo robô até o momento.
- **Colisões**: exibe ou não símbolos vermelhos nos pontos que ocorreram colisões.
- **Capturas**: exibe ou não símbolos verdes nos pontos que ocorreram capturas.
- **Monotonias**: exibe ou não símbolos amarelos nos pontos que ocorreram monotonias.
- **Gráficos:** exibe uma janela com diversos gráficos de informações referentes ao robô e sistema de navegação, em que se pode escolher qual gráfico exibir, para qual robô e, se for o caso, qual regra.

Referências Bibliográficas

ARKIN, R.C., (1993), "Modeling Neural Function at the Schema Level: Implications and Results for Robotic Control", chapter in Biological Neural Networks in Invertebrate Neuroethology and Robotics, ed. R. Beer, R. Ritzmann, and T. McKenna, Academic Press, pp. 383-410.

ARROWSMITH, D. K. & PLACE, C. M., (1990), "An Introduction to Dynamical Systems", Cambridge University Press.

BAKER, G. L. & GOLLUP, J. P., (1990), "Chaotic Dynamics - an Introduction", Cambridge University Press.

BARTLETT, F.C., (1932), "A Study in Experimental and Social Psychology", Cambridge University Press.

BONABEAU, E., DORIGO, M. & THERAULAZ, G., (1999), "Swarm Intelligence: From Natural to Artificial Systems", Oxford University Press.

BOOKER, L. B., GOLDBERG, D. E. & HOLLAND, J. H., (1989), "Classifier Systems and Genetic Algorithms", Artificial Intelligence, vol. 40, pp. 235-282.

BROOKS, R., (1991), "Intelligence without Reason", In Proceedings 12th International Joint Conference on AI, pp. 569-595, Sydney, Australia.

BUTZ, M., GOLDBERG, D. E. & STOLZMANN, W., (1999), "New challenges for an ACS: Hard problems and possible solutions",. Technical Report 99019, University of Illinois at Urbana-Champaign, Urbana, IL, EUA.

BUTZ, M. V. & WILSON, S. W., (2001), "An Algorithmic Description of XCS", Technical Report 2000, University of Illinois at Urbana-Champaign, Urbana, IL, EUA.

CAZANGI, R. R., (2002), "Sistema Autônomo Inteligente Baseado em Computação Evolutiva Aplicado à Navegação de Robôs Móveis", Trabalho de Graduação, Departamento de Informática, UEM, Maringá, PR.

CAZANGI, R. R. & FIGUEIREDO, M. F., (2002), "Simultaneous Emergence of Conflicting Basic Behaviors and Their Coordination in an Evolutionary Autonomous Navigation System", In Proceedings of the World Congress of Computational Intelligence, Conference on Evolutionary Computation, Honolulu, EUA, pp. 466 – 471.

CAZANGI, R. R., VON ZUBEN, F.J., FIGUEIREDO, M. F., (2003a) "Sistema Autônomo Evolutivo em Aplicações Embarcadas para Navegação de Robôs", Anais do VI Simpósio Brasileiro de Automação Inteligente, Bauru, pp. 704 – 709.

CAZANGI, R. R., VON ZUBEN, F.J., FIGUEIREDO, M. F., (2003b), "A Classifier System in Real Applications for Robot Navigation", In Proceedings of the Conference on Evolutionary Computation, Canberra, Australia, vol. 1, pp. 574 – 580.

CLANCEY, W. J., (1997), "Situated Cognition: On Human Knowledge and Computer Representations", Cambridge University Press.

CRESTANI, P.R., (2001), "Sistemas Inteligentes de Navegação Autônoma: Uma Abordagem Modular e Hierárquica Com Novos Mecanismos de Memória e Aprendizagem", Tese de Mestrado, Faculdade de Engenharia Elétrica e de Computação, Unicamp, Campinas, SP.

DAWKINS, R., (1998), "A escalada do monte improvável – uma defesa da teoria da evolução", título original: "Climbing mount Improbable", tradução de Suzana Sturlini Colto, Companhia das Letras, São Paulo.

