UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

NIRVANA DA SILVA ANTONIO

OTIMIZAÇÃO DA FUNÇÃO DE AVALIAÇÃO DO DOMINÓ DE 4 PONTAS UTILIZANDO ALGORITMO GENÉTICO

MANAUS
2011

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

NIRVANA DA SILVA ANTONIO

OTIMIZAÇÃO DA FUNÇÃO DE AVALIAÇÃO DO DOMINÓ DE 4 PONTAS UTILIZANDO ALGORITMO GENÉTICO

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção de título de Mestre em Engenharia Elétrica na área de concentração Controle e Automação de Sistemas.

Orientador: Prof. Dr. Cícero Ferreira Fernandes Costa Filho

Co-orientadora: Profa. Dra. Marly Guimarães Fernandes Costa

MANAUS
2011

Ficha Catalográfica
(Catalogação realizada pela Biblioteca Central da UFAM)

A635o Antonio, Nirvana da Silva

Otimização da função de avaliação do dominó de 4 pontas utilizando algoritmo genético / Nirvana da Silva Antonio .- Manaus: UFAM, 2011.
113f.; il. color.

Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal do Amazonas, 2011.

Orientador: Prof. Dr. Cícero Ferreira Fernandes Costa Filho
Co-orientadora: Profª Drª Marly Guimarães Fernandes Costa

1. Teoria dos jogos 2. Jogo de dominó – Estratégias 3. Algoritmos genéticos. I. Costa Filho, Cícero Ferreira Fernandes (Orient.) II. Costa, Marly Guimarães Fernandes (Co-orient.) III. Universidade Federal do Amazonas IV. Título

CDU (1997) 794.3:004.421(043.3)

NIRVANA DA SILVA ANTONIO

OTIMIZAÇÃO DA FUNÇÃO DE AVALIAÇÃO DO DOMINÓ DE 4 PONTAS UTILIZANDO ALGORÍTMO GENÉTICO

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica na área de concentração Controle e Automação de Sistemas.

Aprovado em 21 de dezembro de 2011.

BANCA EXAMINADORA

Prof. Dr. Cícero Ferreira Fernandes Costa Filho

Universidade Federal do Amazonas- UFAM

Profa. Dra. Marly Guimarães Fernandes Costa

Universidade Federal do Amazonas- UFAM

Prof. Dr. José Raimundo Gomes Pereira

Universidade Federal do Amazonas- UFAM

Prof. Dr. José Francisco de Magalhães Netto

Universidade Federal do Amazonas- UFAM

Dedico este trabalho aos meus pais,

João Bosco e Maria Regina.

Agradecimentos

Ao Prof. Cícero Costa Filho, pela orientação e supervisão durante o desenvolvimento deste trabalho desde a iniciação científica, e à Profa. Marly Guimarães Costa, pelo apoio e incentivo nestes anos de Mestrado.

Aos Professores Ramon Romankevicius e Fernando Lizarralde da COPPE/UFRJ, pela cordial acolhida e auxílio oferecido durante o período de missão de estudo.

Ao Prof. José Raimundo Gomes Pereira, pela valiosa colaboração com a elucidação acerca do tratamento estatístico necessário na etapa final da pesquisa.

Ao Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) da Universidade Federal do Amazonas e, em especial, ao Centro de Pesquisa e Desenvolvimento em Tecnologia Eletrônica e da Informação, CETELI, por fornecer a infraestrutura necessária para minha formação e pela oportunidade de realizar missão de estudo no Programa de Pós-Graduação em Engenharia Elétrica (PEE) da COPPE/UFRJ. À Fundação de Amparo à Pesquisa do Estado do Amazonas, FAPEAM, pelo fundamental apoio financeiro.

Às pessoas que participaram dos testes de avaliação do agente inteligente para o jogo de dominó de 4 pontas.

Aos meus irmãos, Ananda e Carlos Magno, pela inestimável companhia, apoio e sugestões durante a escrita da dissertação, e a toda minha família, pelo incentivo.

À Pamela Levy, pela amizade desde os tempos de ensino médio e pela agradável convivência durante a nossa estada no Rio de Janeiro. Aos amigos do Rio de Janeiro, Ingrid, Luísa e Marcos, pela amizade e companhia nos momentos de descontração, e aos amigos do Mestrado, em especial, Tânia, Clahildek e Rafael, pelo companheirismo e convivência no Laboratório.

A todas as pessoas que contribuíram de alguma forma para a conclusão deste trabalho.

Resumo

O dominó de 4 pontas, popular no Estado do Amazonas, é uma variação do jogo de dominó possivelmente originado na China. No dominó de 4 pontas, as partidas são disputadas entre duas duplas e os jogadores devem elaborar estratégias baseadas em três objetivos principais: pontuar, facilitar jogadas futuras da dupla e dificultar as jogadas dos adversários. Porém, em nenhum momento os jogadores tem conhecimento sobre as peças em posse dos demais jogadores, o que caracteriza o dominó como um jogo de informações imperfeitas, onde a busca no espaço de soluções é uma tarefa mais complexa do que em jogos de informações perfeitas. Este trabalho apresenta o desenvolvimento de um agente inteligente para o jogo de dominó de 4 pontas, cuja escolha das jogadas é feita através de uma função de avaliação. A função de avaliação se baseia em informações sobre o estado presente de jogo para realizar a escolha das jogadas de acordo com os objetivos do jogo. Foram propostas quatro possíveis estratégias a serem adotadas pelo agente inteligente para o dominó de 4 pontas: uma estratégia que considera os três objetivos principais simultaneamente, uma estratégia que prioriza apenas o próprio jogador, uma estratégia que prioriza apenas o parceiro de dupla e uma estratégia que somente visa bloquear as ações adversárias. Por priorizarem diferentes objetivos do jogo, cada estratégia é representada por uma função de avaliação distinta. A escolha dos coeficientes ótimos para estas funções de avaliação foi realizada utilizando Algoritmos Genéticos, uma técnica de busca inspirada na Teoria da Evolução de Darwin. O critério de avaliação usado para determinar a melhor solução foi o número de vitórias em 5000 partidas de dominó e as otimizações foram divididas em três etapas. Inicialmente, para cada estratégia, o algoritmo genético maximizou a quantidade de vitórias contra uma dupla que adotava a estratégia básica de jogo. Na segunda etapa, as estratégias foram otimizadas jogando contra elas mesmas, onde a dupla adversária usou os coeficientes otimizados na primeira etapa. Na última etapa, cada estratégia foi otimizada contra as outras três, onde estas utilizavam os coeficientes otimizados na segunda etapa. Devido à natureza estocástica do algoritmo genético, todas as otimizações foram executadas 10 vezes, permitindo que a média e desvio-padrão dos resultados fossem obtidos. Com estas informações, foram aplicados testes estatísticos para determinar a significância da quantidade de vitórias. O teste mostrou que duas estratégias alcançaram resultados superiores à função desenvolvida em um trabalho anterior. Comparando o desempenho entre as quatro estratégias propostas, concluiu-se que a estratégia que abrange os três objetivos de jogo é superior às demais, as estratégias que priorizam apenas um jogador da dupla apresentam desempenho equivalente e a estratégia que se foca apenas em atrapalhar os adversários possui desempenho inferior às anteriores. A melhor estratégia otimizada nesta pesquisa também foi avaliada contra duplas formadas por jogadores humanos. Contra jogadores experientes, a estratégia não apresentou desempenho satisfatório, ganhando apenas 32% das partidas. Porém, contra jogadores casuais, ganhou em 78% dos jogos disputados.

Palavras-chave: jogo de dominó; função de avaliação; algoritmos genéticos.

Abstract

The 4-sided dominoes game, popular in Amazonas State, is a variation of the dominoes game probably originated in China. In the 4-sided dominoes, the matches are disputed between two teams and the players must elaborate strategies based in three main objectives: scoring, facilitate the future moves of the team and hinder the opponents' moves. On the other hand, at anytime the players have knowledge about the pieces held by the other players, which characterizes the dominoes as an imperfect information game, where the search in the solution space is more complex than in perfect information games. This work presents the development of an intelligent agent for the 4-sided dominoes game, in which the choice of the moves is done through an evaluation function. The evaluation function is based on information about the present game state to make the selection of the moves according to the objectives of the game. We proposed four possible strategies to be adopted by the intelligent agent for the 4-sided dominoes game: a strategy that considers the three main objectives simultaneously, a strategy that prioritizes only the player himself, a strategy that prioritizes only the partner and a strategy that only aims to block the opponents' moves. By prioritizing different objectives of the game, each strategy is represented by a distinct evaluation function. The selection of the optimal coefficients for these evaluation functions was made using Genetic Algorithms, a search technique inspired by Darwin's Theory of Evolution. The evaluation criterion used to determine the best solution was the number of wins in 5,000 matches of dominoes and the optimizations were divided in three steps. Initially, for each strategy, the genetic algorithm maximized the number of wins against a team that adopted the basic strategy. In the second step, the strategies were optimized by playing against themselves, where the opponent team used the coefficients optimized in the first step. In the last step, each strategy was optimized against the other three, where these used the coefficients optimized in the second step. Due to the stochastic nature of the genetic algorithm, all optimizations were performed 10 times, allowing the mean and standard deviation of the results to be obtained. With these informations, statistical tests were applied in order to determine the significance of the number of wins. The test showed that two strategies achieved better results than those obtained by the function developed in a previous work. Comparing the performance between the four proposed strategies, we concluded that the strategy covering the three main objectives of the game is superior to the others; the strategies that emphasize only one team player have equivalent performance; and the strategy that focuses only on hinder the opponents' have inferior performance to the previous ones. The best strategy optimized in this work was also evaluated against teams formed by human players. Against experienced players, the strategy did not show satisfactory performance, winning only 32% of matches. Nevertheless, against casual players, the intelligent agent won in 78% of the disputed matches.

Keywords: dominoes game; evaluation function; genetic algorithms.

Lista de Figuras

Lista de Quadros

Sumário

1 INTRODUÇÃO

O dominó é um jogo formado por peças retangulares divididas em duas metades, as quais são marcadas por pontos indicando um valor numérico. Em um conjunto padrão de peças para o jogo de dominó, os números variam de 0 a 6. A combinação entre estes sete valores possíveis resulta em vinte e oito peças únicas que formam o conjunto padrão.

Possivelmente originado na China, o dominó é um jogo bastante difundido no mundo inteiro e pode ser jogado de diversas maneiras, dependendo da região. No Brasil, a forma mais comum é chamada de dominó de duas pontas. No entanto, no Estado do Amazonas, outra variação do jogo tornou-se mais popular entre seus habitantes, o dominó de quatro pontas. Para o desenvolvimento deste trabalho, utilizamos as regras e estratégias desta variação do jogo.

O dominó de 4 pontas é jogado por quatro pessoas divididas em duas equipes. Por ser disputado em duplas, onde cada dupla tem pontuação independente da adversária, é classificado como um jogo de dois jogadores e de soma não zero. Outra importante característica é ser um jogo de informações imperfeitas, pois o jogador apenas tem conhecimento das peças na mesa de jogo e das peças em sua posse.

Jogos de informações imperfeitas possuem dois problemas principais: a escolha da melhor jogada apesar do conhecimento parcial do estado de jogo e a inferência destas informações de jogo que ainda estão ocultas baseando-se apenas nas ações dos demais jogadores. Devido à complexidade apresentada por jogos de natureza não-determinística, pesquisas envolvendo jogos de informações imperfeitas são relativamente mais recentes que jogos que possuem informações completas do estado de jogo, como xadrez e damas.

Para jogos de informações perfeitas, a utilização de funções de avaliação associadas ao algoritmo alfa-beta para a escolha da melhor opção de jogada foi inicialmente proposta por Shannon (1950), sendo, posteriormente, implementada em jogos de xadrez (CAMPBELL *et*

al, 2002) e de damas (SCHAEFFER, 1997). Para os jogos de informação imperfeita, a utilização da proposta de Shannon envolve raciocínios complexos sobre as estratégias do jogo. Cita-se como exemplo o programa Bridge Baron (SMITH *et al*, 1998), desenvolvido para o jogo de Bridge. Para o jogo de dominó de duas pontas, encontram-se na literatura alguns trabalhos que buscam a melhor estratégia do jogo, como Chlebus (1986), Yen (1992) e, recentemente, Garza (2006). Não encontramos, porém, trabalhos relacionados especificamente ao dominó de 4 pontas.

Em (ANTONIO *et al*, 2008, 2009), desenvolvemos uma metodologia para a escolha da melhor jogada baseada em uma função de avaliação que combina em seus termos informações sobre o estado presente do jogo. Para validar a função de avaliação desenvolvida, foram realizadas simulações do jogo de dominó de quatro pontas, onde uma dupla utilizava a função de avaliação para a escolha da melhor jogada e a outra dupla realizava suas jogadas se baseando apenas nos pontos na mesa. A escolha dos parâmetros da função de avaliação foi feita experimentalmente e como resultado, a primeira dupla ganhou o jogo em mais de 66% das simulações. Em (ANTONIO, 2009), otimizamos a função de avaliação desenvolvida em (ANTONIO *et al*, 2008, 2009) usando um Algoritmo Genético. A otimização foi realizada sobre uma função de avaliação composta por seis parâmetros e o desempenho da dupla que utilizou a função foi de, aproximadamente, 68% dos jogos realizados, sendo superior ao trabalho anterior.

Nesta dissertação, continuamos a pesquisa iniciada nos trabalhos citados anteriormente, porém foram analisadas diversas estratégias que podem ser adotadas por jogadores de dominó de 4 pontas. Para cada estratégia adotada, uma função de avaliação associada foi otimizada por um Algoritmo Genético (AG).

Matematicamente, otimização refere-se ao método de escolher o melhor elemento (solução ótima) dentro de um conjunto de alternativas disponíveis, ou seja, minimizar ou

maximizar uma função objetivo através da escolha de valores para variáveis reais ou inteiras dentro de um conjunto de soluções. Entende-se por otimização o processo pelo qual uma tarefa pode ser realizada da maneira mais eficiente possível, permitindo que alcance um estado de suposta perfeição dentro dos seus próprios limites. Podemos determinar se uma situação ou problema podem ser aperfeiçoados ao analisar, com base em critérios previamente definidos, se uma modificação pode melhorar ou piorar o seu desempenho final. Caso a modificação sugerida introduza algum progresso em relação à situação anterior, é adotada como a melhor opção.

O processo de otimização é baseado em três conceitos principais: a codificação do problema, a função objetivo a ser maximizada (ou minimizada) e o espaço de soluções associado ao problema. Para alcançar um resultado é necessário um procedimento sistemático de busca. Este processo de busca é constituído por iterações, onde são testados os candidatos à solução, permitindo a evolução durante a seleção. As iterações são realizadas até que o valor ótimo seja encontrado ou uma condição seja satisfeita.

Nesta pesquisa, a otimização foi obtida através da técnica de busca e otimização dos Algoritmos Genéticos. Esta técnica se diferencia das técnicas tradicionais de busca, pois fornece um mecanismo de busca e computação paralela (os candidatos à solução são analisados simultaneamente) e adaptativa sem impor muitas limitações ao problema. Portanto, é um método muito eficiente na busca de soluções em uma grande variedade de aplicações. Estas características e os inúmeros resultados de sucesso obtidos nas mais diferentes áreas de aplicação tornam a utilização dos AGs uma alternativa de grande força no trabalho de otimização proposto nesta dissertação.

Com o processo de otimização, esperamos que a dupla que utiliza a função de avaliação para a escolha de suas jogadas obtenha um melhor desempenho em relação à dupla que utiliza apenas a pontuação na mesa como critério de escolha de jogadas, por ser

considerada uma estratégia básica usada por iniciantes. O desempenho dos agentes inteligentes é avaliado de acordo com o número de vitórias em partidas disputadas contra uma dupla adversária usando a estratégia básica de jogo. Posteriormente, a melhor estratégia é utilizada por uma dupla de agentes inteligentes em partidas contra jogadores humanos.

1.1 Objetivo

1.1.1 Geral

O objetivo geral proposto nesta pesquisa é otimizar os parâmetros que compõem a função de avaliação para a escolha da melhor jogada no dominó de quatro pontas. O ajuste destes parâmetros será realizado através de um AG.

1.1.2 Específico

Os objetivos específicos consistem em:

- Identificar diferentes tipos de estratégias usadas por jogadores humanos no jogo de dominó de 4 pontas;
- Propor funções de avaliação para as estratégias identificadas;
- Implementar as funções de aptidão para cada estratégia e o algoritmo genético responsável pela otimização;
- Utilizar o algoritmo genético para otimizar os coeficientes das funções de aptidão propostas;
- Avaliar e comparar o desempenho das estratégias usando os coeficientes otimizados pelo algoritmo genético;
- Avaliar o desempenho de uma dupla formada por agentes inteligentes que utilizam a função de avaliação em partidas contra duplas formadas por jogadores humanos.

1.2 Organização do trabalho

Esta dissertação está dividida em oito capítulos e apresenta o desenvolvimento de um agente inteligente para o jogo de dominó de 4 pontas, como também os fundamentos básicos do método de busca por algoritmos genéticos, implementação computacional e aplicação para otimização das funções de avaliação propostas.

O Capítulo 2 aborda alguns trabalhos encontrados na literatura sobre o uso e a otimização de funções de avaliação em jogos e trabalhos relacionados especificamente ao jogo de dominó. O terceiro capítulo apresenta o jogo de Dominó de 4 pontas, suas regras e principais objetivos. No Capítulo 4, é descrita de forma detalhada a heurística para o desenvolvimento da função de avaliação responsável pela escolha das jogadas. A descrição do funcionamento de um algoritmo genético é feita no Capítulo 5, onde também são apresentados os elementos básicos e seus operadores genéticos. O Capítulo 6 apresenta a metodologia para a otimização da função de avaliação. São definidas diversas estratégias para o jogo de dominó e detalhes sobre a implementação computacional do algoritmo de busca. Os resultados obtidos com o método de otimização proposto são analisados no Capítulo 7 e as conclusões finais são expostas no Capítulo 8.

2 REVISÃO BIBLIOGRÁFICA

Em 1949, Shannon propôs que um programa de computador necessitaria de uma função de avaliação para competir com sucesso contra jogadores humanos em um jogo de xadrez (SHANNON, 1949). Desde então, o desenvolvimento de estratégias e heurísticas que possibilitem computadores a competir contra jogadores humanos tem sido bastante explorado em pesquisas na área da Inteligência Artificial. Jogos possuem características que permitem a implementação e aprendizado de agentes inteligentes: um conjunto bem definido de regras e estado final distinguível (NICOLAI *et al*, 2009). Porém, ao contrário do xadrez, jogos como o dominó exigem a utilização de estratégias baseadas em informações incompletas, tornando o processo de busca mais complicado do que quando aplicado a jogos com informações perfeitas.

No trabalho pioneiro de Shannon, o ajuste dos parâmetros da função de avaliação para o jogo de xadrez foi realizado de maneira manual, através da simulação de várias partidas e análise na mudança de cada parâmetro. Posteriormente, pesquisadores começaram a buscar métodos para automatizar a difícil tarefa de escolher os melhores parâmetros para uma função de avaliação, como a publicação de (SAMUEL, 1959), onde foi apresentado o primeiro método com aprendizado que ajustava iterativamente os pesos de uma função de avaliação em um jogo de damas.

Existem inúmeros métodos criados para o tratamento de problemas de otimização. Entre os algoritmos de busca existentes, a utilização de algoritmos baseados nos princípios evolutivos de Darwin apresentou uma alternativa de fácil representação e implementação na área de jogos, pois possui alto desempenho na busca de soluções no espaço de busca. Estes algoritmos, conhecidos como Algoritmos Evolutivos, pertencem a uma área da Inteligência Artificial chamada Computação Evolutiva ou Evolucionária. A estrutura básica de um algoritmo evolutivo é descrita na Figura 1.

```
procedure evolution program
begin
   t ← 0
   initialize P(t)
   evaluate P(t)
   while (not termination-condition) do
   begin
      t ← t + 1
      select P(t) from P(t − 1)
      alter P(t)
      evaluate P(t)
   end
end
```

Figura 1: Estrutura básica de um Algoritmo Evolutivo
FONTE: Michalewivz, 1996.

Neste contexto estão os Algoritmos Genéticos, técnicas de busca estocástica fundamentadas na teoria de Seleção e Evolução Natural de Charles Darwin e nos mecanismos da Genética. A aplicação de AGs como ferramenta de otimização surgiu em (GOLDBERG, 1989) e atualmente são usados nas áreas de teoria de jogos, reconhecimento de padrões, inferência estatística, controle e otimização de funções matemáticas.

Para o desenvolvimento desta dissertação, foram utilizadas as seguintes referências como principal suporte teórico para estudo e implementação de algoritmos genéticos: (ASHLOCK, 2006), (MICHALEWICZ, 1996) e (NEGNEVITSKY, 2005). Também foram analisados trabalhos realizados na área de jogos envolvendo o uso de algoritmos evolutivos para desenvolvimento de agentes inteligentes, como (NICOLAI *et al*, 1999; BOSKOVIC *et al*, 2006; KENDALL e WHITWELL, 2001). Especificamente para o jogo de dominó, poucos trabalhos foram encontrados na literatura. Entre eles, temos os trabalhos de Chlebus (1986), Yen (1992) e, recentemente, Garza (2006), porém nenhum é aplicado ao dominó de quatro pontas, o qual as únicas referências encontradas foram em nossas pesquisas anteriores (ANTONIO *et al*, 2008, 2009; ANTONIO, 2009).

Nas seções a seguir estão relacionados alguns trabalhos encontrados na literatura sobre a aplicação de algoritmos evolutivos na otimização da função de avaliação de jogos em geral e aos jogos de dominó.

2.1 Otimização da função de avaliação usando Algoritmos Evolutivos

Os primeiros trabalhos voltados para o desenvolvimento de agentes inteligentes na área de jogos foram, principalmente, para xadrez e damas. Como alternativa aos métodos tradicionais de busca determinística, técnicas baseadas em Algoritmos Evolutivos (AE) constituíram uma nova abordagem para o problema de otimização de funções de avaliação nestes jogos.

O trabalho de (WANG e LI, 2008) abordou o desenvolvimento de um agente inteligente para o jogo de xadrez chinês (Xiang Qi), cuja complexidade é maior que o xadrez comum, como também os jogos de damas e Othello. O agente inteligente foi representado por uma rede neural artificial cujos pesos dos neurônios foram ajustados através de um algoritmo genético. Em Kendall e Whitwell (2001), um programa implementando o jogo de xadrez comum foi usado como plataforma de aprendizado e uma versão simplificada da função de avaliação proposta por Shannon foi otimizada por um AE. O método de aprendizado consiste na seleção de dois indivíduos da população para competirem em duas partidas de xadrez. O indivíduo que perde é retirado da população e substituído por uma cópia do indivíduo vencedor que passou pelo processo de mutação. O processo de seleção e mutação do indivíduo vencedor permite que a população seja, posteriormente, formada pelo indivíduo com o melhor conjunto de parâmetros. O processo de mutação é baseado no desvio-padrão dos parâmetros presentes nos indivíduos da população, ou seja, a mutação é controlada pela população e não pelo usuário, o que permite maior exploração do espaço de busca.

Em (BOSKOVIC *et al*, 2006), um algoritmo evolutivo baseado no *Algoritmo de Evolução Diferencial* (ED) é aplicado para otimizar os parâmetros de uma função de

avaliação desenvolvida para a escolha das jogadas em um jogo de xadrez. O algoritmo de Evolução Diferencial, desenvolvido por Price e Storn (1997), é um método evolutivo de otimização global sobre espaços contínuos. Por ser um algoritmo derivado da teoria evolutiva, o ED aplica os processos de mutação, cruzamento e seleção sobre uma população de indivíduos. Porém, neste artigo, o autor propõe a adição da ideia de competição como forma de avaliação da aptidão e otimização dos indivíduos.

Este processo de competição consiste na avaliação de desempenho entre duas populações: a população original da geração atual e uma população modificada após as operações de mutação e cruzamento. No final de cada jogo, o indivíduo que perdeu é modificado através de uma equação de aprendizado, de forma que ele fique com parâmetros mais próximos aos do indivíduo que ganhou a partida. Segundo Boskovic (2006), o parâmetro de aprendizado, que permite a otimização da função de avaliação durante a competição, é responsável pela convergência dos indivíduos mais fracos para os indivíduos mais aptos dentro de uma população, possibilitando rápida convergência do algoritmo para bons valores e menor desvio-padrão entre os indivíduos da população final.

Além de jogos com informações perfeitas como o xadrez, AE também são usados na otimização de agentes inteligentes para jogos de informações imperfeitas (WITTKAMP e BARONE, 2006; FLOM e ROBINSON, 2004), os quais são representados principalmente pelos jogos de cartas como o Poker (BARONE e WHILE, 1999; NICOLAI *et al*, 2009).

Em (NICOLAI *et al*, 2009), é desenvolvido um agente inteligente para uma variante do jogo de Poker, denominada *No-Limit Texas Hold'em*, através de algoritmos evolutivos e redes neurais (*Evolving Neural Network*). Este jogo, além de ser caracterizado como um jogo de informações imperfeitas como o dominó, possui um espaço de estados bastante extenso, tornando a utilização de computação evolutiva mais viável. Os agentes foram implementados como redes neurais, onde suas entradas são um conjunto de recursos disponíveis ao jogador

em uma partida de Poker e novos agentes são gerados através do ajuste dos pesos da rede com a soma ponderada dos pesos dos pais.

Segundo Nicolai, a evolução por si só não é suficiente para gerar um ambiente de aprendizado bem sucedido. Ele propõe a utilização de heurísticas adicionais para melhorar o processo de evolução, a coevolução e o hall da fama. Os experimentos foram realizados para avaliar o desempenho dos agentes gerados a partir do algoritmo evolutivo e os agentes gerados a partir do algoritmo em combinação com coevolução e hall da fama. Como resultado, todos os grupos que utilizavam as heurísticas adicionais geraram agentes com desempenho superior ao agente gerado somente a partir do algoritmo evolutivo, onde o melhor agente obteve desempenho 39% superior ao primeiro agente.

2.2 Publicações na área de jogos de dominó usando Função de Avaliação e AGs

Em (GARZA, 2006), a pesquisa é baseada em um jogo de dominó de duas pontas cujas regras são semelhantes às regras do dominó de 2 pontas mais jogado no Brasil. Em seu trabalho, Garza desenvolveu um agente inteligente para a escolha das jogadas durante uma partida de dominó com quatro jogadores. Os agentes inteligentes foram implementados para agir de maneira independente e cada agente realiza a decisão de jogada através de inferências e suposições baseadas em jogadas anteriores dos demais jogadores. No entanto, os dados armazenados sobre os demais jogadores estão apenas relacionados às pedras que eles mais "gostavam", ou seja, a quantidade de pedras jogadas para cada numeração. Foram implementadas sete estratégias diferentes, incluindo uma estratégia tradicional baseada na estratégia adotada por jogadores reais. Os testes realizados na pesquisa incluíam as seguintes estratégias: tradicional, altruísta, egoísta, aleatória, suicida, tradicional altruísta e tradicional egoísta. Porém, para validar o agente inteligente, apenas um dos jogadores da dupla tinha sua estratégia variada a cada simulação, os demais jogavam com a estratégia tradicional. O melhor resultado obtido neste trabalho foi 54% de vitórias em um total de 100 partidas, não

sendo considerado um resultado significativo, pois ficou dentro do intervalo de tolerância definido pelo autor devido às características aleatórias do jogo.

Para o dominó de 4 pontas, em (ANTONIO *et al*, 2008, 2009), inicialmente foi proposta uma função de avaliação para a escolha da melhor jogada a partir dos estados do jogo. A função de avaliação incorpora em seus termos a soma de pontos possíveis com a jogada e a estratégia de jogo, que incluem facilitar as jogadas da dupla e dificultar as jogadas dos adversários. Nos dois trabalhos, os coeficientes da função de avaliação foram definidos experimentalmente e ambos os jogadores da dupla utilizavam a mesma função de avaliação para decidir suas jogadas. Como resultado, a dupla obteve 66% de vitórias jogando contra uma dupla que utilizava uma estratégia baseada nos pontos possíveis na mesa. O desenvolvimento detalhado da teoria pesquisada neste trabalho será descrito no Capítulo 4.

3 O JOGO DE DOMINÓ

Neste capítulo, apresentaremos os fundamentos necessários para o entendimento do jogo de dominó. Descreveremos os principais elementos, regras e objetivos, bem como algumas definições específicas ao jogo de dominó de 4 pontas.

3.1 Introdução

O jogo de dominó, em sua forma padrão, é composto por vinte e oito peças achatadas e retangulares. Cada peça é dividida em duas metades que, na forma clássica do jogo, contém uma numeração que varia de zero a seis. A Figura 2 mostra todas as combinações possíveis entre esses números, formando vinte e oito peças únicas de dominó.

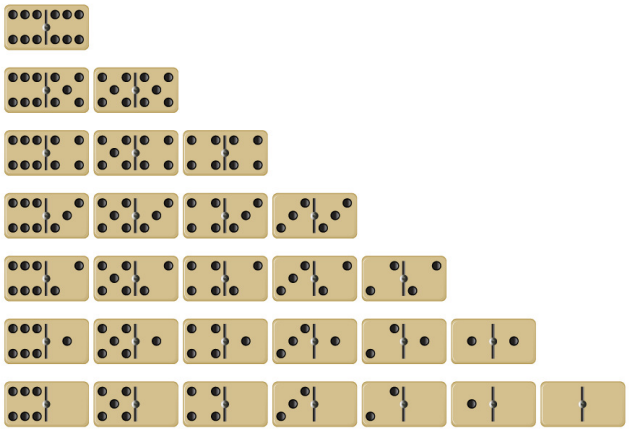


Figura 2: Conjunto padrão do jogo de Dominó formado por 28 peças
FONTE: Retirado de <http://dominoesworlds.com>. Acesso em: 11 de abril de 2011.

As peças, ou pedras, de dominó são nomeadas de acordo com o número pontos em cada metade. Por exemplo, uma pedra com seis pontos em uma metade e quatro pontos na outra metade é referida como “6-4”. Entre os jogadores de dominó, esta pedra é chamada de “sena e quadra”. As numerações, que variam de zero a seis, recebem os seguintes nomes: branco, às, duque, terno, quadra, quina e sena, respectivamente. Se uma pedra tem as duas metades com a mesma numeração, é chamada de “carroça”. Neste caso, temos “carroça de

branco", para a pedra com numeração 0-0, "carroça de às", para a pedra com numeração 1-1, e assim sucessivamente.

3.2 Objetivos e regras

O dominó pode ser jogado por quatro jogadores, em duplas ou individualmente, onde cada jogador recebe sete peças inicialmente ou, alternativamente, pode ser jogado por dois jogadores, onde as 14 pedras restantes são usadas para "comprar" durante a partida. Existem várias formas de jogar dominó e a mais comum no Brasil chamamos de dominó de 2 pontas. No entanto, no Amazonas é jogado outro tipo pouco difundido, o Dominó de 4 pontas.

No Dominó de 2 pontas, o objetivo principal é conseguir colocar sua última peça na mesa antes que os jogadores adversários. Para isso, durante a partida são adotadas estratégias para o jogador "bater", isto é, ser o primeiro a se desfazer de suas pedras. As jogadas visam encaixar alguma peça do jogador nas peças que estão nas pontas do jogo, uma por vez, e minimizar a possibilidade do adversário encaixar uma pedra em uma das pontas. Caso algum jogador tenha batido, sua dupla leva todos os pontos das peças que sobraram nas mãos dos adversários. A partida pode terminar em duas circunstâncias: quando um jogador bater ou quando o jogo fica trancado, ou seja, nenhum dos jogadores tem pedras para jogar em alguma das pontas. No caso de jogo trancado, contam-se todos os pontos das peças que sobraram nas mãos de cada dupla. A dupla que possuir menos pontos é a vencedora e leva os pontos da dupla adversária.

