Eduardo Sacogne Fraccaroli

Análise de Desempenho de Algoritmos Evolutivos no Domínio do Futebol de Robôs

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Ciências, Programa de Pós-graduação em Engenharia Elétrica (Área de Concentração em Sistemas Dinâmicos)

Orientador: Prof. Dr. Ivan Nunes da Silva

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTES
TRABALHOS, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO,
PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

F797a Fraccaroli, Eduardo Sacogne
Análise de desempenho de algoritmos evolutivos no
domínio do futebol de robôs / Eduardo Sacogne
Fraccaroli ; orientador Ivan Nunes da Silva. -- São
Carlos, 2010.

Dissertação (Mestrado-Programa de Pós-Graduação em
Engenharia Elétrica e Área de Concentração em Sistemas
Dinâmicos) -- Escola de Engenharia de São Carlos da
Universidade de São Paulo, 2010.

1. Algoritmos genéticos. 2. Sistemas multiagentes.
3. Robôs móveis inteligentes. I. Título.

FOLHA DE JULGAMENTO

Candidato: Bacharel **EDUARDO SACOGNE FRACCAROLI**.

Dissertação defendida e julgada em 01/09/2010 perante a Comissão Julgadora:

Prof. Associado **IVAN NUNES DA SILVA – (Orientador)**
(Escola de Engenharia de São Carlos/USP)

APROVADA

Profª. Associada **ROSELI APARECIDA FRANCELIN ROMERO**
(Instituto de Ciências Matemáticas e de Computação/USP)

APROVADA

Prof. Dr. **HUMBERTO FERASOLI FILHO**
(Universidade Estadual Paulista “Julio de Mesquita Filho”/UNESP/Campus de Bauru)

APROVADA

Prof. Titular **GERALDO ROBERTO MARTINS DA COSTA**
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica e
Presidente da Comissão de Pós-Graduação

Dedicatória

À Deus, meus pais, meus irmãos e a todos aqueles que acreditaram em mim.

Agradecimentos

À Deus, por conceder força, criatividade, coragem e por cada momento vivido durante a realização deste trabalho.

À minha Família, Brasil, Marisa, Catia e Rafael, por todo apoio oferecido e paciência durante a realização deste trabalho, em especial meu pai, pelo incentivo e pelas correções deste texto.

À Selene, pela paciência, dedicação, coragem, carinho e pela ajuda na realização deste trabalho.

Ao Prof. Dr. Ivan Nunes da Silva, por toda orientação, motivação, apoio, paciência, dedicação, por acreditar e investir no trabalho de seus alunos.

Aos meus amigos do LAIPS: Ricardo, Suetake, Rodrigo, Daniela, Luciano, Spatti, Fernanda e Renato.

Aos meus amigos do GEAR: Ivan, Lang, Lucas, Patrícia, Gustavo, Marcelo, Jonas e Gabriel.

Ao meu amigo do USPDroids Marcelo, pelo tempo concedido às divagações.

Aos diversos professores e funcionários da EESC/SEL pelos serviços prestados.

Às funcionárias da secretaria de pós-graduação do SEL, em especial às secretárias Marisa e Jussara, por estarem sempre dispostas a nos dar atenção.

E, por fim, à CAPES pelo apoio financeiro concedido.

Resumo

Fraccaroli, E. S. Análise de Desempenho de Algoritmos Evolutivos no Domínio do Futebol de Robôs. 2010. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Departamento de Engenharia Elétrica, Universidade de São Paulo, São Carlos, 2010.

Muitos problemas de otimização em ambientes multiagentes utilizam os algoritmos evolutivos para encontrar as melhores soluções. Uma das abordagens mais utilizadas consiste na aplicação de um algoritmo genético, como alternativa aos métodos tradicionais, para definir as ações dos jogadores em um time de futebol de robôs. Entretanto, conforme relatado na literatura, há inúmeras possibilidades e formas de se aplicar um algoritmo genético no domínio do futebol de robôs. Assim sendo, neste trabalho buscou-se realizar uma análise comparativa dos algoritmos genéticos mono-objetivo e multi-objetivo aplicados no domínio do futebol de robôs. O problema padrão escolhido para realizar essa análise foi de desenvolver uma estratégia de controle autônomo, a fim de capacitar que os robôs tomem decisões sem interferência externa, pois, além de sua solução se encontrar ainda em aberto, o mesmo é também de suma relevância para a área de robótica.

Palavras-Chave: Algoritmos Genéticos, Sistemas Multiagentes, *RoboCup Soccer Simulation 2D*.

Abstract

Fraccaroli, E. S. Performance Analysis of Evolutionary Algorithms in the Robot Soccer Domain. 2010. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Departamento de Engenharia Elétrica, Universidade de São Paulo, São Carlos, 2010.

Many optimization problems in multiagent environments adapt evolutionary algorithms to find the best solutions. A widely used approach consists of applying a genetic algorithm as an alternative to traditional methods, in order to define the actions of the players on a soccer team of simulated robots. However, as reported in the literature, there are many possibilities and ways to apply a genetic algorithm in the field of robot soccer. Therefore, this work attempts to make a comparative analysis of mono-objective and multi-objective genetic algorithms applied to control a robot soccer. The standard problem chosen for this analysis was to develop a strategy for autonomous control, in order to enable the robots to make decisions without external interference, because in addition to its solution is still open, it is also of utmost relevance to the area robotics.

Keywords: Genetic Algorithm, Multiagent System, RoboCup Soccer Simulation 2D.

Lista de Figuras

1.1	Sequência de bits utilizados como estratégia do time (Nakashima et al., 2004).	3
1.2	Área utilizada para o exercício de treinamento dos jogadores (Jeong e Lee, 1997).	6
2.1	Robôs da categoria <i>Humanoid</i> da <i>RoboCup</i> (RoboCup, 2009).	10
2.2	Robôs da categoria <i>Middle Size</i> da <i>RoboCup</i> (RoboCup, 2009).	11
2.3	Robôs da categoria <i>Small Size</i> da <i>RoboCup</i> (RoboCup, 2009).	11
2.4	Robôs NAO da categoria <i>Standard Platform</i> da <i>RoboCup</i> (RoboCup, 2009).	12
2.5	Robôs AIBO da categoria <i>Standard Platform</i> da <i>RoboCup</i> (RoboCup, 2009).	12
2.6	Categoria <i>Simulation 2D</i> da <i>RoboCup</i> (RoboCup, 2009).	13
2.7	Ilustração do monitor.	14
2.8	Arquitetura do simulador 2D (Reis, 2003).	15
2.9	Arquitetura do time <i>UvA Trilearn</i> (Boer e Kok, 2002).	16
2.10	Principais componentes da arquitetura do time <i>UvA Trilearn</i> (Boer e Kok, 2002).	17
2.11	Modos de ações presentes no código fonte do time <i>UvA Trilearn</i> (Boer e Kok, 2002).	19
3.1	Exemplo de um cruzamento entre dois cromossomos (Plant e Schaefer, 2008).	27
3.2	Pareto ótimo no espaço de objetivo e a possível relação entre as soluções (Zitzler, 1999).	32
3.3	Modelo esquemático do algoritmo NSGA-II (Deb, 2001).	36
4.1	Representação do campo virtual, dos jogadores e da bola.	42

4.2	Representação dos jogadores.	44
4.3	Fluxo de dados dentro do sistema desenvolvido.	46
4.4	Fluxograma do algoritmo genético mono-objetivo.	48
4.5	Fluxograma do algoritmo genético multi-objetivo.	50
4.6	Resultado de duas mil gerações, AG Mono-Objetivo contra <i>BRAINSTORMERS</i>	54
4.7	Resultado de duas mil gerações, AG Mono-Objetivo contra <i>WRIGHTEAGLE</i>	54
4.8	Resultado de duas mil gerações, AG Mono-Objetivo contra <i>HELIOS</i>	55
4.9	Resultado de cem gerações, AG Multi-Objetivo contra <i>BRAINSTORMERS</i>	56
4.10	Resultado de cem gerações, AG Multi-Objetivo contra <i>WRIGHTEAGLE</i>	57
4.11	Resultado de cem gerações, AG Multi-Objetivo contra <i>HELIOS</i>	57

Lista de Tabelas

3.1	Diferentes modelos de AEMOs.	34
4.1	Tempo total da simulação, sem configurar o servidor.	53
4.2	Tempo total da simulação, com o servidor configurado.	53
4.3	Média dos resultados obtidos da abordagem genética mono-objetivo.	55
4.4	Média dos resultados obtidos da abordagem genética multi-objetivo.	58

Lista de Quadros

3.1	Pseudo-código de um AG clássico.	24
3.2	Pseudo-código do NSGA-II.	37
3.3	Pseudo-código da distância de multidão no NSGA-II.	38

Lista de Abreviaturas e Siglas

AE Algoritmo Evolutivo

AEMO Algoritmos Evolutivo Multi-objetivo

AG Algoritmo Genético

AGMO Algoritmo Genético Multi-Objetivo

NSGA-II *Elitist Non-Dominated Sorting Genetic Algorithm*

PG Programação Genética

POMO Problemas de Otimização Multi-Objetivo

Sumário

Dedicatória	iii
Agradecimentos	v
Resumo	vii
Abstract	ix
Lista de Figuras	xi
Lista de Tabelas	xiii
Lista de Quadros	xv
Lista de Abreviaturas e Siglas	xvii
1 Introdução	1
1.1 Objetivo	2
1.2 Revisão Bibliográfica	2
1.3 Organização da Dissertação	7
1.4 Trabalhos Publicados	8
2 Futebol de Robôs	9
2.1 Aspectos da <i>RoboCup</i>	9
2.2 Categoria de Simulação 2D	13
2.3 <i>UvA Trilearn</i>	16
2.4 Considerações Parciais	20
3 Abordagem Evolutiva Mono-Objetivo e Multi-Objetivo	21
3.1 Teoria da Evolução	22

3.2	Algoritmos Genéticos	23
3.2.1	Representação dos Indivíduos	25
3.2.2	Definição da População Inicial	26
3.2.3	Operadores Genéticos	27
3.2.4	Seleção dos Indivíduos	29
3.3	Otimização Multi-Objetivo	29
3.3.1	Conceito de Dominância e Pareto-Ótimo	31
3.4	Algoritmo Evolutivo Multi-Objetivo	33
3.4.1	Algoritmo NSGA-II	35
3.5	Diferenças Fundamentais entre Otimização Mono-Objetivo e Multi-Objetivo	39
3.6	Considerações Parciais	40
4	Metodologia Proposta	41
4.1	Características do Time Desenvolvido	42
4.2	Codificação dos Indivíduos	44
4.3	Características do AG	45
4.4	Abordagem Mono-Objetivo	47
4.5	Abordagem Multi-Objetivo	48
4.6	Resultados Obtidos	51
4.6.1	Resultado do Algoritmo Genético Mono-Objetivo	53
4.6.2	Resultado do Algoritmo Genético Multi-Objetivo	55
4.7	Considerações Parciais	58
5	Conclusões Gerais	59

Capítulo 1

Introdução

As pesquisas em sistemas multiagentes têm se tornado uma das principais tendências no domínio da inteligência artificial (Weiss, 1999). O conceito de futebol de robôs foi introduzido em 1993 por Itsuki Noda (Kitano et al., 1995). A competição de futebol de robôs tem como pretexto o desenvolvimento de técnicas de inteligência artificial, para aplicações robóticas, utilizando um problema padrão em que um grande leque de possíveis tecnologias possam ser examinadas. Algumas técnicas utilizadas no ambiente do futebol de robôs simulados são as seguintes:

- Algoritmos Genéticos;
- Aprendizado por Reforço;
- Aprendizado por Reforço Acelerado por Heurísticas;
- Sistemas *Fuzzy*;
- Redes Neurais Artificiais.

De fato, já há muito tempo, um dos objetivos mais perseguidos pelo ser humano é dotar uma máquina, construída pelo mesmo, com características inteligentes. Desde então, têm-se observado o surgimento de novos paradigmas simbólicos de aprendizagem, em que muitos deles se tornaram métodos computacionais robustos, como aprendizado baseado em explicações, sistemas classificadores e sistemas evolutivos (Ross, 2004).

Neste trabalho utilizou-se algoritmos genéticos, o qual é inspirado na teoria da evolução de Darwin e pode ser considerado como um método que trabalha com buscas paralelas randômicas direcionadas, em que se utiliza o processo evolutivo para determinar a melhor solução de um problema. São empregados para resolver problemas complexos, tendo como vantagem seu paralelismo. O algoritmo genético percorre o espaço das possíveis soluções usando muitos indivíduos, de modo que são menos suscetíveis de ficarem presos em um mínimo local (Goldberg, 1989).

1.1 Objetivo

O principal objetivo deste trabalho é analisar a eficiência dos algoritmos genéticos aplicados no futebol de robôs da categoria Simulação 2D. Para isto, foram estudados e implementados dois métodos: o primeiro, baseado na aplicação do algoritmo genético mono-objetivo e, o segundo, consiste na aplicação do algoritmo genético multi-objetivo utilizando especificamente o algoritmo *NSGA-II*. Tais implementações visam demonstrar a diferença obtida na aplicação dos respectivos métodos no problema padrão do futebol de robôs simulado.

Outro aspecto deste trabalho que é de suma valia é a inexistência de qualquer tipo de documentação na literatura reportando tal análise comparativa, sendo portanto um trabalho inédito no campo de estudo do futebol de robô.