DE CASTRO, L. N. & TIMMIS, J. I., (2002), "Artificial Immune Systems: A New Computational Intelligence Approach", Springer-Verlag, London.

DE CASTRO, L. N. & VON ZUBEN, F. J. (eds.), (2004), "Recent Developments in Biologically Inspired Computing", Idea Group Publishing.

DEJONG, K.A., (1988), "Using Genetic Algorithms to Learn Task Programs: the Pitt Approach", Machine Learning.

FIGUEIREDO, M., (1997), "Redes neurais nebulosas aplicadas em problemas de modelagem e controle autônomo", Tese de Doutorado, Faculdade de Engenharia Elétrica e de Computação, Unicamp, Campinas, SP.

FIGUEIREDO, M., (1999), "Navegação Autônoma de Robôs", VII Escola de Informática da SBC - Regional Sul, pp. 74 – 106.

FOGEL, D.B., (1994), "An introduction to simulated evolutionary optimization", IEEE Transactions on Neural Networks, vol. 5, no. 1, pp. 3 – 14.

FRANKLIN, S., (1995), "Artificial Minds", MIT Press, Cambridge, MA.

FURUHASHI, T., NAKAOKA, K. & UCHIKAWA, Y., (1994), "A Study on Fuzzy Classifier System for Finding Control Knowledge of Multi-Input Systems", Genetic Algorithms And Soft Computing, pp. 489-502.

GÉRARD, P. & SIGAUD, O., (2001), "YACS: Combining dynamic programming with generalization in classifier systems", In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Advances in learning classifier systems, IWLCS 2000, Springer-Verlag, Berlin Heidelberg pp. 52-69.

GÉRARD, P. & SIGAUD, O., (2003), "Designing Efficient Exploration with MACS: Modules and Function Approximation, In Proceedings of GECCO'03, © Springer Verlag, pp. 1882-1893.

GOLDBERG, D. E., (1989), "Genetic algorithms in search, optimization, and machine learning", Addison-Wesley Reading, MA.

GREFENSTETTE, J.J., (1989), "A system for learning control strategies with genetic algorithms", In Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 183-190.

HAYDU, N. (2003), Uma Abordagem Baseada em Seleção pelas Conseqüências para Aprendizagem de Redes Neurais Multi-camadas Voltadas à Concepção de Sistemas Autônomos Inteligentes, Tese de Mestrado, UFPR, Curitiba, PR.

HENDRIKS-JANSEN, H., (1996), "Catching Ourselves in the Act: Situated Activity", Interactive Emergence, Evolution and Human Thought. The MIT Press, Cambridge, MA.

HOLLAND, J. H., (1975), "Adaptation in natural and artificial systems", The University of Michigan Press, 1st. ed.

HOLLAND J. H., (1986), "Escaping brittleness: the possibility of general-purpose learning algorithms applied to rule-based systems", In Michalski, R. S., Carbonell, J.G, Mitchell, T. M., Machine Learning: An Artificial Intelligence Approach, Morgam Kaufmann, San Mateo, CA, Vol. 2, pp. 593 – 623.

HOLLAND, J. H., (1992), "Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology Control, and Artificial Intelligence", The MIT Press, Ann Arbor, MI.

HOLLAND, J. H., (1995), "Hidden Order", Addison-Wesley Reading, MA.

HOLLEY, J., (2004), "First Investigations of Dream-like Cognitive Processing using the Anticipatory Classifier System", Technical Report UWELCSG04-002.

IYODA, E. M., (2000), "Inteligência Computacional no Projeto Automático de Redes Neurais Híbridas e Redes Neurofuzzy", Tese de Mestrado, Faculdade de Engenharia Elétrica e de Computação, Unicamp, Campinas, SP.

JACKSON, E.A., (1991), "Perspectives of nonlinear dynamics", Cambridge University Press.

KOVACS, T., (2000), "A Learning Classifier Systems Bibliography", http://www.cs.bris.ac.uk/~kovacs/lcs/search.html.