No Dominó de 4 pontas, objeto desta dissertação, a disputa ocorre, em geral, entre duplas. Cada jogador deve ter sete pedras no início de cada rodada. O conjunto de peças que cada jogador possui é chamado de "mão" do jogador. Na primeira rodada, a partida é iniciada pelo jogador que possui a pedra com numeração 6-6, ou carroça de sena. Em seguida, a ordem de jogada ocorre no sentido horário e o próximo jogador deve encaixar uma pedra cuja metade tenha a numeração 6 em uma das faces da pedra inicial. Uma rodada pode terminar

em duas situações distintas: um dos jogadores “bateu”, ou seja, conseguiu jogar todas as pedras da sua “mão” antes dos demais jogadores; o jogo foi trancado, ou seja, os jogadores não podem jogar em nenhuma das pontas na mesa. Nas rodadas seguintes, o jogador que bateu na rodada anterior deverá iniciar com a carroça de sua escolha.

Durante uma partida, os jogadores tem a possibilidade de abrir até quatro pontas de jogo. O conceito de pontas na mesa, no jogo de dominó de 4 pontas, está relacionado à possibilidade dos jogadores encaixarem as peças a partir das quatro faces da carroça que iniciou a partida. Na Figura 3 são apresentadas seis situações que podem ocorrer durante o jogo. Em cada situação, a quantidade de “pontas na mesa” varia, conforme descrito a seguir.

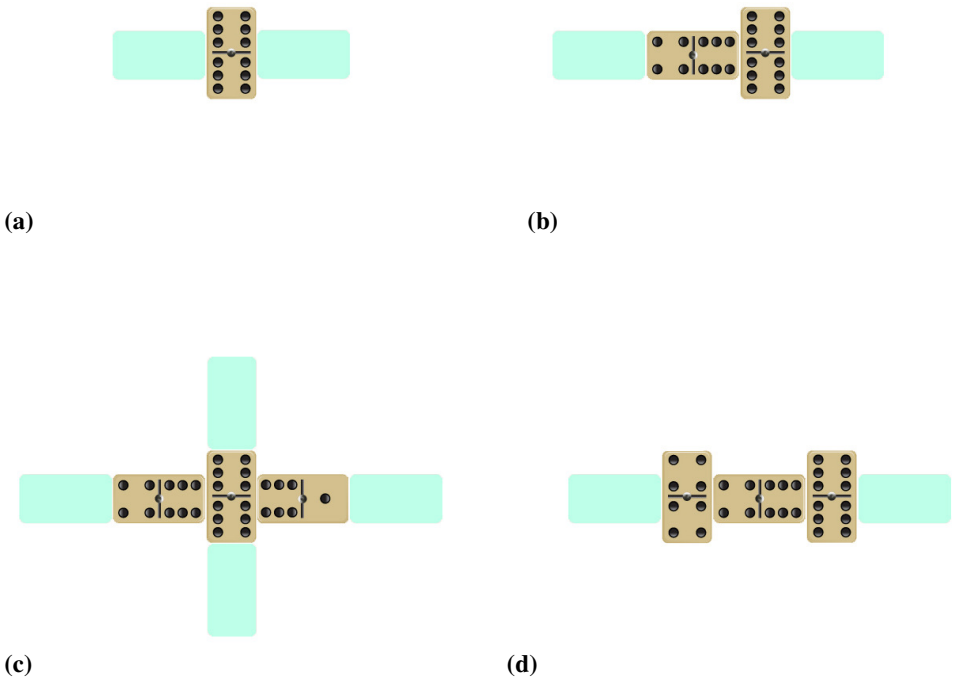


Figura 3: Pontas da mesa no Dominó de 4 pontas
As marcações em verde indicam onde o jogador pode encaixar uma pedra.

Após a jogada inicial, quando existe apenas a carroça na mesa, considera-se que existem **duas** pontas de jogo, pois os demais jogadores tem a possibilidade de jogar em duas laterais da peça, conforme mostra a Figura 3(a). A Figura 3(b) representa o momento em que

o segundo jogador realizou sua jogada em uma das pontas disponíveis inicialmente. Neste caso, o próximo jogador terá ainda duas opções de jogada, uma na ponta de numeração 4 e outra na ponta de numeração 6. Caso o jogador realize uma jogada na ponta de numeração 6, o próximo jogador terá possibilidade de jogar em **quatro** pontas de jogo, conforme mostra a Figura 3(c). Se o jogador realizar a jogada na ponta de numeração 4, as opções de jogada para o jogador seguinte continuam sendo apenas em duas pontas da mesa, conforme mostra a Figura 3(d).

O objetivo principal do dominó de 4 pontas é obter uma pontuação igual ou superior a 200 pontos, podendo ser alcançada em uma ou mais rodadas. Para isso, o jogador pode pontuar 5, 10,..., 50 pontos, ou seja, múltiplos de 5. Definimos abaixo cinco situações P_i onde há a possibilidade do jogador pontuar durante uma partida de dominó de 4 pontas:

- **P_1** : Indica a pontuação realizada na mesa de jogo, calculada através da soma dos pontos nas pontas da mesa. Caso esta soma seja um múltiplo de 5, a pontuação é contabilizada para a dupla que realizou a jogada; caso contrário, nenhum ponto é adicionado. A Figura 4 apresenta várias situações de jogo para exemplificar como é calculada a pontuação na mesa. Na primeira situação, ilustrada pela Figura 4(a), o jogador inicia a rodada com a pedra 5-5, a carroça de quina. Nesta situação, consideramos que existem duas pontas de jogo disponíveis para jogada, conforme ilustrado na Figura 3(a), portanto, os dois lados da pedra são somados, contabilizando 10 pontos para a dupla que realizou a jogada. Na situação apresentada na Figura 4(b), novamente são consideradas apenas duas pontas de jogo para a soma de pontos. Uma ponta possui numeração 0 e a outra ponta possui numeração 10, totalizando 10 pontos. Na Figura 4(c), a soma dos pontos é realizada apenas nas duas pontas que foram “abertas” a partir da carroça inicial, totalizando 5 pontos para a dupla. A situação apresentada na Figura 4(d) é

semelhante à anterior, porém três pontas são usadas para a soma dos pontos, totalizando 5 pontos (0+1+4). Por outro lado, na Figura 4(e) a soma dos pontos (2+1+4) não é um múltiplo de cinco, caracterizando uma jogada onde não se pontua. A Figura 4(f) apresenta a situação onde todas as pontas de jogo foram "abertas", portanto as quatro pontas são usadas para soma de possíveis pontos. Neste exemplo, a dupla que realizou a jogada marcou 10 pontos (2+3+1+4);

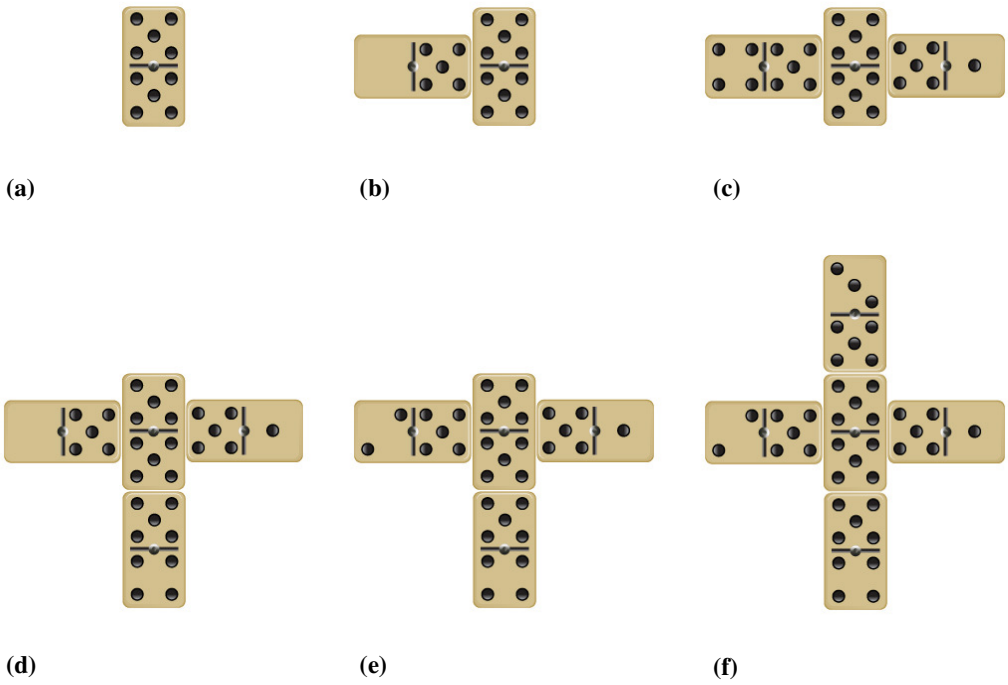


Figura 4: Possibilidades de pontuação na mesa

- **P_2 :** A pontuação P_2 ocorre quando o jogador adversário "passa" em sua vez de jogar, ou seja, não possui pedra que se encaixe em nenhuma das quatro pontas da mesa. O impedimento de jogada do adversário equivale a 20 pontos para a dupla que provocou o passe;
- **P_3 :** Esta possibilidade de pontuação ocorre quando o jogador na vez provoca o passe de todos os demais jogadores (inclusive o parceiro). Esse passe, chamado de "galo", equivale a uma pontuação de 50 pontos;

- **P_4 :** Ocorre quando um jogador “bate” e são somados os pontos das pedras que sobraram nas mãos dos adversários. Essa soma é denominada “garagem”. A pontuação corresponde ao maior múltiplo de 5, menor ou igual a esta soma. A Figura 5 mostra um exemplo em que a garagem é de 10 pontos, pois a soma dos pontos dos adversários é igual a 12;
- **P_5 :** A possibilidade de pontuação P_5 ocorre quando a última peça descartada pelo jogador que bateu é uma carroça (os dois lados tem a mesma numeração). Por ser considerada difícil, esta jogada vale 20 pontos.

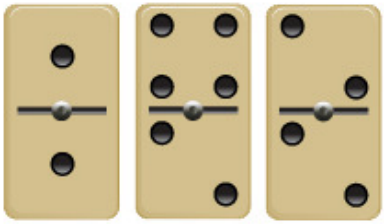


Figura 5: Exemplo de garagem equivalente a 10 pontos

3.3 Estratégias de jogo

O dominó de 4 pontas, por ter objetivos e regras mais elaboradas, adota estratégias mais complexas que as utilizadas no dominó de 2 pontas, pois além de pontuar, o jogador também deve tentar facilitar suas ações futuras (ou do parceiro) e dificultar as ações da dupla adversária.

Para exemplificar as estratégias adotadas durante o jogo, considera-se que, no início de uma rodada, um jogador tem em mãos cinco peças com a numeração 3, conforme ilustrado na Figura 6. Neste caso, a estratégia é tentar fazer com que essa numeração esteja presente no maior número possível de pontas, pois esse tipo de jogada tanto pode fazer seus adversários “passarem” como pode facilitar suas jogadas futuras. Por outro lado, a mão do jogador representada na Figura 6 possui apenas uma pedra com a numeração 2. Então, o jogador deve

evitar que esta numeração esteja presente nas pontas, pois poderia provocar o seu passe. Estas situações também ilustram o papel do parceiro do jogador que também deve estar atento às jogadas do seu companheiro de dupla e perceber as pedras que ele "gosta" ou não.

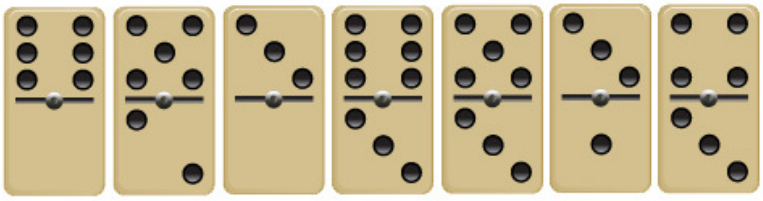


Figura 6: Exemplo da "mão" de um jogador com as sete pedras iniciais

4 FUNÇÃO DE AVALIAÇÃO PARA ESCOLHA DA MELHOR JOGADA

O capítulo aborda a teoria desenvolvida em (ANTONIO *et al*, 2008, 2009) para a escolha da melhor jogada no dominó de 4 pontas que serviu de base para o desenvolvimento da função de avaliação proposta nesta pesquisa.

4.1 Introdução

Em (ANTONIO *et al*, 2008, 2009) foi desenvolvido um agente inteligente para o jogo de dominó de 4 pontas. Este agente realiza a escolha de suas jogadas através de uma função responsável por avaliar o quanto uma peça pode ajudar o jogador ou atrapalhar os adversários, baseando-se em informações sobre as peças que já foram jogadas e as peças na mão do jogador. A função de avaliação é constituída pela soma de dois termos, T_{n1} e T_{n2}, conforme mostra a equação (1).

$$f(\mathrm{n}) = T_{\mathrm{n1}} + T_{\mathrm{n2}} \tag{1}$$

Na expressão definida acima, a variável *n* identifica uma opção de jogada para qual a função de avaliação é calculada. O termo T_{n1} corresponde aos pontos que serão obtidos ao realizar a jogada *n*. O termo T_{n2} incorpora a estratégia do jogo, correspondendo a valores que procuram retratar como a escolha pela jogada *n* pode facilitar ações futuras do jogador e dificultar as ações futuras dos jogadores adversários.

4.2 Os estados de jogo

Para calcular os termos T_{n1} e T_{n2} da função de avaliação da equação (1), utilizamos como parâmetros o estado presente de jogo. O estado presente de jogo é representado por sete vetores, onde cada vetor possui sete coordenadas (cada coordenada corresponde a uma possível numeração presente em uma pedra de dominó), conforme definido a seguir:

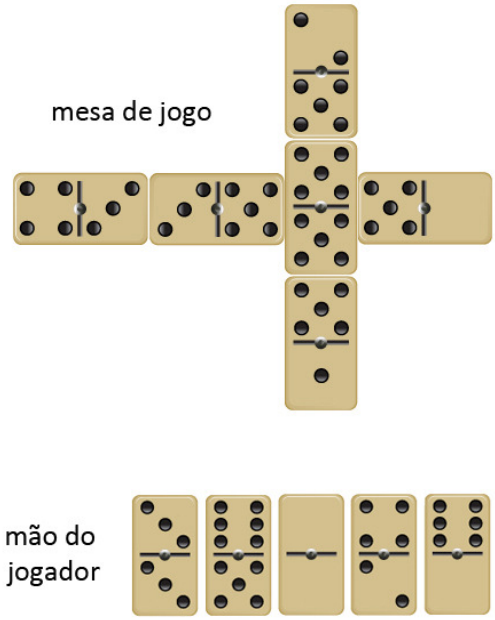
$$V_i = [\ V_{i0}, V_{i1}, V_{i2}, V_{i3}, V_{i4}, V_{i5}, V_{i6}\ ] \qquad i = 0, 1,..., 6$$

No vetor acima, V_{i0} corresponde ao número de pedras com nenhuma numeração; V_{i1} corresponde ao número de pedras com a numeração 1, e assim sucessivamente. Esses vetores expressam estatísticas do jogo e são atualizadas a cada jogada. A definição dos vetores de estado de jogo para o dominó de 4 pontas é apresentada abaixo:

- $\boldsymbol{V_0}$ **:** Representa a quantidade de peças em jogo, ou seja, a quantidade de pedras presentes na mesa, para cada numeração;
- $\boldsymbol{V_1}$ **:** Representa a quantidade de peças na mão do jogador para cada numeração;
- $\boldsymbol{V_2}$ **:** Representa a quantidade de peças nas pontas para cada numeração;
- $\boldsymbol{V_3}$ **:** Representa a quantidade de peças jogadas pelo parceiro de dupla para cada numeração;
- $\boldsymbol{V_4}$ **:** Os vetores V_4, V_5 e V_6 são diferentes dos vetores anteriores, pois recebem apenas os valores 0 ou 1. O vetor V_4 indica as numerações onde o adversário seguinte já passou. Por exemplo, se $V_{40} = 1$, o adversário seguinte (à esquerda do jogador) já passou na numeração 0. Se $V_{40} = 0$, o adversário seguinte ainda não passou na numeração 0, e assim sucessivamente;
- $\boldsymbol{V_5}$ **:** Indica as numerações onde o adversário anterior (à direita do jogador) já passou. Este vetor funciona de maneira análoga ao vetor V_4;
- $\boldsymbol{V_6}$ **:** Indica as numerações onde o parceiro de dupla já passou. Este vetor funciona de maneira análoga ao vetor V_4.

Para demonstrar o funcionamento dos vetores de estado de jogo durante uma partida de dominó de 4 pontas, utilizaremos a situação de jogo exposta na Figura 7. Para este exemplo, os vetores de estado de jogo V_0, V_1 e V_2, referentes ao jogador atual, estariam configurados conforme o quadro exibido na mesma figura. De acordo com os valores indicados neste quadro, o vetor V_0, que corresponde ao número de peças na mesa para cada

numeração, tem coordenada V_{00} = 1, pois existe apenas uma peça cuja metade não possui numeração (representado pelo número zero) na mesa de jogo. Por outro lado, V_{05} = 5 indica que já foram jogadas 5 peças com a numeração 5. De maneira semelhante, os vetores V_1 e V_2 tiveram seus valores contabilizados com base na disposição das peças na mão do jogador e nas pontas da mesa. Os vetores V_3, V_4, V_5 e V_6 não foram mostrados no exemplo, pois necessitaríamos das jogadas realizadas pelos demais jogadores desde o inicio do jogo.



V_i	V_{i0}	V_{i1}	V_{i2}	V_{i3}	V_{i4}	V_{i5}	V_{i6}
V_0	1	1	1	2	1	5	0
V_1	2	0	1	1	1	1	2
V_2	1	1	1	0	1	0	0

Figura 7: Exemplo de uma mesa de jogo e a mão do jogador atual

4.3 Os termos da função de avaliação

Como vimos anteriormente, a função de avaliação é formada pela soma dos termos T_{n1} e T_{n2}. A seguir, serão introduzidos os parâmetros envolvidos no cálculo desses dois termos.

O termo T_{n1} incorpora no cálculo da função de avaliação os pontos obtidos ao realizar a opção de jogada *n*. Os pontos que um jogador pode obter em uma jogada foram definidos

anteriormente como as possibilidades de pontuação P_1, P_2, P_3, P_4 e P_5. Porém, a situação P_4, que corresponde aos pontos de garagem, não será incluída no cálculo de T_{n1}, pois esta pontuação é contabilizada somente no término da rodada.

O termo T_{n1} é definido conforme a equação (2) abaixo:

$$T_{n1} = P_1 + P_2 + P_3 + P_5 \quad (2)$$

A possibilidade de pontuação P_1, que contabiliza a pontuação obtida na mesa com a jogada, é calculada pela soma de todas as pontas disponíveis no jogo. Se a soma resultar em um valor múltiplo de 5, P_1 recebe este valor, caso contrário, $P_1 = 0$.

Os termos P_2 e P_3 são denominados **pontos de passagem** e utilizam um único algoritmo para o seu cálculo. Para o entendimento deste algoritmo, inicialmente são necessárias as definições de algumas variáveis auxiliares. No dominó de 4 pontas, cada pedra possui uma numeração em cada metade. Para que uma jogada seja válida, uma destas numerações deve coincidir com a ponta onde a pedra será "encaixada". Por consequência, a numeração localizada na outra metade da pedra será a "nova ponta" da mesa, substituindo a anterior. Sendo L uma possível opção de jogada, será denominada por L_1 a metade da pedra que coincide com o valor presente em qualquer ponta do jogo e L_2 corresponderá à outra metade da pedra, ou seja, à nova ponta. Portanto, se um jogador tem como opção de jogada uma pedra com numeração 6-0 e a mesa de jogo está disposta conforme mostra a Figura 7, L_1 possui numeração 0, pois coincide com a numeração presente na ponta 2 e L_2, cuja numeração é igual a 6, será a nova ponta na mesa de jogo.

A partir destas definições, os termos P_2 e P_3 são calculados conforme o fluxograma da Figura 8. Neste fluxograma, Np_1, Np_2 e Np_3 são variáveis que correspondem às numerações existentes em três das quatro pontas da mesa e L_2 representa a nova ponta. Os termos C_i indicam se o jogador atual possui todas as pedras restantes com a mesma numeração existente na ponta i, representado por Np_i, ao analisar os vetores V_0 (quantidade de pedras na mesa de

jogo) e V_1 (quantidade de pedras na mão do jogador). Se a soma dos valores V_{0Np_i} e V_{1Np_i} é igual a 7, o algoritmo modifica os vetores V_4, V_5 e V_6 (referentes às numerações onde os jogadores "passam"). Se os vetores V_4, V_5 e V_6 receberem valor 1 para as quatro pontas L_2, Np_1, Np_2 e Np_3, é caracterizado o "galo", jogada que vale 50 pontos. Caso contrário, o algoritmo verifica se o próximo jogador "passa", o que equivale a 20 pontos.

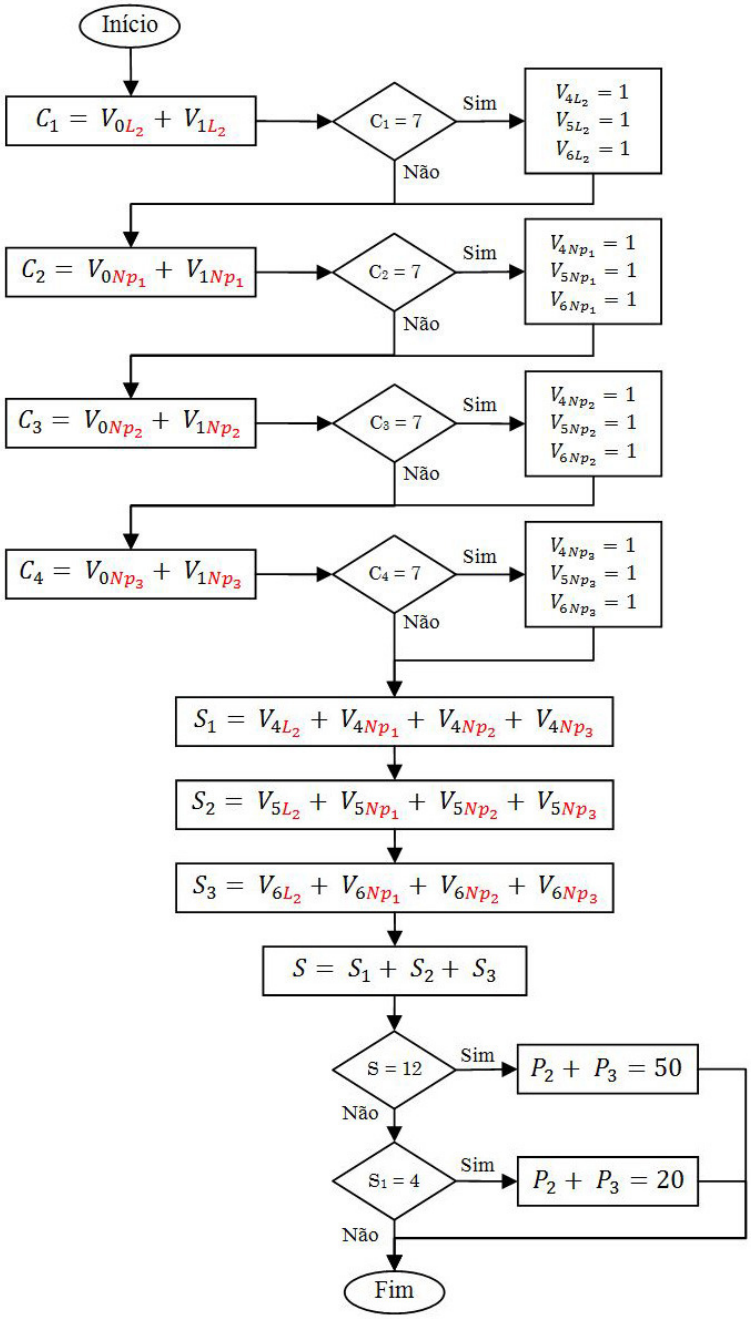


Figura 8: Fluxograma para o cálculo de P_2 e P_3

O termo T_{n2} incorpora no seu cálculo três parcelas distintas relacionadas à estratégia do jogo, conforme mostra a equação a seguir:

$$T_{n2} = \alpha.(- E_1 + E_2 + \delta.E_3) \quad (3)$$

O valor de α permite definir a importância da estratégia, calculada pelo termo T_{n2}, em relação aos pontos obtidos com uma jogada, calculada pelo termo T_{n1}.

A parcela E_1 considera a possibilidade de fazer o adversário seguinte passar na sua vez de jogar. Para o entendimento da parcela E_1, será utilizada a seguinte situação hipotética: considerando que em duas das pontas do jogo exista uma numeração n_1 e que o jogador da vez tenha em mãos mais três peças com essa mesma numeração. Então, cinco das sete peças possíveis com a numeração n_1 não pertencem ao adversário seguinte, sendo grande a probabilidade do mesmo passar se todas as pontas estiverem com essa numeração. Tendo em vista a possibilidade de fazer o adversário seguinte passar, e dessa forma obter 20 pontos, é desejável que não seja descartada nenhuma das peças em que $L_1 = n_1$. Neste caso, a opção de jogar uma peça L em que $L_1 = n_1$ não é desejada do ponto de vista estratégico, justificando o sinal negativo para E_1 na expressão (3). O cálculo de E_1 é realizado conforme a expressão (4).

$$E_1 = K_1 . (V_{0L_1} + V_{1L_1} + V_{2L_1}) \quad (4)$$

Quanto maior o número de peças com a numeração L_1 nas pontas (representado pelo vetor V_2) e nas mãos do jogador (representado pelo vetor V_1), maior será o valor de E_1. Por outro lado, o valor de E_1 também deve ser diretamente proporcional ao número de peças com a numeração L_1 já jogadas (vetor V_0). O coeficiente K_1 controla a importância de cada termo de E_1 em relação às outras parcelas que compõem a função de avaliação.

A parcela E_2 tem por objetivo facilitar as ações futuras do jogador. O cálculo de E_2 é realizado conforme a equação (5).

$$E_2 = K_2 . (V_{0L_2} + V_{1L_2} + V_{2L_2})$$ (5)

Essa expressão é muito semelhante àquela proposta para E_1, porém é utilizado L_2 no lugar de L_1 como argumento, representando a numeração que o jogador quer introduzir na mesa de jogo. O coeficiente K_2 controla a importância de cada termo de E_2 em relação às outras parcelas que compõem a função de avaliação.

A função de avaliação proposta em (ANTONIO *et al*, 2008) também continha uma parcela E_3 que reforçava o objetivo da parcela E_2, facilitando as ações futuras do jogador quando o número de peças na mão é menor ou igual a três. No fluxograma da Figura 9, mostra-se como é realizado o cálculo de E_3. O valor de K_3 controla a importância da parcela E_3 frente às outras parcelas utilizadas para o cálculo de T_{n2}.

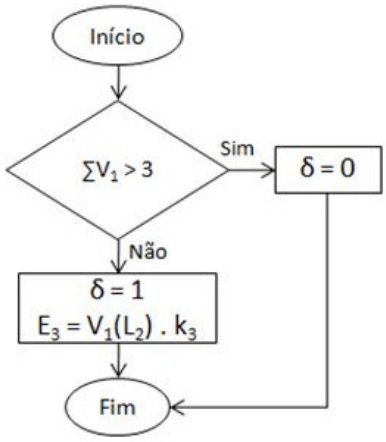


Figura 9: Fluxograma para o cálculo de E_3

A função de avaliação descrita nas equações acima envolve as principais situações consideradas por um jogador ao decidir a melhor jogada a cada rodada, propondo uma possível estratégia a ser adotada no jogo de dominó de 4 pontas. Nesta função, apenas quatro coeficientes, α, K_1, K_2 e K_3, controlam a importância dos vários termos que compõem a equação (3).

Em trabalhos anteriores (ANTONIO *et al*, 2008, 2009), o ajuste dos coeficientes α, K_1, K_2 e K_3 foi realizado através do teste de várias combinações de valores em simulações de

partidas de dominó. Os valores testados estavam na faixa de 0 a 1, com intervalos de 0,1 e na faixa de 1 a 10, com intervalos de 0,5 entre os números. O melhor resultado obtido nestes testes foi de 66% de vitórias para a dupla que utilizou a função de avaliação para a escolha de suas jogadas. Também verificamos que a variação do coeficiente K_3, responsável por determinar a importância do termo E_3 em relação aos outros termos da função de avaliação, não afetava o desempenho final da dupla. Por este motivo, em um trabalho posterior (ANTONIO, 2009), optamos por retirar o termo E_3 da função de avaliação. Neste trabalho, a função de avaliação era formada por seis coeficientes α_i, conforme a equação (6) descrita abaixo, e foi otimizada por um algoritmo genético. O melhor resultado obtido neste trabalho foi de 68,37% em 25000 partidas realizadas.

$$f(\mathrm{n}) = T_{\mathrm{n1}} + T_{\mathrm{n2}} \tag{6}$$

$$T_{\mathrm{n2}} = -E_1 + E_2$$

$$E_1 = \boldsymbol{\alpha_1} \cdot V_{0L_1} + \boldsymbol{\alpha_2} \cdot V_{1L_1} + \boldsymbol{\alpha_3} \cdot V_{2L_1}$$

$$E_2 = \boldsymbol{\alpha_4} \cdot V_{0L_2} + \boldsymbol{\alpha_5} \cdot V_{1L_2} + \boldsymbol{\alpha_6} \cdot V_{2L_2}$$

Nas funções de avaliação apresentadas anteriormente, os termos E_1 e E_2 analisam apenas os vetores de estado de jogo correspondentes ao próprio jogador e aos jogadores adversários. Por ser um jogo disputado em duplas, o jogador também deve considerar o parceiro ao planejar suas jogadas. Este tipo de análise, proporcionado pelo vetor V_3, não foi abordado nos trabalhos anteriores e foi incluído na função de avaliação que é otimizada nesta pesquisa.

5 ALGORITMOS GENÉTICOS

Algoritmos Genéticos (AGs) são as técnicas de busca e otimização mais conhecidas da Computação Evolucionária (MITCHELL, 1999), uma das áreas da Inteligência Artificial que estuda a criação de um sistema inteligente capaz de se adaptar a um ambiente em constante mudança baseando-se no processo de evolução natural. Neste contexto, AGs simulam a evolução natural com o auxílio dos mecanismos de seleção, mutação e reprodução apresentados pela Teoria da Evolução de Darwin (NEGNEVITSKY, 2005).

Os AGs pertencem à classe de algoritmos probabilísticos, mas são bem diferentes de algoritmos de busca aleatória, pois combinam elementos de busca direcionada e estocástica. Estas características tornam os AGs mais robustos que os métodos clássicos de busca. Outra importante propriedade deste método de busca é o processamento paralelo de diversas possíveis soluções do problema, enquanto outras técnicas trabalham apenas com um elemento do espaço de busca (MICHALEWICZ, 1996).

O conceito de AGs foi introduzido e desenvolvido por John Holland e colaboradores na Universidade de Michigan durante as décadas de 60 e 70, o qual resultou na publicação, em 1975, do livro *Adaptation in Natural and Artificial Systems*. O principal objetivo do seu estudo era aplicar a sistemas computacionais os mecanismos de adaptação presentes na seleção natural. Este método de otimização proposto por Holland consiste em transformar uma população de indivíduos (ou cromossomos) em uma nova população selecionando os indivíduos mais aptos (seleção natural) da geração e aplicando os operadores genéticos de cruzamento e mutação.

Segundo Negnevitsky (2005), AGs estão conectados aos problemas através de dois mecanismos: a codificação e a avaliação. A codificação é a representação de uma possível solução para o problema através de uma cadeia de caracteres (cromossomo), sendo o método mais comum a representação por uma cadeia de bits. A função de aptidão desempenha papel

semelhante ao meio ambiente na evolução natural, medindo a capacidade de cada indivíduo de se adaptar ao meio (neste caso, o problema a ser resolvido). Ao calcular a aptidão ou adaptação de cada cromossomo pertencente a uma população, é possível realizar a seleção dos indivíduos que participarão da reprodução, processo responsável pela criação de cromossomos descendentes que contém características dos cromossomos pais. Após diversas iterações do algoritmo, a população, que representa o conjunto de possíveis soluções, vai sendo transformada através de várias reproduções e convergirá para uma situação onde os cromossomos com os melhores valores de adaptação se tornarão predominantes na população, resultando na otimização desejada.