1.2 Revisão Bibliográfica

Os algoritmos genéticos (AG) têm sido utilizados em diversas ocasiões no domínio da *RoboCup*¹. Em Nakashima et al. (2004), um AG foi utilizado para implementar uma estratégia de um time que poderia derrotar a estratégia de um único time adversário. Para isso, a área de jogo foi dividida em 48 áreas diferentes. A localização de um jogador em posse da bola é representada como A , sendo uma figura no intervalo de $[1;48]$. A

¹*RoboCup* é um projeto internacional conjunto para promover a Inteligência Artificial (IA), robótica e campos relacionados. É uma tentativa de promover a IA e pesquisas em robótica inteligente, fornecendo um problema padrão onde uma vasta gama de tecnologias podem ser integradas e examinadas.

distância entre este jogador e um jogador adversário é apresentado como B , e dependendo da distância entre os jogadores, pode-se considerá-lo como próximo ou não próximo (em relação a um valor limiar pré-definido). O resultado de A e de B é mostrado como C , que é a ação escolhida como resposta das possíveis informações. Assim, C pode ser uma das 10 diferentes ações de alto nível, que incluem uma variedade de tipos de passes, drible ou a opção de chutar. Esse processo é representado pelas regras:

$$\text{Se Agente é } A_i \text{ e Adversário está em } B_j \text{ então Ação é } C_k \quad (1.1)$$

A estratégia do time é armazenada como um cromossomo de 960 genes de comprimento (ver Figura 1.1). Como um jogador pode estar em 48 diferentes posições no campo de jogo, com um jogador a ser oposição ou perto ou longe, há então $48 \times 2 = 96$ regras de ação diferentes para cada jogador e, portanto, 960 para todos os 10 jogadores no total. O goleiro não possui um cromossomo, pois apresenta ações básicas. Cada gene é um inteiro no intervalo $[1; 10]$, correspondente a uma das 10 possíveis ações de alto nível.

Figura 1.1: Sequência de bits utilizados como estratégia do time (Nakashima et al., 2004).

Para esta abordagem, foi escolhida população de 1 e a probabilidade de mutação definida como $1/96$. A aptidão do cromossomo é calculada no decorrer de uma partida real. O número de gols marcados pelo time é a principal medida, sendo que se a quantidade de gols marcados for igual na população, o número de gols sofridos será então levado em consideração. Para testar o método evolucionário, dois tipos de experimentos foram

configurados. O primeiro foi colocado para jogar o cromossomo contra a versão mutante do cromossomo. O vencedor será selecionado para, em seguida, ser mutado a fim de assim jogar novamente contra seu predecessor. O outro experimento foi colocar para jogar cada cromossomo contra apenas um único time, ou seja, a *Uva Trilearn* que ganhou a liga de simulação da *RoboCup* 2003. Tanto o cromossomo original como o mutante jogaram com um time pré-determinado, sendo que o mais apto (ou seja, a maior diferença de pontuação) foi selecionado para a mutação. No primeiro experimento, a cada mil gerações, o cromossomo foi medido. Tornou-se claro que, pela geração de seis milésimos, o desempenho do placar tinha melhorado, passando de uma média de 0,9 gols marcados em 10 jogos na milésima geração, para uma média de 3,8 gols por jogo. Da mesma forma, no segundo experimento, os gols por média de jogos melhoraram ao longo do tempo. Os experimentos concluíram que, mesmo com uma população mínima, os resultados podem ser alcançados utilizando técnicas evolutivas.

Um outro caso de empregar os AGs para a aprendizagem de estratégias de um time foi apresentado em Cheung et al. (2005), onde uma técnica semelhante à de Nakashima et al. (2004) foi utilizada. Aqui, os cromossomos consistem de quatro genes, os quais representam um dos quatro jogadores de diferentes tipos (goleiro, zagueiro, meia e atacante). Cada um tem um conjunto de ações de tamanho 50 regras que abrangem as diferentes opções que um jogador pode tomar. No entanto, a seleção dos mais aptos é calculada utilizando um treinador (treinador *off-line*). O treinador conecta-se ao servidor de futebol de maneira muito parecida com um jogador, mas tem uma visão completa e informações sobre o curso do ambiente (Chen et al., 2002). Em Cheung et al. (2005), é utilizado uma classe chamada *Fitness*, que permite ao treinador atribuir pontos para um tipo de jogador com base em seu desempenho (por exemplo, um ponto para marcar um gol). Uma média para cada tipo de jogador é usada como medida de *fitness*.

A abordagem evolutiva em Cheung et al. (2005) é também diferente da abordagem de Nakashima et al. (2004). A técnica utilizada é Selecionar, Multiplicar e Mutar (SMM). Cada cromossomo escolhido é multiplicado por 4, dos quais três são mutantes e o original é mantido. O cromossomo mais apto dos quatro é usado como ponto de partida para a próxima geração. Dois tipos de experimentos foram realizados com cada cromossomo. No primeiro, cada cromossomo joga com o mesmo time adversário, sendo que no segundo

o time adversário foi trocado. A medida para a experiência foi a diferença de gols. Os resultados mostraram que contra um time estático, entre a quinquagésima geração e a centésima geração, houve melhora de 10 gols. Notou-se também que variando o adversário não implicou em nenhuma melhora.

Embora o objetivo inicial fosse implementar um verdadeiro AG neste experimento, esta abordagem foi considerada computacionalmente muito cara e demorada. Portanto, um método evolutivo alternativo foi utilizado, semelhante à abordagem de Nakashima et al. (2004), sem a parte do cruzamento no algoritmo.

Em Raadt et al. (2003), investigou-se como a área de atuação do jogador e parâmetros de localização podem ser manipulados usando um AG. Novamente, foi utilizado um cromossomo para armazenar todas as informações do time. Foi visto como uma vantagem o fato de que todos os membros do time poderiam acessar as informações táticas de cada jogador. Por exemplo, se o lateral direito está na posição de zagueiro, será capaz de lidar com situações relacionadas com o meio de campo. Portanto, foi desenvolvida a classe *Evolver* para ser amplamente utilizada. Primeiro, ela é responsável por determinar a aptidão de cada cromossomo do time. Usando essa informação, a classe *Evolver* pode selecionar os cromossomos mais aptos do time a serem utilizados na próxima geração, ou remover versões inferiores. Para cada geração foram utilizados seis times com cromossomos individuais, que competiram em um torneio de pontos corridos. A classe *Evolver* atribui três pontos por vitória, um ponto por empate e zero por uma derrota.

O vencedor de cada geração do torneio de pontos corridos foi mantido para a próxima geração. Os dois cromossomos do time pior classificado foram completamente retirados da geração e foram também substituídos por mutantes de dois times do topo. Os times restantes foram compostos do cruzamento de quatro times sobreviventes. A formação tática (3 – 5 – 2) foi utilizada no time da primeira geração. Após a septuagésima geração, o time mais forte teve a defesa e o meio-campo aglutinados no centro de seu próprio meio de campo, enquanto os atacantes não mudaram muito durante o jogo. Foi relatado que muitos desses jogos terminaram empatados em 0 – 0, mostrando que os zagueiros estavam bloqueando os atacantes. No entanto, os laterais tinham tomado posições muito à frente. Na geração de número 256, a formação ofensiva do time mais apto mostrou que os laterais

ainda estavam bem posicionados e que os atacantes estavam também em várias posições, no caso o atacante mais avançado (mais perto do gol adversário). Foi concluído que novas formações surgiram com base no cruzamento dos cromossomos entre os times mais forte, sendo que a adaptação dos jogadores para preencher posições diferentes começou a se tornar evidente.

Em Jeong e Lee (1997), uma versão simplificada de um jogo de futebol simulado foi utilizada para testar o sucesso de um AG, em relação à capacidade de um jogador de aprender uma habilidade especial. Um jogo, utilizando células, é implementado para representar o campo. Um passo corresponde ao movimento de cada jogador deslocando-se em cada quadrado. O campo foi constituído por uma área de 6×5 células, como mostrado na Figura 1.2. Há dois times, em que cada um é composto por dois agentes idênticos. Existe também uma bola e dois objetivos, localizados centralmente em ambas as extremidades do campo.

Figura 1.2: Área utilizada para o exercício de treinamento dos jogadores (Jeong e Lee, 1997).

Cada agente tem uma escolha entre três ações: chutar, andar e driblar. Chutar permite que o jogador em posse da bola possa chutar a mesma mais de duas células. Andar permite ao agente mudar para uma célula adjacente, enquanto driblar é o mesmo movimento que andar, mas usado somente quando em posse da bola.

Para a pesquisa do cromossomo usou-se uma *string* com 3.300 genes, obtida a partir de todas as diferentes combinações possíveis do ambiente, como a posição do agente, a posição da bola, ação do agente, etc. Cada solução possível do cromossomo foi medida

usando uma função de *fitness*. No decorrer de um jogo, o time recebeu 100 pontos para cada gol e 10 pontos para cada vez que se encontrava em posse da bola. Os pontos eram negativamente concedidos ao time que sofrer um gol (-90), ou que não estavam em posse da bola (-10).

O cromossomo mais apto, baseado nesta pontuação, foi mantido para a próxima geração. Cada geração consistiu de 50 cromossomos. Os cromossomos restantes foram mutados, com uma probabilidade de 1%, e cruzou-se com uma probabilidade de 50%. O pior dos cromossomos foi retirado a cada geração.

Os resultados para cada caso mostra que o AG foi sempre capaz de encontrar uma solução. O mais bem sucedido foi o caso 2, onde apenas se precisou de 10 gerações para chegar a um resultado. No caso 3, foram necessários mais de 120 gerações para encontrar a solução. Concluiu-se que a maior interação com o time adversário ocorreu no presente caso e, portanto, teve mais tentativas para acontecer isso. Todos os três casos apresentaram uma tendência geral que mostra que, ao longo das gerações, o *ranking fitness* continua aumentando, o que implica também na melhora do time.

1.3 Organização da Dissertação

No Capítulo, 2 será tratado do tema futebol de robôs, explicitando desde os aspectos da competição (*RoboCup*) até as suas categorias. Abordará também o time *Uva Trilearn*, o qual foi utilizado como base para o desenvolvimento do time proposto.

No Capítulo 3, será abordado os aspectos teóricos sobre AG e sua respectiva implementação. Tratará também do problema de otimização multi-objetivo, do conceito de Dominância de Fronteira de *Pareto*.

O Capítulo 4 é referente à metodologia proposta. Além de ser possível verificar o cenário de aplicação do algoritmo apresentado.

Finalmente, o Capítulo 5 serão descritas as conclusões a respeito dessas investigações, assim como, serão apresentados algumas direções para futuros trabalhos.

1.4 Trabalhos Publicados

FRACCAROLI, E. S. ; SILVA, I. N. . Abordagem Fuzzy para Determinação de Estratégia em Ambiente de Futebol de Robôs Simulado. In: 8th Brazilian Conference on Dynamics, Control and Applications, 2009, Bauru - SP. Proceedings of the 8th Brazilian Conference on Dynamics, Control and Applications, 2009. Vol. 8. pp. 1-6.

FRACCAROLI, E. S. ; SILVA, I. N. . Abordagem Genética para Definição de Ações em Ambiente Simulado de Futebol de Robôs. In: IX Simpósio Brasileiro de Automação Inteligente, 2009, Brasília. ANAIS DO IX SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE, (CD-ROM // paper number: 55708 // 06 páginas), 2009.

Capítulo 2

Futebol de Robôs

O conceito de futebol de robôs autônomos foi introduzido em 1993 por Itsuki Noda (Kitano et al., 1995). A competição de futebol de robôs tem como pretexto o desenvolvimento de técnicas de inteligência artificial, para aplicações robóticas, utilizando um problema padrão em que um grande leque de possíveis tecnologias possam ser examinadas. Aprendizado por reforço, aprendizado por reforço baseado em heurísticas, algoritmos genéticos, sistemas *fuzzy* e redes neurais artificiais são as tecnologias mais utilizadas no ambiente de futebol de robôs.

Neste capítulo, a Seção 2.1 tratará sobre os aspectos da *RoboCup* e suas principais categorias. O conceito e arquitetura do ambiente simulado de futebol de robôs será discutida na Seção 2.2. A Seção 2.3 abordará sobre o time *UvA Trilearn* e seu conceito de divisão de camadas de competências.

2.1 Aspectos da *RoboCup*

A *RoboCup* (*Robot World Cup*) teve seu início no Japão e tem como ambicioso objetivo criar um time de robôs humanóides, totalmente autônomos, capazes de ganhar dos campeonatos mundiais de futebol humano (RoboCup, 2009). A *RoboCup* possibilita pesquisas nas áreas de robótica e inteligência artificial, disponibilizando um problema desafiador que oferece a integração de pesquisas nas diversas áreas da robótica e da inteligência artificial,

como exemplo, agentes autônomos, visão computacional, colaboração multiagente, comportamento reativo, aprendizado, controle de movimento, controle inteligente robótico, evolução, resolução em tempo real, entre outros (Kitano et al., 1998).

Existem cinco categorias na *RoboCup*, ou seja, *Simulation*, *Humanoid*, *Middle Size*, *Small Size* e *Standard Platform*, cada uma com o foco em diferentes níveis de abstração para o respectivo problema.

A categoria *Humanoid* (Figura 2.1), possui no máximo dois robôs bípedes em cada time, e tem como foco a pesquisa nas áreas que incluem: andar dinamicamente, correr, chutar a bola, percepção visual da localização de objetos em campo e auto-localização.

Figura 2.1: Robôs da categoria *Humanoid* da *RoboCup* (RoboCup, 2009).

Na categoria *Middle Size* (Figura 2.2), cada time pode jogar com o número máximo de quatro robôs. Cada robô não pode ultrapassar 75cm de altura e 50cm de diâmetro. Suas principais áreas de pesquisas incluem: localização, visão computacional, integração de sensores, percepção distribuída, controle de movimentos do robô e *hardware*.

Figura 2.2: Robôs da categoria *Middle Size* da *RoboCup* (RoboCup, 2009).

A categoria *Small Size* (Figura 2.3) é composta por um time de no máximo cinco robôs, cada um com 15cm de altura e 18cm de diâmetro. Esta possui as mesmas áreas de pesquisas da categoria *Middle Size*.