KOVACS, T. & LANZI, P.L., (1999), "A Learning Classifier Systems Bibliography", Technical Report: CSRP-99-19, University of Birmingham, United Kingdom.

KUO, B. C., (1991), "Automatic Control Systems", Prentice Hall.

LANZI, P. L., STOLZMANN, W. & WILSON, S. W., ed., (2000), "Learning Classifier Systems: From Foundations to Applications", SpringerVerlag, Berlin, vol. 1813 of LNAI.

LANZI, P. L. & WILSON, S. W., (2000), "Toward Optimal Classifier System Performance in Non-Markov Environments", Evolutionary Computation 8, vol. 4, pp. 393-418.

MCFARLAND, D. J. & BOSSER, T, (1993), "Intelligent Behaviours in Animals and Robots", Cambridge, MA, MIT Press.

MICHALEWICZ, Z., (1996), "Genetic Algorithms + Data Structures = Evolution Programs", 3rd edition, Springer.

MICHELAN, R., (2003), "Evolução de Redes Imunológicas para Coordenação Automática de Comportamentos Elementares em Navegação Autônoma de Robôs", Tese de Mestrado, Faculdade de Engenharia Elétrica e de Computação, Unicamp, Campinas, SP.

MINSKY, M.L., (1963), "Steps Toward Artificial Intelligence, Computer and Thought", Feigenbaum, E.A. and Feldman, J. (eds.), McGraw-Hill, New York, NY.

NOLFI , S. & FLOREANO, D. (2000), Evolutionary Robotics, MIT Press.

PFEIFER, R. & SHEIER, C., (1999), "Understanding Intelligence", MIT Press.

RICHARDS, R.A., (1995), "Zeroth-Order Shape Optimization Utilizing Learning Classifier Systems", Ph.D. Thesis, Stanford University.

SHU, L. & SCHAEFFER, J., (1989), "VCS: Variable Classifier System", In ICGA89, pp. 334-339.

SPIESSENS, P., (1990), "PCS: A Classifier System that Builds an Internal World Model". In Proceedings of the Ninths European Conference on Artificial Intelligence, London: Pitman, pp. 622 – 627.

STEELS, L., (1995), "When are robots intelligent autonomous agents?", Journal of Robotics, and Autonomous Systems, vol. 15, pp. 3 - 9.

STOLZMANN, W, (1996), "Learning Classifier Systems using the Cognitive Mechanism of Anticipatory Behavioural Control, detailed version", Proceedings of the First European Workshop on Cognitive Modeling, Berlin, pp. 82-89

STOLZMANN, W., (1997), "Antizipative Classifier System"s. PhD Dissertation. Fachbereich Mathematik /Informatik. Universität Osnabrück. Aachen: Shaker, Verlag. (http://www.shaker.de)

TOMLINSON, A. & BULL, L., (1999), "On Corporate Classifier Systems: Increasing the Benefits of Rule Linkage", GECCO99, pp. 649-656.

VARGAS, P. A., DE CASTRO, L.N. & VON ZUBEN, F. J., (2002), "Artificial Immune Systems as Complex Adaptive Systems", In Proceedings of the First International Conference on Artificial Immune Systems, ICARIS-2002, pp. 115-123.

VARGAS, P.A., DE CASTRO, L.N., MICHELAN, R., VON ZUBEN, F.J., (2003a), "An Immune Learning Classifier Network for Autonomous Navigation", In Timmis, J., Bentley, P. and Hart, E. (eds.) Proceedings of the Second International Conference on Artificial Immune Systems (ICARIS'2003), Lecture Notes in Computer Science, Edinburgh, UK, pp. 69-80.

VARGAS, P.A., DE CASTRO, L.N., MICHELAN, R., VON ZUBEN, F.J. (2003b), "Implementation of an Immuno-Genetic Network on a Real Khepera II Robot", 2003 Congress on Evolutionary Computation (CEC'2003), Canberra, Australia, vol. 1, pp. 420-426.