Como todo método de busca e otimização, os AGs possuem um espaço de soluções, onde são consideradas todas as possibilidades de solução para um determinado problema, e uma função de custo responsável por avaliar cada elemento do espaço de soluções. Em técnicas de busca e otimização tradicionais, o algoritmo inicia-se com um único candidato à solução que, iterativamente, é manipulado através de algumas heurísticas diretamente associadas ao problema a ser solucionado. Por outro lado, um algoritmo genético opera paralelamente sobre uma população de candidatos, proporcionando uma busca em diferentes áreas do espaço de soluções.

5.1 Elementos básicos dos Algoritmos Genéticos

Conforme vimos, AGs são métodos de busca inspirados na teoria biológica da evolução, portanto, muitos termos empregados em sua teoria tem origem em elementos presentes na Genética.

Um dos conceitos emprestados da Genética é o de cromossomos. Cromossomos representam as possíveis soluções dentro do espaço de busca. Na Biologia, cromossomos são cadeias de DNA (em inglês ***d****esoxyribo****n****ucleic* ***a****cid*, ou em português ADN, **á**cido **d**esoxirribo**n**ucleico) responsáveis por determinar as características únicas de cada indivíduo

(MITCHELL, 1999). Um cromossomo é dividido em partes menores denominadas genes. Cada gene é responsável por codificar uma característica do individuo, por exemplo, a cor dos olhos. No contexto de AGs, o cromossomo é o componente principal e representa uma possível solução para o problema. A busca pelo espaço de soluções do problema simula o processo de evolução sobre uma população de cromossomos, os quais também são chamados de indivíduos. Assim como o cromossomo biológico, os cromossomos artificiais são formados por unidades menores, os genes. Cada cromossomo tem um número determinado de genes que descrevem a solução candidata e, geralmente, se apresentam na forma de uma cadeia de bits, podendo assumir os valores 0 ou 1 (Figura 10).

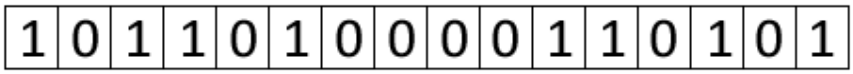


Figura 10: Representação de um cromossomo artificial de 16 bits

Algoritmos Genéticos realizam uma busca pelo espaço de soluções do problema através de operações que simulam a evolução natural. Em uma população, cada indivíduo é avaliado através de uma função de aptidão que indica o quão próximo ele está da solução do problema. Os melhores indivíduos são selecionados para formar uma nova geração a partir de seus descendentes. Estes descendentes serão mais adaptados ao meio, ou seja, estarão mais próximos da solução.

O exemplo a seguir, adaptado de (NEGNEVITSKY, 2005), demonstra de maneira simples o funcionamento de um algoritmo genético. O problema consiste em encontrar o valor máximo para a função abaixo:

$$f(x) = 15x - x^2, \qquad 0 \leq x \leq 15$$

Para simplificar o problema, os candidatos à solução são todos os valores inteiros que a variável *x* pode receber dentro do intervalo definido. Desta forma, os cromossomos podem ser codificados como cadeias de 4 bits. A função de adaptação, neste problema, é a própria função $f(x) = 15x - x^2$ a ser maximizada. Seguindo os critérios de seleção natural, os cromossomos mais adaptados terão mais chances de sobreviverem e, através da reprodução, gerar descendentes mais aptos que os pais. O esquema apresentado na Figura 11 ilustra o procedimento adotado pelo AG para encontrar o melhor valor que maximiza a função $f(x)$.

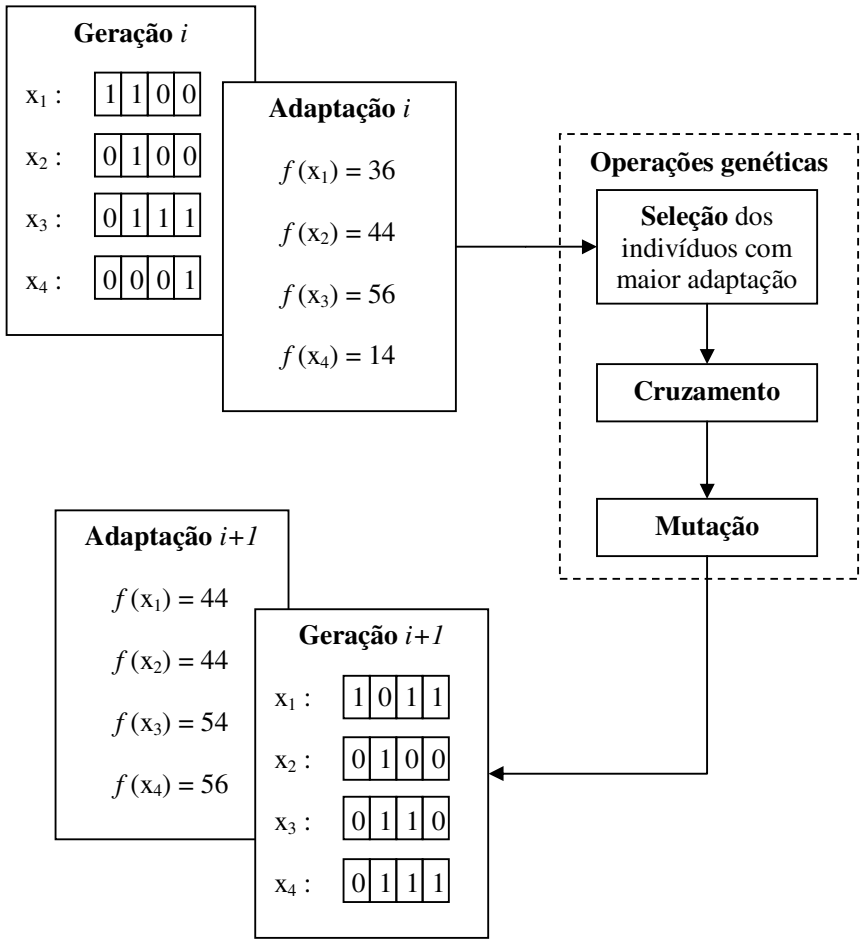


Figura 11: Ciclo de um algoritmo genético

Na Figura 11, são apresentadas as diversas etapas que envolvem a implementação da busca por AG. Estas etapas são denominadas *operadores genéticos*. Operadores genéticos são

os processos responsáveis por manipular a informação genética dos cromossomos, permitindo que novos indivíduos baseados na população original sejam produzidos. Segundo Mitchell (1999), na forma mais simples de um algoritmo genético, os operadores genéticos estão divididos em *seleção*, *cruzamento* e *mutação*.

5.2 Operadores genéticos

5.2.1 Seleção

O operador "seleção" é responsável por escolher, dentro de uma população, os cromossomos com os melhores valores de adaptação para realizar a reprodução, simulando o processo de sobrevivência do mais apto da natureza. Cada cromossomo tem probabilidade de ser selecionado proporcional à sua aptidão.

Existem vários métodos para selecionar os melhores cromossomos dentro de uma população (ASHLOCK, 2006). Entre eles podemos citar:

- **Seleção por torneio:** são formados, de maneira aleatória, vários subgrupos dentro de uma população e o melhor cromossomo de cada subgrupo é selecionado como cromossomo-pai;
- **Seleção por roleta:** é uma das técnicas de seleção mais usadas (GOLDBERG, 1989; DAVIS, 1991 *apud* NEGNEVITSKY, 2005, p. 225), onde os cromossomos-pais são escolhidos com probabilidade proporcional ao seu valor de adaptação. Seja um cromossomo i com adaptação f_i, então a probabilidade de i ser escolhido como cromossomo-pai é de f_i / F, onde F é a soma dos valores de adaptação de toda a população. No exemplo da Figura 12 é ilustrado o método de seleção por roleta. As maiores fatias da roleta correspondem aos cromossomos que obtiveram maior valor de aptidão;

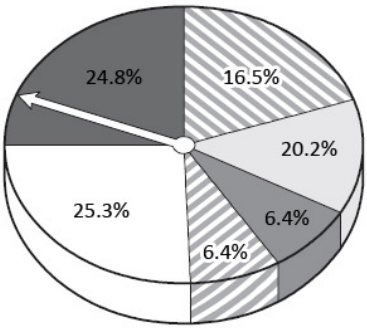


Figura 12: Seleção através do método "roleta"

- **Seleção por classificação:** funciona de maneira similar à roleta, porém os cromossomos são ordenados em função do seu valor de adaptação e a seleção é realizada de acordo com sua posição na lista de classificação.

5.2.2 Cruzamento

O cruzamento é o operador responsável pela troca de informação genética entre dois cromossomos, denominados cromossomos-pais, para geração de um terceiro cromossomo, chamado de cromossomo-filho. Na forma mais simples, o operador escolhe aleatoriamente uma posição no cromossomo, copiando para um dos filhos toda a informação presente antes do ponto escolhido e para o outro filho toda a informação presente depois desse ponto. A mesma operação é realizada no segundo cromossomo-pai para completar a informação genética dos dois filhos.

A operação de cruzamento é controlada pelo parâmetro p_c, chamado de probabilidade de cruzamento. Este parâmetro possibilita que os cromossomos-filhos de uma nova geração sejam cópias exatas de cada cromossomo-pai. Este processo é chamado de "elitismo", pois garante que os melhores indivíduos de uma população permaneçam nas gerações seguintes sem ter seus valores modificados. Os valores da probabilidade de cruzamento p_c geralmente estão entre 0.5 e 0.8 (MITCHELL, 1999), porém Negnevitsky (2005) afirma que o valor de 0.7 geralmente produz bons resultados.

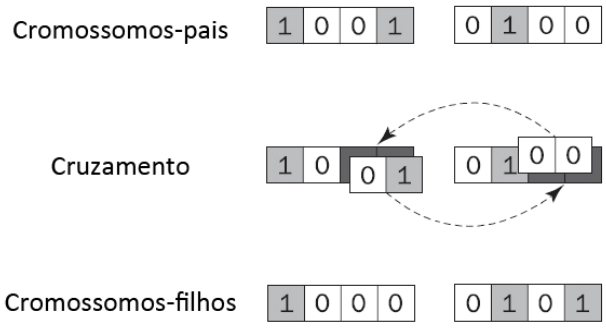


Figura 13: Funcionamento do operador Cruzamento Simples

A técnica de cruzamento ilustrada pelo exemplo da Figura 13 é chamada cruzamento em um ponto, ou cruzamento simples, pois é escolhido apenas um ponto de cruzamento. Existem, porém, vários tipos de operadores de cruzamento, tais como cruzamento em múltiplos pontos, cruzamento uniforme, cruzamento adaptativo, entre outros, que são necessários de acordo com a complexidade do problema ou da estrutura de dados usada para codificar as possíveis soluções do problema (ASHLOCK, 2006).

Em nosso trabalho, utilizaremos os operadores de cruzamento de dois pontos e uniforme. Suas operações serão descritas a seguir:

- **Cruzamento de dois pontos:** seleciona dois pontos aleatórios nos cromossomos-pais, dividindo o cromossomo em três partes. Os cromossomos-filhos são gerados através da concatenação das três partes dos cromossomos-pais. Um exemplo do funcionamento deste método de cruzamento é ilustrado na Figura 14.

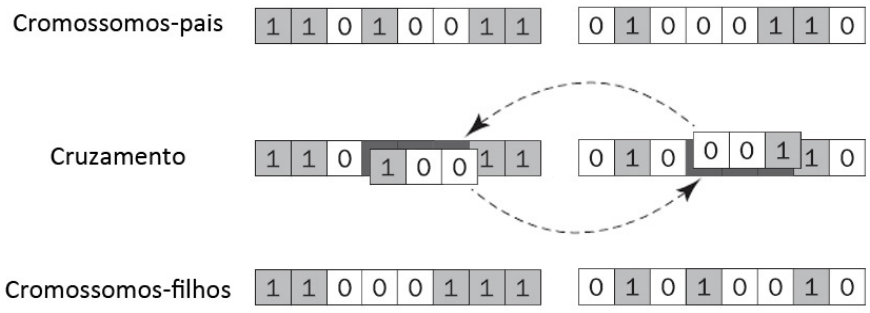


Figura 14: Operador Cruzamento de Dois Pontos

- **Cruzamento uniforme:** o método consiste na geração dos cromossomos-filhos através da seleção aleatória dos genes dos cromossomos-pais. Inicialmente, um vetor binário, denominado máscara de cruzamento, é gerado aleatoriamente. Para cada bit com valor 1 neste vetor, são copiados para o primeiro cromossomo-filho os valores do primeiro cromossomo-pai e para cada bit com valor 0, são copiados os valores do segundo cromossomo-pai. O segundo cromossomo-filho é gerado de maneira semelhante, os bits com valor 1 na máscara de cruzamento copiam as informações do segundo cromossomo-pai e os bits com valor 0 copiam as informações do primeiro cromossomo-pai. Este operador é ilustrado na Figura 15.

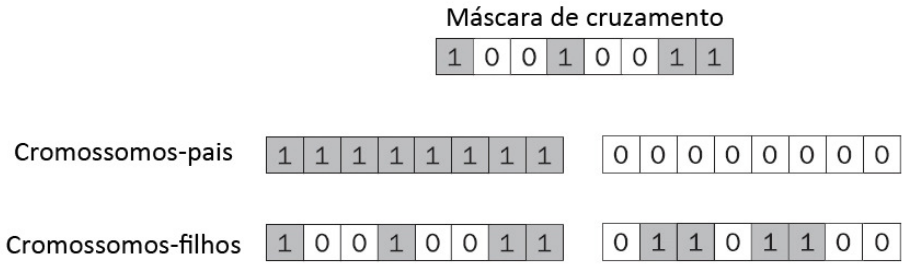


Figura 15: Operador Cruzamento Uniforme

5.2.3 Mutação

O processo de mutação realiza a troca arbitrária de um ou mais bits pertencentes a um cromossomo. Enquanto o cruzamento realiza combinações entre pares de diferentes cromossomos, permitindo uma maior exploração do espaço de busca das soluções do problema, a mutação faz pequenas mudanças no cromossomo. A mutação tem a função de facilitar a busca local e introduzir novos elementos, garantindo a diversidade genética da população (ASHLOCK, 2006).

O processo de mutação é controlado pelo parâmetro p_m que determina a probabilidade de mutação. De maneira semelhante à natureza, onde a mutação ocorre raramente, a operação de mutação em algoritmos genéticos também possui baixa probabilidade de ocorrer, ficando

geralmente entre 0.001 e 0.01 (NEGNEVITSKY, 2005). Caso a probabilidade de mutação seja um valor muito alto, o algoritmo genético pode ser prejudicado, pois se tornaria um simples processo de busca aleatória.

Existem vários operadores para o processo de mutação. Na Figura 16, é ilustrado o funcionamento de um operador mais simples, mutação em um ponto, que modifica apenas um valor do cromossomo em uma posição gerada aleatoriamente. Uma vez que o ponto de mutação é calculado, diferentes tipos de operadores podem ser aplicados, dependendo da complexidade das estruturas de dados usadas no problema. Mutação em múltiplos pontos, mutação probabilística, mutação Lamarckiana e mutação nula são alguns exemplos de operadores de mutação existentes (ASHLOCK, 2006).

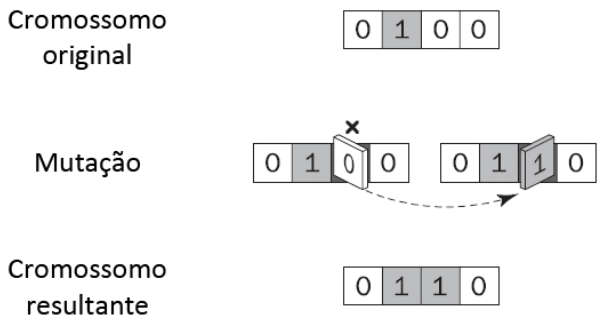


Figura 16: Funcionamento do operador Mutação de um Ponto

5.3 Implementação de um algoritmo genético simples

O algoritmo genético é um processo iterativo, onde cada iteração é chamada de geração. Para a implementação de um algoritmo genético simples, é necessário definir claramente o problema a ser resolvido e representar na forma de uma cadeia de bits as possíveis soluções. O AG pode ser descrito conforme os passos descritos no Quadro 1 (MITCHELL, 1999; NEGNEVITSKY, 2005).

Para Mitchell (1999), o número de gerações utilizado como critério de parada durante a execução de um algoritmo genético está, geralmente, entre 50 a 500. No final da execução,

espera-se encontrar um ou mais cromossomos na população com melhores valores de adaptação.

Quadro 1: Passos para implementação de um Algoritmo Genético Simples

1. Representar os candidatos a solução do problema como um cromossomo de tamanho fixo l, escolher o tamanho N da população de cromossomos, a probabilidade de cruzamento p_c e a probabilidade de mutação p_m;
2. Definir uma função de aptidão para avaliar os cromossomos candidatos à solução;
3. Gerar aleatoriamente uma população com N cromossomos de tamanho l:

   $x_1; x_2;...; x_n$
4. Calcular a adaptação de cada cromossomo x:

   $f(x_1), f(x_2),..., f(x_n)$
5. Repetir os passos a seguir até que N cromossomos filhos sejam gerados para a nova população:
   a. Selecionar um par de cromossomos-pais da população atual;
   b. Gerar dois cromossomos-filhos através do cruzamento e mutação. Com a probabilidade p_c, é realizado o cruzamento dos cromossomos-pais. Em seguida, com a probabilidade p_m, é realizada a mutação dos cromossomos filhos. Os cromossomos gerados são introduzidos na nova população.
6. Substituir a população atual pela nova população;
7. Voltar para o passo 4 e repetir o processo até que o critério de parada seja satisfeito.

Ao observar o fluxograma apresentado na Figura 17 e a sequência de passos definida no Quadro 1, percebemos que a execução de um AG depende de vários parâmetros, tais como tamanho da população, probabilidade de cruzamento, probabilidade de mutação, entre outros. A escolha correta desses parâmetros pode afetar o desempenho e o sucesso do processo de

otimização realizado pelo algoritmo genético. Resolver um problema usando AGs envolve a definição adequada dos parâmetros e critérios de parada, a codificação das soluções do problema como cromossomos, definição de uma função de avaliação e a criação apropriada de operadores de cruzamento e mutação (NEGNEVITSKY, 2005).

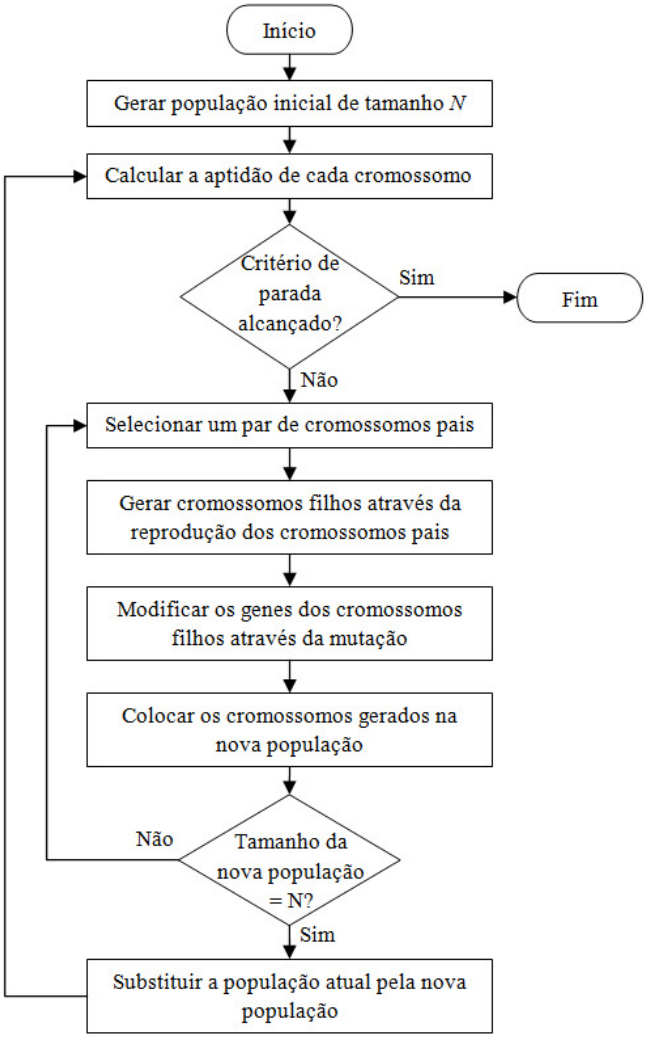


Figura 17: Fluxograma de um Algoritmo Genético Simples

5.3.1 Exemplo de aplicação de um algoritmo genético para otimização

Definidos os principais conceitos que envolvem a implementação de um algoritmo genético, podemos demonstrar o funcionamento de todos os operadores no exemplo citado

anteriormente. Inicialmente, temos a função de avaliação definida como $f(x) = 15x - x^2$ e queremos obter o x que maximiza esta função dentro do intervalo $0 \leq x \leq 15$.

A primeira etapa, segundo o fluxograma da Figura 17, é gerar uma população inicial de tamanho N. Escolhendo N = 6, geramos aleatoriamente uma população inicial com seis cromossomos. Em seguida, é calculado o valor de aptidão de cada cromossomo da população. Uma possível população inicial é apresentada no Quadro 2 com seus valores de adaptação.

Quadro 2: Exemplo de uma população inicial para um problema de maximização

Valor inteiro	Representação binária	Adaptação do cromossomo
12	1100	36
4	0100	44
1	0001	14
14	1110	14
7	0111	56
9	1001	54

Na etapa seguinte, é realizada a seleção dos cromossomos-pais, onde os cromossomos mais adaptados terão mais chances de sobreviver e, através da reprodução, gerar descendentes mais aptos que os pais. No exemplo apresentado no Quadro 2, os candidatos a solução 4, 7 e 9 terão maior probabilidade de serem selecionados e passarem seus genes (bits) para as próximas gerações. Assim, realizando a seleção aleatória de três pares de cromossomos, é possível realizar o cruzamento e, posteriormente, a mutação de cada cromossomo-filho gerado, conforme mostra o Quadro 3.

Quadro 3: Operação de cruzamento e mutação em um problema de maximização

Cromossomos-pais		Cromossomos-filhos			
Pai 1	**Pai 2**	**Filho 1**	**Mutação**	**Filho 2**	**Mutação**
1001	0100	1000	1000	0101	0101
1100	0111	1111	1011	0100	0100
0100	0111	0100	0110	0111	0111

A população é então substituída pelos novos cromossomos gerados através dos processos de cruzamento e mutação. No exemplo apresentado no Quadro 4, observamos que os novos cromossomos da população possuem, em média, valores de adaptação superiores aos valores de adaptação encontrados na população anterior.

O processo de otimização da função de avaliação continua por algumas gerações até que um critério de parada seja atingido. Este critério de parada pode ser definido como um número máximo de gerações ou quando a população não apresentar variação entre algumas gerações.

Quadro 4: Exemplo da nova população gerada após cruzamento e mutação

Valor inteiro	Representação binária	Adaptação do cromossomo
8	1000	56
5	0101	50
11	1011	44
4	0100	44
6	0110	54
7	0111	56

5.4 Ferramenta de Algoritmos Genéticos do MATLAB

Nesta pesquisa, utilizamos o *toolbox* de Algoritmos Genéticos do MATLAB para implementar e executar o algoritmo genético responsável pela otimização das funções de avaliação. O MATLAB, ou ***Mat****rix* ***Lab****oratory*, é um software interativo de alto desempenho voltado para o desenvolvimento de algoritmos, visualização de dados, análise de dados e computação numérica. O software possui uma grande variedade de *toolboxes*, que são coleções de funções para diferentes aplicações como processamento de sinais, processamento de imagens, redes neurais, entre outros.

Para a aplicação de AGs, o MATLAB também possui um *toolbox* específico, o GAtool (***G****enetic* ***A****lgorithm* ***Tool****box*). O GAtool possui funções que implementam os

processos mais importantes para a execução de um AG, como os operadores de seleção, cruzamento e mutação.

A seguir, usamos o exemplo descrito anteriormente para demonstrar o funcionamento do processo de otimização da função de avaliação através de um algoritmo genético implementado no programa MATLAB.

Inicialmente, é necessário criar uma função descrevendo a função de adaptação do algoritmo genético. O Quadro 5 mostra o script *fitness.m* para a função de adaptação do exemplo de maximização. A função calcula a aptidão de um cromossomo, representado pela variável de entrada x, e retorna o valor desta aptidão. Na linha 3, podemos observar que este valor recebe um sinal negativo, pois o *toolbox* de algoritmos genéticos do MATLAB realiza apenas problemas de minimização.

Quadro 5: Exemplo de uma função de aptidão escrita no MATLAB

Função *fitness.m*

```
1  function y = fitness(x)
2  f = 15*x - x^2;
3  y = -f;
4  End
```

Em seguida, desenvolvemos uma função que define os parâmetros desejados para a execução do AG (Quadro 6). Esta função, *GAfunction.m*, tem como parâmetros de entrada o número de variáveis que devem ser otimizados (*nvars*), os valores que estas variáveis podem receber inicialmente (*PopInitRange*), o tamanho da população (*PopulationSize*) e o número máximo de gerações (*Generations*). Os parâmetros relevantes de saída são as variáveis otimizadas (representadas pelo vetor *x*), o valor da função de adaptação das variáveis otimizadas (*fval*), a população final (*population*) e os valores de adaptação da população final (*score*). As demais linhas do código definem outros parâmetros do AG, como os operadores de seleção, cruzamento, mutação e a função de adaptação *fitness.m* do Quadro 5.

Quadro 6: Implementação de um AG utilizando funções do GAtool do MATLAB

Função *GAfunction.m*

```
function [x,fval,exitflag,output,population,score] = ...
GAfunction(nvars,PopInitRange,PopulationSize,Generations)
options = gaoptimset;
options = gaoptimset(options,'PopulationType', 'doubleVector');
options = gaoptimset(options,'PopInitRange', PopInitRange);
options = gaoptimset(options,'PopulationSize',PopulationSize);
options = gaoptimset(options,'Generations', Generations);
options = gaoptimset(options,'CreationFcn', @gacreationuniform);
options = gaoptimset(options,'SelectionFcn', @selectionroulette);
options = gaoptimset(options,'CrossoverFcn',@crossoversinglepoint);
options = gaoptimset(options,'MutationFcn', {@mutationuniform []});
options = gaoptimset(options,'Display', 'off');
options = gaoptimset(options,'Vectorized', 'off');
[x,fval,exitflag,output,population,score] = ...
ga(@fitness,nvars,[],[],[],[],[],[],[],options);
end
```

No Quadro 7, é ilustrada uma sequência de comandos para executar a otimização do problema de maximização proposto. O resultado da otimização utilizando este algoritmo são valores entre 7 e 8, com adaptação por volta de 56,24.

Quadro 7: Script para execução do AG do exemplo de maximização

```
% parâmetros do algoritmo genético
n = 1;           % número de variáveis
range = [0;15]; % variação dos parâmetros
pop = 6;         % tamanho da população
geracoes = 100; % número máximo de gerações

[x,fit,f,out,p,score] = GAfunction(n,range,pop,geracoes);
```

6 MATERIAIS E MÉTODOS

Neste capítulo descreveremos a metodologia adotada para o desenvolvimento do trabalho proposto nesta dissertação. Para otimizar e, posteriormente, avaliar a função de avaliação para o jogo de dominó de 4 pontas, foram realizadas as seguintes etapas descritas na Figura 18 abaixo. Nas seções a seguir, descreveremos detalhadamente cada etapa envolvida nesta pesquisa.

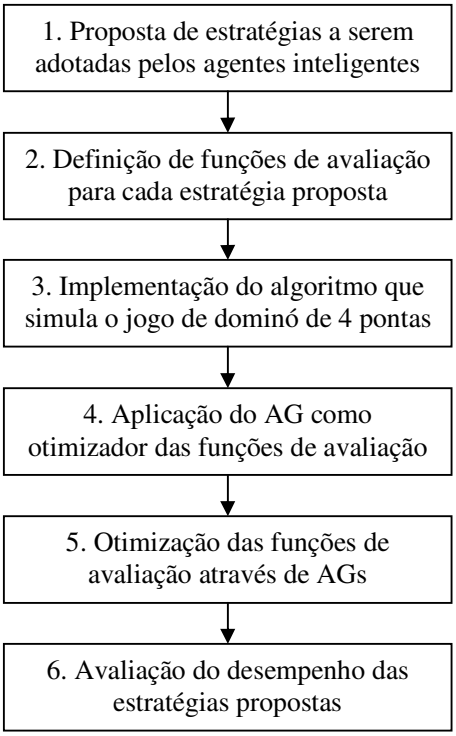


Figura 18: Etapas para o desenvolvimento do trabalho proposto na dissertação

6.1 Proposta de estratégias a serem adotadas pelos agentes inteligentes

Conforme vimos no Capítulo 3, em uma partida de Dominó de 4 pontas, o jogador dispõe de diferentes maneiras para obter a pontuação necessária para ganhar uma partida. Na sua vez de jogar, uma pessoa pode escolher uma peça por ela proporcionar uma grande pontuação na mesa, ou pela possibilidade de fazer o jogador seguinte passar, ou ainda por ser

uma peça que ajuda o seu parceiro de dupla. Determinar a importância que cada uma destas ações representa para a escolha de uma jogada é o que define a estratégia do jogador.

Em (ANTONIO *et al*, 2008, 2009), foi analisada apenas uma estratégia para o agente inteligente onde a escolha das jogadas era baseada somente no que era melhor para o jogador ou no que poderia dificultar as ações adversárias. Nesta dissertação, porém, definimos cinco estratégias com prioridades distintas para a escolha da melhor jogada.

Definimos como "Estratégia 1" a estratégia que pondera entre as três possíveis ações que um jogador pode adotar para escolher a jogada: facilitar o próprio jogo, dificultar as jogadas adversárias e facilitar o jogo do parceiro de dupla. Na segunda estratégia, denominada "Estratégia 2", a escolha da jogada é baseada apenas no favorecimento do próprio jogador, ignorando tanto o parceiro de dupla quanto os adversários ao analisar as futuras jogadas. A "Estratégia 3" prioriza apenas as jogadas que podem favorecer o jogo do parceiro de dupla e a "Estratégia 4" considera apenas jogadas que dificultem o jogo dos adversários. Por fim, na quinta estratégia, a qual denominamos "Estratégia básica" por ser a estratégia utilizada por jogadores iniciantes, o jogador realiza a escolha de suas jogadas apenas pela pontuação que pode obter na mesa ou através da passagem do jogador seguinte.

6.2 Definição das funções de avaliação para cada estratégia proposta

As funções de avaliação que desenvolvemos para descrever cada estratégia proposta na seção anterior utilizam os mesmos conceitos previamente definidos no Capítulo 3 para a escolha da melhor jogada em (ANTONIO *et al*, 2008, 2009), ou seja, as possibilidades de pontuação P_1, P_2, P_3, P_4 e P_5 e os vetores de estado de jogo.

De forma resumida, a função de avaliação apresentada no Capítulo 4 é formada pelas equações apresentadas no Quadro 8. Esta função, formada por um termo responsável pelos pontos obtidos na mesa (T_{n1}) e por um termo responsável pela estratégia de jogo (T_{n2}), considera apenas as jogadas que beneficiam o próprio jogador e as jogadas que prejudicam os

adversários, conforme mencionamos anteriormente. Outra importante observação relativa a esta função refere-se ao termo E_3, que não possui relevância para a escolha das jogadas, pois em testes realizados em (ANTONIO *et al*, 2008, 2009), observamos que a variação sofrida pelo parâmetro K_3 não influenciava o desempenho final da dupla que utilizava a função de avaliação.