Figura 2.3: Robôs da categoria *Small Size* da *RoboCup* (RoboCup, 2009).

A categoria *Standard Platform* possui dois tipos de robôs, um bípede e um quadrúpede, fabricados por diferentes empresas, os quais são utilizados para formar um time de futebol robótico. O robô bípede é chamado de NAO (Figura 2.4), fabricado pela empresa francesa

Aldebaran Robotics. O robô quadrúpede é chamado de AIBO (Figura 2.5), fabricado pela empresa japonesa *Sony*. Suas áreas de pesquisas são as mesmas da categoria *Humanoid*, incluindo somente a complexidade de controlar os movimentos das quatro patas no caso do robô AIBO.

Figura 2.4: Robôs NAO da categoria *Standard Platform* da *RoboCup* (RoboCup, 2009).

Figura 2.5: Robôs AIBO da categoria *Standard Platform* da *RoboCup* (RoboCup, 2009).

A categoria *Simulation* possui onze jogadores e um técnico por time. Suas áreas de pesquisas incluem, a colaboração multiagente, o desenvolvimento de técnicas de otimiza-

ção e a aplicação de sistemas inteligentes. Diferente das outras categorias da *RoboCup*, essa categoria preocupa-se exclusivamente com o desenvolvimento da inteligência coletiva do time.

Figura 2.6: Categoria *Simulation 2D* da *RoboCup* (RoboCup, 2009).

2.2 Categoria de Simulação 2D

O ambiente da *RoboCup* Simulação 2D apresenta as características de ser um ambiente dinâmico, ruidoso¹, cooperativo e coordenado. É composto por dois programas, o *Soccer Server* (servidor) e o *Soccer Monitor* (monitor). O servidor é um ambiente que torna possível o jogo de futebol entre dois times, sendo responsável por simular os movimentos dos jogadores e da bola, a comunicação com cliente (jogador) e o controle do jogo conforme as regras. Para desenvolver times torna-se necessário que se faça uso de ferramentas que viabilizem a comunicação via *UDP/IP* (*User Datagram Protocol/Internet Protocol*), pois toda comunicação feita entre o servidor e cada cliente é feita via *socket UDP/IP*, o qual é um protocolo de comunicação orientado à conexão, incluindo-se mecanismos para iniciar

¹Para tornar as características do ambiente simulado bem próximas daquelas de um ambiente real, o servidor insere um ruído gaussiano na comunicação entre o servidor e o cliente, semelhante ao problema real de comunicação entre os jogadores de um time com seu técnico.

e encerrar a conexão, negociar tamanhos de pacotes e permitir a retransmissão de pacotes corrompidos (Chen et al., 2002).

Figura 2.7: Ilustração do monitor.

O monitor (Figura 2.7) é um programa responsável pela visualização do campo virtual no mesmo, utilizando-se para tanto o sistema *X window*. Por meio dessa interface, é possível observar os agentes jogando de acordo com as informações enviadas pelo servidor. As informações disponíveis no monitor incluem o placar, os nomes dos times, o tempo decorrido e as posições dos jogadores e da bola. É possível conectar vários monitores em um servidor, a fim de ser visualizado o jogo em monitores diferentes (Chen et al., 2002).

O cliente é um processo separado, responsável por atuar no ambiente simulado e por conectar-se com o servidor por meio de uma única porta. Um time pode ter doze clientes, onze jogadores e um técnico.

O cliente é responsável por atuar no ambiente simulado, sendo este o módulo em que são desenvolvidos e aplicados os algoritmos inteligentes para serem avaliados. As ações referentes ao comportamento dos jogadores devem ser executadas obedecendo a seguinte ordem: primeiro, o cliente envia requisições para o servidor; em seguida, o servidor recebe e armazena as requisições, as quais são analisadas para serem executadas (chutar a bola,

virar, correr, etc.); por fim, o ambiente é então atualizado e as ações são realizadas pelos jogadores. Um cliente não pode enviar mais do que três ou quatro comandos por ciclos de simulação para o servidor.

Cada partida realizada tem duração de aproximadamente dez minutos, valor este correspondente a seis mil ciclos de simulação, sendo separado em dois tempos de aproximadamente cinco minutos. Quando a partida se encerra com empate, acrescenta-se então um tempo adicional de três mil ciclos de simulação, sendo que o time que fizer o primeiro gol vencerá a partida. Se mesmo depois do tempo adicional a partida continuar empatada, é então realizada a cobrança de pênaltis, e o time que conseguir uma vantagem maior do que três gols marcados vencerá a partida (Chen et al., 2002).

Figura 2.8: Arquitetura do simulador 2D (Reis, 2003).

Um cliente conecta-se com o servidor utilizando um *socket UDP* (Figura 2.8). Usando o *socket*, o cliente envia comandos para controlar um jogador e recebe informações dos sensores do jogador. Em outras palavras, o programa cliente é o cérebro de um jogador. O cliente recebe informações dos sensores visuais e sonoros do servidor, e envia comandos de controle para o servidor.

Cada cliente pode controlar apenas um jogador. Então, um time consiste de número

igual de clientes e jogadores. A comunicação entre os clientes pode ser feita via servidor usando os protocolos *say* e *hear*.

O ambiente dos agentes simulados é uma estrutura hierárquica, dividida em dois ambientes. O ambiente de baixo nível é responsável por processar informações básicas, recebidas do servidor, tais como: informações visuais, de som, de sensibilidade e, também, é responsável por processar ações básicas como driblar, realizar um passe e chutar. O outro é o ambiente de alto nível que possibilita a decisão do ponto de vista global, tais como a estratégia de jogar de maneira cooperativa (Chen et al., 2002).

2.3 *UvA Trilearn*

O time *UvA Trilearn* foi desenvolvido na Faculdade de Ciência da Universidade de Amsterdam. Foi tema da dissertação de mestrado de Boer e Kok (2002), na qual propunha o desenvolvimento de um time com uma arquitetura dividida em três camadas, de acordo com a Figura 2.9:

Figura 2.9: Arquitetura do time *UvA Trilearn* (Boer e Kok, 2002).

- Camada de Interação: responsável pela interação com o servidor; recebe as informações dos sensores e envia para camada de competência; envia para o servidor as

ações recebidas pela camada de competência, de acordo a respectiva estratégia;

- Camada de Competência: utiliza as funcionalidades enviadas pela camada de interação a fim de construir um modelo abstrato do jogo e para implementar os diversos perfis de cada agente;
- Camada de Controle: responsável pelo raciocínio do sistema; a melhor possibilidade de ação é selecionada da camada de competência, dependendo do estado atual do jogo e da estratégia atual do time.

Figura 2.10: Principais componentes da arquitetura do time *UvA Trilearn* (Boer e Kok, 2002).

Na Figura 2.10 é apresentado o diagrama UML de classes, no qual é possível identificar as relações de associação, composição e generalização entre as três camadas da arquitetura. Em seguida, serão apresentadas todas as classes com mais detalhes.

- Conexão: componente responsável por criar uma conexão *socket UDP* com o servidor, contendo métodos para enviar e receber mensagens por meio dessa conexão;
- Controle dos sensores: esse componente processa as mensagens recebidas do servidor, particiona e envia as informações extraídas para o componente Modelo do Mundo;
- Modelo do Mundo: responsável por criar a representação atual do jogo, do ponto de vista observado pelo agente, determinando-se a posição do agente, da bola e de suas respectivas velocidades;
- Objeto: esse componente contém informações sobre todos os objetos na simulação, como demarcações do campo, jogadores, bola e posição do gol;
- Formações: esse componente contém informações sobre as possíveis formações do time e do método para determinar a posição estratégica;
- Configuração do Jogador: esse componente contém os valores de muitos parâmetros do jogador, tal como a influência para resolver algum processo do agente e de definir métodos para receber e atualizar esses valores.
- Jogador: contém métodos para resolver sobre a possibilidade da melhor ação em uma determinada situação;
- Jogador básico: esse componente contém todas as informações necessárias para um perfil de jogador, como interceptar a bola ou chutar a bola para uma determinada posição do campo;
- Controle da ação: responsável por executar os comandos que o jogador deve realizar, assim que foi recebido do componente Jogador básico.

O time *UvA Trilearn* apresenta como maior contribuição uma arquitetura de três camadas capazes de realizar o processamento paralelo dos dados, um esquema flexível de sincronização do ambiente, métodos eficazes para localização de objetos e estimação de velocidades usando filtros de partículas, uma camada hierárquica de competências, uma política de pontos (gols) para jogadores simulados e uma estratégia para um time eficaz.

Assim, encontram-se disponíveis somente dois modos simples de ações no código do *UvA Trilearn* (Figura 2.11). O primeiro é o modo de manuseio da bola e o segundo é o modo de posicionamento, os quais serão utilizados pelos jogadores dependendo da situação atual durante o jogo. Se o jogador estiver perto da bola, o modo manuseio da bola é então executado. Este modo é executado quando o jogador está perto da bola, o qual avalia a possibilidade de chutar ou não; se existir a possibilidade de chutar, o mesmo realizará uma ação consequente; por outro lado, se não existir tal possibilidade, haverá então a ação de interceptar a bola. A ação de interceptar a bola consiste em posicionar o jogador no ponto de intersecção da trajetória da bola em relação a posição do adversário (Boer e Kok, 2002).

Figura 2.11: Modos de ações presentes no código fonte do time *UvA Trilearn* (Boer e Kok, 2002).

O time *UvA Trilearn* possui 3 formações táticas, (3-4-3), (4-4-2) e (4-3-3). O *UvA Trilearn* utiliza a formação tática (4-3-3), que corresponde em separar os jogadores ao longo do campo em áreas de atuação. No caso, os quatro primeiros jogadores atuarão como zagueiros, os três outros jogadores atuarão como meio-campo e os últimos três jogadores atuarão como atacantes. Todos os jogadores só realizam uma única ação quando em posse de bola, ou seja, chutam em direção ao gol adversário.

Nesse trabalho foi utilizado o time *UvA Trilearn Base* (versão 3.3) como time base para aplicar os modelos evolutivos computacionais. Essa versão encontra-se à disposição para uso comum em *UvA Trilearn* (2003).

2.4 Considerações Parciais

Neste capítulo foi abordado o conceito de futebol de robôs. Apresentou-se a competição *RoboCup* (Seção 2.1) e suas respectivas categorias. Explicitou-se ainda de maneira detalhada a categoria Simulação 2D na Seção 2.2, pois é a categoria na qual esse trabalho faz uso. No time *Uva Trilearn*, apresentado na Seção 2.3, serão aplicados os respectivos AGs (Capítulo 3).

Capítulo 3

Abordagem Evolutiva Mono-Objetivo e Multi-Objetivo

Métodos baseados em computação evolutiva constituem uma classe de algoritmos de busca e otimização estocástica inspirados na teoria da evolução natural.

A grande vantagem dos algoritmos evolutivos (AE) encontra-se na possibilidade de resolver problemas pela simples descrição de sua formulação matemática, sendo desnecessária a descrição explícita de todos os passos até o resultado. Os AEs apresentam características robustas, podendo ser aplicados em diversos tipos de problemas. Os AEs pertencem ao conjunto de técnicas e procedimentos adaptáveis, aplicados em problemas complexos, onde técnicas tradicionais não poderiam ser aplicadas, ou seriam ineficazes.

Neste capítulo, a Seção 3.1 introduz o conceito da teoria da evolução, a qual é a base fundamental para a modelagem dos algoritmos genéticos (Seção 3.2). A Seção 3.3 descreve o conceito e características da otimização multi-objetivo, apresentando o algoritmo NSGA-II (Seção 3.4), sendo utilizado como metodologia computacional os algoritmos genéticos mono-objetivo e multi-objetivo.

3.1 Teoria da Evolução

Charles Darwin¹ propôs a teoria da evolução em 1859 (Darwin, 1859), na qual indivíduos com boas características genéticas possuem maiores chances de sobrevivência e de se reproduzirem, gerando-se, assim, indivíduos cada vez mais aptos. Por outro lado, os indivíduos menos aptos tenderão a desaparecer. O diferencial dessa teoria em relação à teoria Lamarckista² dá-se justamente na explicação de como se realiza a adaptação (seleção natural) dos indivíduos de uma população, pois Lamarck acreditava na herança direta das características adquiridas pelos indivíduos durante a vida. Já a seleção natural vem sendo a principal idéia unificadora nas mais diversas áreas da biologia, pois a mesma é um processo que distingue os processos biológicos dos demais processos físicos e químicos. Darwin apresentou as seguintes hipóteses para explicar o processo de seleção natural:

- Os indivíduos (filhos) tendem a ser em maior número que os indivíduos (pais);
- O número de indivíduos de uma mesma espécie permanece constante;
- A partir do primeiro e do segundo item, pode-se concluir que haverá uma luta pela sobrevivência;
- Dentro de uma mesma espécie, os indivíduos apresentam pequenas diferenças em relação às suas características, sendo que a maioria delas estão presentes nos respectivos indivíduos (pais).

O princípio da seleção natural propõe que os indivíduos, cujas variações se adaptam melhor ao ambiente, terão maior probabilidade de sobreviver e se reproduzir. A evolução Darwiniana é nada mais que a consequência inevitável da competição entre processos de reprodução de informação, operando no interior de uma população em um determinado habitat comum (Francis, 2007).

¹Charles Darwin, naturalista inglês, 1809-1882.

²Jean Baptiste de Lamarck, naturalista francês, 1744-1829.

3.2 Algoritmos Genéticos

O conceito de algoritmos genéticos (AG) foi introduzido por Holland (1992) e sua terminologia é baseada na teoria da evolução. Tem como objetivo simular, por meio de algoritmos computacionais, os processos de adaptação de sistemas naturais utilizando um sistema artificial.