WATANABE, Y., ISHIGURO, A. & UCHIKAWA, H., (1999), "Decentralized Behaviour Arbitration Mechanism for Autonomous Mobile Robot Using Immune Network", In D. Dasgupta (Editor), Artificial Immune Systems and their Applications, Springer.

WILCOX, J. R., (1995), "Organizational Learning within a Learning Classifier System", University of Illinois, Technical Report No. 95003 IlliGAL.

WILSON, S. W., (1994), "ZCS: A zeroth level classifier system", Evolutionary Computation, 1, vol. 2, pp.1-18.

WILSON, S. W., (1995),"Classifier Fitness Based on Accuracy", Evolutionary Computation, 2, vol. 3, pp.149-175.

WILSON, M., KING, C. & HUNT, J., (1997), "Evolving hierarchical robot behaviours", Robotics and Autonomous Systems, v. 22, no 3-4, pp. 215-230.

WINOGRAD, T. & FLORES, F, (1986), "Understanding Computers and Cognition: A New Foundation for Design", Ablex Publishing.

ZHOU, H. H., (1985), "Classifier systems with long term memory", ICGA85, pp. 178-182.

Índice Remissivo

ARKIN, 1993 5

ARROWSMITH & PLACE, 1990 5

BAKER & GOLLUP, 1990 5

BARTLETT, 1932 5

BONABEAU *et al.*, 1999 56

BOOKER *et al*., 1989 ..... 9, 13, 14, 20, 24, 28

BROOKS, 1991 6

BUTZ *et al.*, 1999 30

BUTZ & WILSON, 2001 29

CAZANGI, 2002 11

CAZANGI & FIGUEIREDO, 2002 31

CAZANGI *et al.*, 2003a 31

CAZANGI *et al.*, 2003b 31

CLANCEY, 1997 6

CRESTANI, 2001 ..... 2, 7

DAWKINS, 1998 8

DE CASTRO & TIMMIS, 2002 101

DE CASTRO & VON ZUBEN, 2004 5

DEJONG, 1988 16

FIGUEIREDO, 1997 7

FIGUEIREDO, 1999 ..... 3, 4

FOGEL, 1994 106

FRANKLIN, 1995 6

FURUHASHI *et al.*, 1994 27

GERARD & SIGAUD, 2001 27

GÉRARD & SIGAUD, 2003 27

GOLDBERG, 1989 27

GREFENSTETTE, 1989 8

HAYDU, 2003 7

HENDRIKS-JANSEN, 1996 6

HOLLAND, 1975 ..... 9, 14, 103

HOLLAND, 1986 ..... 14, 27

HOLLAND, 1992 ..... 27, 104

HOLLAND, 1995 ..... 9, 13

HOLLEY, 2004 30

IYODA, 2000 103

JACKSON, 1991 5

KOVACS, 2000 27

KOVACS & LANZI, 1999 27

KUO, 1991 4

LANZI *et al.*, 2000 9

LANZI & WILSON, 2000 29

MCFARLAND & BOSSER, 1993 5

MICHALEWICZ, 1996 ..... 16, 18, 63, 106

MICHELAN, 2003 8

MINSKY, 1963 20

NOLFI & FLOREANO, 2000 ..... 4, 8, 10

PFEIFER & SHEIER, 1999 ..... 2, 3, 5, 6, 9, 10

RICHARDS, 1995 ..... 14, 21, 23

SHU & SCHAEFFER, 1989 27

SPIESSENS, 1990 27

STEELS, 1995 ..... 1, 6

STOLZMANN, 1996 27

STOLZMANN, 1997 30

TOMLINSON & BULL, 1999 27

VARGAS *et al.*, 2002 13

VARGAS *et al.*, 2003a 101

VARGAS *et al.*, 2003b 101

WATANABE *et al.*, 1999 8

WILCOX, 1995 27

WILSON, 1994 27

WILSON, 1995 ..... 27, 29

WILSON *et al.*, 1997 8

WINOGRAD & FLORES, 1986 6

ZHOU, 1985 27