Quadro 8: Função de avaliação desenvolvida em (ANTONIO *et al*, 2008)

$f(\mathrm{n}) = T_{\mathrm{n1}} + T_{\mathrm{n2}}$

onde:

$T_{\mathrm{n1}} = P_1 + P_2 + P_3 + P_5$

$T_{\mathrm{n2}} = \alpha.(- E_1 + E_2 + \delta.E_3)$

$E_1 = K_1 . (V_{0L_1} + V_{1L_1} + V_{2L_1})$

$E_2 = K_2 . (V_{0L_2} + V_{1L_2} + V_{2L_2})$

$E_3 = K_3 . \delta . V_{1L_2}$, onde $\delta = 1$ se $\sum V_{1\mathrm{i}} > 3$ e $\delta = 0$ caso contrário.

Com base nestas informações, inicialmente definimos a função de avaliação para a Estratégia 1 de maneira semelhante à função apresentada no Quadro 8, porém incluímos um componente responsável por analisar as jogadas do parceiro de dupla na equação E_2. Este componente é representado pelo vetor V_3 (quantidade de peças jogadas pelo parceiro) dos estados de jogo.

A função de avaliação que descreve a Estratégia 1 é exibida no Quadro 9. Nesta função, o termo T_{n2}, que representa a estratégia de jogo, possui coeficientes distintos para cada elemento que compõe E_1 e E_2, respectivamente, preocupação com os adversários e preocupação com a dupla. Estes coeficientes, denominados α_1, α_2, α_3, α_4, α_5, α_6 e α_7,

representam os valores que serão determinados pelo Algoritmo Genético no processo de otimização.

Quadro 9: Função de Avaliação para a Estratégia 1

$f(\mathrm{n}) = T_{\mathrm{n1}} + T_{\mathrm{n2}}$

onde:

$T_{\mathrm{n1}} = P_1 + P_2 + P_3 + P_5$

$T_{\mathrm{n2}} = -E_1 + E_2$

$E_1 = \alpha_1.V_{0L_1} + \alpha_2.V_{1L_1} + \alpha_3.V_{2L_1}$

$E_2 = \alpha_4.V_{0L_2} + \alpha_5.V_{1L_2} + \alpha_6.V_{2L_2} + \alpha_7.V_{3L_2}$

Na Estratégia 2, apenas as jogadas que favorecem o jogador tem prioridade. Portanto, os coeficientes α_1, α_2, α_3 (controlam os termos que dificultam as jogadas dos adversários) e α_7 (controla o termo que analisa as jogadas do parceiro) tem valor fixo igual a zero. A função de avaliação que representa a Estratégia 2 é exibida no Quadro 10. Neste caso, a otimização através de algoritmo genético deve encontrar os melhores coeficientes α_4, α_5 e α_6 para esta estratégia.

Quadro 10: Função de Avaliação para a Estratégia 2

$f(\mathrm{n}) = T_{\mathrm{n1}} + T_{\mathrm{n2}}$

onde:

$T_{\mathrm{n1}} = P_1 + P_2 + P_3 + P_5$

$T_{\mathrm{n2}} = -E_1 + E_2$

$E_1 = 0$

$E_2 = \alpha_4.V_{0L_2} + \alpha_5.V_{1L_2} + \alpha_6.V_{2L_2} + 0$

Na Estratégia 3, o jogador dá maior prioridade apenas às jogadas que podem favorecer o jogo do seu parceiro de dupla. Neste caso, os coeficientes α_1, α_2, α_3 (dificultam as jogadas dos adversários) e α_5 (analisa as jogadas que facilitam seu próprio jogo) são definidos com o valor zero. Assim, a função de avaliação a ser otimizada pelo algoritmo genético para a Estratégia 3 é dada pela equação do Quadro 11.

Quadro 11: Função de Avaliação para a Estratégia 3

$f(\mathrm{n}) = T_{\mathrm{n1}} + T_{\mathrm{n2}}$

onde:

$T_{\mathrm{n1}} = P_1 + P_2 + P_3 + P_5$

$T_{\mathrm{n2}} = E_2$

$E_2 = \alpha_4.V_{0L_2} + \alpha_6.V_{2L_2} + \alpha_7.V_{3L_2}$

A Estratégia 4 dá prioridade às jogadas que dificultam as ações dos adversários, ou seja, os coeficientes α_4, α_5, α_6 e α_7 (facilitam as jogadas da dupla) são definidos com o valor zero. A função de avaliação a ser otimizada pelo Algoritmo Genético para a Estratégia 4 é formada apenas pelos termos apresentados no Quadro 12.

Quadro 12: Função de Avaliação para a Estratégia 4

$f(\mathrm{n}) = T_{\mathrm{n1}} + T_{\mathrm{n2}}$

onde:

$T_{\mathrm{n1}} = P_1 + P_2 + P_3 + P_5$

$T_{\mathrm{n2}} = - E_1$

$E_1 = \alpha_1.V_{0L_1} + \alpha_2.V_{1L_1} + \alpha_3.V_{2L_1}$

Para a Estratégia Básica, na qual são considerados para a escolha da melhor jogada apenas os pontos obtidos na mesa e pontos de passagem, os coeficientes α_1, α_2, α_3, α_4, α_5, α_6 e α_7 são definidos com o valor igual a zero. Assim, a função de avaliação relativa a esta estratégia é descrita conforme o Quadro 13 e não possui coeficientes a serem otimizados.

Quadro 13: Função de Avaliação para a Estratégia básica

$f(\mathrm{n}) = T_{\mathrm{n1}} + T_{\mathrm{n2}}$

onde:

$T_{\mathrm{n1}} = P_1 + P_2 + P_3 + P_5$

$T_{\mathrm{n2}} = 0$

6.3 Implementação do algoritmo que simula o jogo de dominó de 4 pontas

Para simular o jogo de dominó de 4 pontas, desenvolvemos um programa em Java, o qual pode ser facilmente executado pela máquina virtual Java presente no MATLAB. O simulador permite a realização de *n* partidas seguindo as regras mais comuns do dominó jogado no Amazonas. Os agentes inteligentes, representando os quatro jogadores, possuem funções independentes para a escolha de suas jogadas. Os coeficientes das funções de avaliação são determinados pelo usuário através dos parâmetros de entrada do simulador.

No Quadro 14 são apresentados comandos no MATLAB para execução do simulador do jogo de dominó. Inicialmente, definimos os coeficientes α_1, α_2, α_3, α_4, α_5, α_6 e α_7 para os jogadores pertencentes às Duplas 1 e 2 através dos vetores *alfa1* e *alfa2*, ou seja, ambas as duplas utilizaram coeficientes iguais a zero neste exemplo. Em seguida, a quantidade de jogos é definida como 1000 partidas e utilizada como parâmetro de entrada pelo método *UsaDomino.realizaJogo*. A saída é definida como o número de partidas vencidas pela Dupla 2.

Quadro 14: Utilização do simulador do jogo de dominó de 4 pontas no MATLAB

```
>> import domino_ga.*
>> jogo = UsaDomino;
>> % definição dos parâmetros 'alfa'
>> alfa1 = [0 0 0 0 0 0 0];
>> alfa2 = [0 0 0 0 0 0 0];
>> % definição da quantidade de jogos
>> qtdade_jogos = 1000;
>> % utilização do método 'realizaJogo' da classe 'UsaDomino'
>> % e retorna o número de vitórias da dupla 2
>> vitorias = jogo.realizaJogo(qtdade_jogos,alfa1,alfa2)

vitorias =

   502
```

6.4 Aplicação do AG como otimizador das funções de avaliação

A aplicação do AG como método de otimização dos coeficientes que formam a função de avaliação proposta para o jogo de dominó de 4 pontas foi realizada com o auxílio do simulador desenvolvido na seção anterior e com funções e *scripts* que descrevem o funcionamento do algoritmo. O esquema da Figura 19 apresenta o funcionamento da otimização por algoritmo genético executada nesta pesquisa.

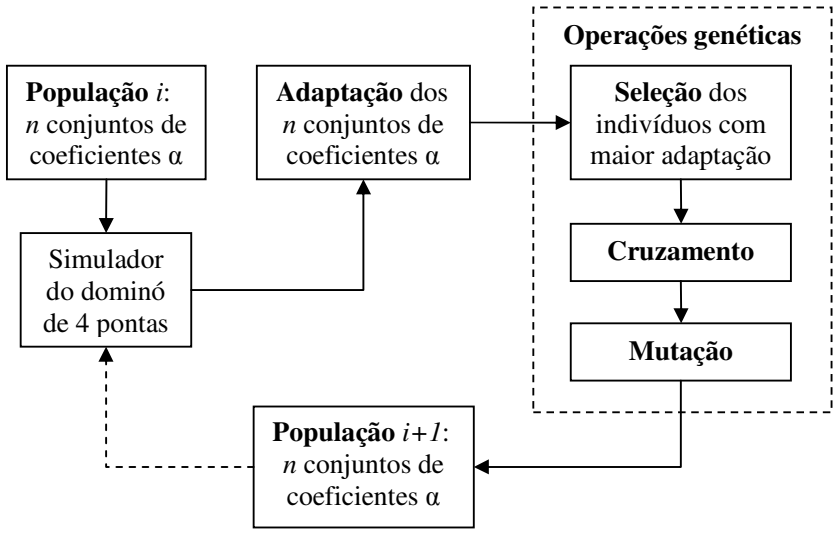


Figura 19: Esquema do processo de otimização dos coeficientes da função de avaliação

Antes de iniciar a implementação no MATLAB, foi necessário definir os parâmetros que seriam utilizados pelo AG durante a otimização dos coeficientes. O desempenho de um algoritmo genético depende da escolha certa de seus parâmetros, que variam de acordo com o tipo de problema. Para o problema proposto nesta dissertação, o algoritmo genético foi executado com diferentes parâmetros, conforme detalhados nas subseções a seguir.

6.4.1 Codificação do cromossomo

Para esta aplicação, os cromossomos serão codificados na forma de um vetor do tipo *Double*, ou seja, os cromossomos representam valores pertencentes ao conjunto dos números reais. Cada cromossomo é um vetor formado por sete genes, os quais representam os coeficientes α_1, α_2, α_3, α_4, α_5, α_6 e α_7 da função de avaliação.

6.4.2 Definição do tamanho da população

É um dos parâmetros que afeta diretamente o desempenho do algoritmo. Populações muito pequenas tem menor diversidade genética e podem levar a uma convergência mais rápida do algoritmo para um mínimo local, por outro lado, populações muito grandes deixam o algoritmo excessivamente lento (ASHLOCK, 2006). Para a otimização realizada neste trabalho, utilizamos os seguintes tamanhos de população: 40, 60, 80 e 100.

6.4.3 Definição dos operadores de cruzamento e mutação e o critério de parada

Para o processo de otimização, foram usadas duas técnicas de Cruzamento: “Dois pontos” e “Uniforme”. A taxa de cruzamento p_c que define a probabilidade com que os cromossomos selecionados passarão pelo cruzamento será de 80%. Para a execução da operação de mutação, o operador escolhido foi o de Mutação Uniforme e o valor de probabilidade de mutação, denominado p_m, recebe os valores de 0,01 e 0,05.

O critério de parada adotado para o Algoritmo Genético será o número total de gerações. Nesta pesquisa, o critério de parada será quando o algoritmo genético alcançar 200 gerações.

6.4.4 Implementação da Função de aptidão

Esta função é responsável por calcular o desempenho de cada cromossomo de uma população. Conforme mencionado anteriormente, esta adaptação é dada pelo número de vitórias da Dupla 2 em um total de *n* partidas. Inicialmente, as partidas são realizadas entre uma dupla que utiliza a Estratégia básica (Dupla 1) e uma dupla que utiliza a estratégia definida pelo cromossomo (Dupla 2).

Nas seções 4.2 e 4.3, definimos cinco estratégias e as funções de avaliação que as representam. O objetivo é otimizar as Estratégias 1, 2, 3 e 4, portanto, são necessárias quatro funções de adaptação, uma para cada estratégia. A função de adaptação da Estratégia 1 foi definida pela função *fitnessFcn1.m* e seu algoritmo é apresentado no Quadro 15.

Quadro 15: Código-fonte da função de Adaptação para a Estratégia 1

Função *fitnessFcn1.m*

```
1   function y = fitnessFcn1(x)
2
3   % parâmetros para a dupla 1
4   alfa_dupla1 = zeros(7,1);
5
6   % parâmetros para a dupla 2
7   alfa_dupla2 = x';
8
9   import domino_ga.*
10  jogo = UsaDomino;
11
12  jogos = 5000;
13  vitorias = ...
14      jogo.realizaJogo(jogos,alfa_dupla1,alfa_dupla2);
15  y = -vitorias;
16  end
```

Na função apresentada no Quadro 15, o parâmetro de entrada *x* representa um cromossomo contendo os sete coeficientes para a Dupla 2. Na linha 4, atribuímos o valor 0 aos coeficientes da Dupla 1 e na linha 7, a Dupla 2 recebe os coeficientes definidos pelo vetor

de entrada *x*. Nas linhas 9 e 10, a classe contendo o simulador para o jogo de dominó em Java é importado e uma instância desta classe é criada para execução das partidas. As últimas linhas de comando definem o número total de partidas que serão realizadas (5000) e a obtenção do número de vitórias resultantes das partidas realizadas. A saída é negativa, pois o GAtool do MATLAB realiza apenas operações de minimização.

Para as demais estratégias, a função de adaptação é semelhante à ilustrada no Quadro 15. Nestas funções, porém, o vetor de coeficientes da Dupla 2 possui componentes iguais a 0, de acordo com a estratégia utilizada. As funções desenvolvidas para estas estratégias encontram-se no Apêndice A.

6.4.5 Implementação de funções no MATLAB para otimização

Conforme mencionamos anteriormente, utilizamos várias combinações de parâmetros para executar o AG com o objetivo de encontrar uma combinação que permitisse alcançar valores ótimos ou quase-ótimos para os coeficientes α_1, α_2, α_3, α_4, α_5, α_6 e α_7 das funções de avaliação. O conjunto de parâmetros definidos nas subseções anteriores nos fornece uma combinação de 16 diferentes tipos de algoritmos para otimizar cada estratégia, totalizando 64 execuções do algoritmo genético, conforme apresentado no Quadro 16.

Quadro 16: Grupos de parâmetros utilizados para otimização por Algoritmos Genéticos

Grupo de Simulações	Função de cruzamento	Probabilidade de mutação	Tamanho da população	Gerações
1	Dois pontos	0,01	40, 60, 80 e 100	200
2		0,05		
3	Uniforme	0,01		
4		0,05		

A partir destas combinações de parâmetros, foi desenvolvido um *script* no Matlab para execução do algoritmo genético, denominado *dominoGA.m*. O código desta função é apresentado no Quadro 17.

Quadro 17: Script para execução do Algoritmo Genético

Função *dominoGA.m*

```
1   function [x,fitness] = ...
2   dominoGA(n,Strategy,PopRange,PopSize,EliteCount,Generations,Crossover2P,
3   MutationRate)
4
5   % Define as opções padrão do Matlab para um algoritmo genético
6   options = gaoptimset;
7
8   options = gaoptimset(options,'PopInitRange', PopRange);
9   options = gaoptimset(options,'PopulationSize', PopSize);
10  options = gaoptimset(options,'EliteCount', EliteCount);
11  options = gaoptimset(options,'Generations', Generations);
12  options = gaoptimset(options,'CreationFcn', @gacreationuniform);
13  options = gaoptimset(options,'MutationFcn', { @mutationuniform
14  MutationRate });
15  options = gaoptimset(options,'Display', 'diagnose');
16  options = gaoptimset(options,'Vectorized', 'off');
17  options = gaoptimset(options,'UseParallel', 'always');
18
19  if (Crossover2P == 1) % twopoint
20      options = gaoptimset(options,'CrossoverFcn', @crossovertwopoint);
21  end
22
23  if (Strategy == 1) % Estratégia 1
24      [x,fitness] = ga(@fitnessFcn1,n,[],[],[],[],[],[],[],options);
25  elseif (Strategy == 2) % Estratégia 2
26      [x,fitness] = ga(@fitnessFcn2,n,[],[],[],[],[],[],[],options);
27  elseif (Strategy == 3) % Estratégia 3
28      [x,fitness] = ga(@fitnessFcn3,n,[],[],[],[],[],[],[],options);
29  else % Estratégia 4
30      [x,fitness] = ga(@fitnessFcn4,n,[],[],[],[],[],[],[],options);
31  end
32  end
```

O *script* descrito no Quadro 17 foi criado com o auxílio das funções pertencentes ao *toolbox* GAtool do Matlab e possui os seguintes parâmetros de entrada:

- Parâmetro *n*: Indica o número de variáveis que serão otimizadas pelo algoritmo genético. Para a Estratégia 1, *n* recebe valor 7 e para as demais estratégias, 3;
- Parâmetro *Strategy*: A definição da estratégia que será adotada pela Dupla 2 permite a definição da função de avaliação que será utilizada pelo algoritmo genético (linhas 23 a 31 do código);
- Parâmetro *PopRange*: Indica o intervalo de valores que as variáveis podem receber. O intervalo utilizado nesta pesquisa foi de -10 a 10;
- Parâmetro *PopSize*: Indica o tamanho da população;

- Parâmetro *EliteCount*: Indica quantos indivíduos da população atual, com os melhores valores de aptidão, serão transmitidos diretamente para a próxima geração sem passar pela reprodução. Nas simulações realizadas, este parâmetro recebeu valores correspondentes a 10% do tamanho da população;
- Parâmetro *Generations*: Indica o número máximo de gerações, ou seja, 200;
- Parâmetro *Crossover2P*: Indica se o algoritmo genético utilizará o operador *twopoint crossover.* Caso contrário, o cruzamento é realizado com o operador padrão do GAtool, *scattered crossover*;
- Parâmetro *MutationRate*: Define a probabilidade de mutação do algoritmo genético;
- Também são feitas outras configurações através da função *gaoptimset* que permite a criação e alteração das opções do algoritmo genético que o GAtool disponibiliza.

Além das funções descritas acima, uma função adicional foi implementada para comandar as execuções do algoritmo genético. A função *runDominoGA.m* (Apêndice A) tem como parâmetros de entrada a Estratégia desejada para otimização, o tamanho da população e o grupo de parâmetros que serão utilizados no algoritmo genético (Quadro 16).

Com os *scripts* descritos nesta seção, podemos executar a otimização das funções de avaliação através de um simples comando no MATLAB, conforme descrito no Quadro 18 abaixo. Neste exemplo, foram usados como parâmetros de entrada a função de avaliação relativa à Estratégia 1 (primeiro parâmetro), uma população com 10 indivíduos (segundo parâmetro) e a configuração 3 de parâmetros do algoritmo genético, que corresponde ao operador de cruzamento uniforme e probabilidade de mutação de 1%, conforme indicado no Quadro 16. A função retorna o tempo total de execução do algoritmo genético, os coeficientes

α_i otimizados (vetor *x*) e o número de vitórias obtidas com esta combinação de parâmetros (variável *adaptacao*).

Quadro 18: Comando no MATLAB para execução da otimização da função de avaliação

```
>> [x fitness] = runDominoGA(1,10,3)

Optimization terminated: maximum number of generations exceeded.
Elapsed time is 688.153225 seconds.

x =

    5.2311   -4.1830   -4.1425   -8.4901    2.9046    8.4806    0.1678


adaptacao =

       -3382
```

6.5 Otimização das funções de avaliação através de algoritmos genéticos

A quinta etapa da pesquisa envolve a execução do algoritmo genético para otimizar as funções de avaliação através das funções implementadas nas seções anteriores. Para executar estas tarefas, utilizamos um microcomputador com processador Intel Core i3 @ 2.13 GHz e 4 GB de RAM, operando sob a plataforma Microsoft Windows 7. A otimização foi realizada no programa MATLAB R2010a operando com a versão 1.6.0_22 da máquina virtual Java.

6.5.1 Metodologia para otimização das funções de avaliação

Para a busca por Algoritmos Genéticos, é necessário que uma das duplas utilize uma estratégia com função de avaliação fixa, enquanto a outra dupla utiliza a estratégia cuja função de avaliação será otimizada. No início, não há uma função de avaliação pré-estabelecida para as estratégias 1, 2, 3 ou 4, portanto, optamos por otimizá-las usando a Estratégia básica como a função fixa, por não possuir coeficientes a serem otimizados.

Após 200 iterações do algoritmo genético, as quais resultam em coeficientes otimizados para as estratégias 1, 2, 3 e 4, realizamos novas otimizações destas estratégias com

o objetivo de encontrar funções cujo desempenho seja superior às funções encontradas nas etapas anteriores. A sequência de passos a seguir descreve cada etapa que envolve o processo de otimização das estratégias para o jogo de dominó de 4 pontas:

- Etapa 1: Para otimizar cada estratégia, uma dupla (Dupla 2) utiliza uma das estratégias 1, 2, 3 ou 4 e a outra dupla (Dupla 1) utiliza a Estratégia Básica, conforme mostra o Quadro 19. Para cada estratégia, serão realizadas 10 otimizações por AGs, pois como se trata de um algoritmo de busca estocástico, não há garantias de que a melhor adaptação será encontrada na primeira tentativa de otimização. Com as 10 execuções do algoritmo genético, serão obtidas as médias e os desvios das funções de avaliação em termos de número de vitórias. Este procedimento também será repetido nas etapas subsequentes.
- Etapa 2: Na segunda etapa da otimização, a Dupla 2 utiliza as Estratégias 1, 2, 3 ou 4 e a Dupla 1 utiliza a mesma estratégia que a Dupla 2, porém com a função de avaliação otimizada no Etapa 1, conforme mostra o Quadro 20. Com este procedimento, esperamos obter uma função de avaliação com desempenho superior à função encontrada no processo de otimização anterior.

Quadro 19: Estratégias usadas pelas Duplas 1 e 2 na primeira etapa da otimização

Dupla 1	x	Dupla 2
Estratégia Básica	x	Estratégia 1
Estratégia Básica	x	Estratégia 2
Estratégia Básica	x	Estratégia 3
Estratégia Básica	x	Estratégia 4

Quadro 20: Estratégias usadas pelas Duplas 1 e 2 na segunda etapa da otimização

Dupla 1	x	Dupla 2
Estratégia 1 (Etapa 1)	x	Estratégia 1
Estratégia 2 (Etapa 1)	x	Estratégia 2
Estratégia 3 (Etapa 1)	x	Estratégia 3
Estratégia 4 (Etapa 1)	x	Estratégia 4

- Etapa 3: No último passo da otimização, a Dupla 2 utiliza a Estratégia *m*, onde $m \in \{1, 2, 3, 4\}$ e a Dupla 1 utiliza uma Estratégia *n*, onde $n \in \{1, 2, 3, 4\}$ e $n \neq m$, porém com as funções de avaliação otimizadas no Etapa 2, conforme mostra o Quadro 21. A otimização proposta nesta etapa permite que cada estratégia seja testada contra adversários que adotam táticas que não priorizam apenas a pontuação que pode ser obtida em uma jogada.

Quadro 21: Estratégias usadas pelas Duplas 1 e 2 na terceira etapa da otimização

Dupla 1	x	Dupla 2
Estratégia 2 (Etapa 2) Estratégia 3 (Etapa 2) Estratégia 4 (Etapa 2)	x	Estratégia 1
Estratégia 1 (Etapa 2) Estratégia 3 (Etapa 2) Estratégia 4 (Etapa 2)	x	Estratégia 2
Estratégia 1 (Etapa 2) Estratégia 2 (Etapa 2) Estratégia 4 (Etapa 2)	x	Estratégia 3
Estratégia 1 (Etapa 2) Estratégia 2 (Etapa 2) Estratégia 3 (Etapa 2)	x	Estratégia 4

6.6 Avaliação do desempenho das estratégias propostas

Para avaliar o desempenho das funções de avaliação otimizadas nas Etapas 1, 2 e 3 descritas na seção anterior, realizamos dois tipos de testes. No primeiro teste, as Estratégias 1, 2, 3 e 4 realizaram 5000 partidas entre si. No segundo teste, foram realizadas partidas entre duplas formadas por jogadores humanos e duplas formadas por jogadores virtuais.

Neste último teste, os coeficientes otimizados pelo AG foram incorporados a um software desenvolvido para o jogo de dominó de 4 pontas. Assim, a função de avaliação otimizada foi testada em partidas contra jogadores humanos, onde uma dupla era formada por dois agentes inteligentes usando a função de avaliação otimizada e a outra dupla era formada

por pessoas. Os jogadores humanos estavam divididos em duas categorias: jogadores experientes e inexperientes.

O aplicativo que utilizamos nesta fase da pesquisa possui interface gráfica, conforme ilustrado na Figura 20, permitindo a interação entre jogadores humanos e virtuais em uma mesa de dominó. Cada jogador humano conecta-se ao jogo de dominó através da internet e o aplicativo cria dois agentes inteligentes independentes entre si para escolher as jogadas da dupla adversária.

Ao iniciar a partida, as sete peças iniciais de cada jogador são escolhidas aleatoriamente e distribuídas para os participantes. Em seguida, a partida começa pelo jogador que possui a peça 6-6, ou carroça de sena, e continua em sentido horário com o jogador que está à esquerda de quem fez a jogada anterior. O jogo acaba quando, ao término de uma rodada, uma das duplas tiver 200 pontos ou mais. Caso as duas duplas tenham pontuação maior que 200, ganha a dupla que tiver mais pontos.

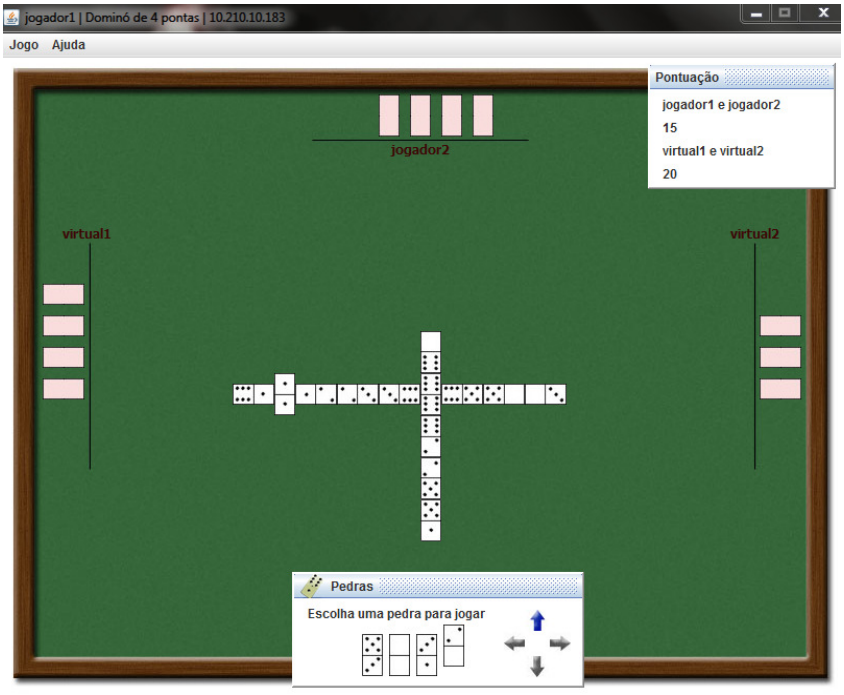


Figura 20: Aplicativo desenvolvido para o jogo de dominó de 4 pontas.

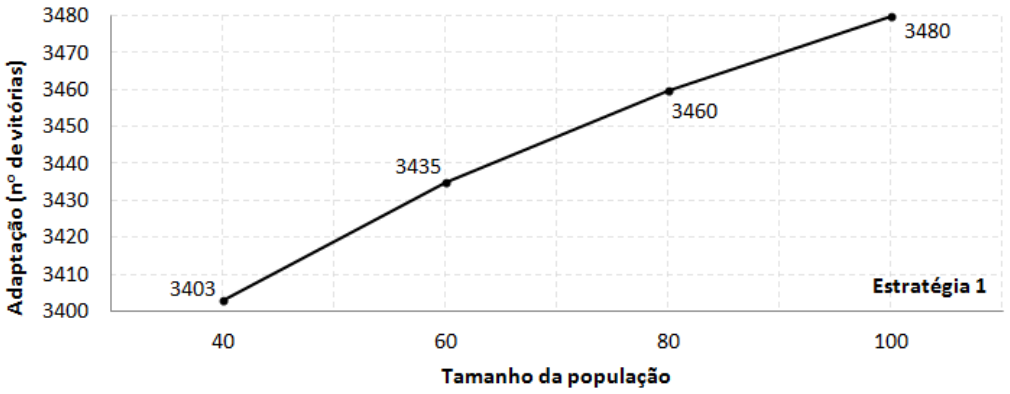
7 RESULTADOS E DISCUSSÕES

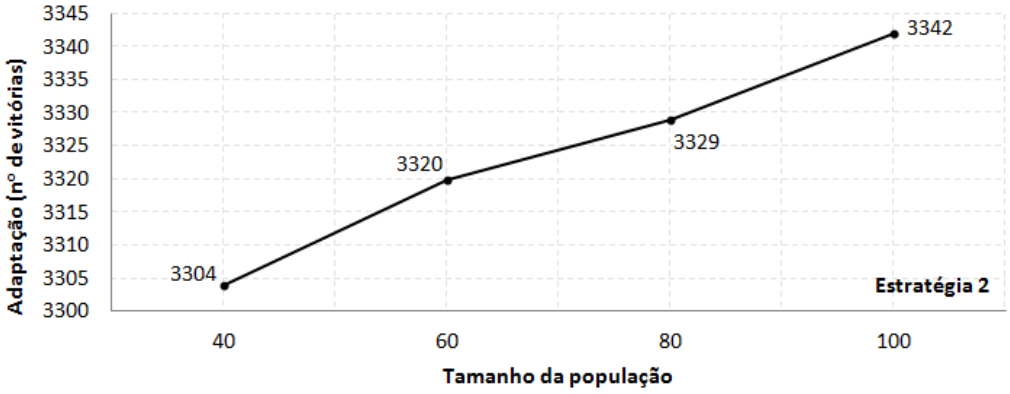
Neste capítulo apresentaremos os resultados e discussões acerca do processo de otimização realizado para as funções de avaliação propostas usando AGs. Para cada estratégia definida anteriormente, a função de avaliação correspondente foi otimizada utilizando diferentes parâmetros para a execução do algoritmo genético. A otimização das funções de avaliação pode resultar em valores ótimos ou quase-ótimos para os coeficientes α_1, α_2, α_3, α_4, α_5, α_6 e α_7, de modo que permitam que a dupla que utiliza esta função otimizada para a escolha da melhor jogada tenha um melhor aproveitamento perante uma dupla que utiliza a estratégia básica de jogo. Para determinar se as funções de avaliação otimizadas produziram resultados significantes, utilizamos os testes estatísticos *t* de Student e χ^2.