O AG é um método de busca estocástica inspirada na teoria de Darwin (Seção 3.1), que consiste em encontrar uma resposta (indivíduo) para um determinado problema com um número grande de possíveis soluções (população). Para isso, utiliza-se dos métodos de seleção de indivíduos, dos operadores genéticos de cruzamento (ou *crossover*) e da mutação. Um indivíduo da população é representado por um único cromossomo, o qual possui a codificação de uma possível solução do problema. Os cromossomos são geralmente implementados na forma de listas de atributos ou vetores de *strings*, onde cada atributo é conhecido como *gene* e os possíveis valores que um determinado *gene* pode assumir são chamados de *alelos* (Holland, 1992).

O processo de evolução executado por um AG corresponde a um procedimento de busca em um espaço de soluções potenciais para o problema. Esta busca requer um equilíbrio entre dois objetivos aparentemente conflitantes: o aproveitamento das melhores soluções e a exploração do espaço de busca. Com isso, os AGs constituem uma classe de métodos de busca de propósito geral, que apresentam um considerável equilíbrio entre aproveitamento das melhores soluções e da exploração do espaço de busca. O processo de busca se dá de maneira multi-dimensional a cada geração; assim, as soluções consideradas boas (mais aptas) se reproduzem, enquanto as soluções consideradas ruins (menos aptas) são descartadas. Uma propriedade importante dos AGs é que esses mantêm uma população de soluções candidatas, enquanto que outros métodos alternativos, como *simulated annealing* (Aarts e Korst, 1989), analisam um único ponto no espaço de busca a cada instante. Além disso, os AGs possuem um paralelismo implícito decorrente da avaliação independente de cada uma das cadeias de bits (cromossomo) que compõem os indivíduos. Emprega-se ainda uma função de avaliação (*fitness*) para distinguir diferentes soluções, simulando aqui o papel da pressão exercida pelo ambiente sobre o indivíduo (Michalewicz, 1996). O Quadro 3.1 apresenta o pseudo-algoritmo do AG clássico proposto por Holland (1992).

Quadro 3.1 Pseudo-código de um AG clássico.

ALGORITMO GENÉTICO

Entrada:

- N (tamanho da população)
 T (número máximo de gerações)
 p_c (probabilidade de cruzamento)
 p_m (taxa de mutação)

Saída:

- A (conjunto não-dominado)

1. **Inicialização:** Conjunto $P_o = \emptyset$ e $t = 0$. PARA $i = 1, \dots, N$ FAÇA
 - a) Escolha $i \in I$ segundo alguma probabilidade de distribuição.
 - b) Conjunto $P_o = P_o + i$.
 2. **Fitness:** PARA cada indivíduo $i \in P_t$, determine o vetor de decisão $x = m(i)$, bem como o vetor objetivo $y = f(x)$ e calcule o valor escalar do *fitness* $F(i)$.
 3. **Seleção:** População temporária P' . Conjunto P' . PARA $i = 1, \dots, N$ FAÇA
 - a) Selecione um indivíduo $i \in P_t$ de acordo com um determinado esquema, e com base no seu valor de *fitness*.
 - b) Conjunto $P' = P' + i$.
 4. **Cruzamento:** Conjunto $P'' = \emptyset$. PARA $i = 1, \dots, N/2$ FAÇA
 - a) Escolha dois indivíduos $i, j \in P'$ e remova ele de P' .
 - b) Recombine i com j . Os indivíduos (filhos) resultantes são $k, l \in I$.
 - c) Adicione k, l em P'' com probabilidade p_c . De outra maneira adicione i, j em P'' .
 5. **Mutação:** Conjunto $P''' = \emptyset$. PARA cada indivíduo $i \in P''$ FAÇA
 - a) Mute i com taxa de mutação p_m . O indivíduo resultante $j \in I$.
 - b) Conjunto $P''' = P''' + j$.
 6. **Conclusão:** Conjunto $P_{t+1} = P'''$ e $t = t + 1$. SE $t \geq T$ ou SE alguma condição de parada for satisfeita ENTÃO conjunto $A = p(m(P_t))$ SE NÃO volte para o Passo 2.
-

O AG deve apresentar os seguintes componentes:

- Um método de seleção genética para soluções candidatas ou potenciais;
- Uma maneira de gerar uma população inicial para as possíveis soluções potenciais;
- Uma função para avaliar sua adaptação ao ambiente;
- Operadores genéticos, como cruzamento e mutação.

Para se fazer uso de um AG, em um determinado problema particular, é necessário considerar a maneira da representação genética para as soluções potenciais (etapa de codificação); metodologia para criar e iniciar uma população inicial; função de avaliação para classificar o quão próximo o indivíduo está da solução ótima; definir os operadores genéticos; e definir os parâmetros livres do AG, como: tamanho da população, tamanho do cromossomo, probabilidade da atuação dos operadores genéticos, condição de parada, número de gerações, entre outros.

3.2.1 Representação dos Indivíduos

Para aplicar o AG a fim de encontrar a solução ótima de um problema qualquer, é necessário considerar o aspecto de como será a representação deste problema, sendo que essa é uma das etapas mais críticas da definição do algoritmo genético, pois dependendo da definição inadequada da codificação, o AG pode ter problemas de convergência prematura (Deb, 2001).

No AG clássico proposto por Holland (1992), os cromossomos são codificados em vetores de *strings* binárias de tamanhos fixos, onde cada gene do vetor de *strings* representa a presença (1) ou ausência (0) de uma determinada característica. A grande motivação para o uso da codificação binária vem da Teoria dos Esquemas (*schemata theory*), utilizada para justificar por que os AGs funcionam. Holland (1992) e Goldberg (1989) argumentam que maximizando o paralelismo inerente do AG, seu desempenho também se maximizará, provando que o alfabeto binário maximiza o paralelismo implícito.

Michalewicz (1996) concluiu, a partir de testes entre um AG codificado em ponto flutuante e um AG codificado em binário, que o desempenho de um AG codificado em binário é pobre quando o espaço de busca possui dimensão elevada. Destaca-se que essa afirmação não representa o consenso entre autores dentro da literatura sobre AGs.

Segundo Michalewicz (1996), em diversas aplicações práticas, a codificação binária aplicada em problemas contendo um espaço de busca contínuo, ou com necessidade de maior precisão, demonstra um desempenho insatisfatório. Uma dificuldade é encontrada com certas *strings*, por exemplo, as *strings* “01111” e “10000”, que para realizar uma

transição para uma solução vizinha no espaço real é necessário alterar muitos *bits* (Deb, 2001).

Outra dificuldade está na incapacidade para armazenar qualquer precisão arbitrária em uma solução ótima. Em AGs codificados em binários, o tamanho da *string* pode ser definido para armazenar uma certa precisão de uma solução. Quanto maior a precisão, maior será o tamanho da *string*. Quanto maior for o tamanho da *string*, maior será o tamanho da população, aumentando-se consequentemente a complexidade do algoritmo (Deb, 2001).

Em problemas de otimização restrita, a codificação inadequada pode gerar indivíduos modificados inválidos por cruzamento ou mutação.

3.2.2 Definição da População Inicial

A população inicial pode ser gerada utilizando um método de inicialização aleatória dos cromossomos. Quando houver conhecimento de alguma heurística, a respeito do problema, pode-se utilizá-la na criação, tendo cuidado para não gerar indivíduos inválidos nesta fase. Cada um dos cromossomos deve ser avaliado de acordo com seu sucesso em direção à solução do problema, sendo que tal avaliação é obtida por meio da função de avaliação (*fitness*), cuja missão é estabelecer uma relação que permita medir o desempenho de cada solução (Deb, 2001).

O tamanho da população afeta diretamente no desempenho dos algoritmos genéticos, isto é:

- Se a população gerada for pequena, a cobertura do espaço de busca do problema também será pequena, obtendo-se assim um baixo custo computacional;
- Se a população gerada for grande, a cobertura do espaço de busca do problema também será grande, o que pode prevenir convergências prematuras para soluções locais; porém, aumenta-se o custo computacional.

Depois de ter escolhido uma forma de representação dos esquemas de *strings* e ter gerado a população de *strings* aleatórias, é necessário aplicar os operadores genéticos

para *strings* semelhantes a fim de se encontrar a melhor população de soluções.

3.2.3 Operadores Genéticos

Os operadores genéticos são aplicados aos indivíduos esperando-se obter uma população sucessiva cada vez mais apta, ou seja, mais próxima da solução ótima. Os operadores genéticos mais utilizados são cruzamento (recombinação gênica) e mutação. São gerados novos indivíduos (filhos) a partir das características herdadas dos indivíduos (pais).

O operador genético de cruzamento (*crossover*) produz novas soluções a partir do cruzamento de dois ou mais indivíduos da geração atual, trocando assim, informações entre diferentes soluções candidatas. Existem muitas variações de cruzamento, muitas delas específicas para um determinado problema, não existindo então um tipo de cruzamento que seja mais eficiente. Cada variação de cruzamento pode ser particularmente eficiente para uma determinada classe de problemas e extremamente ineficiente para outras (Rezende, 2003).

Figura 3.1: Exemplo de um cruzamento entre dois cromossomos (Plant e Schaefer, 2008).

No AG clássico, o operador de cruzamento mais utilizado é o cruzamento de um ponto. Antes de aplicar esse operador, primeiro são selecionados dois indivíduos (pais), depois seus cromossomos são recombinados, gerando então dois novos indivíduos (filhos). De acordo com a Figura 3.1, após a seleção de dois indivíduos (pais), é necessário definir um ponto de corte aleatório igual para os indivíduos (pais). Em seguida, os segmentos gerados a partir do ponto de corte serão trocados, gerando-se assim dois novos indivíduos (filhos).

Cada indivíduo (filho) contém, em seu próprio cromossomo, metade do cromossomo do Pai 1 e outra metade do cromossomo do Pai 2.

O operador de cruzamento é utilizado como um operador de recombinação dos melhores indivíduos tentando preservar as características boas de cada um. Para preservar as *strings* boas durante a aplicação do operador de recombinação, nem todas as *strings* de uma população serão utilizadas no cruzamento. Se uma probabilidade de cruzamento de p_c é utilizada, então $100p_c\%$ de *strings* da população serão usadas na operação de cruzamento e $100(1 - p_c)\%$ da população serão simplesmente copiadas para a nova população (Deb, 2001).

O operador de mutação modifica aleatoriamente um ou mais genes de um cromossomo, sendo o mesmo essencial por dois motivos. Primeiro, na natureza, mutações podem ocorrer de maneira aleatória. Segundo, e mais importante, em um contexto de um problema que visa atingir o objetivo ótimo, isso impede que a população torne-se uniforme e incapaz de se desenvolver e, também, prevê a possibilidade de saltar pontos de mínimos ou máximos locais. Define-se então qual será a probabilidade de mutação p_m , ou seja, quantos genes estarão sujeitos à mutação em um cromossomo. Geralmente a probabilidade de mutação é um valor baixo, para manter a diversidade na população sem destruir o progresso já obtido. Para indivíduos codificados binariamente, o operador de mutação troca o valor do gene escolhido para 1, se for de valor 0, ou vice-versa (Rezende, 2003).

O procedimento de mutação de um *bit* necessita a criação de um número aleatório para cada *bit*. Para reduzir o esforço computacional, Goldberg (1989) sugeriu um operador de mutação baseado no tempo, onde depois de um *bit* ser mutado, a localização do próximo *bit* a ser mutado é determinada por uma distribuição exponencial. A característica da distribuição é dada por $\mu = 1/p_m$. Primeiro, é criado um número aleatório $r \in [0, 1]$. Então, estima-se o próximo *bit* a ser mutado a partir de saltos de $\eta = -p_m \ln(1 - r)$ bits do *bit* corrente (Deb, 2001).

Os operadores de cruzamento e de mutação são simples e claros. Após selecionar boas *strings*, o operador de cruzamento recombina boas *substrings* a partir de duas outras boas *strings* de forma a se esperar uma *substring* melhor. Já o operador de mutação altera apenas um ponto da *string* esperando criar uma *string* melhor. Espera-se que, se *strings*

ruins forem criadas, que elas sejam eliminadas pelo operador de reprodução na próxima geração e, se boas *strings* forem criadas, que elas sejam enfatizadas.

3.2.4 Seleção dos Indivíduos

São utilizados diversos métodos para seleção dos indivíduos, sendo que os mais utilizados são: seleção por roleta (*roulette wheel*), seleção elitista e seleção baseada em torneio.

O método de seleção por roleta (Michalewicz, 1996) atribui a cada indivíduo da população um valor correspondente à probabilidade de estar presente na próxima geração. Esta probabilidade é proporcional ao valor da função de avaliação medido de cada indivíduo, em relação à somatória dos valores da função de avaliação de todos indivíduos de uma população. Assim, quanto maior o valor da função de avaliação de um indivíduo, maior a probabilidade dele estar presente na próxima geração. Este método de seleção apresenta o seguinte problema: por ser um método de seleção aleatório, no momento da seleção o melhor indivíduo da população pode não ser selecionado.

O método de seleção elitista foi proposto para solucionar o problema do método de seleção por roleta. Esse método seleciona o melhor indivíduo da população e o insere na próxima população (Michalewicz, 1996).

No método de seleção baseada em torneio, os indivíduos são ordenados de acordo com o valor correspondente do *fitness* de cada indivíduo da população, e a posição dos indivíduos é utilizada para determinar a probabilidade de seleção (Deb, 2001). Uma variação desse método é simplesmente passar os n melhores indivíduos para a próxima geração.

3.3 Otimização Multi-Objetivo

O problema de otimização multi-objetivo (POMO)³ pode também ser definido como um problema de busca. Como o próprio nome sugere, um POMO apresenta um incontável conjunto de soluções que, quando avaliadas, produzem vetores cujos componentes

³Também chamado de otimização de multi-critérios, multi-desempenho ou problema de otimização de vetores.

representam as soluções conflitantes no espaço objetivo (Coello et al., 2002).

Um POMO possui um conjunto de funções objetivos a serem otimizadas e restrições que devem ser satisfeitas por qualquer solução viável⁴ (Deb, 2001). O conjunto de todas as soluções viáveis é conhecido como *espaço de busca* ou *região viável*.

Um POMO genérico será formalmente apresentado a seguir.

Definição de POMO: Encontrar o vetor $\vec{x}^* = [x_1^*, x_2^*, \dots, x_n^*]^T$ que satisfaça as m restrições de desigualdade:

$$g_i(\vec{x}) \geq 0 \quad i = 1, 2, \dots, m \quad (3.1)$$

as p restrições de igualdade:

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \dots, p \quad (3.2)$$

e que otimize o vetor função:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})]^T \quad (3.3)$$

O objetivo é determinar, dentre o conjunto de todos os números que satisfaçam os dois conjuntos de restrições, o conjunto particular $x_1^*, x_2^*, \dots, x_n^*$ que produza o melhor valor da função objetivo.

As restrições impostas definem a região viável Ω e qualquer ponto $\vec{x}$ em Ω é considerado uma solução viável. O vetor função $\vec{f}(\vec{x})$ é uma função que mapeia o conjunto Ω dentro do conjunto Λ , que representa todos os valores possíveis da função objetivo, chamado de fronteira de *Pareto* (Seção 3.3.1). Os k componentes do vetor função $\vec{f}(\vec{x})$ representam o critério não comensurável que devem ser considerados. As constantes $g_i(\vec{x})$ e $h_i(\vec{x})$ representam as restrições impostas sobre as variáveis de decisão. O vetor $\vec{x}^*$ é reservado para designar as soluções ótimas e que, geralmente, apresentam mais de uma solução

⁴Uma solução x é viável se, e somente se, satisfazer todas as restrições. Caso contrário, a solução não será viável.

ótima (Coello et al., 2002).

Existem três possibilidades em POMO: minimizar todas as funções objetivo; maximizar todas as funções objetivo, ou ainda, minimizar algumas e maximizar outras. Por razões de simplificação, normalmente todas as funções são convertidas para a forma de maximização ou minimização $\max f_i(\vec{x}) = -\min(-f_i(\vec{x}))$.

3.3.1 Conceito de Dominância e Pareto-Ótimo

Na maioria dos algoritmos de otimização multi-objetivo é usado o conceito de dominância. Nesses algoritmos, duas soluções são comparadas para identificar se uma solução domina a outra ou não.

Para definir, tanto em minimização ou maximização de funções objetivo, se uma solução u é melhor do que uma solução v , é utilizado o operador $\preceq$ entre as duas soluções, como em $u \preceq v$. Se uma solução u é pior do que uma solução v , é então utilizado o operador $\succeq$, como em $u \succeq v$ (Deb, 2001).

Definição de Dominância: Uma solução x_1 é dita dominada por outra solução x_2 , se as condições 1 e 2, especificadas a seguir, forem ambas verdadeiras:

- 1) A solução x_1 é não pior que x_2 em todos os objetivos, ou seja:

$$f_j(x_1) \succeq f_j(x_2), \quad \text{para todo} \quad j = 1, 2, \dots, M \quad (3.4)$$

- 2) A solução x_1 é estritamente melhor que x_2 , em pelo menos um objetivo, isto é:

$$f_j(x_1) \preceq f_j(x_2), \quad \text{para pelo menos um} \quad j = 1, 2, \dots, M \quad (3.5)$$

No conjunto de soluções encontradas, nenhuma delas é melhor do que as demais em pelo menos um objetivo. Estas são chamadas de soluções não dominadas, também conhecidas como soluções de *Pareto*⁵. Além dessas soluções, existem também aquelas soluções

⁵Vilfredo Pareto, político, sociólogo e economista italiano, 1848-1923.

dominadas, para as quais todos os objetivos encontrados são inferiores a pelo menos uma outra solução encontrada. A Figura 3.2 apresenta essas soluções.

Figura 3.2: Pareto ótimo no espaço de objetivo e a possível relação entre as soluções (Zitzler, 1999).

Assim, para propósitos de elucidação, considera-se aqui o exemplo de um POMO com dois objetivos e com cinco soluções distintas no espaço de objetivo, apresentado na Figura 3.2. Suponha que a função objetivo f precise ser maximizada, enquanto que a função objetivo g precise ser minimizada. São então apresentadas na Figura 3.2, cinco possíveis soluções. Como ambos objetivos são importantes, geralmente é difícil encontrar uma solução que seja a melhor em ambos os objetivos. O conceito de dominância é aplicado para decidir qual solução é melhor para ambos os objetivos. Comparando-se a solução E com a solução C , é possível notar que a solução C é melhor do que a solução E em relação à função objetivo f , e também, a solução C é melhor do que a solução E em relação à função objetivo g . Quando ambas as condições de dominância forem satisfeitas, é portanto possível afirmar que a solução C domina a solução E , ou $C \preceq E$.

Utilizando o mesmo exemplo da Figura 3.2, é comparada a solução A com a solução C . Observa-se que a solução A é melhor do que a solução C em relação à função objetivo f , enquanto a solução A é pior do que a solução C em relação à função objetivo g . Neste caso, a primeira condição de dominância não foi satisfeita para essas soluções. Portanto, não é possível afirmar que a solução A domina a solução C , ou que a solução C domina a solução A . As soluções A e C são chamadas nesse caso de soluções não dominadas. Com respeito a ambos objetivos, não se pode então afirmar qual das duas soluções é a melhor delas.

Como se pode observar, as soluções dominadas não são melhores em nenhum aspecto que as não dominadas, sendo assim deverão ser descartadas. O conjunto das soluções não dominadas pode ser representado no espaço cartesiano, formando-se então a chamada frente de *Pareto* ou fronteira de *Pareto*. As soluções *Pareto* ótimas ou conjunto *Pareto* ótimo, ou ainda, fronteira ótima de *Pareto*, formam o conjunto de soluções não dominadas em relação a todas as soluções possíveis.

No caso de AEMOs (Algoritmos Evolutivo Multi-objetivo) (Seção 3.4), o uso de vários objetivos conflitantes muda a definição de ótimo: em lugar de se encontrar uma solução ótima (usando um objetivo), geralmente, encontra-se um conjunto de soluções ótimas. É importante ressaltar que, ao contrário de um AG mono-objetivo, em que resulta em uma única solução ótima no final da execução, o AG multi-objetivo resulta em várias soluções ótimas (soluções não-dominadas).

3.4 Algoritmo Evolutivo Multi-Objetivo

O primeiro AEMO a ser implementado foi proposto por Schaffer (1984). Ele foi chamado de VEGA (*Vector Evaluated Genetic Algorithm*), por se tratar de um AG que avalia vetores de objetivos, com cada elemento do vetor representando cada função objetivo. O VEGA tem como vantagem ser de fácil implementação, pois requer pequenas mudanças em um simples AG a fim de convertê-lo em um AEMO, sem necessitar de qualquer complexidade computacional. Porém, tem-se como desvantagem o fato de não se obter uma boa diversidade nas soluções da fronteira de *Pareto* (Seção 3.3.1), pois geralmente as soluções tendem a ficar muito próximas da solução ótima.

A manutenção de uma população bem diversificada é de fundamental importância para a eficácia de um AEMO. Face a esta constatação, Goldberg (1989) propôs o conceito de não dominância e explicitou o operador que preserva a diversidade de uma população, eliminando-se assim o problema encontrado no VEGA. No conceito de dominância, insere-se mais cópias de indivíduos não dominados em uma população; já a diversidade de uma população, foi tratada com a estratégia de nichos entre uma classe de soluções não dominadas (Deb, 2001).

O primeiro algoritmo a dar ênfase para o conceito de dominância e a diversidade da população foi o MOGA (*Multi-Objective Genetic Algorithm*), proposto por Fonseca e Fleming (1993). O MOGA difere de um AG clássico pela forma de atribuição dos valores de *fitness* para cada solução em uma população. Para cada solução, associa-se um valor de *ranking*, igual ao número de soluções n_i que a domina mais um, ou seja:

$$r_i = 1 + n_i \quad (3.6)$$

Assim, o valor de *ranking* das soluções não dominadas é igual a um, fazendo com que pelo menos um indivíduo de uma população possua o valor de $r_i = 1$. O valor máximo de r_i não excede o tamanho N da população (Deb, 2001).

Com base nas idéias iniciais propostas por Goldberg (1989), foram então propostos diversos modelos de AEMOs. A Tabela 3.1 apresenta os principais modelos de AEMOs encontrados na literatura e seus respectivos autores.

Tabela 3.1: Diferentes modelos de AEMOs.

Sigla	Nome do Modelo	Autores
VEGA	Vector Evaluated Genetic Algorithm	(Schaffer, 1984)
WBGA	Weight Based Genetic Algorithm	(Hajela e Lin, 1992)
MOGA	Multiple Objective Genetic Algorithm	(Fonseca e Fleming, 1993)
NSGA	Non-Dominated Sorting Genetic Algorithm	(Srinivas e Deb, 1994)
NPGA	Niched-Pareto Genetic Algorithm	(Horn et al., 1994)
NSGA-II	Elitist Non-Dominated Sorting Genetic Algorithm	(Deb et al., 2000)
SPEA, SPEA-2	Strenght Pareto Evolutionary Algorithm 1 e 2	(Zitzler, 1999), (Zitzler et al., 2001)
PAES	Pareto-Archived Evolutionary Strategy	(Knowles e Corne, 1999)
MONGA-I, MONGA-II	Multi-Objective Messy Genetic Algorithm	(Veldhuizen, 1999)
Micro-GA	Multi-Objective Micro-Genetic Algorithm	(Coello e Pulido, 2001)
PESA-I, PESA-II	Pareto Envelope-Base Selection Algorithm	(Corne et al., 2001)
DRMOGA	Divided-Range Multi-Objective Genetic Algorithm	(Hiroyasu et al., 2000)

SNSGA	Steady-State Non-Dominated Sorting Genetic Algorithm	(Gao et al., 2000)
FMOGA	Fuzzy Multi-Objective Genetic Algorithm	(Zhang et al., 2004)
RMOGA	Relational Multi-Objective Genetic Algorithm	(Lee e Tsui, 2004)

O modelo de AEMO utilizado neste trabalho, para realizar a comparação entre AEs aplicados no domínio do futebol de robôs, foi o NSGA-II (*Elitist Non-Dominated Sorting Genetic Algorithm*). Dentre os AEMOs da Tabela 3.1, o NSGA-II é o mais utilizado atualmente na literatura em virtude de suas características. A Seção 3.4.1 descreve o algoritmo NSGA-II.

3.4.1 Algoritmo NSGA-II

O algoritmo NSGA-II não possui semelhanças com o original NSGA, proposto por Srinivas e Deb (1994). Manteve-se o nome simplesmente para destacar o local e origem da criação. O NSGA-II foi proposto por Deb et al. (2000) e trabalha de modo semelhante ao AG clássico, gerando uma população filha Q a partir de uma população pai P . Tanto o tamanho da população P , quanto da população Q , são de tamanho fixo N . Na primeira iteração gera-se a população inicial P_t , ordenando-a por não dominância (Seção 3.3.1). Em seguida, aplica-se os operadores de seleção por torneio (Seção 3.2.4), cruzamento e mutação para se obter a população filha Q_t .

A união da população P_t com a população Q_t se dá por R_t , com $|R| = 2N$. As fronteiras $F_1, F_2, \dots$, são obtidas ordenando R_t por meio de não dominância e inseridas todas na nova população P_{t+1} . Somente N soluções serão inseridas na nova população P_{t+1} , outras N soluções de R_t serão rejeitadas. Todas fronteiras F_i devem ser inseridas em P_{t+1} , enquanto $P_{n+1} + |F_i| \leq N$. Ao inserir um F_j , tal que $|F_j| > N - |P_{n+1}|$, o algoritmo NSGA-II seleciona as soluções de F_j que estejam melhor distribuídas. A Figura 3.3 apresenta o esquema do algoritmo NSGA-II.

Figura 3.3: Modelo esquemático do algoritmo NSGA-II (Deb, 2001).

O NSGA-II tem como base dois princípios, ou sejam:

- Ordenação das soluções não dominadas da população (Quadro 3.2);
- Determinação da métrica definida pelos autores como distância de multidão ou distância de agrupamento (Quadro 3.3).

Seleção de Torneio por Multidão

A ordenação das soluções não dominadas da população baseia-se simplesmente na identificação das soluções não dominadas de uma população a cada iteração, e no modo de armazenar estas em um conjunto de soluções não dominadas de acordo com o nível de dominância.

Definição da Seleção de Torneio por Multidão: é inserido no método de seleção por torneio o operador de comparação $<_c$, para comparar duas soluções. A solução i será considerada melhor que a solução j , se ambas condições 1 e 2 forem verdadeiras, isto é:

- 1) Se a solução i for melhor classificada em relação ao nível de não dominância, $r_i < r_j$;
- 2) Se ambas possuírem o mesmo nível de não dominância, mas i possui uma distância da população maior, $d_i > d_j$.

Quando as iterações são iniciadas, a primeira solução da população é armazenada no conjunto de soluções não dominadas e as soluções restantes são avaliadas. Se uma solução dominar alguma solução que esteja armazenada no conjunto de soluções não dominadas, estas soluções dominadas são retiradas do conjunto de soluções dominantes e repete-se este procedimento até que todas as soluções da população sejam avaliadas. Desta forma, determina-se a frente de nível 1. Esse procedimento repete-se até que todas as frentes com diversos níveis de dominância sejam montadas, ordenando-as assim, todas as soluções da população com base no conceito de dominância. O Quadro 3.2 apresenta o pseudo-algoritmo do NSGA-II proposto por Deb et al. (2000).

Quadro 3.2 Pseudo-código do NSGA-II.