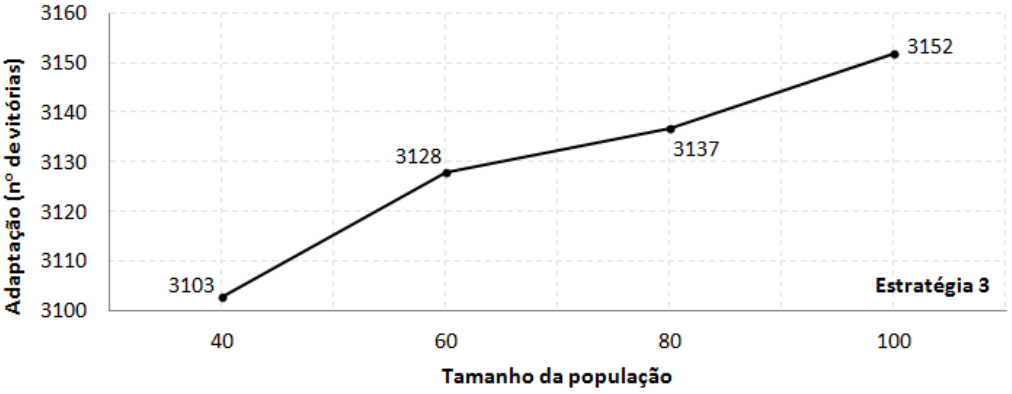
7.1 Análise de desempenho do Algoritmo Genético

Segundo a metodologia apresentada no Capítulo 6, foram propostas diferentes combinações de parâmetros que definem o funcionamento do AG. Para a escolha de um conjunto de parâmetros para serem adotados no decorrer do processo de otimização, inicialmente avaliamos o desempenho do AG durante a variação de parâmetros como o tamanho da população, o operador de cruzamento e a probabilidade de mutação, conforme os valores propostos anteriormente no Quadro 16. A variação destes parâmetros permite analisar se o AG consegue realizar a busca por uma grande área do espaço de estados dentro de um limite pré-determinado, ou seja, durante as 200 gerações.

O primeiro teste avaliou o desempenho do AG ao realizar a busca com diferentes tamanhos de população: 40, 60, 80 e 100. Para cada estratégia, identificamos que o aumento na população também proporcionou maiores valores de adaptação para o indivíduo resultante do processo de otimização. A Figura 21 ilustra a evolução da adaptação, representada pelo número de vitórias, em função da variação do tamanho da população.







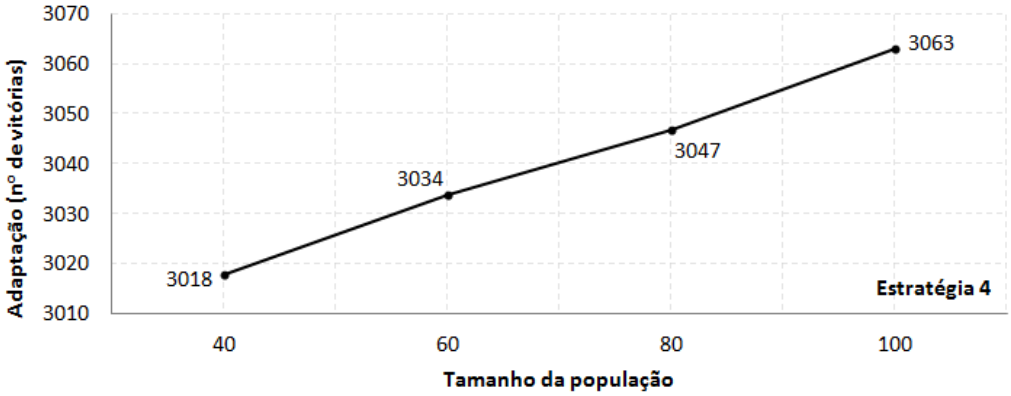


Figura 21: Otimização das Estratégias 1, 2, 3 e 4 para diferentes tamanhos de população

O primeiro gráfico da Figura 21 apresenta o desempenho na otimização da Estratégia 1, onde a média de vitórias obtidas na otimização com 40 indivíduos é de 3403 em um total de 5000 partidas, 77 partidas a menos que a melhor adaptação para uma população de 100 indivíduos. O mesmo comportamento pode ser observado nos demais gráficos apresentados para as otimizações das Estratégias 2, 3 e 4.

Em relação ao tempo de computação observado nestes testes, utilizamos o processamento paralelo do MATLAB para reduzir o período gasto com cada execução do AG, pois o algoritmo pode analisar, simultaneamente, todos os cromossomos que compõem a população. A variação no tamanho da população foi o principal responsável pelo tempo total de execução do algoritmo genético. Para uma combinação de parâmetros com população igual a 40 e total de 200 gerações, o AG foi executado em aproximadamente 28 minutos usando o processamento paralelo e em 45 minutos sem o processamento paralelo. A "pior" combinação, ou seja, população com 100 cromossomos e total de 200 gerações, teve tempo total de execução de aproximadamente 65 minutos. Sem o processamento paralelo, o tempo total sobe para 3 horas de execução ou mais, conforme observamos em (ANTONIO, 2009).

Em um segundo momento, avaliamos o desempenho do AG em função da variação dos parâmetros que definem as operações genéticas de cruzamento e mutação. No Quadro 22 são exibidos os resultados obtidos em otimizações com uma população de tamanho 100.

Quadro 22: Desempenho da otimização em função das operações de cruzamento e mutação

Operador de cruzamento	Probabilidade de Mutação	Adaptação (número de vitórias)			
		Estratégia 1	Estratégia 2	Estratégia 3	Estratégia 4
Dois pontos	1%	3466	3325	3137	3038
	5%	3480	3332	3148	3063
Uniforme	1%	3474	3307	3142	3047
	5%	3480	3342	3152	3053

É possível notar que a variação de parâmetros apresentada no Quadro 22 não proporcionou nenhuma diferença significativa entre as otimizações. Em relação à operação de

mutação, o AG que aplicou probabilidade de mutação de 5% apresentou resultados ligeiramente superiores que a mutação com probabilidade de 1%, sendo 35 vitórias a maior diferença observada. A variação na operação de cruzamento não apresentou grande diferença entre os resultados finais das otimizações. Com base nestes resultados, constatamos que seria necessário apenas realizar a otimização das funções de avaliação utilizando uma população de 100 indivíduos. Em relação aos operadores de cruzamento e mutação, optamos por utilizar o cruzamento uniforme e probabilidade de mutação de 5% durante as 200 gerações da otimização.

Definidos os parâmetros para execução do AG, prosseguimos com as etapas propostas para a otimização da função de avaliação. Cada otimização foi repetida dez vezes, de forma a permitir o cálculo da média e o desvio-padrão das adaptações, as quais representam o número de vitórias em um total de 5000 partidas. A descrição e análise das otimizações realizadas em cada etapa são apresentadas nas seções a seguir.

7.2 Resultados da primeira etapa de otimização

Na Etapa 1, conforme descrevemos no Capítulo 6, as Estratégias 1, 2, 3 e 4 são otimizadas contra a Estratégia Básica, cujos coeficientes α_i são definidos com o valor zero. O AG operou com tamanho fixo de 200 gerações e população de 100 indivíduos. A avaliação de desempenho das otimizações foi feita através do teste de hipóteses *t* de Student.

O teste de hipóteses *t* de Student, ou teste *t*, permite determinar se a diferença entre a média de vitórias obtida experimentalmente e a média esperada de vitórias é significativa (WASSERMAN, 2003). Inicialmente, definimos as duas hipóteses a seguir:

- Hipótese nula: A média experimental é igual à média esperada.

$$H_0 : \mu = \mu_0$$

- Hipótese alternativa: A média experimental é diferente da média esperada.

$$H_1 : \mu \neq \mu_0$$

Se a média de vitórias da dupla que utilizou a estratégia otimizada é muito acima, ou muito abaixo, da quantidade de vitórias esperada, a hipótese nula é rejeitada, pois os resultados favorecem a hipótese alternativa proposta. O valor crítico é o que determina a região de rejeição da hipótese nula e depende do teste estatístico aplicado, dos graus de liberdade do problema e do nível de significância adotado.

Cada teste estatístico fornece um valor que é comparado ao valor crítico para determinar se a hipótese nula é rejeitada ou não. Para o teste *t* de Student, este valor *t* é calculado através da equação abaixo:

$$t = \frac{\mu - \mu_0}{s/\sqrt{N}}$$

Nesta equação, s e N representam, respectivamente, o desvio-padrão e o tamanho da amostra e as variáveis μ e μ_0 representam a média experimental e a média esperada do problema. Nas otimizações que realizamos, μ é a média de vitórias obtidas em N = 10 otimizações, o que também indica 9 graus de liberdade para o teste realizado. Com base no valor *t* calculado, podemos compará-lo ao valor crítico t_c. Assim, a hipótese nula é rejeitada se $t \geq t_c$ ou $t \leq -t_c$ e a média da amostra é considerada significativamente maior, para $t \geq t_c$, ou menor, para $t \leq -t_c$, que a média esperada μ_0. Neste trabalho, para um nível de significância de 1% e 9 graus de liberdade, o valor crítico t_c é de 2,821.

Quadro 23: Resultados da Etapa 1 do processo de otimização

Estratégia	Adaptação		H_0: $\mu = 2500$	H_0: $\mu = 3300$
	μ	s	Valor *t*	Valor *t*
Estratégia 1 x Est. Básica	3469,1	6,12	500,88	87,4
Estratégia 2 x Est. Básica	3316,7	12,17	208,32	4,20
Estratégia 3 x Est. Básica	3260,2	10,6	226,74	-11,87
Estratégia 4 x Est. Básica	3041,9	6,74	254,23	-121,08

O Quadro 23 mostra a média de vitórias em 10 otimizações e o desvio-padrão calculado para cada estratégia. É possível notar que a Estratégia 1 apresentou desempenho

superior em relação às demais estratégias quando disputou partidas contra a Estratégia Básica, com média de 3469,1 vitórias. Por outro lado, a Estratégia 4, que apenas busca dificultar as jogadas dos adversários, obteve o pior desempenho entre as quatro estratégias.

Para verificar se as vitórias alcançadas pelas estratégias perante a Estratégia Básica podem ser consideradas significativas, inicialmente comparamos os resultados com uma média esperada de $\mu_0 = 2500$ vitórias, o que equivale a 50% do total de partidas disputadas. Os valores *t* calculados para as quatro estratégias estão exibidos na quarta coluna do Quadro 23 e nos permitem afirmar que todas as estratégias apresentaram resultados significativamente superiores ao esperado. Nesta comparação, o valor *t* calculado para a Estratégia 1 também foi superior aos outros por apresentar maior diferença para a média μ e um pequeno desvio entre as otimizações.

Em seguida, uma segunda comparação foi feita com o melhor resultado obtido em (ANTONIO *et al*, 2008, 2009). Neste trabalho, em partidas realizadas contra a Estratégia Básica, a função de avaliação desenvolvida ganhou em 66% das partidas, o que corresponde a $\mu_0 = 3300$ vitórias em 5000 jogos. Os valores *t* calculados para esta comparação estão na quinta coluna do Quadro 23. As Estratégias 1 e 2 apresentaram desempenho significativamente superior. Por outro lado, as Estratégias 3 e 4 não alcançaram a média de vitórias estabelecida anteriormente, apresentando desempenho significativamente inferior.

Em relação aos resultados gerais da primeira etapa de otimização das funções de avaliação, para a Estratégia 1, o melhor valor de adaptação foi de 3480 vitórias, ou 69,6% das partidas. O melhor desempenho da Estratégia 2 foi de 3341 vitórias (66,82%). A Estratégia 3 obteve 3281 vitórias (65,62%) e a Estratégia 4 ganhou em 60,98% das partidas. Na Figura 22 são exibidas as curvas de evolução do melhor indivíduo da população e a adaptação média dos indivíduos da população para cada estratégia, onde podemos observar que, ao final de 200 gerações, a adaptação média da população não convergiu totalmente para o melhor indivíduo.

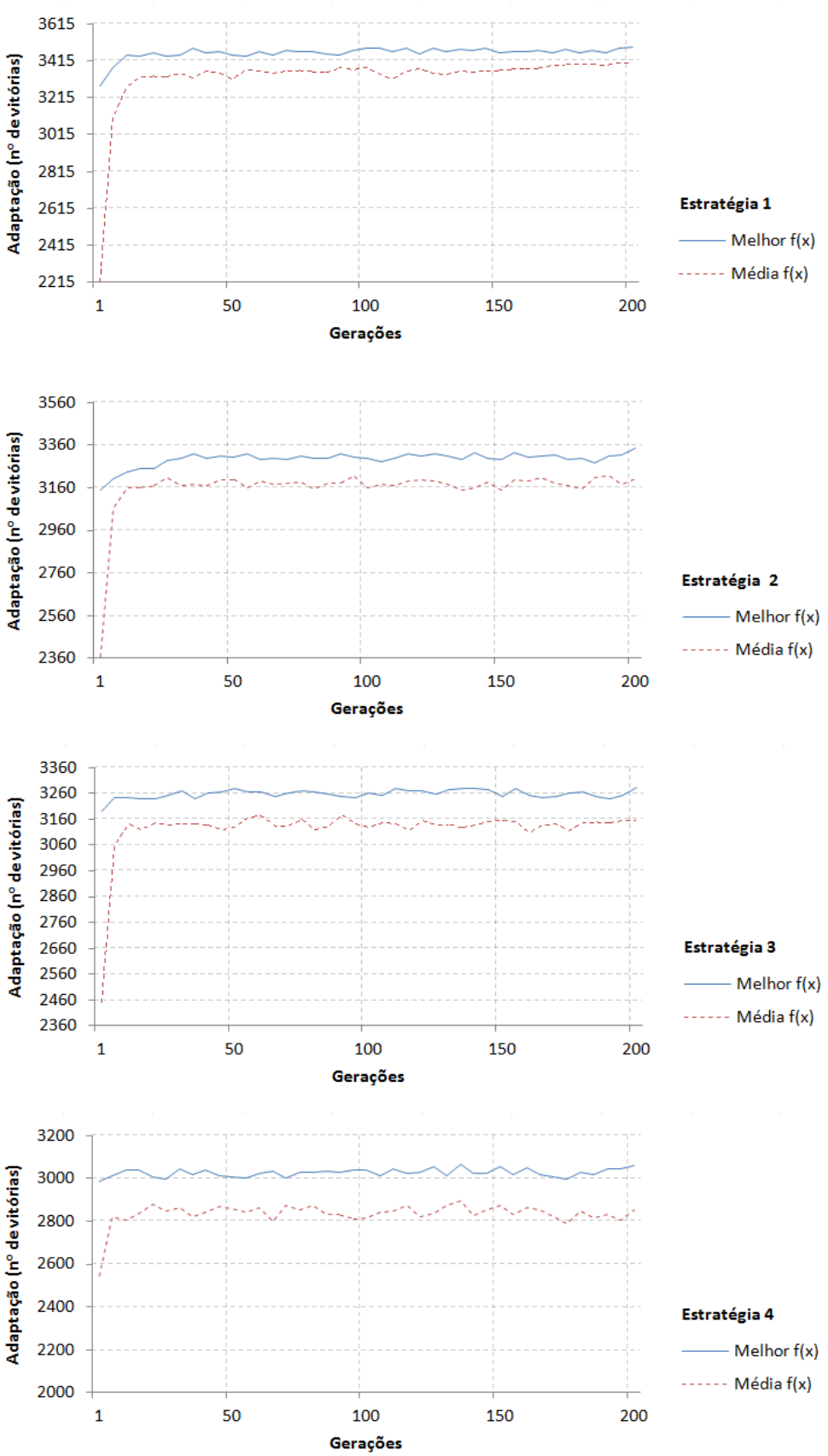


Figura 22: Evolução da população na Etapa 1 de otimização

Os coeficientes otimizados para cada estratégia estão indicados no Quadro 24. Para as Estratégias 2, 3 e 4, apenas os coeficientes α_i com valor diferente de zero foram otimizados pelo AG.

Quadro 24: Coeficientes otimizados na Etapa 1

Estratégia	α_1	α_2	α_3	α_4	α_5	α_6	α_7
Estratégia 1	-2,802	-3,196	-4,199	-6,34	1,093	6,58	1,341
Estratégia 2	0	0	0	-9,3674	1,8851	5,4356	0
Estratégia 3	0	0	0	-8,8029	0	4,1434	2,0617
Estratégia 4	3,4773	-2,6242	-4,7903	0	0	0	0

7.3 Resultados da segunda etapa de otimização

Na Etapa 2 da otimização, cada estratégia foi otimizada em partidas onde as duas duplas utilizavam a mesma estratégia, porém a dupla com estratégia fixa usou a função de avaliação otimizada no Etapa 1. Assim como na Etapa 1, as estratégias foram otimizadas dez vezes pelo Algoritmo Genético, permitindo a obtenção das médias e desvios, conforme mostra o Quadro 25.

Quadro 25: Resultados da Etapa 2 do processo de otimização

Estratégia	Adaptação		H_0: $\mu = 2500$
	μ	s	Valor *t*
Estratégia 1 x Estratégia 1	2608,9	21,3	16,16
Estratégia 2 x Estratégia 2	2599	10,5	29,81
Estratégia 3 x Estratégia 3	2607,6	19,2	17,72
Estratégia 4 x Estratégia 4	2593,5	14,51	20,37

Neste gráfico, a variável **μ** representa a média de adaptação (quantidade de vitórias em 5000 partidas) em dez otimizações para cada estratégia e a variável **s** representa o desvio-padrão calculado para estes resultados. O teste *t* foi novamente aplicado de forma a comparar a média experimental **μ** a uma média esperada $\boldsymbol{\mu_0}$ de 2500 vitórias, por representar 50% das partidas realizadas.

Assim, temos novamente as hipóteses:

- $H_0 : \mu = 2500$ (hipótese nula)
- $H_1 : \mu \neq 2500$ (hipótese alternativa)

Para 9 graus de liberdade e nível de significância de 1%, os valores *t* apresentados na quarta coluna do Quadro 25 superam o valor crítico $t_c = 2{,}821$, rejeitando, portanto, a hipótese nula e comprovando que as otimizações realizadas na segunda etapa produziram resultados significativamente superiores aos anteriores.

Em geral, as otimizações da Etapa 2 proporcionaram um desempenho de, aproximadamente, 52% de vitórias de cada estratégia contra elas mesmas. O Quadro 26 mostra os coeficientes resultantes das otimizações desta etapa para as quatro estratégias. Porém, para as Estratégias 2, 3 e 4, apenas os coeficientes com valor diferente de zero foram otimizados.

Quadro 26: Coeficientes otimizados na Etapa 2

Estratégia	α_1	α_2	α_3	α_4	α_5	α_6	α_7
Estratégia 1	-6,7612	-5,2781	-6,3282	-9,3417	2,0431	8,3193	0,0535
Estratégia 2	0	0	0	-8,6452	2,1204	7,2673	0
Estratégia 3	0	0	0	-9,9581	0	4,7058	1,3813
Estratégia 4	8,2625	-6,6377	-4,5438	0	0	0	0

Os gráficos exibidos na Figura 23 a seguir mostram as curvas de evolução da população para as Estratégias 1, 2, 3 e 4. Em cada gráfico são representadas as adaptações do melhor indivíduo e a média da população durante as 200 iterações do algoritmo. Podemos observar através destes gráficos que, assim como na otimização da Etapa 1, a quantidade de iterações realizadas não permitiu a convergência da população para o melhor indivíduo.

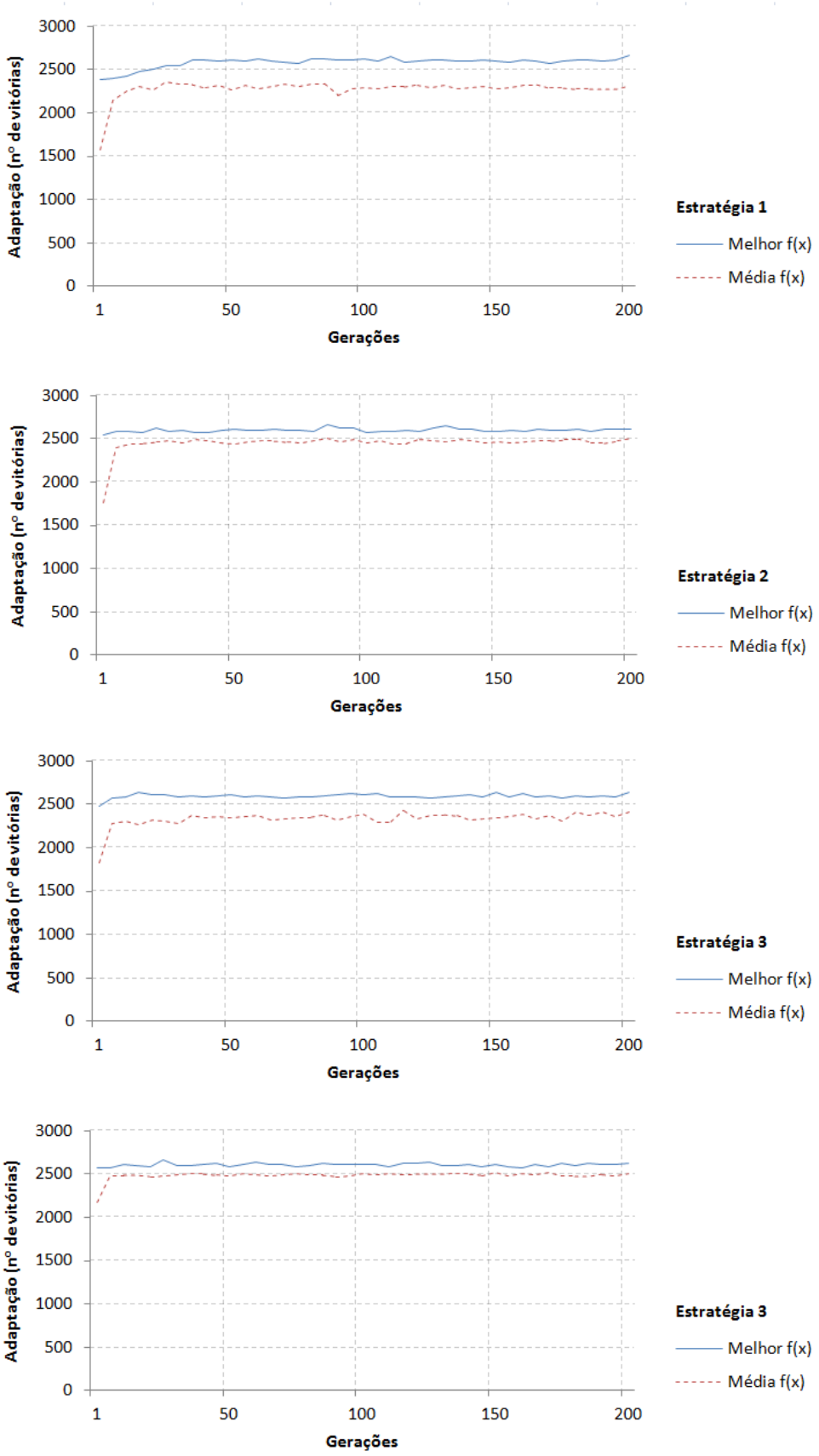


Figura 23: Evolução da população das estratégias na Etapa 2 da otimização

7.4 Resultados da terceira etapa de otimização

A Etapa 3 compreende a realização de três otimizações diferentes para as Estratégias 1, 2, 3 e 4. Cada estratégia foi testada contra as três estratégias restantes com o objetivo de maximizar seu desempenho contra estas estratégias específicas. Deste modo, a Estratégia 1 realizou partidas contra as Estratégias 2, 3 e 4, as quais usavam as funções otimizadas na Etapa 2 (Quadro 26). O mesmo ocorreu na otimização das demais estratégias.

No Quadro 27 são apresentadas as médias e os desvios obtidos após dez otimizações das Estratégias 1, 2, 3 e 4. O teste de significância aplicado foi semelhante aos anteriores, comparando a média das otimizações à média esperada de 2500 vitórias.

Quadro 27: Resultados da Etapa 3 do processo de otimização

Estratégia		Adaptação		H_0: $\mu = 2500$
		μ	s	Valor *t*
Estratégia 1	x Estratégia 2	2727,1	9,78	73,42
	x Estratégia 3	2800,5	7,25	131,14
	x Estratégia 4	3095,2	13,39	140,56
Estratégia 2	x Estratégia 1	2470,4	18,23	-5,13
	x Estratégia 3	2656,9	10,63	46,67
	x Estratégia 4	2965,8	17,64	83,51
Estratégia 3	x Estratégia 1	2411,4	14,49	-19,33
	x Estratégia 2	2529,7	15,61	6,01
	x Estratégia 4	2904,5	9,29	137,71
Estratégia 4	x Estratégia 1	2092,2	13,34	-96,66
	x Estratégia 2	2224,6	7,57	-114,97
	x Estratégia 3	2297,7	10,59	-60,38

Em relação à Estratégia 1, de acordo com os valores *t* calculados, as três otimizações proporcionaram desempenho significativamente superior à média esperada, em um nível de significância de 1%. O melhor desempenho desta estratégia foi observado na otimização contra a Estratégia 4, com média de 61,9% de vitórias em 5000 partidas.

Nas otimizações onde a Estratégia 2 jogou contra as Estratégias 3 e 4, o número total de vitórias alcançado foi superior à média esperada de 2500, porém, na otimização contra a

Estratégia 1, o valor *t* calculado foi de -5,13. O valor crítico, conforme mencionado anteriormente, é de, aproximadamente, 2,821. Uma vez que $-5{,}13 \leq -t_c$, podemos concluir que a diferença entre a média experimental e a média esperada é significativa e inferior ao desempenho desejado.

A Estratégia 3 também apresentou desempenho inferior quando foi otimizada contra a Estratégia 1. Contra as Estratégias 2 e 4, o resultado foi significantemente superior às 2500 vitórias esperadas. Na otimização da Estratégia 4, a dupla que usou esta estratégia alcançou resultados bem abaixo do esperado contra as demais estratégias.

A Etapa 3, ao contrário do que ocorreu nas etapas anteriores, produziu três funções de avaliação para cada estratégia devido aos três adversários distintos que foram usados nas otimizações. Os coeficientes que proporcionaram o maior número de vitórias para cada estratégia estão listados no Quadro 28. Vale ressaltar que estes coeficientes foram otimizados de forma a aumentar o desempenho contra adversários específicos. A evolução da população que resultou nos coeficientes listados no Quadro 28 pode ser vista nas Figuras 24, 25, 26 e 27 para as Estratégias 1, 2, 3 e 4, respectivamente. Em cada gráfico estão representadas as curvas do melhor indivíduo e da média de adaptação dos indivíduos por geração.

Quadro 28: Coeficientes otimizados na Etapa 3

Estratégia		α_1	α_2	α_3	α_4	α_5	α_6	α_7
Estratégia 1	x Est.2	7,6223	-3,582	-3,9637	-5,6579	1,5242	7,0482	0,1583
	x Est.3	-5,7202	-3,1482	-4,4601	-6,449	1,1343	6,5739	0,4437
	x Est.4	-3,658	-4,8026	-4,8387	-5,9034	0,5429	7,3988	1,0559
Estratégia 2	x Est.1	0	0	0	-7,2479	1,7106	7,1819	0
	x Est.3	0	0	0	-8,5045	2,0396	6,2316	0
	x Est.4	0	0	0	-9,1656	1,9965	7,3109	0
Estratégia 3	x Est.1	0	0	0	-8,5071	0	5,7713	1,2141
	x Est.2	0	0	0	-7,9689	0	5,2823	1,6954
	x Est.4	0	0	0	-8,1196	0	6,8578	1,8949
Estratégia 4	x Est.1	2,0528	-5,3762	-5,2952	0	0	0	0
	x Est.2	7,7727	-3,2953	-4,1464	0	0	0	0
	x Est.3	-3,7709	-4,044	-4,6423	0	0	0	0

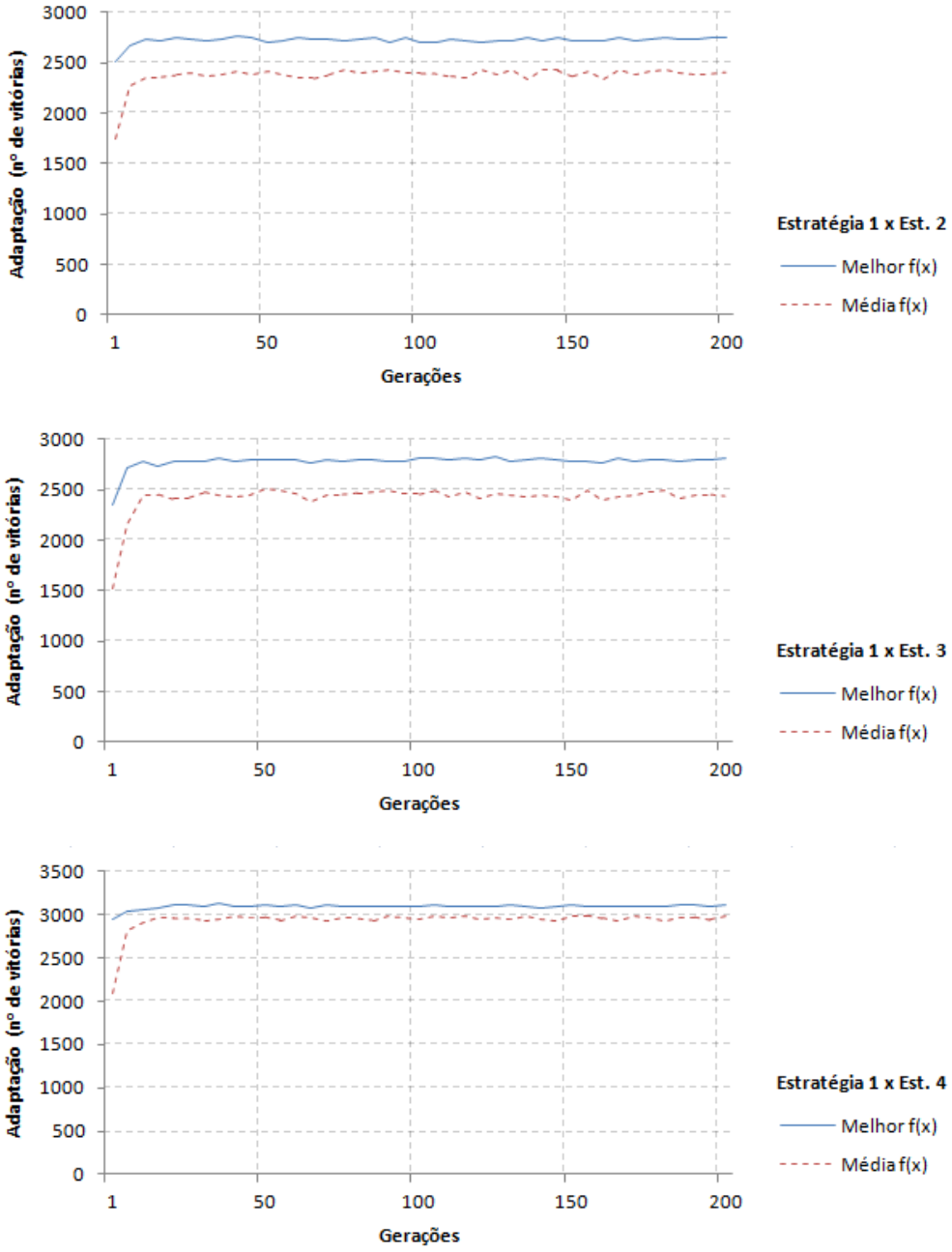


Figura 24: Evolução da população da Estratégia 1 na Etapa 3 da otimização

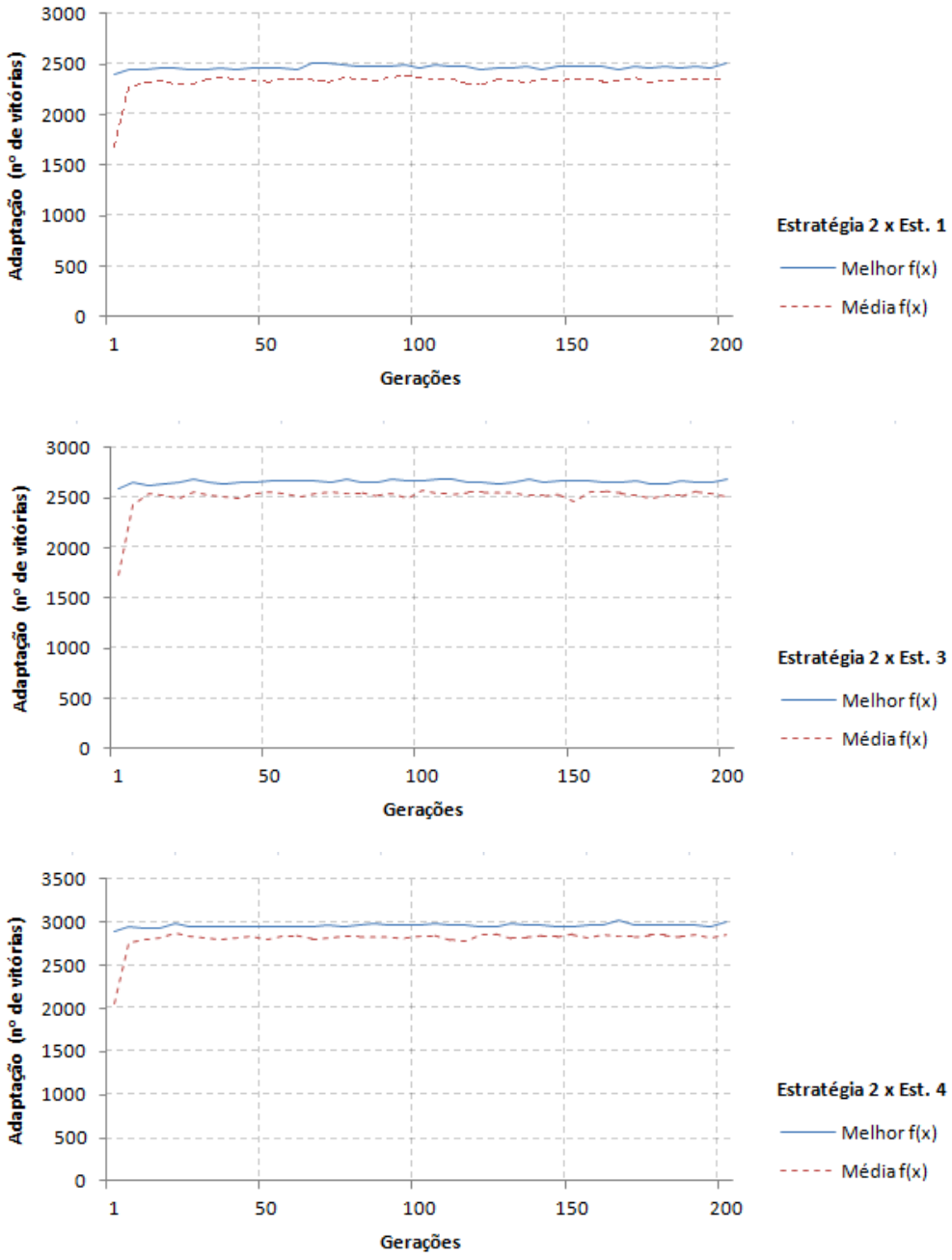


Figura 25: Evolução da população da Estratégia 2 na Etapa 3 da otimização

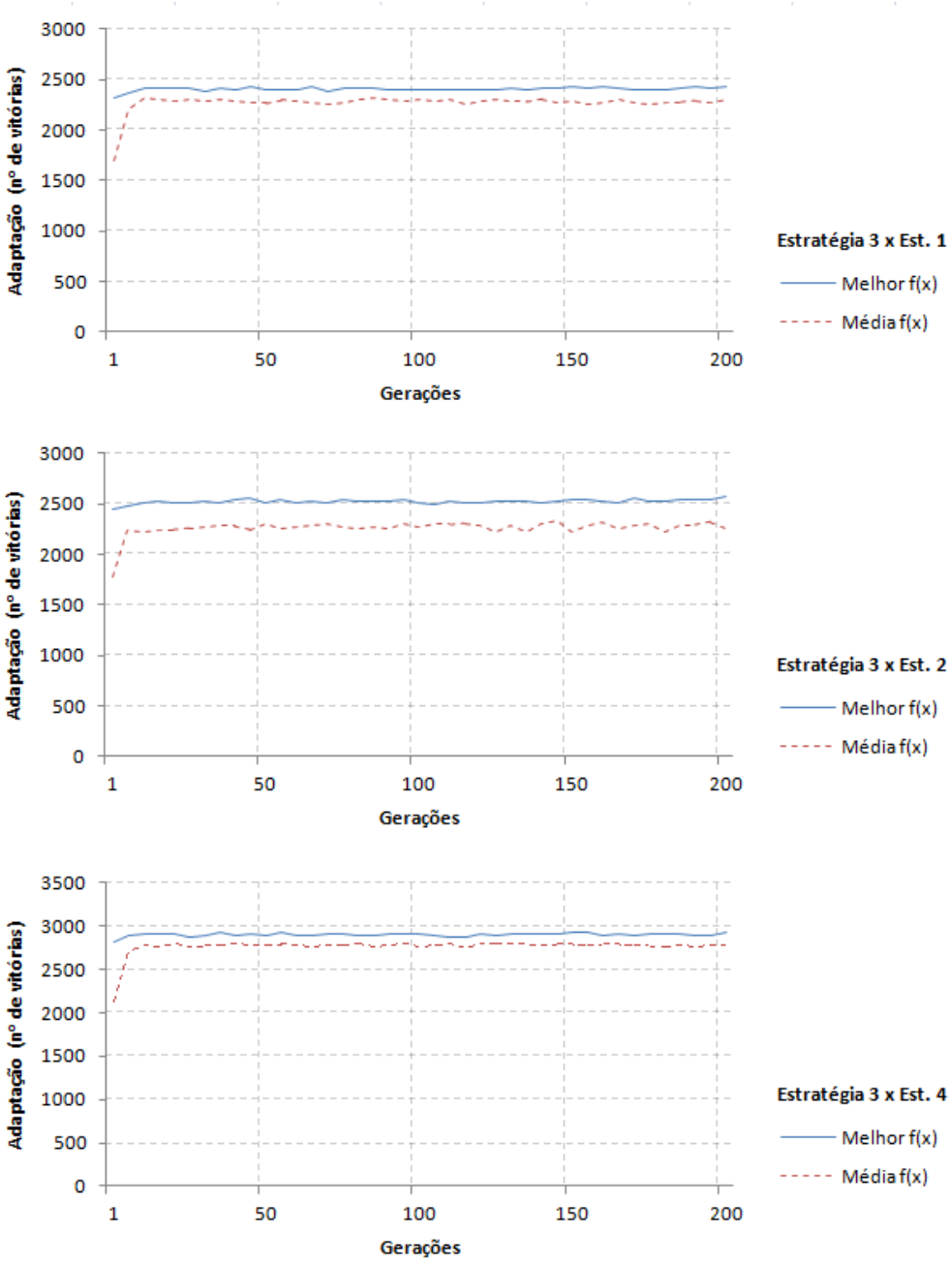


Figura 26: Evolução da população da Estratégia 3 na Etapa 3 da otimização

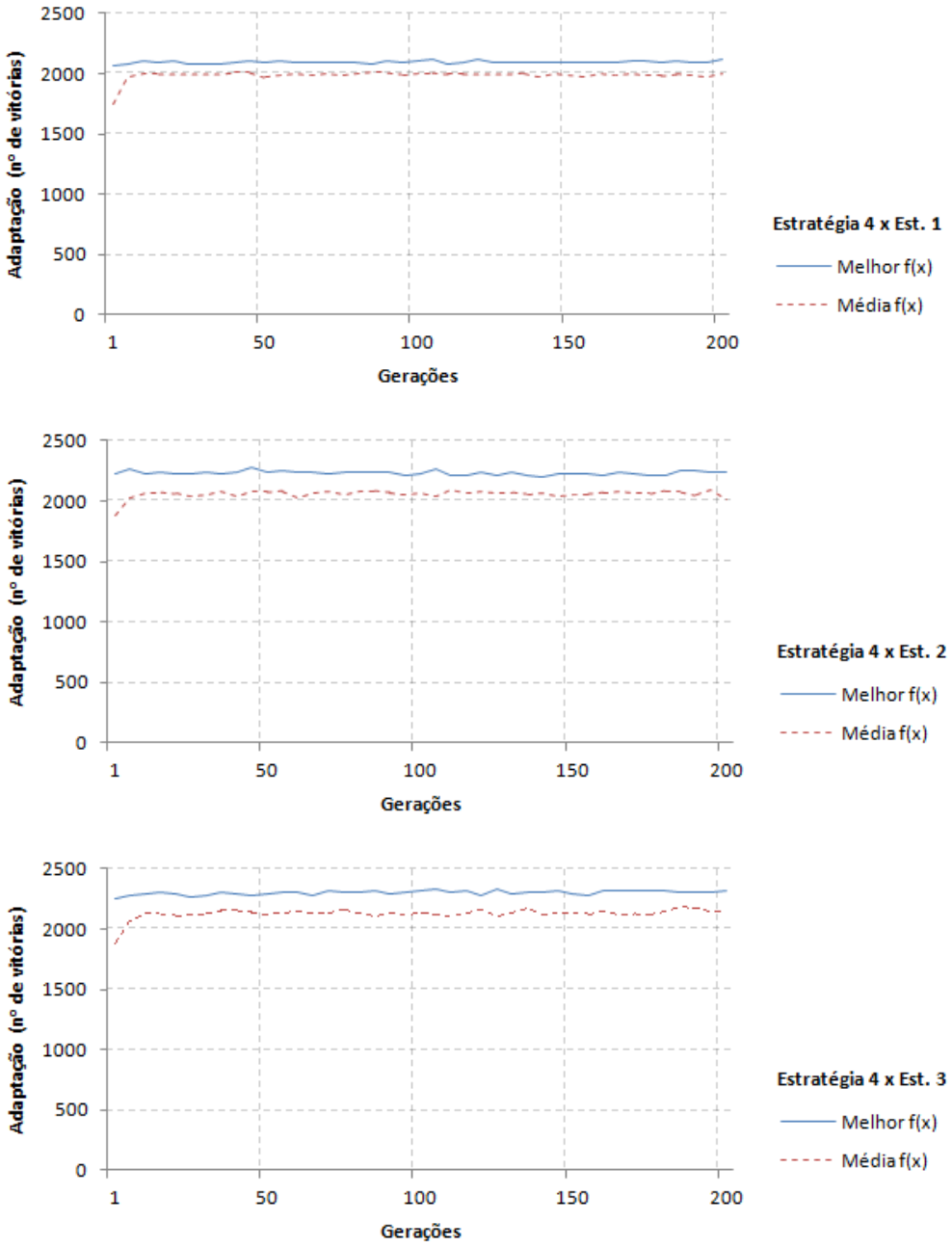


Figura 27: Evolução da população da Estratégia 4 na Etapa 3 da otimização

Conforme ilustrado nas Figuras 24, 25, 26 e 27, a evolução da população nas otimizações da Etapa 3 também apresentou uma curva onde a adaptação média da população não convergiu para a melhor adaptação.

Após as três etapas de otimização, observamos que a população rapidamente convergia para um valor médio de adaptação nas primeiras iterações do algoritmo genético. Nas gerações posteriores, a população foi evoluindo lentamente até a última geração.

7.5 Análise dos resultados obtidos nas Etapas 1, 2 e 3

Nas seções anteriores, foram mostrados os resultados das três etapas de otimização propostas nesta dissertação. Estas etapas de otimização geraram diferentes funções de avaliação para as Estratégias 1, 2, 3 e 4.

Nesta seção, avaliamos o desempenho destas funções otimizadas através da análise dos resultados obtidos em partidas de dominó de 4 pontas. Para cada estratégia, foram simuladas 5000 partidas contra todas as demais estratégias, incluindo a Estratégia Básica. Os resultados para estas simulações estão sumarizados no Quadro 29, onde a quantidade de vitórias para cada dupla é apresentada nas colunas D1 (Dupla 1) e D2 (Dupla 2).

Quadro 29: Desempenho das Estratégias 1, 2, 3 e 4 contra a Estratégia Básica

Teste			Função da Etapa 1		Função da Etapa 2		Funções de Avaliação da Etapa 3					
							Função 1		Função 2		Função 3	
			Vitórias		Vitórias		Vitórias		Vitórias		Vitórias	
Dupla 1	x	Dupla 2	D1	D2	D1	D2	D1	D2	D1	D2	D1	D2
Estratégia 1	x	Est. Básica	3452	1548	3405	1595	3447	1553	3459	1541	3459	1541
Estratégia 2	x	Est. Básica	3291	1709	3299	1701	3296	1704	3322	1678	3331	1669
Estratégia 3	x	Est. Básica	3260	1740	3273	1727	3265	1735	3264	1736	3247	1753
Estratégia 4	x	Est. Básica	3021	1979	2992	2008	2991	2009	3035	1965	3025	1975
Estratégia 1	x	Estratégia 2	2639	2361	2644	2356	2658	2342	2645	2355	2634	2366
Estratégia 1	x	Estratégia 3	2702	2298	2703	2297	2713	2287	2722	2278	2709	2291
Estratégia 1	x	Estratégia 4	3016	1984	3009	1991	3005	1995	3030	1970	3036	1964
Estratégia 2	x	Estratégia 3	2581	2419	2577	2423	2564	2436	2582	2418	2572	2428
Estratégia 2	x	Estratégia 4	2884	2116	2891	2109	2867	2133	2861	2139	2894	2106
Estratégia 3	x	Estratégia 4	2817	2183	2828	2172	2802	2198	2806	2194	2849	2151

Com estes dados, aplicamos o teste estatístico Chi-quadrado (WASSERMAN, 2003), também representado por χ^2, com o objetivo de verificar se a quantidade de vitórias que uma dupla obteve é significativamente superior à quantidade de vitórias da dupla adversária.

O teste χ^2 é um teste de hipóteses cujo objetivo é verificar o quanto as frequências observadas se distanciam da frequência esperada. Para a avaliação de desempenho desejada, observamos que cada partida apresenta duas possíveis saídas: uma vitória para a Dupla 1 ou uma vitória para a Dupla 2, cada uma com 50% de chance de ocorrer. Queremos testar duas hipóteses:

- $H_0 : p = p_0$ (hipótese nula)
- $H_1 : p \neq p_0$ (hipótese alternativa)

Onde **p** representa a probabilidade observada no experimento e $\mathbf{p_0}$ representa a probabilidade de vitória para a dupla. As hipóteses são testadas através da seguinte equação:

$$\chi^2 = \sum_{j=1}^{2} \frac{\left(X_j - np_{0j}\right)^2}{np_{0j}}$$

Na equação acima, X_j é a frequência observada para as duas saídas possíveis, ou seja, as vitórias das Duplas 1 e 2. O termo np_{0j} representa a frequência esperada para cada equipe e equivale a 2500 vitórias para cada dupla, pois temos que *n* é a quantidade de experimentos realizados (5000 partidas) e p_{0j} é a probabilidade de cada saída, sendo 0,5 nos dois casos. Para o teste χ^2, o valor crítico é obtido de acordo com o grau de liberdade do problema e o nível de significância adotado. Neste problema, para um nível de significância de 0,1% e 1 grau de liberdade, o valor crítico é de 10,827, aproximadamente.

Aplicando a equação para o cálculo dos valores χ^2 nos dados apresentados no Quadro 29, temos os valores apresentados no Quadro 30 a seguir.

Quadro 30: Valores do teste χ^2 para os dados apresentados no Quadro 29

Teste			Função da Etapa 1	Função da Etapa 2	Funções de Avaliação da Etapa 3		
					Função 1	Função 2	Função 3
Dupla 1	x	Dupla 2	D1 x D2 χ^2	D1 x D2 χ^2	D1 x D2 χ^2	D1 x D2 χ^2	D1 x D2 χ^2
Estratégia 1	x	Est. Básica	725,04	655,22	717,44	735,74	735,74
Estratégia 2	x	Est. Básica	500,54	510,72	506,89	540,54	552,44
Estratégia 3	x	Est. Básica	462,08	478,02	468,18	466,95	446,4
Estratégia 4	x	Est. Básica	217,15	193,65	192,86	228,98	220,5
Estratégia 1	x	Estratégia 2	15,45	16,58	19,97	16,82	14,36
Estratégia 1	x	Estratégia 3	32,64	32,96	36,29	39,42	34,94
Estratégia 1	x	Estratégia 4	213	207,26	204,02	224,72	229,83
Estratégia 2	x	Estratégia 3	5,24	4,74	3,27	5,37	4,14
Estratégia 2	x	Estratégia 4	117,96	122,3	107,75	104,25	124,18
Estratégia 3	x	Estratégia 4	80,39	86,06	72,96	74,9	97,44

Comparando o valor crítico de 10,827 com os valores χ^2 no Quadro 30, observamos que apenas a disputa entre as Estratégias 2 e 3 não apresentou diferença significativa para a quantidade de vitórias esperadas. Em relação às demais estratégias, a Estratégia 1 se mostrou superior às demais estratégias e as Estratégias 2 e 3 apresentaram desempenho superior à Estratégia 4.

O melhor resultado obtido neste trabalho contra a Estratégia Básica produziu um valor médio de 69,18% de aproveitamento. Esta média de vitórias foi proporcionada pela função de avaliação otimizada na Etapa 3 para a Estratégia 1 e mostrou-se superior aos resultados obtidos em (ANTONIO *et al*, 2008, 2009), em (ANTONIO, 2009) e em (GARZA, 2006).

Em (ANTONIO *et al*, 2008, 2009), o agente inteligente para a escolha das jogadas usou a função de avaliação descrita no Capítulo 4, a qual também serviu de base para o desenvolvimento das estratégias propostas nesta dissertação. Seu melhor resultado contra a Estratégia Básica foi de 66% do total de partidas disputadas.

Em um trabalho posterior (ANTONIO, 2009), a mesma função de avaliação do Capítulo 4 foi otimizada por um AG. Esta função, porém, não continha todos os coeficientes α_i presentes nas funções da Estratégia 1. O melhor resultado obtido nesta pesquisa foi de 68%.

Em (GARZA, 2006), onde a pesquisa é baseada em uma variação do dominó de 2 pontas, também foi desenvolvido um agente inteligente para a escolha das jogadas. O melhor resultado obtido neste trabalho foi 54% de vitórias, não sendo considerado um resultado significativo, pois ficou dentro do intervalo de tolerância entre 45% e 55% definido pelo autor devido às características aleatórias do jogo.

7.6 Teste do agente inteligente contra jogadores reais

O último experimento deste trabalho teve como objetivo avaliar o desempenho do agente inteligente em partidas contra duplas formadas por jogadores humanos. Porém, devido à falta de pessoas dispostas a jogar a quantidade de partidas necessárias, as estatísticas exibidas a seguir referem-se apenas à função de avaliação otimizada na primeira etapa para a Estratégia 1.

Para testar a Estratégia 1 em partidas contra jogadores humanos, foram selecionadas seis duplas, as quais foram distribuídas em dois grupos:

- Grupo A: Formado por jogadores considerados experientes no jogo de dominó de 4 pontas. As duplas pertencentes a este grupo serão denominadas A1, A2 e A3;
- Grupo B: Formado por jogadores iniciantes ou casuais. Neste grupo são esperados jogadores que não avaliam adequadamente todas as possibilidades que envolvem o jogo ou que apenas escolhem as jogadas que pontuam na mesa. As duplas do grupo B serão denominadas B1, B2 e B3.

A dupla virtual utilizou a função de avaliação definida no Quadro 9 para a escolha de suas jogadas. Os coeficientes α_1, α_2, α_3, α_4, α_5, α_6 e α_7 foram definidos conforme os valores otimizados para a Estratégia 1 no Quadro 23.

Cada dupla formada por jogadores humanos disputou 50 partidas contra a dupla virtual através do software apresentado no Capítulo 6. A Figura 28 mostra o percentual de vitórias da dupla virtual contra cada dupla formada por jogadores humanos.

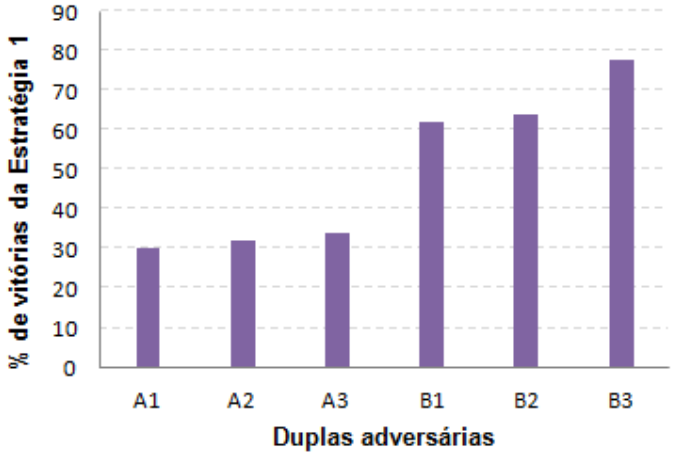


Figura 28: Desempenho da Estratégia 1 contra jogadores humanos

Nas partidas realizadas contra as duplas do grupo A, ou seja, A1, A2 e A3, a dupla virtual ganhou, em média, 16 partidas em 50 jogos, o que corresponde a 32% do total de partidas disputadas. Durante as partidas, observamos que os jogadores experientes rapidamente se adaptaram à estratégia adotada pela dupla virtual. Estas duplas, formadas por pessoas que jogavam diariamente entre si, estavam habituadas com a maneira de jogar de seus parceiros, sendo este outro fator que possivelmente contribuiu para o grande percentual de vitórias deste grupo.

Por outro lado, nas partidas realizadas contra as duplas do grupo B, a estratégia alcançou desempenho superior a 60%. As duplas B1 e B2, formadas por pessoas que não jogavam com muita frequência, perderam, em média, 63% dos jogos disputados contra a Estratégia 1. Contra a dupla B3, formada por pessoas que apenas conheciam as regras do jogo, ou seja, jogadores iniciantes, a Estratégia 1 foi superior em 78% das partidas realizadas.

8 CONCLUSÃO

Neste trabalho, apresentamos diferentes tipos de estratégias a serem adotadas por um agente inteligente para o jogo de dominó de 4 pontas. Quatro estratégias foram propostas com prioridades distintas em relação à escolha de suas jogadas. As estratégias foram representadas por funções de avaliação, cujos coeficientes foram treinados por um Algoritmo Genético que avaliava uma combinação específica de parâmetros através da quantidade de jogos que o agente inteligente era capaz de ganhar contra um oponente com função de avaliação fixa.

Propomos quatro estratégias para o jogo de dominó de 4 pontas. Na Estratégia 2, o jogador prioriza jogadas que favorecem apenas o seu próprio jogo, ignorando tanto as jogadas do parceiro quanto as dos adversários. Para a Estratégia 3, as jogadas buscam favorecer o jogo do parceiro, baseando-se nos movimentos realizados por ele. A Estratégia 4, por outro lado, escolhe os movimentos que tem maior probabilidade de atrapalhar as jogadas dos adversários, o que também é feito com base nas jogadas anteriores da dupla rival. Por fim, a Estratégia 1 compreende as três estratégias citadas anteriormente, ponderando entre as três possíveis ações: facilitar o próprio jogo, dificultar as jogadas adversárias e facilitar o jogo do parceiro.

Para a implementação do método de busca por AGs, os cromossomos da população foram codificados como um vetor formado pelos sete coeficientes das funções de avaliação e uma função de adaptação foi implementada de forma a determinar o número de vitórias em n partidas. Também testamos diversos parâmetros do AG de forma a verificar a combinação que permitiria a melhor busca dentro do espaço de soluções. A utilização de processamento paralelo permitiu menor tempo de computação em relação ao tempo verificado em trabalhos anteriores (ANTONIO, 2009), onde o processamento foi serial.

Escolhidos os parâmetros do AG, a otimização das funções de avaliação correspondentes às quatro estratégias passou por três etapas. Inicialmente, as estratégias foram otimizadas contra a Estratégia Básica, cujos coeficientes da função de avaliação são iguais a

zero. Este processo de otimização foi denominado Etapa 1 e gerou uma função de avaliação para cada estratégia. Em seguida, na Etapa 2, as estratégias foram otimizadas contra elas mesmas, em partidas onde a dupla adversária utilizou as funções treinadas da Etapa 1. A otimização da Etapa 2 resultou em novas funções de avaliação para as Estratégias 1, 2, 3 e 4. Na última etapa, as estratégias tiveram como adversários as outras estratégias propostas. Neste caso, as funções utilizadas pelos jogadores rivais foram as resultantes da Etapa 2 de otimização. A otimização desta Etapa 3 produziu três funções de avaliação para cada estratégia.

Os resultados da otimização foram analisados através de testes de significância para determinar se eram superiores a uma quantidade de vitórias esperada. Na otimização da Etapa 1, constatamos que todas as Estratégias tiveram desempenho superior a média de vitórias esperada, porém apenas as Estratégias 1 e 2 foram superiores ao melhor resultado obtido em (ANTONIO *et al*, 2008, 2009). Na Etapa 2, a otimização gerou funções de avaliação que superaram as funções da Etapa 1 em número de vitórias. Por outro lado, na otimização da Etapa 3, algumas estratégias não conseguiram obter vitórias acima da média esperada contra suas adversárias. Verificamos que a Estratégia 1 é superior às demais estratégias, enquanto as Estratégias 2 e 3 possuem desempenho semelhante e ambas são superiores à Estratégia 4.

Em testes contra jogadores humanos, apenas a Estratégia 1 otimizada na primeira etapa pode ser avaliada, devido à falta de voluntários. A estratégia disputou partidas contra duplas formadas por jogadores experientes e duplas formadas por jogadores inexperientes. A Estratégia 1 foi superior aos jogadores casuais, porém apresentou resultado muito abaixo da média contra os jogadores mais experientes.

8.1 Trabalhos futuros

Para aperfeiçoamento da função de avaliação, sugere-se a aplicação de outras técnicas de busca e otimização, tais como:

- Aplicação de Programação Genética para produção, com maior liberdade, de uma função de avaliação mais adequada para a solução do problema;
- Aplicação do algoritmo Expectiminimax, uma variação da técnica de busca Minimax para jogos com informações imperfeitas, como o dominó;
- O jogo de dominó de 4 pontas, conforme estabelecido anteriormente, dispõe de diversos objetivos distintos para alcançar a pontuação necessária para ganhar uma partida. Assim, a escolha dos coeficientes da função de avaliação pode ser caracterizada como um problema multiobjetivo, pois é necessário otimizar a pontuação que o jogador pode realizar durante a partida, maximizar a pontuação do parceiro de dupla e minimizar a pontuação dos adversários. A aplicação de Algoritmos Evolutivos para Otimização Multiobjetivo especificamente adaptados a este problema poderia proporcionar melhor desempenho do agente inteligente.

Referências Bibliográficas

ANTONIO, N. S.; COSTA FILHO, C. F. F.; COSTA, M. G. F. Proposta de uma heurística para o jogo de dominó de 4 pontas. In: VII Brazilian Symposium on Computer Games and Digital Entertainment, 2008, Belo Horizonte, MG. Proceedings of SBGames'08 – Computing Track, 2008, 24-30.

ANTONIO, N. S.; COSTA FILHO, C. F. F.; COSTA, M. G. F. Pesquisa de um Agente Inteligente para o Jogo de Dominó de 4 pontas. In: XVIII CONIC – Congresso de Iniciação Científica da UFAM, 2009, Manaus, AM. Anais do XVIII Congresso de Iniciação Científica da UFAM, 2009.

ANTONIO, N. S. Otimização da função de avaliação do jogo de dominó de 4 pontas utilizando algoritmo genético. 2009. 53f. Monografia (Graduação em Engenharia da Computação) – Faculdade de Tecnologia, Universidade Federal do Amazonas, Manaus.

ANTONIO, N. S.; COSTA FILHO, C. F. F.; COSTA, M. G. F.; PADILLA, R. Optimization of an Evaluation Function of the 4-Sided Dominoes Game Using a Genetic Algorithm. In: IEEE Conference on Computational Intelligence and Games (CIG), 2011, Seoul, South Korea. Proceedings of CIG'11, 2011, 24-30.

ASHLOCK, D. Evolutionary Computation for Modeling and Optimization. Springer, 2006.

BARONE, L.; WHILE, L. An Adaptive Learning Model for Simplified Poker Using Evolutionary Algorithms. University of Western Australia, Perth. 1999. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.41.4618&rep=rep1&type=pdf>. Acesso em: 20 de setembro de 2011.

BENNATON, J. F. O que é otimizar?. Disponível em: <http://www.lps.usp.br/neo/jocelyn/que_e_otimizar.htm>. Acesso em: 07 de dezembro de 2009.

BITTENCOURT, G. Inteligência Computacional. Disponível em: <http://www.das.ufsc.br/gia/softcomp/softcomp.html>. Acesso em: 08 de dezembro de 2009.

BOSKOVIC, B.; GREINER, S.; BREST, J.; ZUMER, V. A Differential Evolution for the Tuning of a Chess Evaluation Function. In: IEEE Congress on Evolutionary Computation, 2006, Vancouver, Canada, pp. 1851-1856.

CAMPBELL, M. S.; HOANE, A. J.; HUS, F. H. Deep Blue. Artificial Intelligence, 134(1-2), 57-83, 2002.

CHLEUBS, B. S. Domino-tiling games. Journal of Computer and System Sciences, v. 32, n.3, 374-392, 1986.

Differential Evolution. Disponível em: <http://drdobbs.com/database/184410166>. Acesso em: 14/03/2011.

Dominoes. Disponível em: <http://www.gamecabinet.com/rules/DominoIntro.html>. Acesso em: 09 de dezembro de 2009.

Dominoes Worlds. Disponível em: <http://dominoesworlds.com>. Acesso em: 11 de abril de 2011.

FLOM, L.; ROBINSON, C. Using a Genetic Algorithm to Weight an Evaluation Function for Tetris. 2004. Disponível em: <http://www.mendeley.com/research/using-genetic-algorithm-weight-evaluation-function-tetris/>. Acesso em: 20 de setembro de 2011.

GARZA, A. G. de S. Evaluating Individual Player Strategies in a Collaborative Incomplete-Information Agent-Based Game Playing Environment. In: Symposium on Computational Intelligence and Games, 2006, pp. 211-216.

GOLDBERG, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, Reading, MA, 1989.

HOLLAND, J. H. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, 1975.

Introduction to Global Optimization. Disponível em: <http://www.mat.univie.ac.at/~neum/glopt/intro.html>. Acesso em: 07 de dezembro de 2009.

MATHWORKS. MATLAB – Global Optimization Toolbox™ 3 User's Guide. Disponível em: <http://www.mathworks.com/help/pdf_doc/gads/gads_tb.pdf>. Acesso em: 13 de julho de 2011.

MICHALEWICZ, Z. Genetic Algorithms + Data Structures = Evolution Programs. 3rd rev. and extended ed. New York: Springer, 1996.

MIRANDA, M. N. Algoritmos Genéticos: Fundamentos e Aplicações. Disponível em: < http://www.gta.ufrj.br/~marcio/genetic.html>. Acesso em: 27 de novembro de 2009.

MITCHELL, M. An Introduction to Genetic Algorithms. 5. ed. Cambridge: MIT Press, 1999.

NEGNEVITSKY, M. Artificial Intelligence: a guide to intelligent systems. 2. ed. Addison Wesley, 2005.

NICOLAI, G.; HILDERMAN, R. J. No-Limit Texas Hold'em Poker Agents Created with Evolutionary Neural Networks. In: Symposium on Computational Intelligence and Games, 2009, 125-131.

OBITKO, M. Genetic Algorithms: 1998. Tradução de Hermelindo Pinheiro Manoel em 2004. Disponível em: <http://www.obitko.com/tutorials/genetic-algorithms/portuguese/index.php>. Acesso em: 27 de novembro de 2009.

PRICE, K.; STORN, R. Differential Evolution: A Simple Evolution Strategy for Fast Optimization. Dr. Dobb's Journal of Software Tools, 22(4):18-24, 1997.

RUSSEL, S.; NORVIG, P. Inteligência Artificial : tradução da 2ª edição. Rio de Janeiro: Elsevier, 2004.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, Vol. 3, 211-229, 1959.

SHAEFFER, J. Jump Ahead: Challenging Human Supremacy in Checkers. One, Springer-Verlag, Berlin, 1997.

SHANNON, C. E. Programming a computer for playing chess. Philosophical Magazine, 41(4), 256-275, 1950.