ALGORITMO NSGA-II

Variáveis:

P	(população pai)
Q	(população filha)
N	(tamanho fixo da população P e Q)
F_j	(conjunto de soluções na fronteira j)
$nMax$	(número máximo de gerações)
n	(número da geração atual)

1. Gerar a população inicial P_0 e $Q_0 = \emptyset$.
 2. Atribuir $n = 0$.
 3. Realizar a seleção, o cruzamento e mutação para gerar a população filha Q_0 .
 4. Fazer $R_n = P_n \cup Q_n$.
 5. Realizar a ordenação por não dominância em R_n .
 6. Criar $P_{n+1} = \emptyset$.
 7. ENQUANTO $|P_{n+1} + F_j| \leq N$ FAÇA.
 - a) Copiar as soluções de F_j em P_{n+1} .
 8. Calcular as distância de multidão em F_j .
 9. Ordenar F_j de acordo com as distâncias d_j .
 10. Copiar as primeiras $N - |P_{n+1}|$ soluções de F_j para P_{n+1} .
 11. Aplicar a seleção de torneio por multidão para os indivíduos de P_{n+1} .
 12. Aplicar os operadores de cruzamento e de mutação para gerar a nova população Q_{n+1} .
 13. SE $n > nMax$ ENTÃO parar
 14. SENÃO atribuir $n = n + 1$ e retornar para o passo 4.
-

Distância de Multidão

A distância de agrupamento ou distância de multidão (*crowding distance*) fornece uma estimativa da densidade de soluções próximas de uma solução i de uma população. Primeiramente, mede-se a distância média entre as duas soluções para ambos os lados da solução i , ao longo de cada um dos objetivos. Em seguida, todas as soluções extremas⁶ recebem um valor infinito de distância. Para todas as outras soluções intermediárias atribui-se um valor de distância, que corresponde à diferença do valor da função objetivo do restante das soluções. Este procedimento se repete para todas as funções objetivos, sendo que o valor da distância de agrupamento é a soma das distâncias individuais para cada objetivo.

Quadro 3.3 Pseudo-código da distância de multidão no NSGA-II.

ALGORITMO Distância de Multidão

Variáveis:

F_j (conjunto de soluções na fronteira i)

1. l denota o número de soluções em F_j .
2. PARA cada solução em F_j atribui-se $d_i = 0$ FAÇA.
3. PARA cada função objetivo $m = 1, 2, \dots, M$ FAÇA.
 - a) Ordenar de forma decrescente as soluções por f_m na lista I^m
4. PARA cada solução extrema (mínimo e máximo) em cada um dos M objetivos FAÇA
 - a) Fazer $d_{I_1^m} = d_{I_l^m} = \infty$
5. PARA as soluções $i = 2, \dots, l - 1$ calcular:

$$d_{I_i^m} = d_{I_i^m} + \frac{f_m^{I_{i+1}^m} - f_m^{I_{i-1}^m}}{f_m^{max} - f_m^{min}} \quad (3.7)$$

Este algoritmo é descrito no Quadro 3.3, onde o índice I_j representa o índice da i -ésima solução na lista ordenada pelo objetivo m . Para qualquer objetivo, I_1 e I_l são os elementos da lista com o menor e o maior m , respectivamente. $f_m^{I_{i+1}^m}$ e $f_m^{I_{i-1}^m}$ são os valores dos vizinhos de i na m -ésima função objetivo. Os f_m^{max} e f_m^{min} são parâmetros dos valores da população máxima e da população mínima da m -ésima função objetivo.

⁶Soluções testadas com valores maiores ou menores da função objetivo.

A Equação (3.7) garante que as soluções mais afastadas tenham d_i maior do que aquelas mais próximas.

3.5 Diferenças Fundamentais entre Otimização Mono-Objetivo e Multi-Objetivo

Nesta seção ilustra-se a diferença fundamental entre otimização mono e multi-objetivo por meio de um problema de otimização com dois objetivos. Para dois objetivos conflitantes, cada objetivo corresponde a uma solução ótima diferente. No problema de decisão da Figura 3.2, as soluções A e C são soluções ótimas. Se for assumir em sacrificar a função g para estender um pouco até a solução A , é possível encontrar uma outra solução em relação à função g que seja melhor. Idealmente, o aumento do sacrifício em relação à função g é relacionado com o ganho em relação à função f . Assim, pode-se visualizar um conjunto de soluções ótimas (semelhantes às soluções A , B , C , D e E), onde um ganho em um objetivo significa sacrificar o outro objetivo.

Com todas estas soluções conflitantes, pode-se dizer que uma solução é a melhor no que diz respeito a ambos os objetivos? A verdade reside no fato de que nenhuma solução deste conjunto, para ambos os objetivos (função objetivo f e função objetivo g), parece ser melhor do que qualquer outra solução do conjunto. Assim, em problemas de otimização que possuem mais de um objetivo conflitante, não existe uma única solução ótima. Existe um conjunto de soluções que são todas soluções ótimas. Sem possuir alguma informação adicional, nenhuma solução a partir do conjunto de soluções ótimas pode ser dita como sendo melhor do que qualquer outra solução. Uma vez que são um conjunto de soluções ótimas, em um POMO, muitas dessas soluções ótimas são importantes. Esta é a diferença entre uma otimização de problemas mono-objetivo e multi-objetivo. O importante em uma otimização mono-objetivo é uma única solução ótima, enquanto que, na otimização multi-objetivo, uma série de soluções ótimas resultantes entre os objetivos conflitantes são importantes (Deb, 2001).

A outra diferença entre multi-objetivo e mono-objetivo é o espaço de busca: no multi-objetivo é multi-dimensional, sendo que cada solução x , no espaço de decisão, possui $f(x)$

em Z ; e no mono-objetivo, é unidimensional.

3.6 Considerações Parciais

Neste capítulo foram abordados os conceitos dos AEs, explicitando-se as características básicas de um AG clássico mono-objetivo (Seção 3.2). Também foi abordado o conceito da otimização multi-objetivo (Seção 3.3) e seu respectivo algoritmo (Seção 3.4). Foi detalhado ainda o algoritmo NSGA-II (Seção 3.4.1). A Seção 3.5 deixa claro que a diferença fundamental entre multi-objetivo e mono-objetivo é o espaço de busca: o multi-objetivo é multi-dimensional, onde cada solução x , no espaço de decisão, possui $f(x)$ em Z ; e o mono-objetivo é unidimensional. Este trabalho utiliza o AG clássico e o NSGA-II como estratégia evolutiva para selecionar as ações aplicadas em um time de futebol de robôs (Capítulo 4), a fim de medir o desempenho de ambos os métodos.

Capítulo 4

Formulação do Time para Abordagem Mono-Objetivo e Multi-Objetivo

Neste trabalho foram utilizados dois métodos evolutivos para se obter a estratégia para os jogadores, sendo estes eficientes para o propósito de se jogar futebol. O primeiro método trata de uma abordagem Mono-Objetivo, onde cada indivíduo será avaliado apenas em relação a uma função objetivo. Já a abordagem Multi-Objetivo avalia cada indivíduo em relação a duas funções objetivo. Cada método evolutivo foi aplicado em um time de futebol de robôs simulados, possibilitando, assim, a comparação dos métodos em relação à eficiência para se jogar futebol.

O foco é tentar encontrar o melhor conjunto de ações para os dez jogadores. Cada jogador possui seu próprio conjunto de ações que será utilizado quando estiver de posse da bola. Toda estratégia está armazenada dentro de um vetor de números inteiros. Ressalta-se que o foco não é otimizar um jogador unicamente e sim o time como um todo.

As características do desenvolvimento do time estão descritas na Seção 4.1. A estrutura de codificação dos indivíduos está representada na Seção 4.2, sendo que as características dos AG estão explanadas na Seção 4.3. Os algoritmos dos métodos evolucionários serão apresentados e descritos na Seção 4.4 e Seção 4.5.

4.1 Características do Time Desenvolvido

Nesse trabalho foi utilizado o time *UvA Trilearn* como base para aplicar os métodos evolutivos computacionais. O código fonte do time *UvA Trilearn* contém apenas a implementação do comportamento de baixo nível, como por exemplo, as informações trocadas entre o servidor e os jogadores. O comportamento de alto nível, como a estratégia do time, foi omitida no código fonte.

O time de futebol de robôs é composto por dez jogadores, um goleiro e um técnico. Os dez jogadores utilizam-se da divisão da área de atuação, para que cada qual possa se comportar de acordo com a posição estratégica definida durante uma partida. A partida de futebol de robôs é disputada em um campo virtual retangular por dois times, de onze jogadores cada, que têm como objetivo colocar a bola dentro do gol adversário, o que é chamado de gol. Não é permitido o uso do comando para manipular a bola com as mãos, exceto pelos goleiros.

Figura 4.1: Representação do campo virtual, dos jogadores e da bola.

Os jogadores foram divididos em três diferentes posições, atuando-se assim de maneira diferente dentro de campo. O jogador que desempenha a função de zagueiro atua na área do campo conhecida como campo de defesa, sendo que tem como principal função marcar

os jogadores atacantes do time adversário, evitando-se que estes criem a possibilidade de gol. O jogador que desempenha a função de meio-campista atua de forma a intermediar os zagueiros aos atacantes, jogando tanto no campo de defesa quanto no campo de ataque. O jogador que desempenha a função de atacante atua na área de ataque desempenhando função ofensiva, com objetivo de fazer gols ou dar assistência para seus companheiros de time.

O time proposto foi dividido da seguinte forma: um goleiro, quatro zagueiros, três meio-campistas e três atacantes. A Figura 4.1 apresenta o campo virtual onde uma partida de futebol de robôs é exibida, assim como os jogadores e a bola.

Foi então proposta a criação de dez ações, que correspondem às jogadas previamente codificadas para serem utilizadas pelos jogadores durante uma partida. As ações serão realizadas assim que for executada a parte do código da *UvA Trilearn*, onde há a condição de verificação, no qual avalia se o jogador está ou não de posse da bola (Figura 2.11). Essas ações são sintetizadas por:

1. Realizar um passe com velocidade média para o companheiro número seis;
2. Realizar um passe com velocidade média para o companheiro número sete;
3. Realizar um passe com velocidade média para o companheiro número oito;
4. Levar a bola para uma área sem oponentes. Esta ação será usada quando não conseguir driblar ou realizar o passe para o companheiro;
5. Chutar em direção ao gol adversário;
6. Driblar em direção à linha lateral com um ângulo positivo;
7. Driblar em direção à linha lateral com um ângulo negativo;
8. Realizar um passe com velocidade alta para o companheiro número nove;
9. Realizar um passe com velocidade alta para o companheiro número dez;
10. Realizar um passe com velocidade alta para o companheiro número onze.

As jogadas irão compor a estrutura de um jogador, sendo inseridas dentro de cada posição de um vetor de inteiros, o qual representa a estrutura cromossômica de um jogador (Seção 4.2).

4.2 Codificação dos Indivíduos

Para a codificação dos indivíduos (jogadores), foi necessário a criação de um vetor de inteiros. Cada jogador é representado por um vetor de inteiros de dez posições. Em cada posição deste vetor será inserida uma das dez possíveis ações propostas (Seção 4.1), as quais são inseridas aleatoriamente de acordo com a geração dos jogadores no AG.

Assim que o AG é iniciado, tanto no AG mono-objetivo, como no AG multi-objetivo, são gerados dez jogadores. Em cada uma das dez posições do vetor de inteiros de um jogador é inserido o valor aleatório no intervalo de $[1,10]$, de acordo com as dez possíveis ações descritas na Seção 4.1. Um time formado por dez jogadores¹ contém então dez vetores de inteiros de dez posições cada.

A Figura 4.2 representa os jogadores gerados e suas respectivas ações. Cada jogador possui um conjunto de ações de 1 a 10. Gera-se então de 1 a 10 jogadores para cada geração.

Ação 1	Ação 2	Ação 3	Ação 4	...	Ação 10	
8	6	3	4	...	2	Jogador 1
5	7	10	2	...	9	Jogador 2
...						
3	1	9	6	...	5	Jogador 10

Figura 4.2: Representação dos jogadores.

¹O goleiro foi excluído, pois apresenta características de ações diferente dos outros jogadores, como a ação de manuseiar a bola com as mãos.

4.3 Características do AG

O algoritmo genético foi aplicado somente nos dez jogadores, sendo que o goleiro foi excluído. Considerando o total de ações por jogador, utilizaram-se então os dez vetores de inteiros para representar o time.

O algoritmo genético foi configurado com as seguintes características:

- Processo de avaliação exclusivo para cada método evolutivo: na abordagem Mono-Objetivo foi utilizada apenas uma função objetivo, sendo que na abordagem Multi-Objetivo foram utilizadas duas funções objetivo;
- Cruzamento usando apenas um ponto em 60% dos indivíduos;
- Mutação em apenas 1% dos indivíduos;
- Método de seleção do tipo elitista utilizado na abordagem Mono-Objetivo, e o método baseado na dominância de *Pareto* utilizado na abordagem Multi-Objetivo.

Na operação de cruzamento foi selecionada 60% da população pelo método do elitismo, respeitando-se os indivíduos com melhores valores da função de avaliação, para ambos os métodos evolutivos. Tal método elenca os indivíduos de acordo com o valor da função de avaliação, do maior valor para o menor valor desta função, mantendo-se os indivíduos mais aptos na população.

Depois de se selecionar 60% dos indivíduos, escolhe-se aleatoriamente um ponto de secção para realizar a operação de cruzamento, em que serão trocados as duas metades dos dois vetores selecionados anteriormente. Dois indivíduos (pais) geram, a partir do método de cruzamento, dois novos indivíduos (filhos).

Para a operação de mutação, um indivíduo é escolhido aleatoriamente. Em seguida, escolhe-se um ponto dentro do seu vetor de inteiros para realizar a mutação, obedecendo-se os seguintes critérios:

- Se o indivíduo for um zagueiro, priorizar então as ações 1, 2, 3, 4, 5, 6 e 7;
- Se o indivíduo for do tipo meio-campo, priorizar então as ações 4, 5, 6, 7, 8, 9 e 10;

- Se o indivíduo for um atacante, priorizar então as ações 4, 5, 6 e 7.

O sistema desenvolvido funciona de acordo com a Figura 4.3. Primeiramente, geram-se de maneira aleatória dez indivíduos (jogadores), os quais são escritos dentro de um arquivo texto (indiv.txt). O módulo Gera Indivíduos é executado apenas uma vez na primeira geração, sendo que nas gerações subsequentes os indivíduos serão gerados a partir das operações genéticas realizadas pelo módulo Algoritmo Genético. Os dez indivíduos gerados apresentam a estrutura representada pela Figura 4.2. Para a primeira geração, não é realizada nenhuma operação genética, pois os indivíduos são lidos do arquivo texto (indiv.txt) pelo módulo Time e os colocam para disputar uma partida no simulador de futebol.

Figura 4.3: Fluxo de dados dentro do sistema desenvolvido.

Logo após o término da partida, o simulador retorna o placar do jogo, sendo que o mesmo é escrito pelo módulo Time no arquivo texto (placar.txt). Em seguida, o módulo Algoritmo Genético lê o placar desse arquivo e inicia os operadores genéticos que irão avaliar, cruzar e mutar os indivíduos. O módulo Algoritmo Genético é composto por três diferentes AGs, cada um responsável pelo conjunto de jogadores de acordo com suas respectivas áreas de atuações. O que diferencia um AG de outro é a função objetivo e o operador genético de mutação. Em seguida, os indivíduos são escritos dentro do arquivo

texto (indiv.txt), em que o processo se reinicia até que o número de gerações seja igual ao valor definido.

O módulo AG, quando utilizado na abordagem Mono-Objetivo, faz uso de funções objetivo descritas pelas Equações (4.1), (4.2) e (4.3), conforme apresentadas na Seção 4.4. Na abordagem Multi-Objetivo, utiliza-se as funções objetivo dadas pelas Equações (4.4), (4.5) e (4.6), descritas na Seção 4.5.

4.4 Abordagem Mono-Objetivo

Para o algoritmo Mono-Objetivo foi proposto o método de avaliação baseado na quantidade de gols prós (GP) e gols contra (GC), em relação à área de atuação do jogador em campo (zagueiro, meio-campo ou atacante). O jogador zagueiro recebe os valores de recompensa (GP_{ZG}) e de punição (GC_{ZG}), o jogador meio-campo recebe os valores de recompensa (GP_{MC}) e de punição (GC_{MC}) e o jogador atacante recebe os valores de recompensa (GP_{AT}) e de punição (GC_{AT}).

Os valores de recompensa e punição, para cada tipo de jogador, foram pré-estipulados para auxiliar no cálculo da função objetivo. Foi proposto o cálculo da função objetivo para os jogadores zagueiros, meios-campos e atacantes da seguinte maneira:

- Para o jogador zagueiro, utiliza-se a função FIT_{ZG} :

$$FIT_{ZG} = ((GP_{ZG} * GP) + (GC_{ZG} * GC)) \quad (4.1)$$

- Para o jogador meio-campo, utiliza-se a função FIT_{MC} :

$$FIT_{MC} = ((GP_{MC} * GP) + (GC_{MC} * GC)) \quad (4.2)$$

- Para o jogador atacante, utiliza-se a função FIT_{AT} :

$$FIT_{AT} = ((GP_{AT} * GP) + (GC_{AT} * GC)) \quad (4.3)$$

O método de avaliação prioriza a quantidade de gols prós em relação à quantidade de gols contra. Quando a partida é finalizada, dois valores são então retornados para a função de avaliação, isto é, a quantidade de gols prós e a quantidade de gols contra. A função de avaliação recebe o placar da partida e pontua cada indivíduo de acordo com sua posição tática. Em seguida, realiza-se o cruzamento e a mutação dos indivíduos selecionados de acordo com suas respectivas pontuações.

Figura 4.4: Fluxograma do algoritmo genético mono-objetivo.

A Figura 4.4 apresenta o fluxograma do algoritmo genético proposto. Primeiramente, dez indivíduos são gerados aleatoriamente. Então, esse time gerado é colocado para disputar uma partida de futebol contra os times que estarão descritos na Seção 4.6. Após o término da partida, o referido time é avaliado de acordo com o placar resultante.

4.5 Abordagem Multi-Objetivo

Para o AGMO foi proposto o método de avaliação baseado também na quantidade de gols prós (GP) e gols contra (GC), em relação à área de atuação do jogador em campo

(zagueiro, meio-campo ou atacante). A diferença em relação ao método Mono-Objetivo se dá principalmente pelo fato de serem utilizadas duas funções objetivo, uma para avaliar a quantidade de gols prós (GP) e outra para avaliar a quantidade de gols contra (GC), de forma distinta.

O método proposto tem como objetivo principal aumentar a quantidade de GP ao mesmo tempo que diminui a quantidade de GC .

Como no método Mono-Objetivo, o jogador zagueiro recebe os valores de recompensa (GP_{ZG}) e de punição (GC_{ZG}), o jogador meio-campo recebe os valores de recompensa (GP_{MC}) e de punição (GC_{MC}) e o jogador atacante recebe os valores de recompensa (GP_{AT}) e de punição (GC_{AT}).

Para auxiliar o cálculo das funções objetivo, foram propostos valores de recompensa e punição, para cada tipo de jogador (zagueiros, meios-campos e atacantes), da seguinte maneira:

- Para o jogador zagueiro, utilizam-se duas funções objetivo. A $FIT_{ZG_{GP}}$ avalia os gols prós e a $FIT_{ZG_{GC}}$ avalia os gols contra:

$$\begin{aligned} FIT_{ZG_{GP}} &= (GP_{ZG} * GP) \\ FIT_{ZG_{GC}} &= (GC_{ZG} * GC) \end{aligned} \tag{4.4}$$

- Para o jogador meio-campo, utilizam-se duas funções objetivo. A $FIT_{MC_{GP}}$ avalia os gols prós e a $FIT_{MC_{GC}}$ avalia os gols contra:

$$\begin{aligned} FIT_{MC_{GP}} &= (GP_{MC} * GP) \\ FIT_{MC_{GC}} &= (GC_{MC} * GC) \end{aligned} \tag{4.5}$$

- Para o jogador atacante, utilizam-se duas funções objetivo. A $FIT_{AT_{GP}}$ avalia os gols prós e a $FIT_{AT_{GC}}$ avalia os gols contra:

$$\begin{aligned} FIT_{AT_{GP}} &= (GP_{AT} * GP) \\ FIT_{AT_{GC}} &= (GC_{AT} * GC) \end{aligned} \tag{4.6}$$

O método de avaliação prioriza a quantidade de gols prós em relação à quantidade de gols contra. Quando a partida é finalizada, dois valores são então retornados para a função de avaliação, isto é, a quantidade de gols prós e a quantidade de gols contra. A função de avaliação recebe o placar da partida e pontua cada indivíduo de acordo com sua posição tática. Cada tipo de jogador terá dois valores atribuídos pela função objetivo, um valor para gols prós (FIT_{GP}) e outro com o valor de gols contra (FIT_{GC}). Em seguida, ocorre a reprodução do tipo elitista, na qual se seleciona os indivíduos para a próxima geração de acordo com o modelo esquemático do algoritmo NSGA-II (Figura 3.3). Por último, realiza-se o cruzamento e a mutação dos indivíduos selecionados de acordo com suas respectivas pontuações.

Figura 4.5: Fluxograma do algoritmo genético multi-objetivo.

A Figura 4.5 apresenta o fluxograma do algoritmo genético proposto. Primeiramente, dez indivíduos são gerados aleatoriamente. Então, esse time gerado é colocado para disputar uma partida de futebol contra os times que estarão descritos na próxima seção.

Após o término da partida, o referido time é avaliado de acordo com o placar resultante.

4.6 Resultados Obtidos

Nesta seção são apresentados os resultados obtidos, no domínio do futebol de robôs simulados, utilizando ambas abordagens evolutivas. Foram realizados 30 experimentos, utilizando a abordagem mono-objetivo e a abordagem multi-objetivo. Para avaliar a eficiência para ambas abordagens genéticas, foram escolhidos os três melhores times que participaram das competições mundiais da *RoboCup* de 2007 à 2009, são eles:

- *BRAINSTORMERS* - Esse time foi desenvolvido por Martin Reidmiller em conjunto com Thomas Gabel, no Instituto de Ciência Cognitiva, pertencente a Universidade de Osnabruek, Alemanha. O time utiliza técnicas conhecidas de aprendizado de máquina, particularmente o método de aprendizado por reforço. Foram campeões da categoria simulação 2D da *RoboCup* em 2007 e 2008, respectivamente realizadas em Atlanta, Estados Unidos da América e Suzhou, China. Durante a última *RoboCup* 2009, realizada em Graz na Áustria, obtiveram a quarta colocação (Gabel e Riedmiller, 2009).
- *WRIGHTEAGLE* - O time foi desenvolvido por Ke Shi, Aijun Bai, Yunfang Tai e Xiaoping Chen, no Laboratório de Sistemas Multi-Agentes, na Escola de Ciência da Computação e Tecnologia da Universidade de Ciência da Computação, China. Esse time tem como objetivo, a longo prazo, realizar uma pesquisa utilizando planejamento de decisão teórica e outros projetos em inteligência artificial. Desenvolveram uma estrutura nova do time, baseado na teoria da observação parcial estocástica de um jogo, como também desenvolveram novas técnicas de baixo e alto nível. Obtiveram na *RoboCup*, em 2007 e 2008, o segundo lugar na categoria simulação 2D. Na *RoboCup* em 2009 foram campeões (Shi et al., 2009).
- *HELIOS* - Esse time foi desenvolvido por Hidehisa Akiyama, Hiroki Shimora e Itsuki Noda, na Universidade de Tokyo, Japão. Utilizam técnicas que aplicam o conhecimento humano dentro da tomada de decisões de um agente. Dentre essas

técnicas, destacam-se o mecanismo de posicionamento que utiliza a triangularização *Delaunay*. Durante a *RoboCup*, em 2007 e 2008, obtiveram o terceiro lugar e foram vice-campeões na *RoboCup* em 2009 na categoria simulação 2D (Akiyama et al., 2009).

Em ambas abordagens evolutivas utilizou-se pesos no processo de avaliação para auxiliar a convergência dos AG's. Não foi encontrado na literatura nenhum estudo relacionado a escolha dos pesos atribuídos aos valores prós e valores contra da função de avaliação, portanto, foram testados diversos valores até se chegar em uma pontuação que fosse eficiente, auxiliando-se assim, no processo de convergência do algoritmo. Foram atribuídos os pesos da seguinte maneira:

- Para a função de avaliação dos zagueiros, foram atribuídos os valores de:

$$\begin{aligned}GP_{ZG} &= 60; \\ GC_{ZG} &= -55;\end{aligned}\tag{4.7}$$

- Para a função de avaliação do meio-campo, foram atribuídos os valores de:

$$\begin{aligned}GP_{MC} &= 80; \\ GC_{MC} &= -60;\end{aligned}\tag{4.8}$$

- Para a função de avaliação dos atacantes, foram atribuídos os valores de:

$$\begin{aligned}GP_{AT} &= 100; \\ GC_{AT} &= -70.\end{aligned}\tag{4.9}$$

O problema mais crítico foi em relação ao tempo total de duração das simulações, pois cada partida tem duração de aproximadamente dez minutos (Seção 2.2). Seria então inviável realizar trinta testes, simulando as duas mil partidas. A Tabela 4.1 ilustra detalhadamente o problema em questão.

Tabela 4.1: Tempo total da simulação, sem configurar o servidor.

	Tempo da Partida	Quant. de Testes	Quant. de Gerações	Tempo Total
Mono-Objetivo	10 min	30	2000	417 dias
Multi-Objetivo	10 min	30	100	21 dias

O servidor apresenta diversos recursos, os quais podem ser habilitados ou não. Para solucionar o problema do tempo elevado de simulação, foi habilitado o recurso de acelerar o tempo de execução de uma partida, fazendo com que pudesse ser concluída em no máximo um minuto. O tempo resultante após a configuração do servidor está presente na Tabela 4.2.

Tabela 4.2: Tempo total da simulação, com o servidor configurado.

	Tempo da Partida	Quant. de Testes	Quant. de Gerações	Tempo Total
Mono-Objetivo	1 min	30	2000	41 dias
Multi-Objetivo	1 min	30	100	2 dias

4.6.1 Resultado do Algoritmo Genético Mono-Objetivo

Neste experimento foram feitos trinta testes com o time que possui a abordagem genética mono-objetivo. Cada experimento corresponde a duas mil partidas contra cada um dos times escolhidos, totalizando cento e oitenta mil partidas.

A Figura 4.6 ilustra a média das curvas de evolução realizada por duas mil partidas contra o time *BRAINSTORMERS*. Nota-se que, por volta da quadringentésima geração, a quantidade de gols prós começa a aumentar, atingindo o valor máximo de dezessete gols por volta da milésima geração. Nota-se também que a quantidade de gols sofridos permanece praticamente zerada.

Figura 4.6: Resultado de duas mil gerações, AG Mono-Objetivo contra *BRAINSTORMERS*.

A Figura 4.7 ilustra a média das curvas de evolução realizada por duas mil partidas contra o time *WRIGHTEAGLE*. Nota-se que por volta da tricentésima geração a quantidade de gols contra começa a diminuir, atingindo o valor máximo de dezessete gols, por volta da geração mil e seiscentos. Nota-se também que a quantidade de gols prós mantém-se praticamente zerada.

Figura 4.7: Resultado de duas mil gerações, AG Mono-Objetivo contra *WRIGHTEAGLE*.