SMITH, S. J. J.; NAU, D. S.; THROOP, T. A. Success in spades: Using AI planning techniques to win the world championship of computer bridge. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, 1079-1086, Madison, Wisconsin, AAAI Press, 1998.

SYCARA, K. P. Multiagent systems. AI Magazine, v. 19, n. 2, 79-92, 1998.

WANG, J.; LI, S. Representing Evaluation of Computer Chinese Chess by Artificial Neural Network Using Genetic Algorithm. In: Chinese Control and Decision Conference, 2008, 1226-1232.

WASSERMAN, L. All of Statistics: A Concise Course in Statistical Inference. Springer, 2003.

WITTKAMP, M.; BARONE, L. Evolving Adaptive Play for the Game of Spoof Using Genetic Programming. In: IEEE Symposium on Computational Intelligence and Games, 2006, 164-172.

YEN, H. C. A multiparameter analysis of domino tiling with an application to concurrent systems. Theoretical Computer Science, v. 98, n. 2, 263-287, 2002.

Apêndice A – Scripts e funções do MATLAB

1. Função de Adaptação para a Estratégia 1 (*fitnessFcn1.m*):

```
% ------------------------------------------
% Função de Adaptação do Algoritmo Genético
% Estratégia 1
% ------------------------------------------
function y = fitnessFcn1(x)

% parâmetros para a dupla 1
alfa_dupla1 = zeros(7,1);
% parâmetros para a dupla 2
alfa_dupla2 = x';

import domino_ga.*
jogo = UsaDomino;
jogos = 5000;
vitorias = jogo.realizaJogo(jogos,alfa_dupla1,alfa_dupla2);
y = -vitorias;
end
```

2. Função de Adaptação para a Estratégia 2 (*fitnessFcn2.m*):

```
% ------------------------------------------
% Função de Adaptação do Algoritmo Genético
% Estratégia 2
% ------------------------------------------
function y = fitnessFcn2(x)

% parâmetros para a dupla 1
alfa_dupla1 = zeros(7,1);
% parâmetros para a dupla 2
alfa_dupla2 = [0 0 0 x(1) x(2) x(3) 0]';

import domino_ga.*
jogo = UsaDomino;
jogos = 5000;
vitorias = jogo.realizaJogo(jogos,alfa_dupla1,alfa_dupla2);
y = -vitorias;
end
```

3. Função de Adaptação para a Estratégia 3 (*fitnessFcn3.m*):

```
% ------------------------------------------
% Função de Adaptação do Algoritmo Genético
% Estratégia 3
% ------------------------------------------
function y = fitnessFcn3(x)

% parâmetros para a dupla 1
alfa_dupla1 = zeros(7,1);
% parâmetros para a dupla 2
alfa_dupla2 = [0 0 0 x(1) 0 x(2) x(3)]';
```

```
import domino_ga.*
jogo = UsaDomino;
jogos = 5000;
vitorias = jogo.realizaJogo(jogos,alfa_dupla1,alfa_dupla2);
y = -vitorias;
end
```

4. Função de Adaptação para a Estratégia 4 (*fitnessFcn4.m*):

```
% ------------------------------------------------
% Função de Adaptação do Algoritmo Genético
% Estratégia 4
% ------------------------------------------------
function y = fitnessFcn4(x)

% parâmetros para a dupla 1
alfa_dupla1 = zeros(7,1);
% parâmetros para a dupla 2
alfa2 = [x(1) x(2) x(3) 0 0 0 0]';

import domino_ga.*
jogo = UsaDomino;
jogos = 5000;
vitorias = jogo.realizaJogo(jogos,alfa_dupla1,alfa_dupla2);
y = -vitorias;
end
```

5. Função *dominoGA.m*:

```
function [x,fitness] = ...
dominoGA(n,Strategy,PopRange,PopSize,EliteCount,Generations,Crossover
2P,MutationRate)

% Define as opções padrão do Matlab para um algoritmo genético
% como o tipo de população 'Double Vector'
% Probabilidade de Crossover 0.8
% Função de seleção - Roleta
options = gaoptimset;

options = gaoptimset(options,'PopInitRange', PopRange);
options = gaoptimset(options,'PopulationSize', PopSize);
options = gaoptimset(options,'EliteCount', EliteCount);
options = gaoptimset(options,'Generations', Generations);
options = gaoptimset(options,'CreationFcn', @gacreationuniform);

if (Crossover2P == 1) % twopoint
    options = gaoptimset(options,'CrossoverFcn', @crossovertwopoint);
end

options = gaoptimset(options,'MutationFcn', {  @mutationuniform
MutationRate });
options = gaoptimset(options,'Display', 'diagnose');
options = gaoptimset(options,'Vectorized', 'off');
options = gaoptimset(options,'UseParallel', 'always');
```

```
if (Strategy == 1) % Estratégia 1
    [x,fitness] = ga(@fitnessFcn1,n,[],[],[],[],[],[],[],options);
elseif (Strategy == 2) % Estratégia 2
    [x,fitness] = ga(@fitnessFcn2,n,[],[],[],[],[],[],[],options);
elseif (Strategy == 3) % Estratégia 3
    [x,fitness] = ga(@fitnessFcn3,n,[],[],[],[],[],[],[],options);
else % Estratégia 4
    [x,fitness] = ga(@fitnessFcn4,n,[],[],[],[],[],[],[],options);
end
end
```

6. Função *runDominoGA.m*:

```
function [x,fitness] = runDominoGA(estrategia,populacao,grupo)

% Ativa o processamento paralelo
matlabpool

range = [-10;10];       % variação dos parametros
elite = populacao/10;   % elite count data - 10% da população
geracoes = 200;         % numero de gerações

crossover2P = 0;
mutationRate = 0.01;
if (grupo == 1) % twopoint crossover | mutação 1%
    crossover2P = 1;
elseif (grupo == 2) % twopoint crossover | mutação 5%
    crossover2P = 1;
    mutationRate = 0.05;
elseif (grupo == 4) % scattered | mutação 5%
    mutationRate = 0.05;
end

% define o numero de variaveis e a função de avaliação
if (estrategia == 1)
    n = 7; % numero de variaveis para a estratégia 1
else
    n = 3; % numero de variaveis para as estratégias 2, 3, 4
end

[x,fitness] = ...
dominoGA(n,estrategia,range,populacao,elite,geracoes,crossover2P,mutationRa
te);

% Desativa o processamento paralelo
Matlabpool close
end
```

Apêndice B – Artigos publicados

Este apêndice contém os artigos publicados resultantes desta pesquisa:

1. "**Proposta de uma heurística para o jogo de dominó de 4 pontas**", publicado em *Proceedings of SBGames'08: Computing Track*, pp. 24-30, em novembro de 2008, Belo Horizonte, Minas Gerais.
2. "***Optimization of an Evaluation Function of the 4-sided Dominoes Game Using a Genetic Algorithm***", publicado em *Proceedings of 2011 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 24-30, em setembro de 2011, Seul, Coreia do Sul.

SBC - Proceedings of SBGames'08: Computing Track - Full Papers Belo Horizonte - MG, November 10 - 12

Proposta de uma heurística para o jogo de dominó de 4 pontas

Nirvana da S. Antônio* Cicero F.F. Costa Filho Marly G. F. Costa

Universidade Federal do Amazonas, Centro de Tecnologia Eletrônica e da Informação, Brasil

Abstract

This paper presents a new methodology for the choice of the best move in the four extremity domino game. The proposed methodology can be divided in two parts. First of all is defined the game state. This definition comprises the use of seven vectors. These vectors map some aspects of the game such as the number of pieces played, the number of pieces in hand, etc. Second is proposed a heuristic evaluation function to the choice of the best move, comprised of two terms. The first term takes into account the number of points marked in a move. The second one incorporates the game strategy. In the results section are shown some simulation results to games of four persons, grouped in two groups. The group that used the heuristic function proposed in this paper wins the game in 66% of the simulations. Future works are proposed that makes use of a new heuristic function associated with genetic algorithm as an optimization tool.

Keywords: domino game, heuristic function, game state.

Authors' contact:
{ccosta,mcosta}@ufam.edu.br
* nirvana.sa@gmail.com

1. Introdução

O dominó é composto de 28 peças (pedras) chatas, retangulares As 28 pedras têm duas metades, cada uma dessas metades contém uma numeração que varia de zero (vazio) a seis pontos, formando várias combinações. Na forma clássica do jogo, são sete números (de zero a seis), combinados entre si. Matematicamente: C(7,2) + 7 = C(8,2) = 28. O dominó pode ser jogado em duplas adversárias onde cada jogador recebe 7 peças ou alternativamente pode ser jogado por apenas 2 jogadores com 7 pedras cada um e 14 pedras para comprar no caso do oponente não ter a pedra da vez. Existem várias formas de se jogar dominó, a mais comum é o dominó de 2 pontas. No estado do Amazonas joga-se outro tipo pouco difundido, o dominó de 4 pontas, objeto desse trabalho.

No dominó de 2 pontas, o objetivo do jogo é tão somente conseguir colocar a última peça no tabuleiro do jogo. Durante a partida, são adotadas algumas estratégias para o jogador "bater", isto é, ser o primeiro a desfazer-se de todas as suas pedras. As jogadas visam encaixar alguma peça nas peças que estão nas pontas do jogo, uma por vez e minimizar a possibilidade do adversário encaixar uma pedra em uma das pontas e passe. Caso algum jogador tenha batido o jogo, sua dupla leva todos os pontos das peças que estão nas mãos dos adversários. A partida pode terminar em duas circunstâncias: quando um jogador consegue bater o jogo, ou quando o jogo fica trancado. Caso o jogo fique trancado, contam-se todos os pontos conseguidos por cada dupla. A dupla que possuir menos pontos é a vencedora, e leva todos os pontos da dupla adversária.

No dominó de 4 pontas, a disputa ocorre, em geral, entre duplas. Cada jogador deve ter 7 pedras em mãos no início de uma rodada. Na primeira rodada da competição, a primeira pedra a ser jogada deverá ser a de numeração 6-6, chamada de "carroça de sena" por ter suas duas metades iguais a seis. Os jogadores têm a possibilidade de abrir até quatro pontas de jogo a partir da carroça 6-6. Na figura 1 mostra-se um jogo onde já foram abertas 3 pontas. O objetivo principal dessa versão do jogo é alcançar uma pontuação igual ou superior a 200 pontos, que pode ser alcançada em uma ou mais rodadas. Em cada jogada existe a possibilidade de se pontuar 5, 10, 15,..., 50 pontos. Pode-se marcar pontos (um múltiplo de 5) em cinco situações distintas:

P_1) A soma dos pontos das pontas da mesa é múltiplo de 5, sendo a pontuação igual a essa soma. Na figura 1 mostra-se um exemplo de jogada que rende 15 pontos equivalente a soma dos pontos das pontas abertas (2+5+8). Na figura 2 mostra-se um exemplo de jogada onde não se pontua, pois a soma dos pontos nas pontas abertas é 14 (1+5+8).

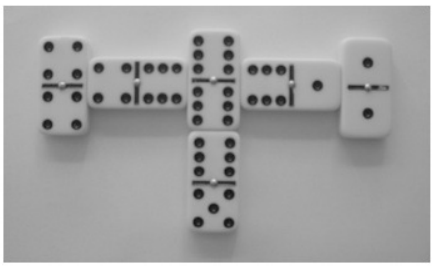


Figure 1: Exemplo de jogada do dominó de 4 pontas onde pontua-se 15 pontos.

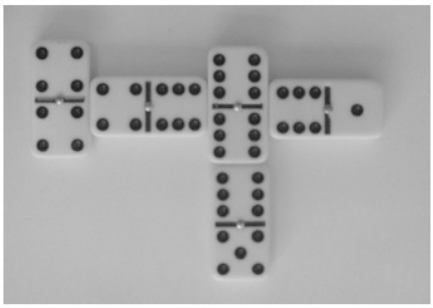


Figure 2: Exemplo de jogada do domino de 4 pontas onde não se pontua, pois a soma dos pontos nas pontas abertas é igual a 14.

P_2) O jogador adversário "passa" em sua vez de jogar, ou seja, não possui pedra que se encaixe em nenhuma das 4 pontas. O impedimento de jogada do adversário rende a dupla que provocou o passe 20 pontos.

P_3) O jogador provoca o passe de todos os demais jogadores (inclusive o parceiro) e a vez volta para o jogador ele. Esse passe geral, chamado de "galo", equivale a uma pontuação de 50 pontos.

P_4) Quando um jogador "bate" (consegue encaixar todas as pedras que dispunha). Nesse caso, somam-se todos os pontos das peças que estão nas mãos dos adversários. Essa soma é denominada de "garagem". A pontuação da jogada corresponde ao maior múltiplo de 5 menor ou igual a essa soma. Na figura 3 mostra-se um exemplo em que a soma dos pontos dos adversários é igual a 12, sendo a pontuação da jogada igual a 10 pontos;

P_5) A última peça descartada pelo jogador que bateu é uma "carroça", ou seja, uma pedra que tem os dois lados com numerações iguais, sendo a pontuação da jogada igual a 20 pontos.

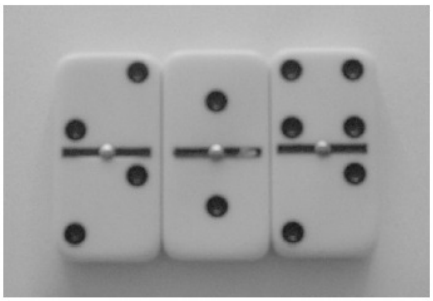


Figure 3: Exemplo de "garagem" onde a soma de pontos da dupla adversária é igual a 12. A pontuação da jogada é igual a 10 pontos.

Nas rodadas seguintes a primeira, o jogador que "bateu" na rodada anterior deverá iniciar com a carroça de sua escolha.

No dominó de quatro pontas, devido ao fato do objetivo e das regras serem mais elaboradas, as estratégias de jogo são mais complexas que às utilizadas no dominó de duas pontas.

As estratégias utilizadas durante o jogo do dominó de 4 pontas, além do objetivo de pontuar, almejam também facilitar as ações futuras de quem joga (ou de seu parceiro) e dificultar a ação dos jogadores da dupla adversária. Quando no início de uma rodada, um jogador tem em mãos quatro ou mais peças que apresentem a mesma numeração em um das metades (vide exemplo na figura 4), a estratégia recomendada é fazer com que essa numeração esteja presente no maior número possível de pontas, com o intuito tanto de fazer seus adversários "passarem" como de facilitar suas jogadas futuras. Para o conjunto de pedras mostrado na figura 4 é interessante buscar uma configuração de jogo onde exista um maior número de pontas com a numeração 5.

O jogo de domino de 4 pontas caracteriza-se por ser um problema multiagente [Sycara 1998]. Trata-se de um jogo de vários jogadores, de soma não zero, de informações imperfeitas. Na sua modalidade mais jogada, duas duplas, caracteriza-se como sendo de dois jogadores, pois só existem duas pontuações. Por ser um jogo com informações imperfeitas, um algoritmo de busca para selecionar a melhor jogada explora um espaço de estados de crença (crenças sobre quem tem determinadas peças e com que probabilidade) [Russel e Norvig 2004]. Tais algoritmos utilizam raciocínios probabilísticos complexos.

Nesse trabalho propõe-se a pesquisa de um heurística que possibilite o desenvolvimento de um agente inteligente para o jogo do dominó de 4 pontas. Procurou-se uma alternativa mais simples para a escolha da melhor jogada, baseada na utilização de uma função de avaliação que combina nos seus termos informações sobre o estado presente do jogo. Tal função incorpora informações estratégicas que visam facilitar as ações futuras da dupla do jogador que joga e dificultar as ações futuras da dupla adversária.

Figure 4: Exemplo de um conjunto de 7 peças iniciais com 4 peças contendo a numeração 5.

Na seção trabalhos relacionados analisa-se alguns trabalhos relacionados com a aplicação de funções de avaliação a jogos. Na seção metodologia propõe-se um conceito de estado para o jogo de dominó de 4 pontas e propõe-se uma função heurística para o mesmo em função do estado proposto. Na seção de resultados mostra-se o desempenho de um jogo de duas duplas em função de algumas escolhas feitas para os parâmetros da função heurística. Na seção de discussão propõe-se a continuidade do trabalho através da exploração de outros valores para os parâmetros da função heurística e através da utilização de algoritmos genéticos.

2. Trabalhos Relacionados

A utilização de funções de avaliação associadas ao algoritmo alfa-beta para a escolha da melhor opção de jogada em jogos de informações perfeitas foi inicialmente proposta por Shannon [Shannon 1950]. Tal proposta foi implementada para os jogos de xadrez [Campbell et al. 2002] e de damas [Schaeffer, 1997].

Para os jogos de informação imperfeita a utilização da proposta de Shannon envolve raciocínios complexos sobre as estratégias do jogo. Cita-se como exemplo o programa Bridge Baron [Smith et al. 1998], desenvolvido para o jogo de Bridge. Para o jogo de dominó de 2 pontas, encontram-se na literatura alguns trabalhos que buscam a melhor estratégia do jogo, como Yen e Chlebus [Yen 1992, Chlebus, 1986]. Não encontrou-se, porém trabalhos relacionados ao dominó de 4 pontas.

3. Metodologia

A função de avaliação proposta para a escolha da melhor jogada é constituída pela soma de dois termos, conforme mostra a expressão (1). A variável n identifica uma opção de jogada para a qual a função de avaliação é calculada. O termo T_1 corresponde aos pontos obtidos ao se efetuar a opção de jogada n. O termo T_2 incorpora a estratégia do jogo. O mesmo corresponde a valores inteiros positivos que procuram retratar como a escolha pela jogada n pode facilitar ações futuras do jogador que efetua a jogada e dificultar ações futuras dos jogadores adversários.

$$f(n) = T_1 + T_2 \quad (1)$$

Em que:
n – opção de jogada a ser avaliada;
T_1 – soma dos pontos obtidos com a jogada n;
T_2 – termo que incorpora a estratégia do jogo.

Antes que seja feito o detalhamento de como se calculam os termos T_1 e T_2 da função f(n) é necessário definir-se um estado para o jogo de dominó de 4 pontas. O estado do jogo é definido nesse trabalho através de um conjunto de vetores, V_i , onde $0 \leq i \leq 6$. Esses vetores expressam estatísticas que são atualizadas em cada jogada. Os quatro primeiros vetores contêm valores inteiros enquanto que os três últimos contêm valores binários. A definição dos mesmos é feita a seguir.

1) V_0: Vetor contendo a quantidade de peças em jogo para cada numeração. Esse vetor é expresso por: $V_0 = \{a_0, b_0, c_0, d_0, e_0, f_0, g_0\}$. Em que: a_0 corresponde ao número de peças já jogadas com a numeração 0; b_0 corresponde ao número de peças já jogadas com a numeração 1, e assim sucessivamente;

2) V_1: Vetor contendo a quantidade de peças na mão para cada numeração. Esse vetor é expresso por: $V_1 = \{a_1, b_1, c_1, d_1, e_1, f_1, g_1\}$. Em que: a_1 corresponde ao número de peças na mão com a numeração 0; b_1 corresponde ao número de peças na mão com a numeração 1, e assim sucessivamente;

3) V_2: Vetor contendo a quantidade de peças nas pontas para cada numeração. Esse vetor é expresso por: $V_2 = \{a_2, b_2, c_2, d_2, e_2, f_2, g_2\}$. Em que: a_2 corresponde ao número de peças nas pontas com a numeração 0; b_2 corresponde ao número de peças nas pontas com a numeração 1, e assim sucessivamente;

4) V_3: Vetor contendo a quantidade de peças já jogadas pelo parceiro de dupla para cada numeração . Esse vetor é expresso por: $V_3 = \{a_3, b_3, c_3, d_3, e_3, f_3, g_3\}$. Em que: a_3 corresponde ao número de peças já jogadas pelo parceiro com a numeração 0; b_3 corresponde ao número de peças já jogadas pelo parceiro com a numeração 1, e assim sucessivamente;

5) V_4: Vetor que indica as numerações onde o adversário seguinte já passou. Esse vetor é expresso por: $V_4 = \{a_4, b_4, c_4, d_4, e_4, f_4, g_4\}$. Em que: a_4 indica se o adversário seguinte já passou ou não para a numeração 0. Se a_4 for igual a 1 o adversário seguinte já passou para a numeração 0. Se a_4 for igual a 0 o adversário seguinte ainda não passou para a numeração 0; b_4 indica se o adversário seguinte já passou ou não para a numeração 1. Se b_4 for igual a 1 o adversário seguinte já passou para a numeração 1. Se b_4 for igual a 0 o adversário seguinte ainda não passou para a numeração 0, e assim sucessivamente;

6) V_5: Vetor que indica as numerações onde o adversário anterior já passou. Esse vetor é expresso por: $V_5 = \{a_5, b_5, c_5, d_5, e_5, f_5, g_5\}$ Em que: a_5 indica se o adversário anterior já passou ou não para a numeração 0. Se a_5 for igual a 1 o adversário anterior já passou para a numeração 0. Se a_5 for igual a 0 o adversário anterior ainda não passou para a numeração 0; b_5 indica se o adversário anterior já passou ou não para a numeração 1. Se b_5 for igual a 1 o adversário anterior já passou para a numeração 1. Se b_5 for igual a 0 o adversário anterior ainda não passou para a numeração 0, e assim sucessivamente;

7) V_6: Vetor que indica as numerações onde o parceiro de dupla já passou. Esse vetor é expresso por: V_6= {a_6, b_6, c_6, d_6, e_6, f_6, g_6}. Em que: a_6 indica se o parceiro de dupla já passou ou não para a numeração 0. Se a_6 for igual a 1 o parceiro de dupla já passou para a numeração 0. Se a_6 for igual a 0 o parceiro de dupla ainda não passou para a numeração 0; b_6 indica se o parceiro de dupla já passou ou não para a numeração 1. Se b_6 for igual a 1 o parceiro de dupla já passou para a numeração 1. Se b_6 for igual a 0 o parceiro de dupla ainda não passou para a numeração 0, e assim sucessivamente;

A seguir analisar-se-á como são calculados os termos T_1 e T_2. O termo T_1 incorpora no cálculo da função de avaliação os pontos obtidos ao se realizar a opção de jogada n. Na seção de introdução foram listadas 5 situações distintas em que se pode pontuar numa jogada. O termo T_1 incorpora no seu cálculo os pontos que podem ser obtidos através das situações P_1, P_2, P_3 e P_5. Assim T_1 é dado por (2):

$$T_1 = P_1 + P_2 + P_3 + P_5 \qquad (2)$$

Observar que T_1 não incorpora a situação P_4, que corresponde aos pontos de garagem, pois os mesmos, na medida em que se encontram nas mãos da dupla adversária, não podem ser contabilizados. Os termos P_2 e P_3 são calculados através de um único algoritmo. Para o cálculo dos mesmos são utilizados os vetores V_4, V_5 e V_6. Na figura 5 mostra-se um fluxograma desse algoritmo. Para o entendimento desse fluxograma e do texto que se segue as seguintes definições são necessárias:

Np_1, Np_2, Np_3 e Np_4 – valores inteiros correspondentes as numerações existentes nas pontas 1, 2 ,3 e 4 do jogo, respectivamente. A numeração das pontas não obedece a nenhuma ordem pré-definida;
L – peça a ser inserida no jogo na opção de jogada n.
L_1 - valor inteiro correspondente a numeração da metade da peça L que coincide com um ou mais dos valores Np_1, Np_2, Np_3 ou Np_4.
L_2 – valor inteiro correspondente a numeração da metade da peça L que não coincide com um ou mais dos valores Np_1, Np_2, Np_3 ou Np_4.

Para o fluxograma mostrado na figura 5 supõe-se que L_1=Np_4.

O termo T_2 incorpora no seu cálculo três parcelas distintas relacionadas com a estratégia do jogo. A expressão para T_2 é mostrada em (3).

$$T_2 = \alpha \cdot (-E_1 + E_2 + \delta E_3) \qquad (3)$$

O valor de α permite modelar a importância da estratégia (T_2) em relação aos pontos obtidos com uma jogada (T_1).

A parcela E_1 considera a possibilidade de se fazer o adversário seguinte passar na sua vez de jogar. Para o entendimento da parcela E_1 será utilizada a seguinte situação hipotética: considere que em duas das pontas do jogo exista uma numeração n_1 e que o jogador da vez tenha em mãos mais três peças com essa mesma numeração. Então, 5 das 7 peças possíveis com a numeração n_1 não pertencem ao adversário seguinte, sendo grande a possibilidade do mesmo passar se todas as pontas estiverem com essa numeração. Tendo em vista a possibilidade de fazer o adversário seguinte passar, e dessa forma obter 20 pontos, é desejável que não seja descartada nenhuma das peças em que L_1=n_1. Assim sendo a opção de jogada de uma peça L em que L_1=n_1 não é desejada do ponto de vista estratégico e daí o sinal negativo para E_1 na expressão (3). A determinação de E_1 é realizada conforme mostrado na expressão (4). Quanto maior o número de peças com a numeração L_1 nas pontas e nas mãos de quem joga maior será o valor de E_1. Por outro lado, o valor de E_1 deve ser tanto maior quanto maior for o número de peças com a numeração L_1 já jogadas. O valor de K_1 controla a importância da parcela E_1 frente às outras parcelas utilizadas no cálculo de T_2.

$$E_1 = K_1 \cdot (V_1(L_1) + V_2(L_1) + V_0(L_1)) \qquad (4)$$

A parcela E_2 tem por objetivo facilitar as ações futuras de quem joga. A determinação de E_2 é realizada conforme mostrado na expressão (5). Essa expressão é muito semelhante àquela proposta para E_1, sendo a única diferença que ao invés de L_1 utiliza-se L_2 como argumento independente. O valor de K_2 controla a importância da parcela E_2 frente às outras parcelas utilizadas no cálculo de T_2.

$$E_2 = K_2 \cdot (V_1(L_2) + V_2(L_2) + V_0(L_2)) \qquad (5)$$

A parcela E_3 reforça o objetivo da parcela E_2, visando facilitar as ações futuras de quem joga quando o número de peças na mão é menor ou igual a 3. No fluxograma da figura 6 mostra-se como é realizado o cálculo de E_3. O valor de K_3 controla a importância da parcela E_3 frente as outras parcelas utilizadas para o cálculo de T_2.

4. Resultados

Para se testar a heurística proposta foram realizadas simulações de um jogo com duas duplas de jogadores. A primeira dupla, denominada dupla 1, realizou a escolha da melhor jogada utilizando a função f_1 dada pela expressão (6).

$$f_1 = T_1 \qquad (6)$$

A segunda dupla, denominada dupla 2, realizou a escolha da melhor jogada utilizando a função f_2 dada pela expressão (7):

$$f_2 = T_1 + T_2 \qquad (7)$$

Os valores de α na expressão para f_2 foram variados entre 0 e 10, em incrementos de 0,1 entre 0 e 1 e em incrementos de 0,25 entre 1 e 10. Aos valores de K_1, K_2 e K_3 foram atribuídos o mesmo valor, 2, fazendo com que E_1, E_2 e E_3 assumissem igual importância no cálculo de T_2.

Para cada valor de α realizou-se um total de 100.000 partidas e anotou-se o número de vitórias da dupla 1 e da dupla 2. Nas tabelas 1 e 2 mostram-se esses resultados.

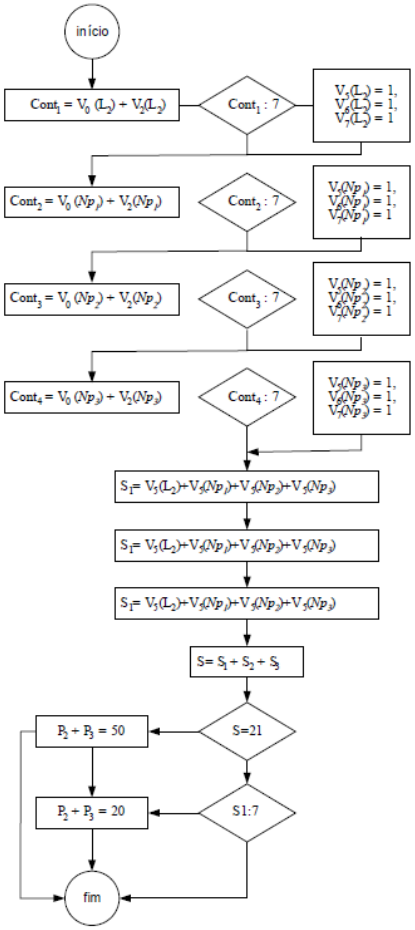


Figure 5: Fluxograma para o cálculo de $P_2 + P_3$.

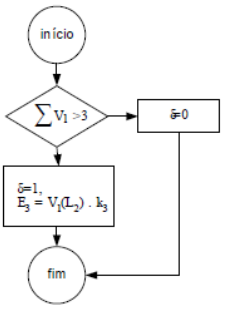


Figure 6: Fluxograma para o cálculo de E_3.

Tabela 1: Resultados da simulação com α variando no intervalo entre 0 e 5.

Coef. α	Número de Vitórias em 100.000 partidas	
	Dupla 1	Dupla 2
0	50248	49752
0,01	36541	63459
0,02	36541	63459
0,03	36541	63459
0,04	36541	63459
0,05	36504	63496
0,10	36504	63496
0,15	36504	63496
0,20	36504	63496
0,25	36579	63421
0,50	36278	63722
0,75	35565	64435
1,00	35052	64948
1,25	34659	65341
1,50	34233	65767
1,75	33998	66002
2,00	34076	65924
2,25	33981	66019
2,50	34300	65700
2,75	34383	65617
3,00	34507	65493
3,25	34574	65426
3,50	35018	64982
3,75	35267	64733
4,00	35510	64490
4,25	35571	64429
4,50	35574	64426
4,75	35574	64426
5,00	36378	63622

Tabela 2: Resultados da simulação com α variando entre 5,25 e 10.

Coeficiente α	Número de Vitórias em 100.000 partidas	
	Dupla 1	Dupla 2
5,25	36907	63093
5,50	36907	63093
5,75	36907	63093
6,00	36909	63091
6,25	36937	63063
6,50	36948	63052
6,75	37162	62838
7,00	37162	62838
7,25	37163	62837
7,50	37734	62266
7,75	38236	61764
8,00	38236	61764
8,25	38236	61764
8,50	38265	61735
8,75	38265	61735
9,00	38265	61735
9,25	38270	61730
9,50	38270	61730
9,75	38273	61727
10,00	38839	61161

4. Discussão

Através da observação das tabelas 1 e 2 observa-se que: A dupla 2, a exceção de α=0, sempre obteve um número de vitórias maior que a dupla 1; O maior número de vitórias obtidas pela dupla 2 foi 66002, tendo ocorrido para um valor de $\alpha = 2.25$; Para $\alpha = 0$, quando as duas funções tornam-se iguais, f_1=f_2, o número de vitórias das duas duplas é praticamente igual; Quando α varia de 0 para 0,01, o número de vitórias da dupla 2 cresce de forma abrupta, de 49752 para 63459. As outras variações observadas para o número de vitórias da dupla 2 em função de α são mais suaves.

As observações anteriores permitem concluir que a dupla que escolheu a melhor jogada utilizando a função de avaliação f_2 teve um desempenho superior a dupla que escolheu a melhor jogada utilizando a função de avaliação f_1. A responsabilidade por esse melhor desempenho cabe ao termo de estratégia T_2 presente na função f_2 e ausente na função f_1. O melhor resultado alcançado foi de 66% de vitórias para a dupla que escolheu a melhor jogada utilizando a função f_2.

Para obtenção desses resultados utilizou-se um mesmo valor para K_1, K_2 e K_3 igual a 2, escolhido de forma empírica.

Em trabalhos futuros pretende-se: Realizar outras simulações com a função de avaliação f_2 associando α a diferentes valores de K_1, K_2 e K_3; Utilizar algoritmo genético para otimização de uma outra função de avaliação onde o termo T_2 seja dado pela expressão (8). Essa expressão reúne as parcelas presentes nos termos E_1, E_2 e E_3. Através do algoritmo genético serão obtidos os valores de α_1, α_2, α_3, α_4, α_5 e α_6 que otimizarão o número de vitórias da dupla que utilizar a função f_2 para a escolha da melhor jogada.

$$T_2 = \alpha_1.V_1(L_1) + \alpha_2.V_2(L_1) + \alpha_3.V_0(L_1) + \alpha_4.V_1(L_2) + \alpha_5.V_2(L_2) + \alpha_6.V_0(L_2) \quad (8)$$

5. Conclusão

Propôs-se nesse trabalho uma nova metodologia para a escolha da melhor jogada no dominó de 4 pontas, utilizando uma função heurística que, para o seu cálculo, utiliza informações provenientes do "estado do jogo", definido como um conjunto de 7 vetores atualizados a cada jogada. Os parâmetros utilizados na função heurística não foram otimizados nesse trabalho. Propõe-se isso seja feito em trabalho futuro. Os resultados foram avaliados através de simulações com jogos de duplas. O melhor resultado alcançado em 100.000 partidas foi de 66% de vitórias para a dupla que utilizou a função de avaliação proposta nesse trabalho. Espera-se que o processo de otimização proposto para trabalhos futuros resulte em uma estatística de vitórias mais elevada. Acredita-se que a metodologia apresentada nesse trabalho para a escolha da melhor jogada seja original. Preetende-se, em trabalhos futuros, explorar a aplicação dessa metodologia em outros tipos de jogos.

Agradecimentos

Esse trabalho teve o apoio financeiro da SUFRAMA (convênios 068/2001 e 069/2001) e do CNPq (bolsa de IC).

Referências

CAMPBELL, M.S., HOANE, A.J., HUS, F.H. 2002, Deep Blue, Artificial Intelligence, 134(1-2), 57-83.

CHLEUBS, B.S., 1986, Domino-tiling games, Journal of Computer and System Sciences, v. 32, n.3, 374-392.

RUSSEL, S., NORVIG, P., 2004, Inteligência artificial, Elsevier, 2ª edição;

SHAEFFER, J., 1997, One Jump Ahead: Challenging Human Supremacy in Checkers, Springer-Verlag, Berlin.

SHANNON, C.E. , 1950, Programming a computer for playing chess. Philosophical Magazine, 41(4), 256-275.

SMITH, S.J.J, NAU, D.S., THROOP, T.A., 1998, Success in spades: Using ai planning techniques to win the world championship of computer bridge. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, 1079-1086, Madison, Wisconsin, AAAI Press.

SYCARA, K.P., 1998. Multiagent systems, AI Magazine, v. 19, n. 2, 79-92.

YEN, H.C., 2002. A multiparameter analysis of domino tiling with an application to concurrent systems, Theoretical Computer Science, v. 98, n. 2, 263-287.

Optimization of an Evaluation Function of the 4-sided Dominoes Game Using a Genetic Algorithm

Nirvana S. Antonio, Cícero F. F. Costa Filho, Marly G. F. Costa, Rafael Padilla

***Abstract*— In 4-sided dominoes, the popular way of playing dominoes in Amazonas State, the strategies used for the game are more complex than those adopted in the more traditional 2-sided dominoes, the most popular dominoes game played in Brazil. This work presents the optimization of an evaluation function for the best move in 4-sided dominoes using a genetic algorithm. The evaluation function is composed of terms that incorporate the game's strategies and are defined as: punctuating, facilitating future moves and complicating opponents' moves. Coefficients were defined to determine the importance of each term of the evaluation function and a set of parameters and operators for implementation of the genetic algorithm. The players' ability was calculated by the number of wins in 5,000 matches. The results obtained during the simulations showed that the team (composed of 2 players) that used the evaluation function with its coefficients optimized by the genetic algorithm won in more than 70% of the total matches.**

I. Introduction

Dominoes is a game consisting of rectangular pieces divided in 2 halves, which are marked with black dots indicating a numeric value. Possibly having originated in China, dominoes is a popular game all over the world and can be played in many different ways, depending on the region. In Brazil, the most popular way is called "2-sided dominoes". However, in Amazonas State, another variation of the game is more popular among the locals, dominoes played on four sides.

4-sided dominoes is characterized by being a multi-agent problem [1], because it is not a zero sum game, played by four players divided in two pairs, which have imperfect information about the possible moves of the other. When played by 2 pairs, it is considered a two-player game, as there are only two scores. Therefore, each player must develop strategies based on incomplete information in order to win the game with his/her partner.

N.S. Antonio is with *Universidade Federal do Amazonas/Centro de P&D em Tecnologia Eletrônica e da Informação* – UFAM/CETELI. Manaus. Amazonas. Brasil (tel: +55 92 8414 1406; e-mail: nirvana.sa@gmail.com).

C. F. F. Costa Filho is with *Universidade Federal do Amazonas/Centro de P&D em Tecnologia Eletrônica e da Informação* – UFAM/CETELI. Manaus. Amazonas. Brasil (tel:: +55 92 91464954; e-mail: ccosta@ufam.edu.br).

M. G. F. Costa is with *Universidade Federal do Amazonas/Centro de P&D em Tecnologia Eletrônica e da Informação* – UFAM/CETELI. Manaus. Amazonas. Brazil (tel: +55 92 9128 2404; e-mail: mcosta@ufam.edu.br).

R. Padilla is with *Universidade Federal do Amazonas/Centro de P&D em Tecnologia Eletrônica e da Informação* – UFAM/CETELI. Manaus. Amazonas. Brasil (tel: +55 92 9165 1123; e-mail: eng.rafaelpadilla@gmail.com).

The use of an evaluation function associated with the alpha-beta algorithm for choosing the best move in games of perfect information was first proposed by Shannon [3]. This proposal was implemented for chess [4] and checkers [5]. For games of imperfect information, the use of Shannon's proposal involves complex reasoning about game strategy. It is mentioned for example in the Bridge Baron program [6], developed for Bridge. For 2-sided dominoes, some studies can be found in the literature seeking the best game strategy [7] [9].

In [2], we developed a methodology for choosing the best move based on an evaluation function that combines in its terms information about the present state of the game. To evaluate the developed evaluation function, simulations were conducted for 4-sided dominoes games, in which a pair used the evaluation function for choosing the best move while the other pair performed their moves based only on the dots of the dominoes on the table. The choice of the parameters of the evaluation function was done manually and as result, the first pair won the game in more than 66% of the simulations.

In this study, we will perform the optimization of the parameters that make up the evaluation function applied to some strategies that can be adopted by the player. The optimization will be achieved by the optimization and search technique of the genetic algorithms (GA). GA is a very efficient technique for searching for optimal or near-optimal solutions. Moreover, it is easy to implement and allows for parallel computing. With the optimization process, we hope that the pair that uses the evaluation function to choose their moves performs better than their opponents that use only the punctuations of the pieces on the table as a criterion to choose their moves, considered a basic strategy used by beginners.

II. DOMINOES GAME

Dominoes is a game consisting of 28 rectangular and flat pieces (or tiles). These pieces are divided in two halves that, in the classic form of the game, contain numbers ranging from zero to six dots, marking all combinations between these numbers. It can be played by four players, in pairs or individually, where each player receives seven pieces or, alternatively, can be played by two players, where the 14 remaining tiles are "bought" during the match. There are several ways to play dominoes and the most common form in Brazil is called 2-sided dominoes. However, in Amazonas State, it is played in another less popular way, 4-sided dominoes.

978-1-4577-0011-8/11/$26.00 ©2011 IEEE

In 4-sided dominoes, the games are played by two pairs. Each player must have seven tiles in hand at the beginning of each turn. In the first round, the first piece to be displayed is the 6-6, called "double six", having two halves numbered with 6 dots each. The players have the possibility to play off the 4 sides of this first piece. In Figure 1, we can see a game where 3 points have been opened from the "double six".

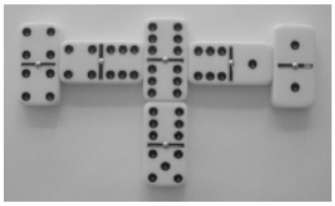


Fig. 1. Example of a move that is scored 15 points

In the following rounds, the player who "hit" (used up all their "stones" (tile pieces) before the other players) in the previous round will start with the double of his choice. The main objective of 4-sided dominoes is to achieve a score of 200 or more points, in one or more rounds of play. During the game, there is the possibility of scoring 5, 10, …, 50 points, in other words, multiples of 5. The chances of scoring can be defined by the following conditions:

P1) The sum of the dots on the ends is a multiple of 5 and the score recorded for the pair is equal to this sum. Figure 1 exemplifies a move equivalent to 15 points, because the sum of the dots on the ends of the double that started the game (2+5+8) is a multiple of 5. However, Figure 2 shows an example of a move where there is no score, because the sum of the dots on the ends is 14 (1+5+8).

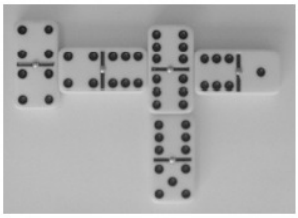


Fig. 2. Example of a move with no score

P2) The opponent player has no piece to fit in any of the 4 ends, so he "skips" his turn instead of playing, impeding an opponents' play is equivalent to a score of 20 points for the pair who caused the skipping.

P3) When the player causes all other players to pass (including his partner). This pass, called a "rooster" is equivalent to a score of 50 points.

P4) When a player "hits", the dots of the pieces left in the hands of the opponents are totaled. This sum is called "the garage". The score corresponds to the highest multiple of 5 less than or equal to this sum. Figure 3 shows an example where the "garage" is worth 10 points since the score of the opponents is 12.

Fig. 3. "Garage" example of 10 points

P5) The last piece discarded by the player who goes out is a "double". The score of this move is equal to 20 points.

The game of 4-sided dominoes, having more elaborated goals and rules, adopts more complex strategies than those used in 2-sided dominoes, because in addition to scoring, the player should also try to facilitate his (or his partner's) future moves and impede the moves of their opponents. To illustrate the strategies adopted during the game, it is considered that at the beginning of a round, a player has in hand four or more pieces with the same number (Figure 4). In this scenario, the strategy is to try to make this same number present on the most possible sides, since this type of move could force his opponent to skip his turn or may facilitate future moves.

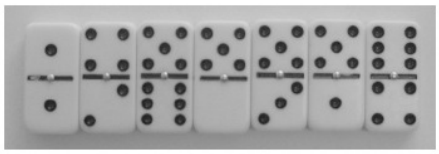


Fig. 4. Example of a set of initial tiles of a player where four of them have the number 5

III. EVALUATION FUNCTION FOR SELECTING THE BEST MOVE

Previously we developed an intelligent agent for 4-sided dominoes that chooses the best move by an evaluation function [2]. The evaluation function consists of the sum of two terms, T_{n1} and T_{n2}, as defined by equation (1).

$$f(n) = T_{n1} + T_{n2} \quad (1)$$

In the expression above, the variable n identifies a move for which the evaluation function is calculated. The term T_{n1} corresponds to points obtained by performing move n. The term T_{n2} incorporates the strategy of the game, corresponding to values that represent how choosing move n can facilitate future actions of the player and make future moves of the opponents even more difficult.

To calculate the terms T_{n1} and T_{n2} of the evaluation function, the current state of the game was used. The state of the game was defined as a set of 7 vectors, $V_i = \{a_i, b_i, c_i, d_i, e_i, f_i, g_i\}$, where a_i corresponds to the number of pieces (tiles) played with numbers 0; b_i corresponds to the number of pieces played with numbers 1, and so on. These vectors express statistics updated every move. The definition of states of 4-sided dominoes is shown below:

- V_0: quantity of pieces in the game for each number.
- V_1: quantity of pieces in the hands of the players for each number
- V_2: quantity of pieces for each number at the tile ends.
- V_3: quantity of pieces already played by the pair for each number.
- V_4: indicates the numbers where the following opponent has already passed. If a_4 is equal to 1, the following opponent has already passed the numbering 0. If a_4 is equal to 0, the following opponent still has not passed the numbering 0, and so on.
- V_5: indicates the numbers where the previous opponent has already passed.
- V_6: indicates the numberings where the partner has already passed.

The term T_{nl} incorporates in the calculation of the evaluation function the points obtained when the move option n is made through situations P_1, P_2, P_3 and P_5, as shown in equation (2).

$$T_{n1} = P_1 + P_2 + P_3 + P_5 \quad (2)$$

It is noteworthy that the T_{n1} does not incorporate situation P_4, which corresponds to the points of the "garage", because they cannot be counted while the round is not complete.

To understand the text that follows, let us define L as a possible option to play, which means, a tile in the player's hand. Each tile has numbers in its two halves. L_1 corresponds to the numbering of the half of the stone that matches one of the values in any tile ends of the game, and L_2 corresponds to the other half that does not match the values at the ends. Thus, if a player has the option to place a piece numbered 5-3 and the game on the table is prepared as illustrated in Figure 1, L_1 will receive the number 5, as it matches one side and L_2 will receive the number 3, which does not match the side.

The term T_{n2} includes in its calculation two distinct portions related to the game's strategy, as shown in equation (3):

$$T_{n2} = -E_1 + E_2 \quad (3)$$

Portion E_1 considers the possibility of forcing the subsequent opponent to skip his turn. To understand portion E_1, it will be used in the following hypothetical situation: let us suppose that in two of the tile ends there is the number n_1 and the player taking a turn has three pieces in his hands with the same numbering. So, five of the seven possible pieces with numbering n_1 do not belong to the opponent playing next; therefore, the probability of the opponent skipping his turn is higher if all the ends have this numbering. Given the possibility of making the next opponent skip his turn and, thus score 20 points, it is desirable that no piece with numbering $L_1 = n_1$ is discarded. In this case, the option to play a tile in which $L_1 = n_1$ is not desired from a strategic perspective, the negative sign for E_1 is given in the expression (3). The calculation of E_1 is performed according to expression (4).

$$E_1 = \alpha_1.V_0(L_1) + \alpha_2.V_1(L_1) + \alpha_3.V_2(L_1) \quad (4)$$

The more pieces with numbering L_1 on the ends (represented by vector V_2) and in the hands of the player (represented by the vector V_1), the greater the value of E_1 is. On the other hand, the value of E_1 should also be directly proportional to the number of pieces with the numbering L_1 already played (vector V_0). The coefficients α_1, α_2 and α_3 control the importance of each term of E_1 in relation to other parcels compromising the evaluation function.

The portion E_2 aims to facilitate future actions of the player. The calculation of E_2 is performed according to equation (5).

$$E_2 = \alpha_4.V_0(L_2) + \alpha_5.V_1(L_2) + \alpha_6.V_2(L_2) + \alpha_7.V_3(L_2) \quad (5)$$

This expression is very similar to the one proposed for E_1, but L_2 is used instead of L_1 as independent argument, which is the number the player wants to add on the table. There is also a term related to facilitating his partner's play, represented by the vector V_3. In this case, the term examines the pieces played by his partner to the numbering L_2 presented at the tile ends. The coefficients α_4, α_5, α_6 and α_7 control the importance of each term in E_2 in relation to other parcels that compose the evaluation function.

The evaluation function described in equations (1) - (5) involves the main cases considered by a player while deciding the best move at each round. In other words, it proposes a possible strategy to be adopted in 4-sided dominoes. In this work, this strategy will be called "Strategy 1" and its evaluation function will be used to calculate the fitness of the chromosomes during the execution of the genetic algorithm. At the end of the optimization, the genetic algorithm will find the coefficients α_1, α_2, α_3, α_4, α_5, α_6 and α_7 optimized for this strategy.

Besides the strategy outlined above, other strategies were analyzed in this work. In the second strategy, the player gives higher priority only to moves that can promote his own game; in other words, the coefficients α_1, α_2, α_3 (makes the opponents' moves more difficult) and α_7 (analyzes the moves that make his partner's game easier) are set to zero. Therefore, the evaluation function to be optimized by the genetic algorithm for Strategy 2 is made only by the terms of

equation (6). At the end of optimization, the genetic algorithm will find the coefficients α_4, α_5 and α_6 optimized for this strategy.

$$f(n) = T_{n1} + \alpha_4.V_0(L_2) + \alpha_5.V_1(L_2) + \alpha_6.V_2(L_2) \quad (6)$$

In the third strategy, the player only gives higher priority to moves that can promote his partner's game, in other words, the coefficients α_1, α_2, α_3 (makes his opponents' moves harder) and α_5 (analyzes moves that make his own game easier) are set to zero. Therefore, the evaluation function optimized by the genetic algorithm for Strategy 3 is formed only by the terms of equation (7). At the end of optimization, the genetic algorithm will find the coefficients α_4, α_6 and α_7 optimized for this strategy.

$$f(n) = T_{n1} + \alpha_4.V_0(L_2) + \alpha_6.V_2(L_2) + \alpha_7.V_3(L_2) \quad (7)$$

The fourth strategy gives higher priority to moves that make the opponents' actions more difficult, in other words, coefficients α_4, α_5, α_6 and α_7 (facilitates the moves of the pair) are set to zero. Therefore, the evaluation function to be optimized by genetic algorithm for Strategy 4 is formed only by the terms of equation (8). At the end of optimization, the genetic algorithm will find coefficients α_1, α_2 and α_3 optimized for this strategy.

$$f(n) = T_1 + \alpha_1.V_0(L_1) + \alpha_2.V_1(L_1) + \alpha_3.V_2(L_1) \quad (8)$$

The fifth strategy to be considered in this work is the basic strategy of 4-sided dominoes, being the strategy used by novice players. This strategy considers only the term T_{n1} of the evaluation function described in equation (1); in other words, only those points are considered for choosing the best move.

IV. OPTIMIZATION OF THE EVALUATION FUNCTION

A. Parameters of the Genetic Algorithm

In order to optimize the evaluation function using GA, we use a combination of parameters for better results, as described below.

Coding: the chromosomes are encoded as a vector of type double, which represent parameters α_1, α_2, α_3, α_4, α_5, α_6 and α_7 of the evaluation function.

Fitness function (*fitness*): the fitness function is responsible for measuring the performance of a chromosome during the simulation of domino matches. The fitness value of a chromosome is given by the number of wins resulting from the performed matches.

Population size: the size of the population directly affects the performance of the algorithm. Very small populations have less genetic diversity and can lead to a faster convergence of the algorithm to a local minimum. On the other hand, very large populations make the algorithm too slow [10]. For the optimization performed in this work, we use the following population sizes: 40, 60, 80 and 100.

Crossover for the optimization process, two techniques of crossing were used: *Two-point Crossover and Uniform Crossover*. The rate of crossover p_c, which defines the probability that the selected chromosomes pass by the process of crossing, will be 0.8 in this work.

Mutation: To implement the mutation process, the operator we chose was the Uniform Mutation and the value of mutation probability, p_m, takes the values 0.1 and 0.5.

Stopping criterion: as stop criterion of the genetic algorithm, we used the total number of generations in the following sizes: 50, 100, 150 and 200.

B. Implementation

For the simulation of 4-sided dominoes, a program was developed in Java with four agents acting as the four players required for the match. In this program, the first pair uses a basic strategy for choosing the best move and the second pair may have its parameters α_1, α_2, α_3, α_4, α_5, α_6 and α_7 adjusted for each simulation. Thus, the simulator of 4-sided dominoes has as input parameters the total number of desired matches and the set of coefficients α of the second pair, returning the number of wins of this pair.

To implement the optimization by genetic algorithm, we use the Genetic Algorithm toolbox of *MATLAB*, the GAtool. A function was created in MATLAB (dfitness.m) responsible for calculating the fitness of each tested chromosome, by adjusting the input parameters of the domino simulator in Java. Besides the fitness function (or *fitness*), the population type was defined as a *Double Vector*, receiving values in the range of -10 to 10.

In the selection process, we used the Roulette method, which simulates a roulette wheel with the area corresponding to each chromosome proportional to its fitness value; in other words, the probability of a chromosome being selected is directly proportional to its area on the roulette wheel.

For the configuration of the reproduction process, the parameter *Elite count*, which indicates how many individuals with the best fitness values will be transmitted directly to the next generation without going through the reproduction, receives values corresponding to 10% of the population size. We used two crossover operators (or crossover): *Two-point* and *Scattered* (also known as *Uniform crossover*).

The purpose of performing various optimizations is to find a combination of the genetic algorithm parameters required to achieve the optimal or near-optimal values for the coefficients α_1, α_2, α_3, α_4, α_5, α_6 and α_7 of the evaluation function, in a way that they allow the pair that uses this function to choose the best move, has a better performance than the pair that only uses the strategy of adding points during the match (coefficient α equal to zero).

The set of parameters defined previously gives us a combination of 64 executions of the genetic algorithm optimization for each strategy, in a total of 256 runs of the genetic algorithm. A summary of all variations of these

parameters is shown in Figure 5:

Population size	Crossover function	Mutation rate	Generations
• 40 • 60 • 80 • 100	• Two-point • Scattered	• 0.1 • 0.5	• 50 • 100 • 150 • 200

Fig. 5. Parameters used for the optimization of genetic algorithm.

V. Results and Discussion

The following results show the optimizations obtained for Strategies 1, 2, 3 and 4 playing against the "basic strategy" game. For Strategy 1, the genetic algorithm performed the optimization of the seven coefficients α. On the other hand, for Strategies 2, 3 and 4, the genetic algorithm performed the optimization of only three coefficients α_i, because the others are set to zero, depending on the type of strategy.

A. *Optimization of the evaluation function of Strategy 1*

The optimization of the evaluation function corresponding to the first strategy was divided into four groups according to the size of the population, receiving the values 40, 60, 80 and 100. For each parameter combination, the amount of wins was registered for pair 2 and the coefficients resulting from the optimization process. Table 1 and Figure 6 show the best results for each optimization group of the evaluation function of Strategy 1 together with the parameters used in the genetic algorithm during the optimization process.

According to the simulations, the best results were obtained in the optimizations with a total of 100 generations, regardless of population size. On the other hand, an\ increase in population size led to an improvement of the results.

TABLE I
Optimization results of the evaluation function of Strategy 1

Population	Generations	Crossover Operator	p_m	Wins (pair 2)
40	100	Two-point	0.5	69.24 %
60	100	Scattered	0.1	69.68 %
80	100	Scattered	0.5	69.76 %
100	100	Two-point	0.1	70.06 %

The best results obtained by pair 2, which uses Strategy 1, are displayed with the parameters used in Genetic Algorithm.

The first group of the optimizations corresponds to the population of 40 chromosomes. In this group, a total of 5.000 matches were held: pair 2 won 3,462 matches, which represents 69.24% of wins. The second group corresponds to the combinations with population size equals to 60. In this group, the best result is 3,484 winnings in 5,000 matches, which corresponds to a 69.68% yield. In the third group of optimizations for the population of 80 chromosomes, the best fitness value achieved was 3,488 winnings (or 69.76%) for the pair that used the evaluation function with the parameters optimized by the genetic algorithm. The last group of optimizations of the evaluation function performed tests for a population of 100 chromosomes. The best result is 3,503 winnings in 5,000 matches, which corresponds to a 70.06% yield for the pair that used the evaluation function to choose the best move. This was also the best overall result obtained for Strategy 1 and the coefficients α_1, α_2, α_3, α_4, α_5, α_6 and α_7 of the evaluation function are shown in Table 5, in the end of the section.

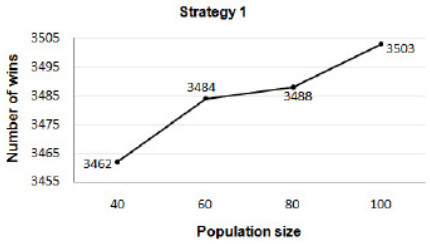


Fig. 6. Overall results of the optimization of the evaluation function of Strategy 1

B. *Optimization of the evaluation function of Strategy 2*

To optimize the coefficients of the evaluation function of Strategy 2, the values for a α_1, α_2, α_3 and α_7 were defined as 0. Once more, the optimization of the evaluation function was divided into groups according to the population size.

Table 2 and Figure 7 show the best results obtained in the optimization of the evaluation function of Strategy 2. We can see that this strategy, where the player focuses only on his own game, the results were worse than those in Strategy 1. The best overall result was 3,355 victories in 5,000 games, which represents 67.1% success for the pair that used the evaluation function to choose the best move. The coefficients α_4, α_5 and α_6 optimized in the evaluation function are shown in Table 5, at the end of this section.

TABLE 2
Optimization results of the evaluation function of Strategy 2

Population	Generations	Crossover Operator	p_m	Wins (pair 2)
40	150	Two-point	0.5	66.22 %
60	150	Scattered	0.5	67.10 %
80	100	Scattered	0.1	66.30 %
100	150	Two-point	0.1	66.46 %

The best results obtained by pair 2, which uses Strategy 2, displayed along with parameters used in the Genetic Algorithm.

C. *Optimization of the evaluation function of Strategy 3*

To optimize the coefficients of the evaluation function of Strategy 3, the values for α_1, α_2, α_3 and α_5 were defined as 0. As in the previous optimizations, the results were divided into groups according to the population size. Table 3 shows

the best results for each group of the optimization of the evaluation of Strategy 3 and the parameters used in the Genetic Algorithm.

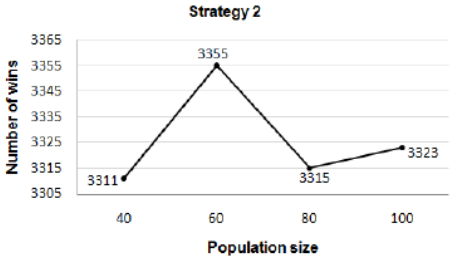


Fig. 7. Overall results of the optimization of the evaluation function of Strategy 2

TABLE 3
Optimization results of the evaluation function of Strategy 3

Population	Generations	Crossover Operator	p_m	Wins (pair 2)
40	50	Two-point	0.5	62.38 %
60	150	Scattered	0.1	63.08 %
80	150	Two-point	0.1	63.20 %
100	150	Scattered	0.5	62.90 %

The best results obtained by pair 2, which uses Strategy 3, displayed along with parameters used in the Genetic Algorithm.

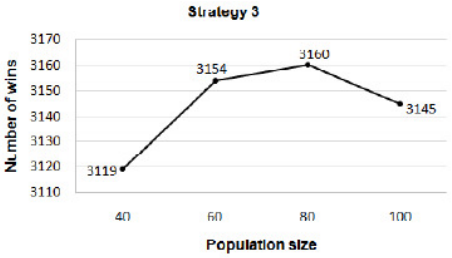


Fig. 7. Overall results of the optimization of the evaluation function of Strategy 3

D. *Optimization of the evaluation function of Strategy 4*

In Strategy 4, the player wants only to impede the moves of his opponents, so the values for the α_4, α_5, α_6 and α_7 were defined as 0. Table 4 and Figure 9 present the best results for each group of the optimization of the evaluation of Strategy 4 and the parameters of the genetic algorithm.

TABLE 4
Optimization results of the evaluation function of Strategy 4

Population	Generations	Crossover Operator	p_m	Wins (pair 2)
40	150	Two-point	0.1	60.98 %
60	200	Scattered	0.1	60.82 %
80	150	Scattered	0.1	61.04 %
100	150	Scattered	0.5	60.98 %

The best results obtained by pair 2, which uses Strategy 4, are displayed along with the parameters used in the Genetic Algorithm.

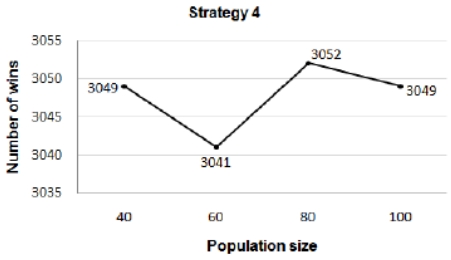


Fig. 8. Overall results of the optimization of the evaluation function of Strategy 4

Observing the above results, we note that Strategy 4 showed the worst results among all tested strategies, whereas the best result was the 61.04% of wins of pair 2, or 3,052 victories in 5,000 matches. The optimized coefficients α_1, α_2 and α_3 in the evaluation function are shown in Table 5, as well as the optimized coefficients for the other strategies.

All of the tested strategies showed significant results compared to the basic strategy, used by beginner players. However, the evaluation function proposed initially represented by Strategy 1, achieved the best results because it combines the three strategies in its equation.

The best result obtained in this study, 70.06% yield, was also higher than the results obtained in [2] and [9]. In [2], the same intelligent agent was used to choose the best move, but the definition of the coefficients α_i was done manually. His best result was 66% wins. In [9], where the research was based on 2-sided dominoes, an intelligent agent was also developed to choose the moves. The agents conducted the decision of the move by inferences and assumptions based on previous moves of other players. However, the stored data was only related to the numbers the players played. Seven different strategies were implemented, including a traditional strategy based on the strategy adopted by real players. However, to validate the intelligent agent, only one of the two players varied his strategy in each simulation; the others played with the traditional strategy. The best result obtained in this study was 54% wins and is not considered a significant result because it was within the tolerance limits set by the author due to the random characteristics of the game.

TABLE 5
Optimized coefficients for Strategies 1, 2, 3 and 4

Est.	α_1	α_2	α_3	α_4	α_5	α_6	α_7
1	-6.63	-3.828	-3.858	-4.694	0.543	6.052	2.419
2	0	0	0	-9.417	2.262	7.065	0
3	0	0	0	1.18	0	7.265	5.811
4	-3.733	-3.233	-4.912	0	0	0	0

VI. Conclusions

This work presented the results of the coefficients optimization process of the evaluation function to choose the best move in 4-sided dominoes. To adjust the coefficients, we used the technique of searching and optimization of Genetic Algorithms applied to find the best combination that allowed the largest possible number of wins. Four different strategies were evaluated and to obtain the best combination of coefficients of the evaluation functions, the genetic algorithm was run several times with different settings. The optimization results of the evaluation function were superior to the results obtained in previous work, where 66% was obtained. Among the tested strategies, Strategy 1, which is a combination of the other three strategies, had the best results. Given these results, we conclude that the performance of the genetic algorithm was excellent, being superior to the others. Analyzing and comparing our results with results from other studies, ours proved to be more effective in the choice of the best moves in 4-sided dominoes. It is believed that the methodology applied to choose the best move can also be applied to other games where players have imperfect information.

References

[1] K. P. Sycara, "Multiagent systems". AI Magazine, 1998, v. 19, n. 2, pp. 79–92.

[2] N. S. Antonio, C. F. F. Costa Filho and M. G. F. Costa, "Proposta de uma heurística para o jogo de domino de 4 pontas," in *Proc. VII Brazilian Symposium on Computer Games and Digital Entertainment – Computing Track*, Belo Horizonte, 2008, pp. 24–30.

[3] C. E. Shannon, "Programming a computer for playing chess". Philosophical Magazine, 1950, 41(4), pp. 256–275.

[4] M. S. Campbell, A. J. Hoane, F. H. Hus, "Deep Blue," Artificial Intelligence, 2002, 134(1-2), pp. 57–83.

[5] J. Shaeffer, *One Jump Ahead: Challenging Human Supremacy in Checkers*. Berlin: Springer-Verlag, 1997.

[6] S. J. J. Smith, D. S. Nau, T. A. Throop, "Success in spades: Using AI planning techniques to win the world championship of computer bridge," *in Proceedings of the Fifteenth National Conference on Artificial Intelligence*, Madison, Wisconsin, AAAI Press, 1998, pp. 1079–1086.

[7] B. S. Chlebus, "Domino-tiling games", *Journal of Computer and System Sciences*, v. 32, n. 3, 1986, pp. 374–392.

[8] H. C. Yen, "A multiparameter analysis of domino tiling with an application to concurrent systems". Theoretical Computer Science, 2002, v. 98, n. 2, pp. 263–287.

[9] A. G. de S. Garza, "Evaluating Individual Player Strategies in a Collaborative Incomplete-Information Agent-Based Game Playing Environment," in *IEEE Symposium on Computational Intelligence and Games*, 2006, pp. 211–216.

[10] D. Ashlock, *Evolutionary Computation for Modeling and Optimization*, Springer, 2006.