A Figura 4.8 ilustra a média das curvas de evolução realizada por duas mil partidas contra o time *HELIOS*. Nota-se que por volta da ducentésima geração a quantidade de gols contra começa a diminuir, atingindo o valor máximo de doze gols por volta da geração mil e cem. Nota-se neste caso que a quantidade de gols prós apresenta significativa melhora após a quadringentésima geração, atingindo o valor máximo de um gol.

Figura 4.8: Resultado de duas mil gerações, AG Mono-Objetivo contra *HELIOS*.

Na Tabela 4.3 ilustra-se de maneira geral os resultados obtidos aplicando a abordagem genética mono-objetivo contra os respectivos adversários, descritos acima.

Tabela 4.3: Média dos resultados obtidos da abordagem genética mono-objetivo.

	BRAINSTORMERS	WRIGHTEAGLE	HELIOS	NÚMERO DE GERAÇÕES
Mono-Objetivo	5 X 0	0 X 21	0 X 29	1
Mono-Objetivo	17 X 0	0 X 17	1 X 12	2000

4.6.2 Resultado do Algoritmo Genético Multi-Objetivo

Neste experimento foram feitos trinta testes com o time que possui a abordagem genética multi-objetivo. Cada experimento corresponde a cem partidas contra cada um

dos times escolhidos, totalizando nove mil partidas.

A Figura 4.9 ilustra a média das curvas de evolução realizada por cem partidas contra o time *BRAINSTORMERS*. Nota-se que desde as primeiras gerações a quantidade de gols prós começa a aumentar, atingindo o valor máximo de dezenove gols, por volta da octogésima geração. Nota-se também que a quantidade de gols sofridos permanece praticamente zerada.

Figura 4.9: Resultado de cem gerações, AG Multi-Objetivo contra *BRAINSTORMERS*.

A Figura 4.10 ilustra a média das curvas de evolução realizada por cem partidas contra o time *WRIGHTTEAGLE*. Nota-se que desde as primeiras gerações a quantidade de gols contra começa a diminuir, atingindo o valor máximo de cinco gols, por volta da octogésima geração. Nota-se neste caso que a quantidade de gols prós apresenta significativa melhora após a décima quinta geração, atingindo o valor máximo de um gol.

Figura 4.10: Resultado de cem gerações, AG Multi-Objetivo contra *WRIGHTEAGLE*.

A Figura 4.11 ilustra a média das curvas de evolução realizada por cem partidas contra o time *HELIOS*. Nota-se que desde as primeiras gerações a quantidade de gols contra começa a diminuir, atinge o valor máximo de dois gols, por volta da septuagésima geração. Nota-se neste caso que a quantidade de gols prós, após a trigésima geração, atingindo o valor máximo de três gols. Sendo este o caso mais interessante em relação aos resultados obtidos durante os experimentos.

Figura 4.11: Resultado de cem gerações, AG Multi-Objetivo contra *HELIOS*.

Na Tabela 4.4 ilustra-se de maneira geral os resultados obtidos aplicando a abordagem genética multi-objetivo contra os respectivos adversários, descritos acima.

Tabela 4.4: Média dos resultados obtidos da abordagem genética multi-objetivo.

	BRAINSTORMERS	WRIGHTEAGLE	HELIOS	NÚMERO DE GERAÇÕES
Multi-Objetivo	6 X 0	0 X 10	0 X 7	1
Multi-Objetivo	19 X 0	1 X 5	3 X 2	100

4.7 Considerações Parciais

Neste capítulo foram abordados as características do time (Seção 4.1), o modelo de codificação dos indivíduos (Seção 4.2) e as características do AG (Seção 4.3). Também foram apresentados as abordagens Mono-Objetivo e Multi-Objetivo, explicitando cada uma delas nas respectivas Seções 4.4 e 4.5 em conjunto com seus respectivos resultados apresentados nas Subseções 4.6.1 e 4.6.2.

Capítulo 5

Conclusões Gerais

Neste trabalho foi realizado uma análise comparativa dos AGs aplicados no domínio do futebol de robôs. O primeiro é o AG mono-objetivo e o segundo é o AG multi-objetivo.

No AG multi-objetivo foi utilizado o algoritmo NSGA-II para selecionar os indivíduos por meio do conceito de dominância de *Pareto*.

Em Fraccaroli e Silva (2009b) foi aplicado um algoritmo genético para definir ações no domínio do futebol de robôs. Entretanto, não foi utilizado o método de atribuição de pesos, e com isso notou-se a convergência tardia do AG. O método de atribuição de pesos no processo de avaliação foi proposto para auxiliar o tempo convergência dos AG's.

Foram realizados testes e foi utilizado o método de atribuição de pesos para acelerar a convergência dos AG's. Adotando-se este, foram necessários aproximadamente mil gerações para a convergência do AG mono-objetivo e oitenta gerações para a convergência do AG multi-objetivo.

Foi provado que o AG multi-objetivo, por utilizar-se de mais de um objetivo, percorre melhor o espaço de busca, auxiliando no tempo de convergência; porém, dependerá do tipo de modelagem realizado para o determinado time.

Uma outra contribuição importante do trabalho foi o desenvolvimento de uma estratégia baseada em AG aplicado a um time de futebol de robôs simulados, constituído de onze robôs simulados, o qual poderá ser utilizado em competições nacionais e internacionais de futebol de robôs simulados. Este ambiente fornece um ambiente muito rico de temas

de pesquisas a serem investigadas.

Os dois times implementados, um com AG mono-objetivo, outro com AG multi-objetivo foram criados utilizando o time base *UvA Trilearn*. A partir desses times criados, é possível que outros pesquisadores que desejam trabalhar com AE's aplicados no ambiente de futebol de robôs simulados possam utilizar seus métodos sem ter que implementar um time inteiro.

O desenvolvimento deste trabalho possibilitou a publicação de dois artigos durante o ano de 2009, (Fraccaroli e Silva, 2009a) e (Fraccaroli e Silva, 2009b). Também foi possível a conquista do campeonato de futebol de robôs simulados, realizado em Brasília, durante o Simpósio Brasileiro de Automação Inteligente, SBAI, em novembro de 2009.

Como direções futuras deste trabalho é interessante realizar experimentos com a utilização de outra modelagem, aplicando-se diversas outras possíveis ações. Outro ponto a ser investigado seria a aplicação de um sistema *Fuzzy* para definir a formação tática, a partir de diversas variáveis de entrada, tais como: posição da bola; tempo decorrido da partida; tempo de posse de bola e placar. Finalmente, seria ainda interessante realizar experimentos com um AG adaptativo, possibilitando que ajustes automáticos nas ações dos times sejam realizados no decorrer de uma partida.

Referências Bibliográficas

- Aarts, E. e J. Korst (1989). *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley and Sons.
- Akiyama, H., H. Shimora, e I. Noda (2009). Oficial site of helios, team description paper. In *Disponível em <http://romeo.ist.tugraz.at/robocup2009/tdps/helios-rc09-tdp.pdf>*. Visited in 12/08/2009.
- Boer, R. e J. Kok (2002). The incremental development of a synthetic multi-agent system: The uva trilearn 2001 robotic soccer simulation team. Dissertação de Mestrado, University of Amsterdam.
- Chen, M., E. Foroughi, F. Heintz, Z. Huang, S. Kapetanakis, K. Kostiadis, J. Kummeneje, I. Noda, O. Obst, P. Riley, T. Steffens, Y. Wang, e X. Yin (2002). *RoboCup Soccer Server*. Disponível em <http://sourceforge.net/projects/sserver>.
- Cheung, W., A. Kentish, e A. Law (2005). Sabotage fc: Implementing a simulated robot football team using evolutionary algorithms. In *CO600 Group Project Reports 2004/05*.
- Coello, C. e G. Pulido (2001). Multiobjective optimization using a micro-genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*. Morgan Kaufmann Publishers.
- Coello, C. A. C., D. A. V. Veldhuizen, e G. B. Lamont (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic/Plenum Publishers.
- Corne, D., N. Jerram, J. Knowles, e M. Oates (2001). Pesa-ii: Region-based selection in evolutionary multiobjective optimization. In L. Spector, E. Goodman, A. Wu, W. Langdon, H. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, e E. Burke (Eds.),

- Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pp. 283–290. Morgan Kaufmann Publishers.
- Darwin, C. (1859). *On the Origin of Species By Means of Natural Selection*.
- Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons.
- Deb, K., S. Agrawal, A. Pratab, e T. Meyarivan (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. KanGAL report 200001, Indian Institute of Technology, Kanpur, India.
- Fonseca, C. e P. Fleming (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, California, pp. 416–423. University of Illinois at Urbana-Champaign.
- Fraccaroli, E. S. e I. N. Silva (2009a). Abordagem fuzzy para determinação de estratégia em ambiente de futebol de robôs simulado. In *Brazilian Conference on Dynamics, Control and Applications*, Volume 8, pp. 1–6.
- Fraccaroli, E. S. e I. N. Silva (2009b). Abordagem genética para definição de ações em ambiente simulado de futebol de robôs. In *Simpósio Brasileiro de Automação Inteligente*. CD-ROM // paper number: 55708 // 06 páginas.
- Francis, K. A. (2007). *Charles Darwin and The origin of species*. Greenwood Press.
- Gabel, T. e M. Riedmiller (2009). Oficial site of brainstormers, team description paper. In *Disponível em <http://romeo.ist.tugraz.at/robocup2009/tdps/brainstormers-rc09-tdp.pdf>*. Visited in 12/08/2009.
- Gao, Y., L. Shi, e P. Yao (2000). Study on multi-objective genetic algorithm. In *Intelligent Control and Automation*, Volume 1, pp. 646–650.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley Publishing Company, Inc.

-
- Hajela, P. e C. Y. Lin (1992). Genetic search strategies in multicriterion optimal design. *Structural and Multidisciplinary Optimization* 4, 99–107.
- Hiroyasu, T., M. Miki, e S. Watanabe (2000). The new model of parallel genetic algorithm in multi-objective optimization problems - divided range multi-objective genetic algorithm. In *Evolutionary Computation*, Volume 1, pp. 333–340.
- Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.
- Horn, J., N. Nafpliotis, e D. Goldberg (1994). A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, Volume 1, pp. 82–87. IEEE Service Center.
- Jeong, I. K. e J. J. Lee (1997). Evolving multi-agents using a self-organization algorithm. Volume 88, pp. 293–303. Applied Mathematics and Computing.
- Kitano, H., M. Asada, Y. Kuniyoshi, I. Noda, e E. Osawa (1995). Robocup: The robot world cup initiative. In *IJCAI-95 Workshop on Entertainment and AI/Alife*, Volume 1, pp. 532.
- Kitano, H., M. Asada, Y. Kuniyoshi, I. Noda, e E. Osawa (1998). Robocup: A challenge problem for ai and robotics. In *Lecture Notes in Computer Science*, Volume 1395, pp. 73–85.
- Knowles, J. e D. Corne (1999). The pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation. In *1999 Congress on Evolutionary Computation*, pp. 98–105. IEEE Service Center.
- Lee, S. W. e H. T. Tsui (2004). A relational multi-objective genetic algorithm. In *Neural Networks*, Volume 1.
- Michalewicz, Z. (1996). *Genetic algorithms + Data Structures = Evolution Programs*. Springer-Verlag New York, Inc.

- Nakashima, T., M. Takatani, M. Udo, e H. Ishibuchi (2004). An evolutionary approach for strategy learning in robocup soccer. In *International Conference on Systems, Man and Cybernetics*, Volume 2, pp. 2023–2028.
- Plant, W. R. e G. Schaefer (2008). An overview of genetic algorithms in simulation soccer. In *IEEE Congress on Evolutionary Computation (CEC 2008)*, pp. 3897 – 3904.
- Raadt, M., M. Prokopenko, e M. Butler (2003). Evolving tactical formations on the robocup field. In *Workshop on Adaptability in Multi-Agent Systems at The First RoboCup Australian Open 2003*.
- Reis, L. P. (2003). *Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico*. Tese de Doutorado, Faculdade de Engenharia da Universidade do Porto.
- Rezende, S. O. (2003). *Sistemas Inteligentes: Fundamentos e Aplicações*. Manole.
- RoboCup (2009). *Official Site of RoboCup World Championship*. In <http://www.robocup.org>. Visited in 12/01/2009.
- Ross, T. J. (2004). *Fuzzy Logic with Engineering Applications*. John Wiley.
- Schaffer, J. D. (1984). *Some Experiments in Machine Learning using Vector Evaluated Genetic Algorithms*. Tese de Doutorado, TN: Vanderbilt University.
- Shi, K., A. Bai, Y. Tai, e X. Chen (2009). Official site of wrighteagle, team description paper. In *Disponível em <http://ai.ustc.edu.cn/en/robocup/2D/tdp/WrightEagle20092DSoccerSimulationTeamDescriptionPaper.pdf>*. Visited in 12/08/2009.
- Srinivas, N. e K. Deb (1994). *Multiobjective optimization using nondominated sorting in genetic algorithms*. *Evolutionary Computation* 2, 221–248.
- UvA Trilearn (2003). *Official site of uva trilearn 2003 - soccer simulation team*. In <http://staff.science.uva.nl/jellekok/robocup/2003/trilearnrc2003bin.tar.gz>. Visited in 20/05/2008.
- Veldhuizen, D. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Tese de Doutorado, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology.

- Weiss, G. (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The University of Michigan Press.
- Zhang, P., B. Zhao, Y. Cao, e S. Cheng (2004). A novel multi-objective genetic algorithm for economic power dispatch. In *Universities Power Engineering Conference*, Volume 1, pp. 422–426.
- Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Tese de Doutorado, Swiss Federal Institute of Technology Zurich.
- Zitzler, E., M. Laumanns, e L. Thiele (2001). Spea2: Improving the strength pareto evolutionary algorithm. Technical Report No. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich.