UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA – PATOS DE MINAS
ENGENHARIA ELETRÔNICA E DE TELECOMUNICAÇÕES

DÉBORA AMORIM FERREIRA SCHMIELE

ALGORITMOS GENÉTICOS APLICADOS À ANÁLISE ESTRATÉGICA DE JOGOS DE TABULEIRO

Patos de Minas - MG
2023

DÉBORA AMORIM FERREIRA SCHMIELE

ALGORITMOS GENÉTICOS APLICADOS À ANÁLISE ESTRATÉGICA DE JOGOS DE TABULEIRO

Trabalho apresentado à banca examinadora como requisito parcial para obtenção do título de bacharel em Engenharia Eletrônica e de Telecomunicações da Universidade Federal de Uberlândia - Campus Patos de Minas.

Orientador: Prof. Dr. Laurence Rodrigues do Amaral

Patos de Minas - MG
2023

DÉBORA AMORIM FERREIRA SCHMIELE

ALGORITMOS GENÉTICOS APLICADOS À ANÁLISE ESTRATÉGICA DE JOGOS DE TABULEIRO

Trabalho apresentado à banca examinadora como requisito parcial para obtenção do título de bacharel em Engenharia Eletrônica e de Telecomunicações da Universidade Federal de Uberlândia - Campus Patos de Minas.

Orientador: Prof. Dr. Laurence Rodrigues do Amaral

Patos de Minas, 7 de Dezembro de 2023

BANCA EXAMINADORA:

Prof. Dr. Laurence Rodrigues do Amaral
Universidade Federal de Uberlândia
Orientador

Prof. Dr. Pedro Luiz Lima Bertarini
Universidade Federal de Uberlândia
Examinador

MSc. Daniel de Oliveira Ferreira
Universidade Federal de Uberlândia
Examinador

Dedico este trabalho em especial à minha mãe, Dôra Amorim, minha irmã, Fabiana Amorim e ao meu marido, Eric F. Schmiele.

AGRADECIMENTOS

Agradeço ao meu Deus, por seu carinho e cuidado comigo. Muitos foram os que abandonaram suas crenças por não conseguirem conciliar fé e razão, e diferente destes, encerro este ciclo crendo ainda mais no poder de Deus e reconhecendo que Ele é a fonte de toda a sabedoria.

À minha mãe, Dôra Amorim, agradeço por ter me incentivado a ser uma criança curiosa e ter me permitido colecionar um amontoado de equipamentos eletrônicos ao invés de bonecas, a despeito do que qualquer outra mãe pudesse dizer.

À minha irmã, Fabiana Amorim, agradeço pelo incentivo no decorrer destes longos anos. Seu exemplo de luta e dedicação me incentivaram a continuar.

Ao meu marido, Eric Ferreira Schmiele, eu agradeço por ser meu parceiro nos altos e baixos nesta e em outras jornadas, aliviando meus fardos e comemorando cada vitória. Seu amor pelo conhecimento me inspira em uma busca pelo desconhecido.

Agradeço aos amigos que estiveram comigo, Mateus Silvério e Priscila Borges. A amizade de vocês, adquirida na faculdade e fortalecida em tantos outros momentos é com toda certeza um dos maiores presentes que eu poderia receber.

Ao meu orientador, Prof. Dr. Laurence Rodrigues do Amaral por sua dedicação e paciência, me auxiliando a concluir esta etapa tão importante.

Ao corpo docente, que contribuiu imensamente em minha formação acadêmica, agradeço pelo carinho dentro e fora das salas de aula.

E por fim agradeço à Universidade Federal de Uberlândia pelas experiências que vivi e pelas pessoas incríveis que de outra forma não teria conhecido.

RESUMO

Algoritmos Genéticos (AGs) são uma técnica de obtenção de soluções aproximadas, que envolve conceitos de evolução como: hereditariedade, mutação, seleção natural e recombinação. Algoritmos genéticos têm sido utilizados em jogos de tabuleiro como "Xadrez" e "Damas" para encontrar melhores estratégias, incluindo a análise do comportamento do adversário. Jogos de tabuleiro são, além de passatempos, uma forma de desenvolvimento lógico-matemático. Um dos jogos mais conhecidos é Monopoly, jogado em todo mundo, por pessoas de diferentes idades e perfis. Esse é um jogo baseado em turnos, em que a estratégia é um fator determinante, embora o estado de um jogador em curto prazo seja altamente influenciado pelo lançamento de dados. Esse trabalho visa a análise de agentes inteligentes (utilizando AGs) para identificar a melhor estratégia de jogo. Um simulador com regras simplificadas de Monopoly foi desenvolvido para ambientar a ação dos agentes inteligentes. A implementação exigiu algumas restrições, dado que certos mecanismos do jogo exigem maior poder de processamento, pois o jogo passa a ter mais estados aumentando a complexidade da análise.

Palavras-chave: Jogos de tabuleiro. Algoritmos Genéticos. Inteligência Artificial. Modelagem de jogadores. Estratégia. Monopoly. Banco imobiliário. Jogos multiplayers.

ABSTRACT

Genetic Algorithms (GAs) are a technique to obtain approximate solutions that consist of evolution concepts such as: heredity, mutation, natural selection and recombination. Genetic Algorithms have been used in board games such as “Chess” and “Checkers” to find better strategies, including the analysis of the adversary’s behavior. Board games are, besides hobbies, a form of logical and mathematical development. One of the most known board games is Monopoly, played all over the world by people of different ages and profiles. This is mainly a turn-based strategy game, although the state of a player in the short term is highly influenced by dice rolling. This work intends to analyze intelligent agents (using GAs) to identify the best game strategy. A Monopoly simulator with simplified rules was developed to set the actions of the intelligent agents. The implementation demanded some restrictions given that certain game mechanisms demand a higher processing, because the game starts having more states, enhancing its analysis complexity.

Keywords: Board games. Genetic Algorithms. Artificial Intelligence. Gamer modeling. Strategy. Monopoly. Multiplayer games.

LISTA DE FIGURAS

LISTA DE EQUAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AG - Algoritmo Genético

AE - Algoritmo Evolutivo

DS - Deterministic Sampling

FF - Função *Fitness*

IA - Inteligência Artificial

Ng - Número de gerações

Tc - Taxa de *crossover*

Tm - Taxa de mutação

Tp - Tamanho da população

NPC – Non-player character

SUMÁRIO

CAPÍTULO I

1.1. INTRODUÇÃO

Jogos de tabuleiro, em sua maioria, exigem que o jogador tenha uma estratégia superior a de seus adversários para atingir as metas e/ou pontuações necessárias para a vitória. Contudo, nos jogos, assim como na Economia, os resultados obtidos não dependem de um único indivíduo, mas dependem também das decisões de outros indivíduos que interagem com o primeiro. É neste contexto que a Teoria dos Jogos se destaca como uma área de interesse da Computação e Matemática.

Muitos estudos de estratégia dos mais variados jogos têm sido desenvolvidos[1, 2, 3, 4]. Nos últimos anos, com o aumento do poder computacional, e as ferramentas de Inteligência Artificial (IA), tornou-se possível testar jogos mais rapidamente, encontrando caminhos indesejados, ou até mesmo desenvolvendo *Non-player characters* (NPCs) com decisões mais realistas em jogos digitais.

Monopoly é um clássico jogo de compra e venda de propriedades e está entre os jogos de tabuleiro mais jogados em todo o mundo. Trata-se de um jogo de estratégia e uma boa dose de sorte. Análises estratégicas do jogo original são bem comuns de se encontrar [5, 6]. Muitas dessas estratégias buscam traduzir o jogo em um modelo matemático para legitimar (ou não) estratégias adotadas pelos jogadores, optando por versões simplificadas.

Em princípio, para a simulação de jogos, alguns perfis de jogadores foram pré-programados, com base em características comuns de jogadores (impulsividade, análise de valor de uma propriedade, valor do aluguel, dentre outras). Esses perfis possibilitaram a realização de testes do simulador e a decisão de quais mecanismos do jogo seriam mais relevantes, com o objetivo de simplificar o modelo matemático.

A otimização por Algoritmos Genéticos (AGs) permite encontrar valores para parâmetros utilizados na tomada de decisão em comprar ou não uma propriedade e então entender quais deles são mais relevantes: valor da propriedade, formar um conjunto completo, valor de aluguel, manter saldo suficiente para pagar aluguéis ao percorrer o tabuleiro, dentre outros.

É importante considerar que jogos são ambientes de testes ricos para pesquisas na área

de IA, dadas algumas características em comum com o mundo real: as ações controladas ou não pelo jogador modificam o ambiente. Além disso, o ambiente responde às ações tomadas pelos jogadores.

Tendo em vista a complexidade destes ambientes e sua consequente semelhança com o mundo real, jogos são classificados como simuladores para testes de algoritmos de gerenciamento de recursos, definição de rotas, colaboração e planejamento. Contudo, nem sempre são utilizados porque os algoritmos usados para implementar técnicas de IA tem um custo computacional que por vezes os tornam inviáveis.

1.2. TEMA DO PROJETO

Utilização de AG para análise de estratégias em jogos de tabuleiro.

1.3. PROBLEMATIZAÇÃO

Testar as possíveis estratégias e/ou comportamentos de jogadores é de grande interesse para o desenvolvimento de jogos pois possibilita a análise de engajamento, dificuldade, estudo de possíveis melhorias, mudanças nos mecanismos de jogo, entre outros. Porém, tais testes demandam muito tempo se feitos pessoalmente com um grupo de jogadores de teste. A aplicação de IA diminui consideravelmente esse tempo e abre novas possibilidades que talvez não seriam contempladas por um grupo de jogadores de teste.

1.4. OBJETIVOS

1.4.1. Objetivos Gerais

Desenvolver um estudo a fim de simular um jogo de tabuleiro, onde se possa, por meio de AGs, ter agentes inteligentes, capazes de tomar decisões sobre eventos do jogo e analisar quais estratégias possuem melhor desempenho.

1.4.2. Objetivos Específicos

Visando atender ao objetivo geral proposto é necessário cumprir os seguintes objetivos específicos:

- Implementar a geração de agentes inteligentes por meio de AGs;
- Implementar versão simplificada do jogo;
- Implementar a interação entre agentes inteligentes e o jogo;
- Utilizar algoritmos de otimização;
- Analisar as estratégias de melhor desempenho.

1.5. JUSTIFICATIVAS

Nos últimos anos houve grandes avanços na área da Computação que permitiram então que técnicas de IA fossem desenvolvidas e implementadas. Diante disso, tarefas repetitivas puderam ser automatizadas e seu tempo de execução tornou-se incrivelmente menor.

O setor de jogos deve estar em constante atualização devido à exigência dos jogadores por jogos cada vez mais dinâmicos e realistas. Para tanto, os testes de mecanismos, comportamento de jogadores, caminhos, dentre outros, são de grande importância. A produção de jogos é hoje vista como área de interesse para pesquisadores de IA, dados os desafios no seu desenvolvimento, visando proporcionar ambientes que se assemelham às situações do cotidiano quanto à tomada de decisões.

Diante das possibilidades oferecidas no desenvolvimento de jogos aliado a técnicas de IA, esse trabalho objetiva encontrar estratégia(s) otimizada(s) para o jogo Monopoly, por meio de sua simulação computacional, utilizando-se de AGs.

1.6. CONSIDERAÇÕES FINAIS

Esse trabalho de conclusão de curso tem como objetivo estudar a aplicação de IA no jogo de tabuleiro Monopoly para a análise de comportamento de jogadores. Também será abordado nesse projeto o estudo teórico de IA e algoritmos de otimização. A organização do trabalho é proposta da seguinte maneira:

- O primeiro capítulo traz o tema proposto com uma pequena introdução sobre a aplicação de IA e algoritmos de otimização e os motivos que levaram ao estudo.

- O segundo capítulo apresenta o jogo Monopoly e as mecânicas do jogo que serão implementadas, bem como o conceito de AGs.
- O terceiro capítulo irá apresentar a metodologia utilizada, além dos recursos que serão utilizados na aplicação de IA ao jogo em questão.
- O quarto capítulo é destinado aos resultados obtidos das simulações e discussões a respeito desses resultados;
- O quinto capítulo sintetiza as conclusões do trabalho e traz propostas de trabalhos futuros.

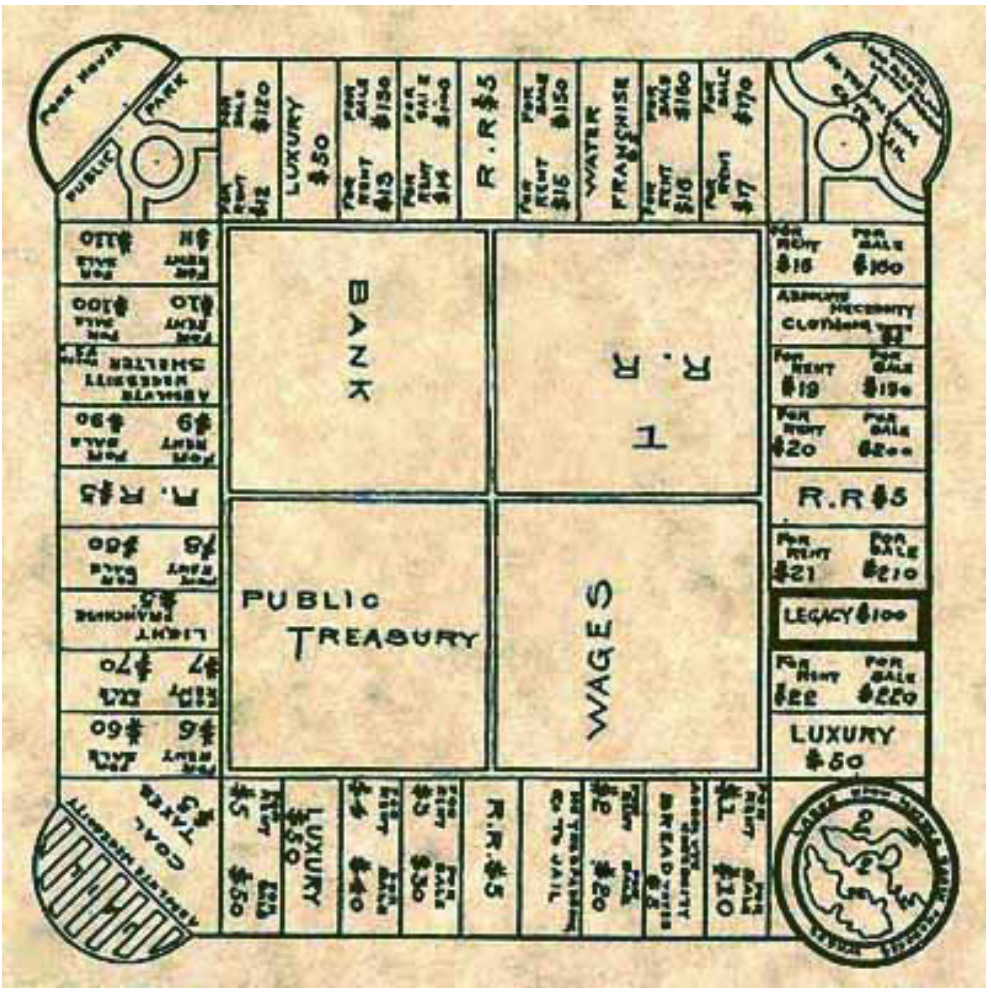
CAPÍTULO II

2.1. MONOPOLY

2.1.1. Histórico

Monopoly foi publicado em 1935, nos Estados Unidos por Charles Darrow. O jogo foi baseado em uma versão prévia, *The Landlord's Game*, de Elizabeth J. Magie Phillips, cujo tabuleiro é apresentado na figura 2.1. Elizabeth desejava que o jogo fosse uma forma de ensinar sobre as teorias econômicas de Henry George, economista político, criador da teoria do imposto único [7,8].

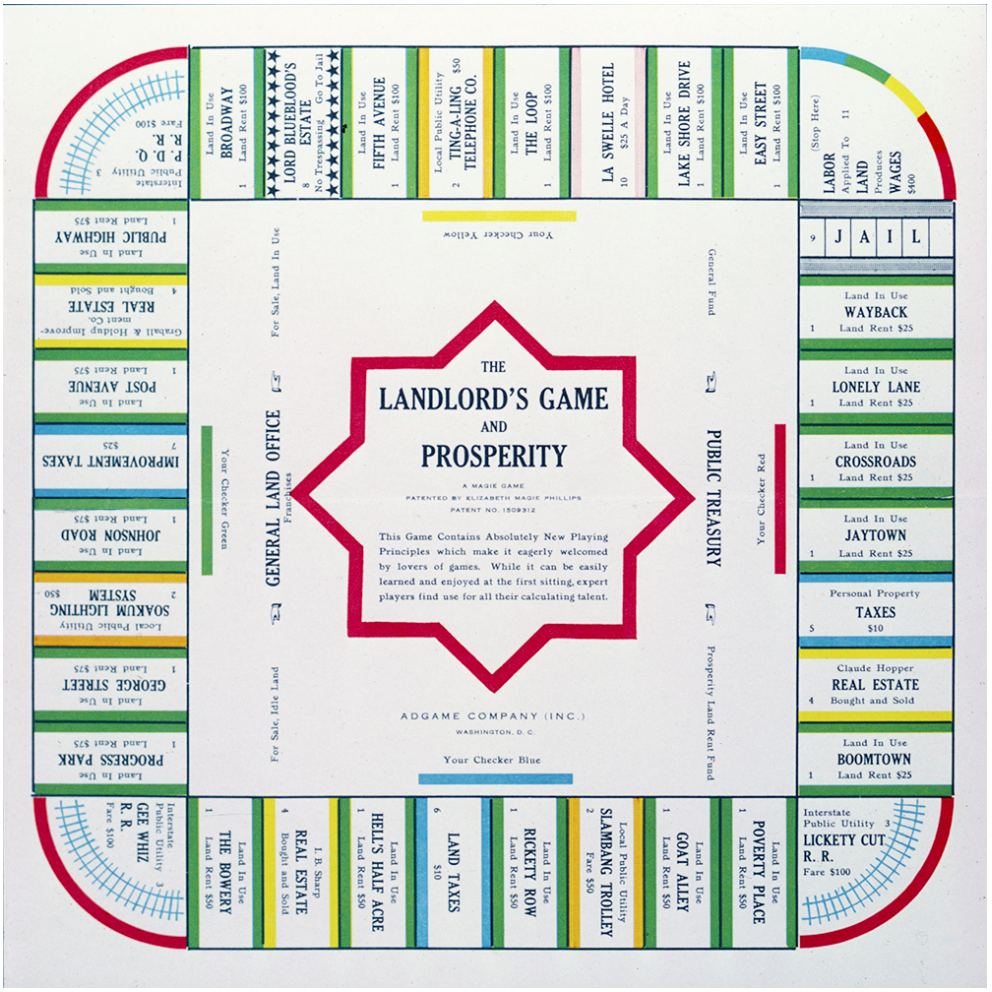
Figura 2.1 - The Landlord's Game



Fonte: [10]

A primeira versão comercial de *The Landlord's Game* foi publicada pela Economic Game Company of New York, fundada por Elizabeth e outros seguidores de Henry George, em 1906. Em 1932, uma nova versão foi publicada (Figura 2.2) incluindo nomes de ruas e um novo conjunto de regras. Além disso, um segundo nome foi adicionado: *Prosperity.*

Figura 2.2 - The Landlord's Game And Prosperity



Fonte: [10]

O jogo era conhecido em uma comunidade de defensores das ideias de Henry George, que começaram a alterar as regras. A principal delas foi a de, ao invés de apenas pagarem aluguel ao pararem em uma propriedade, eles também podiam realizar um leilão para comprá-la. Logo novos tabuleiros começaram a ser produzidos pelos próprios jogadores e as propriedades eram substituídas pelas de suas regiões. Em algum momento o jogo deixou de

ser chamado *"The Landlord's Game"*, passando a se chamar *"Auction Monopoly"* e depois apenas *"Monopoly"* [7].

2.1.2. *Game Play*

Há versões regionais do jogo, como por exemplo a versão comercializada pela Estrela no Brasil sob o nome de Banco Imobiliário. As regras podem variar um pouco entre versões regionais e até mesmo entre edições comemorativas lançadas pela Hasbro, atual detentora dos direitos de venda do jogo, que adicionam novos mecanismos de acordo com a temática. Hoje, além das versões de tabuleiro (Figura 2.3), o jogo também conta com versões mobile, Facebook e Chrome.

Figura 2.3 - Tabuleiro de Monopoly

Fonte: [11]

2.2. INTELIGÊNCIA ARTIFICIAL

A IA é uma área de estudos que visa aplicar técnicas computacionais inspiradas em diferentes sistemas, entre eles, a natureza. O objetivo é gerar um modelo que seja capaz de aprender ou lidar com novas situações em um dado contexto no qual o sistema possui habilidades de reação, como generalizar, descobrir e abstrair. Isso é, desenvolver sistemas computacionais que ajam de forma semelhante a certos sistemas biológicos ao ponto de considerar que possuem inteligência no dado contexto.

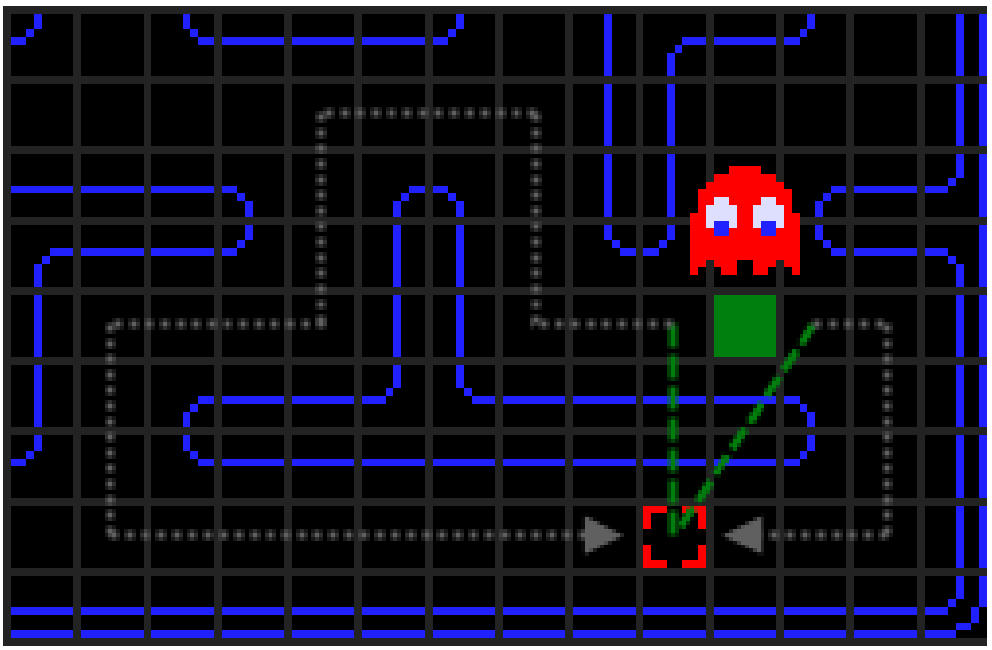
Tais sistemas são aplicados para solucionar problemas matemáticos e computacionais de uma forma análoga a comportamentos encontrados na natureza. Assim, surgiram várias técnicas que compõem a IA baseadas em computação natural, como os AGs que se baseiam na teoria da evolução natural de Charles Darwin [9]. As diferentes técnicas da IA têm sido aplicadas para desenvolver sistemas eficazes, de simples manipulação, robustos e capazes de apresentar soluções satisfatórias para problemas complexos. Essas técnicas são largamente utilizadas no desenvolvimento de sistemas inteligentes para tomada de decisões [10], reconhecimento de padrões [11], otimização [12], controle, etc., em diversas áreas do conhecimento.

2.2.1. Utilização em jogos

No jogo Pac-Man, desenvolvido por Toru Iwatani, temos quatro fantasmas com cores, nomes e personalidades diferentes. Cada fantasma seguia um padrão próprio de como perseguir o Pac-Man dentro do labirinto, ou até mesmo como fugir [13]. Foi necessário simplificar o comportamento de cada fantasma para fazê-lo parecer mais esperto e ainda manter a experiência fluida. Assim, poucas decisões eram tomadas por eles.

Em um corredor, os fantasmas apenas seguiam em frente. Porém, quando em um cruzamento, cada um se comportava de forma diferente. O fantasma vermelho (Figura 2.4) escolhia o caminho de menor distância até o Pac-Man e ganhou o nome de Perseguidor. O fantasma rosa seguia um pouco mais a frente, tornando-se o Emboscador. O azul era o fantasma Instável e hora ia mais para perto, hora ia mais pra longe. Já o laranja era o Perdido, e podia ir para qualquer lado.

Figura 2.4 - Fantasma Vermelho



Fonte: [13]

Mesmo que os fantasmas de Pac-Man apresentem comportamentos diferentes, não foram utilizadas técnicas de IA. Hoje, graças à crescente capacidade de processamento, as técnicas de IA vêm sendo aplicadas a jogos com diferentes finalidades. Inicialmente, foram utilizadas técnicas de IA como um desafio para ganhar de um campeão mundial de xadrez, o que foi alcançado em 1996 [14], e posteriormente com objetivos diversos como: gerar inimigos/desafios automaticamente e em tempo real, ter inimigos com diferentes níveis de habilidades, tomada de decisão semelhante à de um jogador humano, análise de possíveis estratégias para vencer um jogo, análise de engajamento e níveis de dificuldade de um jogo, entre tantos outros.

Consequentemente, nos anos 90, a IA se tornou um diferencial para a venda de jogos [15], proporcionando até mesmo a criação de novas categorias de jogos virtuais como os jogos de estratégia em tempo real (RTS, *Real Time Strategy*), entre tantas outras mecânicas e funcionalidades de jogos que hoje são utilizadas corriqueiramente. É seguro dizer que as técnicas de IA revolucionaram os jogos e continuarão o fazendo.

2.3. ALGORITMOS GENÉTICOS

A complexidade na resolução de um problema está diretamente relacionada ao seu espaço de busca e, em diversos contextos, algoritmos exatos acabam se tornando inviáveis pelo alto custo computacional que impõe. Diante disto, algoritmos de otimização se tornam uma opção.

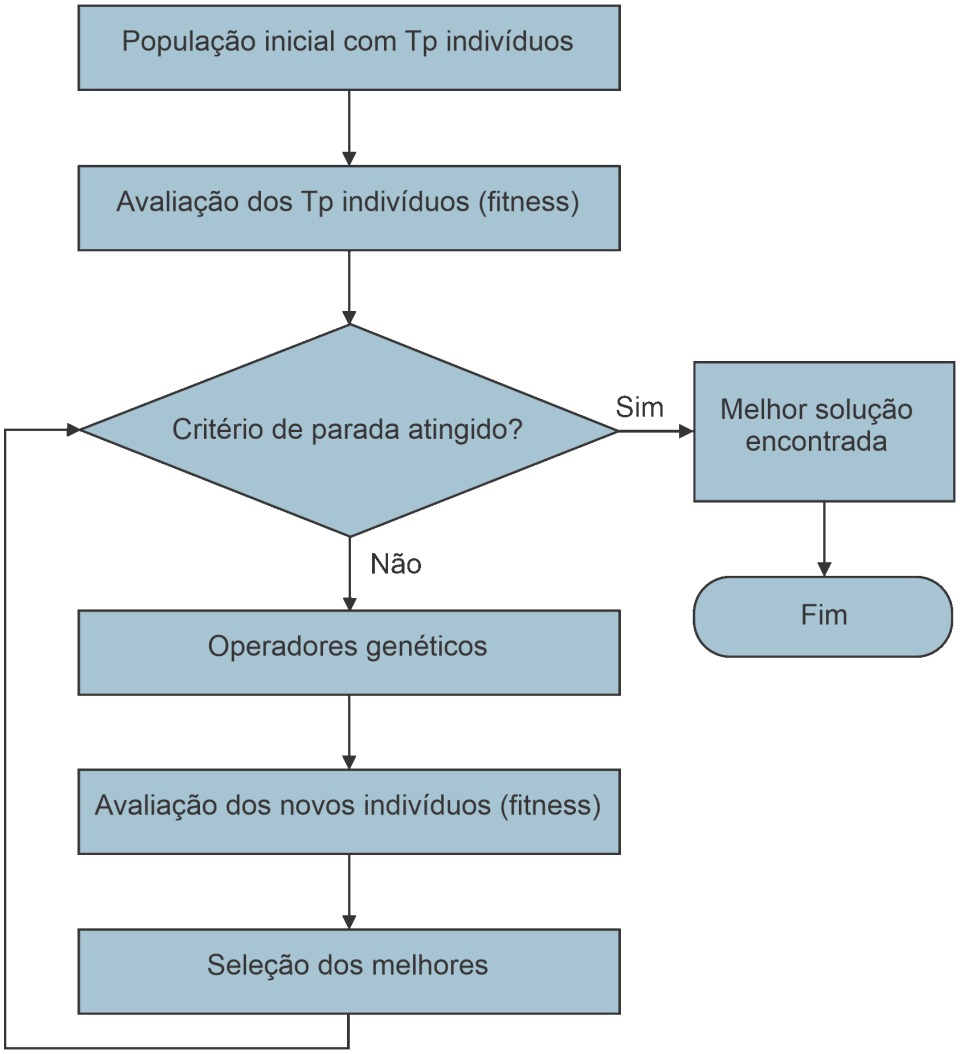
Algoritmos Genéticos, propostos por Holland [16], são algoritmos evolucionários, ou seja, se baseiam na teoria da evolução de Charles Darwin [9]. Esses algoritmos são métodos de busca e otimização guiados pela seleção natural. Utilizam probabilidades e possuem um mecanismo de busca paralela e adaptativa baseado na reprodução das espécies e no princípio de sobrevivência dos mais aptos. Os AGs buscam encontrar uma solução otimizada para um determinado problema manipulando uma população de soluções candidatas [17].

2.3.1. Funcionamento

Assim como outros Algoritmos Evolutivos (AEs), os AGs adotam estratégias guiadas pela teoria da evolução das espécies, buscando uma solução otimizada para um dado problema manipulando uma população de possíveis soluções. Nesse sentido, cada população é avaliada e as melhores são selecionadas com a finalidade de se reproduzirem para formar a próxima geração [18].

O indivíduo é uma codificação de uma possível solução para um dado problema. Essa solução é incorporada em uma estrutura semelhante à de um cromossomo em que são aplicados os operadores de seleção, cruzamento e mutação com o objetivo de manter informações relativas à solução do problema [18]. Os cromossomos são implementados como vetores, sendo que cada componente deste vetor é um gene. Os prováveis valores que um gene pode assumir são denominados alelos. O algoritmo básico de um AG pode ser representado por meio dos passos mostrados na Figura 2.5[19]:

Figura 2.5: Execução de um AG



Fonte: Autora

Considerando uma explicação mais abrangente, um AG deve possuir os seguintes componentes [18]:

a) Definição do problema a ser solucionado

Os AGs são aplicados para resolver problemas complexos de otimização, seja por ter vários parâmetros a serem combinados ou várias restrições difíceis de serem representadas matematicamente, ou por ter um espaço de possíveis soluções muito amplo. Em cada caso, a abordagem e o contexto podem variar.

b) Representação e codificação dos indivíduos

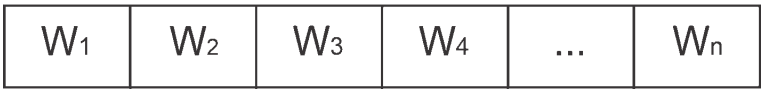
Como os problemas diferem, as representações das possíveis soluções também irão variar de acordo com a aplicação. Assim será definida a estrutura dos cromossomos a serem manipulados dependendo do tipo de informação pertinente a cada solução. Temos duas principais representações de informações no cromossomo: binária e vetor de pesos.

A representação binária (Figura 2.6) é simples e fácil de ser transformada em inteiro ou real. Ela facilita não só a prova de determinados teoremas como também a manipulação dos cromossomos por meio dos operadores genéticos. Já a representação por vetor de pesos (figura 2.7) tem um desempenho mais robusto quando o problema é numérico.

Figura 2.6: Representação Binária de um AG

Fonte: Autora

Figura 2.7: Representação de um AG por vetor de pesos



Fonte: Autora

A codificação dos indivíduos é de suma importância para o funcionamento do AG, visto que, se feita de maneira equivocada, pode convergir para uma solução não otimizada ou mesmo encontrar soluções não factíveis.

c) Definição da população inicial

A primeira geração de indivíduos pode ser gerada de forma aleatória diminuindo a influência de possíveis vieses. Duas maneiras de evitar este problema é a utilização de uma população grande o suficiente e executar o algoritmo mais de uma vez para verificar se a solução encontrada é a mesma, isto é, o mesmo indivíduo é eleito como o melhor ao final.

d) Avaliação dos indivíduos

Consiste em definir uma função que melhor represente o problema, para assim ser utilizada como métrica para verificar a aptidão de cada um dos indivíduos da população atual [20].

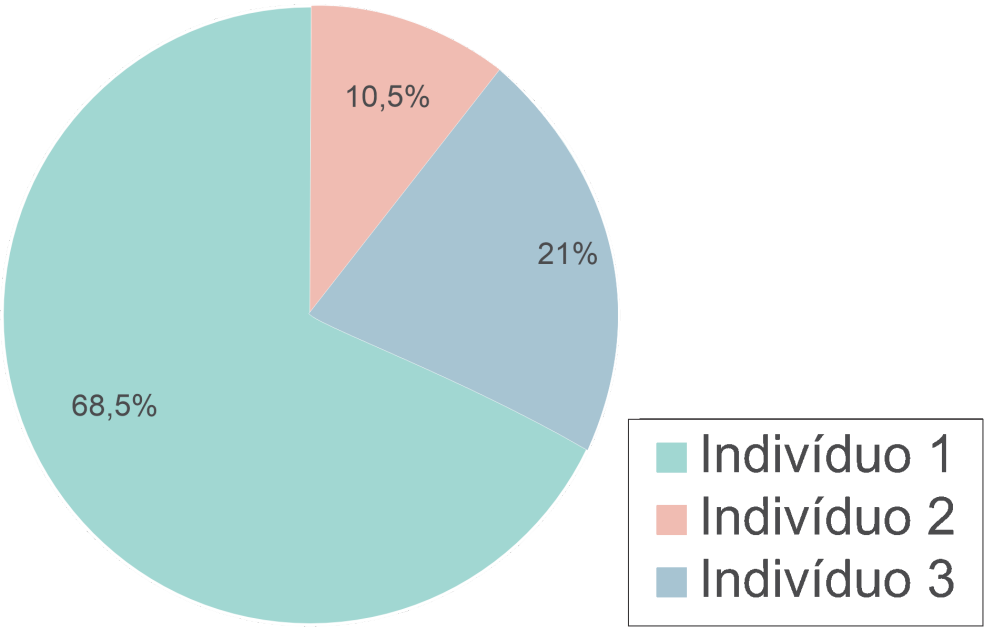
A partir desta função, Função *Fitness* (FF), os indivíduos são classificados de acordo com o seu resultado. A FF deve ser feita de maneira que penalize soluções indesejáveis, portanto deve ser implementada com cuidado.

e) Seleção para a próxima geração

Considerando a aptidão dos indivíduos, ou seja, o resultado de cada indivíduo frente à FF, alguns indivíduos são selecionados para a próxima geração, assim como na seleção natural acontece a sobrevivência dos mais aptos. Aqueles com melhor resultado têm maior probabilidade de serem selecionados.

Um dos métodos mais utilizados para a seleção é a roleta. Uma roleta virtual é criada na qual cada indivíduo recebe um pedaço proporcional à sua avaliação e essa é sua probabilidade de ser selecionado [21]. Para ilustrar o funcionamento da roleta, a Figura 2.8 representa a avaliação de uma população de 3 indivíduos.

Figura 2.8 - Roleta



Fonte: A autora

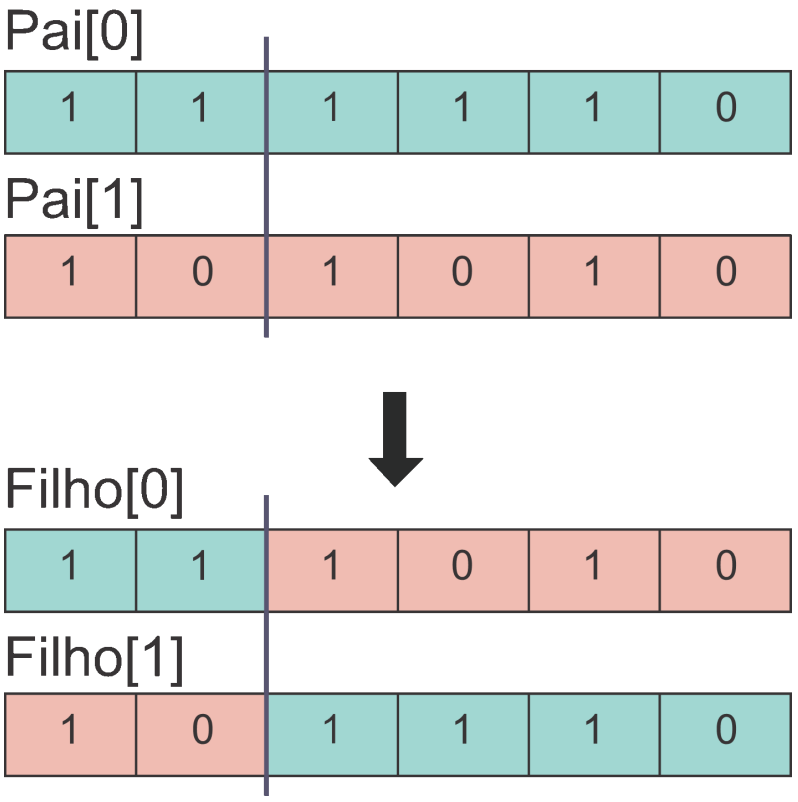
Existem também outras métricas para selecionar os melhores indivíduos como: torneio, estocástico uniforme, roleta uniforme, Deterministic Sampling (DS) entre outros.

f) Definição dos operadores genéticos

Os operadores genéticos mais utilizados são o *crossover* (cruzamento) e a mutação. Os indivíduos selecionados, conhecidos como pais, são recombinados segundo um operador de cruzamento. Os pais são separados em pares e de acordo um operador de seleção e passarão pelo processo de *crossover*. A partir deles, novos indivíduos são criados (filhos) trocando o material genético entre eles. Os filhos gerados são diferentes dos anteriores, porém possuem características genéticas de seus pais.

O *crossover* mais utilizado é o de cruzamento de um ponto [22], aplicado principalmente em representações binárias (Figura 2.9). Nele, os cromossomos dos pais são particionados em um ponto aleatório (ponto de corte). Dois novos indivíduos são gerados trocando-se a metade inicial de um cromossomo com a metade inicial do outro.

Figura 2.9 - *Crossover* de 1 ponto

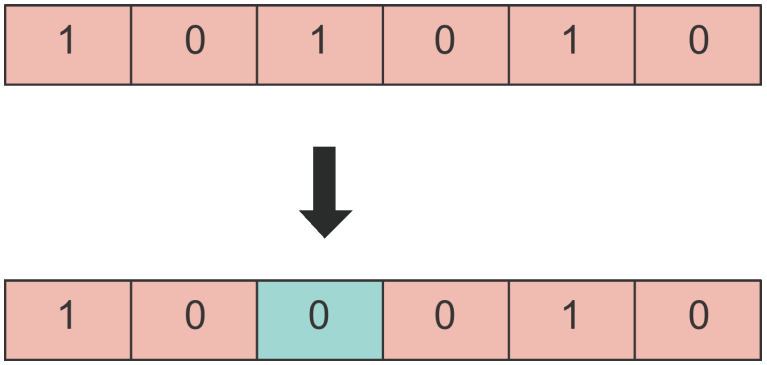


Fonte: A autora

Para evitar que indivíduos muito parecidos componham a nova geração após o *crossover*, o que levaria a soluções ótimas locais, utiliza-se do operador de mutação.

Alguns indivíduos da nova geração são selecionados, de acordo com uma taxa de mutação, e então um cromossomo ou gene sofre mutação. No exemplo da Figura 2.10, em um caso de representação binária, um dos seus valores é trocado de 1 para 0.

Figura 2.10 - Mutação



Fonte: A autora

g) Elitismo

Em alguns casos é interessante que um ou dois dos melhores indivíduos da geração anterior passem para a próxima sem *crossover* e mutação, a fim de garantir a preservação de suas informações.

h) Critério de Parada

Este critério determina quando a execução do AG será encerrada e pode ser definido de diferentes formas:

1. Avaliação de um indivíduo atinge o valor esperado;
2. O número de gerações predefinido é alcançado;
3. Não acontecem melhoras no desempenho da população a *n* iterações.

i) Parâmetros Genéticos

Os parâmetros genéticos vão influenciar nas características do AG em diferentes momentos: inicialização da população, crossover, mutação, tamanho da população e o número de gerações.

O tamanho da população (*Tp*) afeta de forma direta o desempenho do AG. Se o *Tp* é grande o suficiente para cobrir o espaço de busca do problema, evita-se convergências prematuras. Em contrapartida serão necessários maiores recursos computacionais.

Quando a taxa de *crossover* (*Tc*) é alta, maior será o número de pais escolhidos na atual geração para produzirem filhos para a próxima geração. A taxa de mutação (*Tm*) indica a probabilidade de que cada gene de um indivíduo seja alterado. A *Tm*, ao contrário da *Tc*, não deve ser muito alta, uma vez que isso tornaria a busca essencialmente aleatória ao invés de evolutiva, descartando as boas características presentes na atual geração.

Cada iteração do AG compreende uma geração. O número de gerações (*Ng*) deve ser determinado com cautela a fim de que as soluções encontradas atendam aos requisitos desejados, sem penalizar a evolução das soluções ou resultar em uma execução desnecessariamente longa.

2.4. CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os conceitos teóricos utilizados no desenvolvimento deste trabalho. Foram apresentadas as regras para a variação de Monopoly utilizada e os conceitos de AG. Todos esses conceitos formam a base para compreensão dos capítulos seguintes.

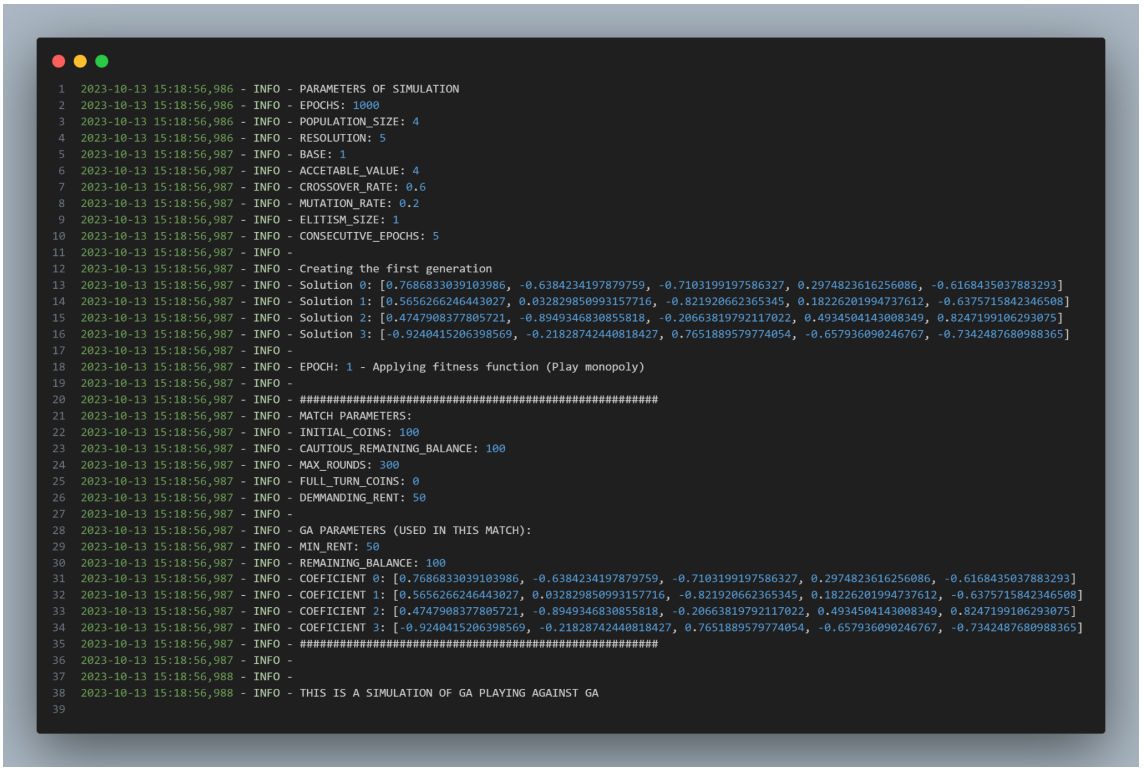
CAPÍTULO III

3.1. MATERIAIS E MÉTODOS

Neste trabalho foi utilizado uma variação do jogo “Monopoly” para fazer uma análise de possíveis estratégias de jogadores utilizando AG, a fim de encontrar uma que seja otimizada. Todo o sistema foi implementado utilizando a linguagem de programação em Python [23]. Todas as funções da implementação do AG foram desenvolvidas, sem fazer uso de nenhuma biblioteca do tipo.

Para facilitar o entendimento e verificação das execuções foi desenvolvido um sistema de logs, como mostrado na Figura 3.1. Nele, há a geração de um arquivo a cada execução, onde se pode verificar os parâmetros da execução, o desempenho de cada indivíduo ao ser submetido à FF e também como o jogo se desenvolveu, indicando a rodada, posição do jogador e as ações do mesmo no seu turno, como exemplificado na Figura 3.2.

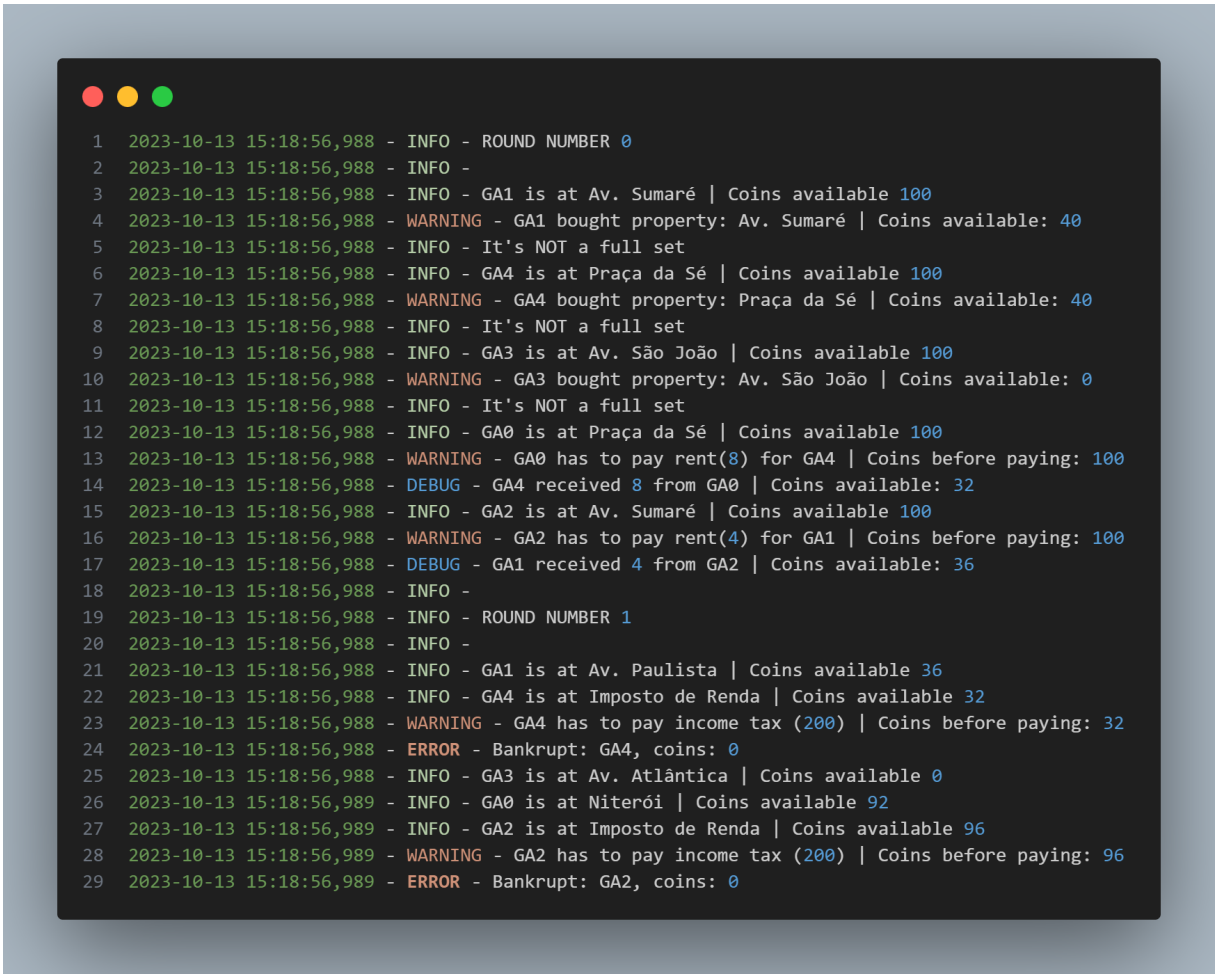
Figura 3.1 - Logs de parâmetros da simulação



```
2023-10-13 15:18:56,986 - INFO - PARAMETERS OF SIMULATION
2023-10-13 15:18:56,986 - INFO - EPOCHS: 1000
2023-10-13 15:18:56,986 - INFO - POPULATION_SIZE: 4
2023-10-13 15:18:56,986 - INFO - RESOLUTION: 5
2023-10-13 15:18:56,987 - INFO - BASE: 1
2023-10-13 15:18:56,987 - INFO - ACCETABLE_VALUE: 4
2023-10-13 15:18:56,987 - INFO - CROSSOVER_RATE: 0.6
2023-10-13 15:18:56,987 - INFO - MUTATION_RATE: 0.2
2023-10-13 15:18:56,987 - INFO - ELITISM_SIZE: 1
2023-10-13 15:18:56,987 - INFO - CONSECUTIVE_EPOCHS: 5
2023-10-13 15:18:56,987 - INFO -
2023-10-13 15:18:56,987 - INFO - Creating the first generation
2023-10-13 15:18:56,987 - INFO - Solution 0: [0.7686833039103986, -0.6384234197879759, -0.7103199197586327, 0.2974823616256086, -0.6168435037883293]
2023-10-13 15:18:56,987 - INFO - Solution 1: [0.5656266246443027, 0.032829850993157716, -0.821920662365345, 0.18226201994737612, -0.6375715842346508]
2023-10-13 15:18:56,987 - INFO - Solution 2: [0.4747908377805721, -0.8949346830855818, -0.20663819792117022, 0.4934504143008349, 0.8247199106293075]
2023-10-13 15:18:56,987 - INFO - Solution 3: [-0.9240415206398569, -0.21828742440818427, 0.7651889579774054, -0.657936090246767, -0.7342487680988365]
2023-10-13 15:18:56,987 - INFO -
2023-10-13 15:18:56,987 - INFO - EPOCH: 1 - Applying fitness function (Play monopoly)
2023-10-13 15:18:56,987 - INFO -
2023-10-13 15:18:56,987 - INFO - ##################################################
2023-10-13 15:18:56,987 - INFO - MATCH PARAMETERS:
2023-10-13 15:18:56,987 - INFO - INITIAL_COINS: 100
2023-10-13 15:18:56,987 - INFO - CAUTIOUS_REMAINING_BALANCE: 100
2023-10-13 15:18:56,987 - INFO - MAX_ROUNDS: 300
2023-10-13 15:18:56,987 - INFO - FULL_TURN_COINS: 0
2023-10-13 15:18:56,987 - INFO - DEMMANDING_RENT: 50
2023-10-13 15:18:56,987 - INFO -
2023-10-13 15:18:56,987 - INFO - GA PARAMETERS (USED IN THIS MATCH):
2023-10-13 15:18:56,987 - INFO - MIN_RENT: 50
2023-10-13 15:18:56,987 - INFO - REMAINING_BALANCE: 100
2023-10-13 15:18:56,987 - INFO - COEFICIENT 0: [0.7686833039103986, -0.6384234197879759, -0.7103199197586327, 0.2974823616256086, -0.6168435037883293]
2023-10-13 15:18:56,987 - INFO - COEFICIENT 1: [0.5656266246443027, 0.032829850993157716, -0.821920662365345, 0.18226201994737612, -0.6375715842346508]
2023-10-13 15:18:56,987 - INFO - COEFICIENT 2: [0.4747908377805721, -0.8949346830855818, -0.20663819792117022, 0.4934504143008349, 0.8247199106293075]
2023-10-13 15:18:56,987 - INFO - COEFICIENT 3: [-0.9240415206398569, -0.21828742440818427, 0.7651889579774054, -0.657936090246767, -0.7342487680988365]
2023-10-13 15:18:56,987 - INFO - ##################################################
2023-10-13 15:18:56,987 - INFO -
2023-10-13 15:18:56,988 - INFO -
2023-10-13 15:18:56,988 - INFO - THIS IS A SIMULATION OF GA PLAYING AGAINST GA
```

Fonte: Autora

Figura 3.2 - Logs de execução



```
2023-10-13 15:18:56,988 - INFO - ROUND NUMBER 0
2023-10-13 15:18:56,988 - INFO -
2023-10-13 15:18:56,988 - INFO - GA1 is at Av. Sumaré | Coins available 100
2023-10-13 15:18:56,988 - WARNING - GA1 bought property: Av. Sumaré | Coins available: 40
2023-10-13 15:18:56,988 - INFO - It's NOT a full set
2023-10-13 15:18:56,988 - INFO - GA4 is at Praça da Sé | Coins available 100
2023-10-13 15:18:56,988 - WARNING - GA4 bought property: Praça da Sé | Coins available: 40
2023-10-13 15:18:56,988 - INFO - It's NOT a full set
2023-10-13 15:18:56,988 - INFO - GA3 is at Av. São João | Coins available 100
2023-10-13 15:18:56,988 - WARNING - GA3 bought property: Av. São João | Coins available: 0
2023-10-13 15:18:56,988 - INFO - It's NOT a full set
2023-10-13 15:18:56,988 - INFO - GA0 is at Praça da Sé | Coins available 100
2023-10-13 15:18:56,988 - WARNING - GA0 has to pay rent(8) for GA4 | Coins before paying: 100
2023-10-13 15:18:56,988 - DEBUG - GA4 received 8 from GA0 | Coins available: 32
2023-10-13 15:18:56,988 - INFO - GA2 is at Av. Sumaré | Coins available 100
2023-10-13 15:18:56,988 - WARNING - GA2 has to pay rent(4) for GA1 | Coins before paying: 100
2023-10-13 15:18:56,988 - DEBUG - GA1 received 4 from GA2 | Coins available: 36
2023-10-13 15:18:56,988 - INFO -
2023-10-13 15:18:56,988 - INFO - ROUND NUMBER 1
2023-10-13 15:18:56,988 - INFO -
2023-10-13 15:18:56,988 - INFO - GA1 is at Av. Paulista | Coins available 36
2023-10-13 15:18:56,988 - INFO - GA4 is at Imposto de Renda | Coins available 32
2023-10-13 15:18:56,988 - WARNING - GA4 has to pay income tax (200) | Coins before paying: 32
2023-10-13 15:18:56,988 - ERROR - Bankrupt: GA4, coins: 0
2023-10-13 15:18:56,988 - INFO - GA3 is at Av. Atlântica | Coins available 0
2023-10-13 15:18:56,989 - INFO - GA0 is at Niterói | Coins available 92
2023-10-13 15:18:56,989 - INFO - GA2 is at Imposto de Renda | Coins available 96
2023-10-13 15:18:56,989 - WARNING - GA2 has to pay income tax (200) | Coins before paying: 96
2023-10-13 15:18:56,989 - ERROR - Bankrupt: GA2, coins: 0
```

Fonte: Autora

3.1.1. Jogo

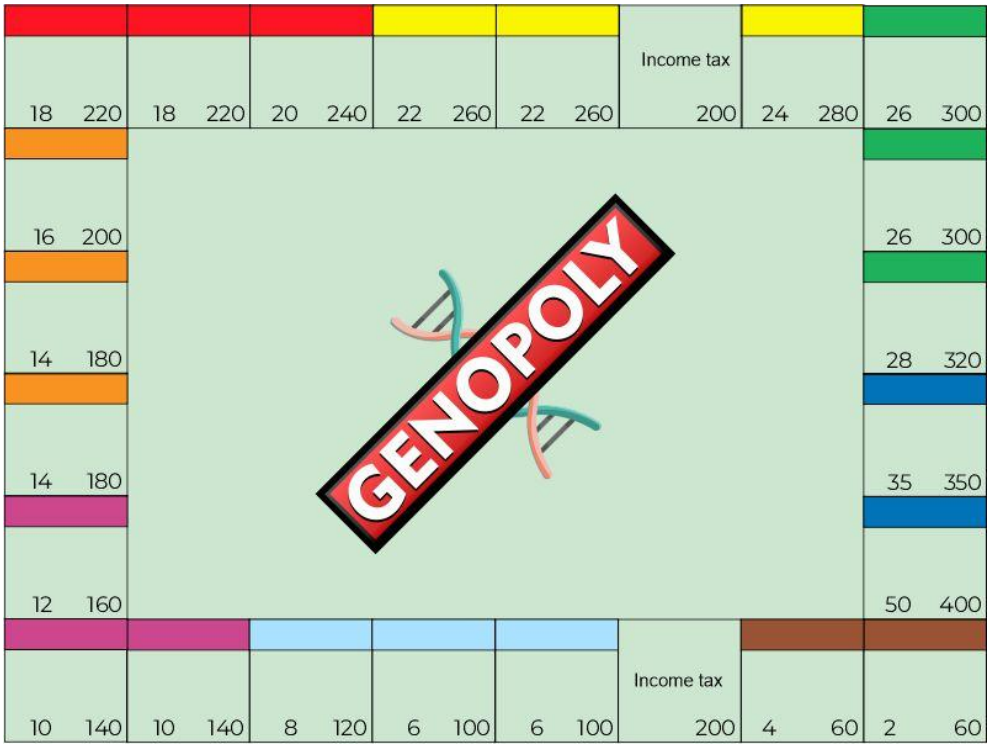
Os principais mecanismos do jogo serão descritos a seguir. Alguns mecanismos do jogo original foram simplificados para possibilitar as simulações para testes com agentes inteligentes.

a. Tabuleiro: Há no tabuleiro 24 espaços, sendo desses, 22 propriedades e 2 espaços onde acontece o pagamento obrigatório de taxas. Esses 24 espaços compõe a trilha percorrida pelo jogador;
b. Propriedades: Cada propriedade é identificada por um nome e possui um valor de aluguel, valor de venda e cor. Além disso, a propriedade pode ou não ser parte de um conjunto completo. As propriedades podem ser compradas pelo jogador que esteja na mesma posição do tabuleiro, caso ainda não tenha um proprietário;

c. Objetivo: Tornar-se o único jogador com recursos;
d. Dinheiro: Os jogadores começam a partida com o mesmo saldo. Ao longo do jogo esse valor será utilizado por ele para comprar propriedades e realizar pagamentos;
e. Início da partida: A ordem dos jogadores é definida pelo lançamento de um dado equiprovável de 6 faces com todos iniciam fora do tabuleiro. No seu turno, o jogador lança o dado para determinar a quantidade de espaços a ser percorrida;
f. Ações do jogador: Ao parar em uma propriedade o jogador pode decidir comprá-la, caso ela ainda não tenha um proprietário. Se a comprar, o valor é abatido de seu saldo. Se o jogador optar por não comprar a propriedade, nada acontece. Seu turno se encerra. Caso a propriedade tenha dono, o jogador deve compulsoriamente pagar a ele o valor de aluguel. O jogador tem o valor do aluguel abatido de seu saldo atual e o dono da propriedade recebe o valor dessa transação. No caso do jogador parar em um espaço que corresponde ao pagamento de taxas, ele paga o valor da mesma forma como compra uma propriedade. Quando o jogador não possui saldo suficiente para pagamento da dívida, suas propriedades são vendidas seguindo uma classificação de valor da propriedade, em ordem crescente, até que ele tenha saldo suficiente. Se após as vendas seu saldo não for suficiente, todo seu saldo será transferido ao seu credor e ele entrará em falência;
g. Falência: Se um jogador não possui recursos suficientes para pagar o valor do aluguel ou taxa o mesmo é eliminado do jogo.

Na Figura 3.3 temos a representação gráfica do tabuleiro implementado, onde se verifica os valores das propriedades (canto inferior direito da propriedade) e os respectivos valores de aluguel (canto inferior esquerdo da propriedade):

Figura 3.3 - Tabuleiro de Genopoly



Fonte: Autora

Mecânicas como: *"Vá para prisão"*, *"Cartas Chance", "Leilão", "Troca",* etc, não foram implementadas pois essas mecânicas implicam em um considerável aumento no número de possíveis estados no jogo. Os mecanismos de Banco e Banqueiro também não fazem parte dessa versão, assim sendo o pagamento de impostos não tem um destinatário. Também não é possível fazer empréstimos ou hipotecar as propriedades. Pela falta de tais mecanismos o jogador pode vir à falência mais rapidamente durante as simulações.

O código fonte do jogo possui um arquivo de configurações que foi utilizado para definir os parâmetros de cada simulação:

- *Initial coins*: saldo inicial de cada jogador;
- *Cautious remaining balance*: saldo mínimo restante após uma compra;
- *Max number of rounds*: o número máximo de rodadas caso não haja um vencedor;
- *Full turn coins*: valor recebido pelo jogador ao completar a volta no tabuleiro;
- *Demmanding rent*: valor mínimo de aluguel da propriedade a ser comprada;

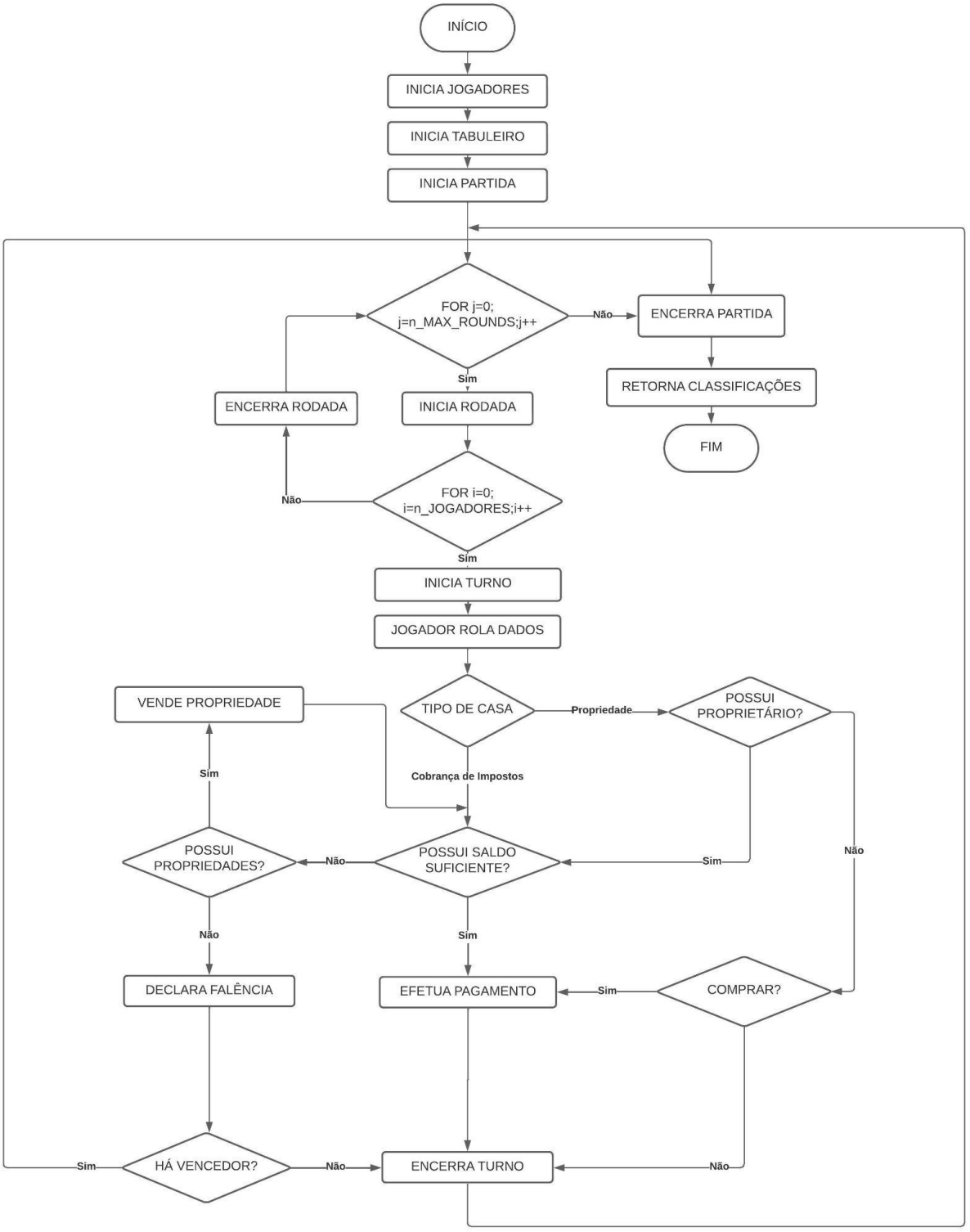
- *Rent base value modifier*: valor base de modificação do valor do aluguel de todas as propriedades;
- *Full set rent modifier*: valor base de modificação do valor do aluguel de uma propriedade que seja parte de um conjunto completo.

O jogo foi modelado de forma a receber as informações necessárias para a tomada de decisões de um jogador (*inputs* do AG) e para que possa receber as decisões do mesmo (*outputs* do AG) por meio de um sistema desenvolvido em Python. A única decisão que os jogadores devem tomar é comprar ou não uma propriedade disponível na posição do tabuleiro em que se encontram a cada turno. Assim, o *output* será binário. Já os *inputs* do AG são 4:

1. Coeficiente de relevância de cor da propriedade;
2. Coeficiente de relevância de saldo após a compra;
3. Coeficiente de relevância de valor do aluguel da propriedade;
4. Coeficiente de relevância de impulsividade.

Na Figura 3.4 pode-se verificar a execução do jogo, independente se é uma partida entre indivíduos de mesma geração do AG, ou se um indivíduo disputa a partida contra os perfis pré-estabelecidos.

Figura 3.4 - Execução do jogo



Fonte: Autora

3.1.2. Jogadores

Há dois tipos de jogadores interagindo na simulação. Os jogadores artificiais evolutivos são representados por cada indivíduo das gerações do AG. As ações desses jogadores mudam a cada geração e tendem a convergir para uma solução otimizada.

Já os jogadores artificiais estáticos têm seu comportamento definido por uma função imutável:

- Comportamento Impulsivo: compra qualquer imóvel desde que tenha saldo;
- Comportamento Cauteloso: compra um imóvel apenas se o dinheiro restante após a compra em sua carteira for maior ou igual a um valor pré-estabelecido;
- Comportamento ambicioso: compra um imóvel apenas se o aluguel do mesmo for maior ou igual a um valor pré-estabelecido;
- Comportamento aleatório: não é possível prever quando o jogador fará uma compra.

Os jogadores com perfis pré-estabelecidos foram representados de forma binária, conforme a Tabela 3.1. Cada um destes cromossomos determinava a influência que as características da propriedade à venda (ou do próprio jogador) teriam no momento de decisão da compra.

Tabela 3.1: Representação binária de perfis de jogadores

Perfil	Cor	Saldo	Valor de aluguel	Impulsividade
Impulsivo	0	0	0	1
Cauteloso	0	1	0	0
Exigente	1	0	1	0

Fonte: Autora

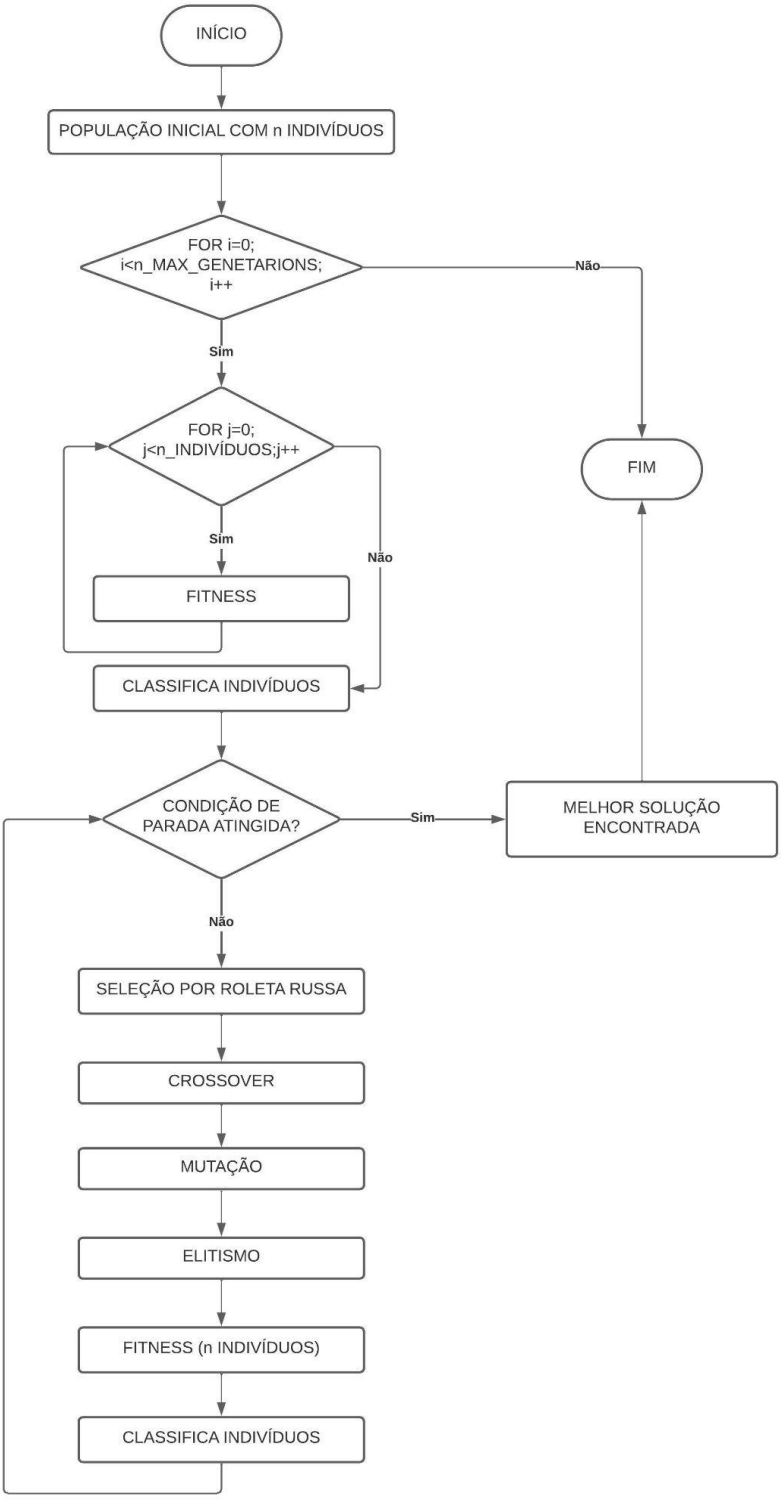
O jogador com perfil Aleatório era representado por um vetor de pesos, com valores aleatórios entre -1 e 1. Mas diferentemente do jogador representado pelo indivíduo gerado no AG, que também também era representado por vetor de pesos, o jogador aleatório era gerado pelo jogo ao início da partida e nunca passou por algoritmos evolutivos.

3.1.3. Algoritmo genético

Foram feitas simulações considerando jogos entre 5 jogadores. A princípio cada jogador representado pelo AG disputava uma partida contra os jogadores de perfil pré-estabelecido. Levando em consideração que não se joga sempre contra os mesmos jogadores e que esses nem sempre adotam a mesma estratégia, também foram feitas simulações onde os indivíduos de uma mesma geração do AG se enfrentaram, uma vez que se deseja encontrar uma estratégia robusta.

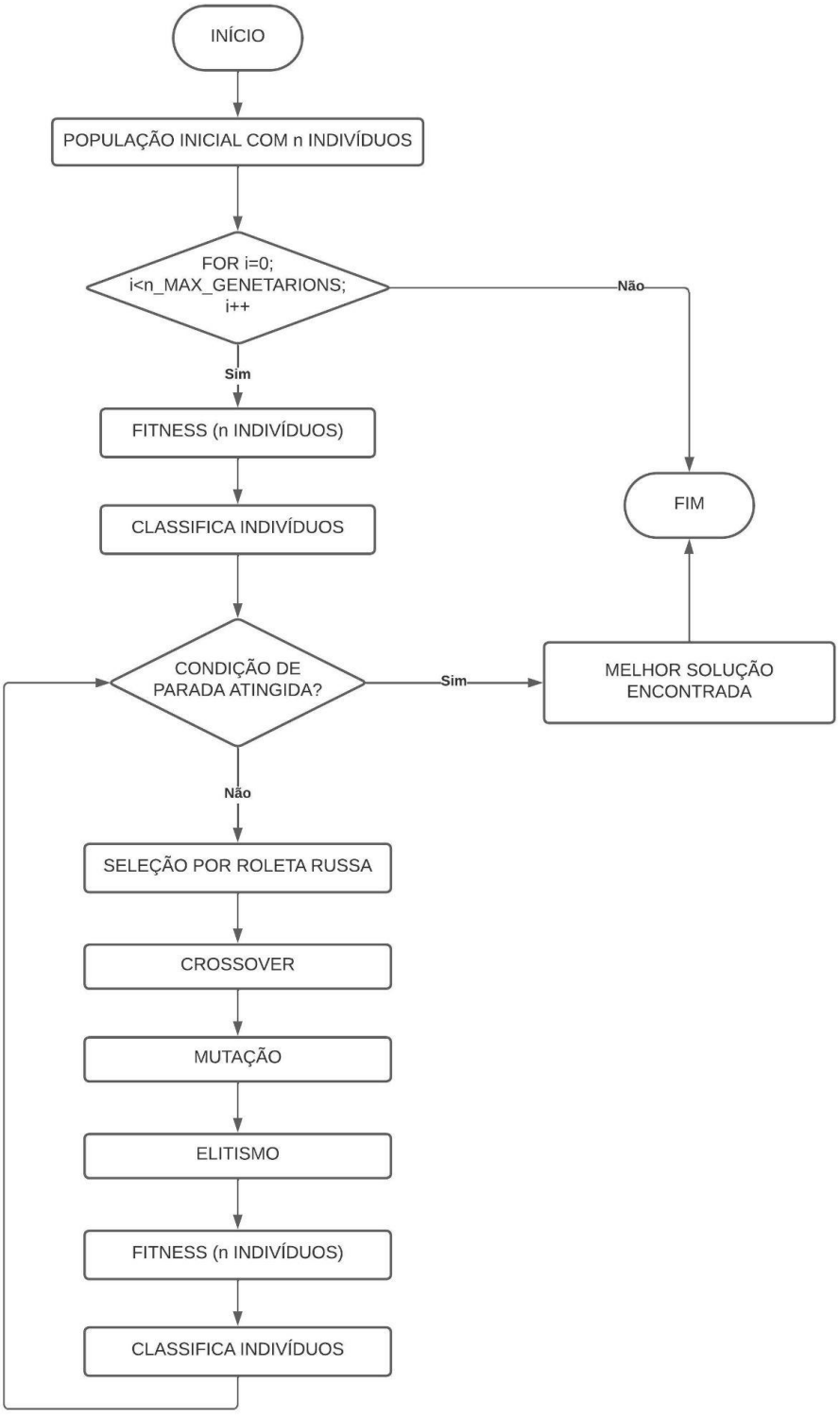
Na Figura 3.5 pode-se verificar a execução do AG contra perfis pré-estabelecidos. Nesse caso, além de iterar sob cada geração, havia também uma iteração sob o número de jogadores, ou seja, sob *Tp*. Na Figura 3.6 verifica-se a execução do AG quando todos os indivíduos de mesma geração disputavam a mesma partida. Neste caso o algoritmo iterava apenas sob *Ng* estipulado nos parâmetros da simulação, caso o critério de parada não fosse atingido antes.

Figura 3.5 - Execução do AG contra perfis pré-estabelecidos



Fonte: Autora

Figura 3.6 - Execução do AG contra indivíduos da mesma geração



Fonte: Autora

A implementação do AG possui um arquivo de configurações que foi utilizado para definir os parâmetros de cada simulação:

- *Epochs*: o número máximo de gerações (Ng);
- *Population size*: o número de indivíduos a cada geração (Tp);
- *Resolution*: a quantidade de cromossomos de cada indivíduo;
- *Base*: intervalo de valor de cada cromossomo;
- *Acceptable value*: Pontuação do indivíduo na partida disputada;
- *Crossover rate*: taxa para crossover a ser aplicada nos indivíduos;
- *Mutation rate*: Taxa de mutação dos indivíduos;
- *Elitism size*: Quantidade de indivíduos a serem selecionados para compor a próxima geração;
- *Consecutive epochs*: quantidade de gerações consecutivas onde o melhor indivíduo não mudou;
- *Number of players per match*: Quantidade de jogadores (indivíduos) disputando cada jogo;

Uma vez definidos os valores para cada parâmetro, a população era iniciada, formando então a primeira geração de indivíduos a serem avaliados. Os indivíduos da primeira geração eram então submetidos à FF. O vetor de pesos de cada indivíduo era enviado para a FF como *input* do jogo.

Ao longo do jogo, a única decisão que devia ser tomada pelos jogadores é comprar ou não uma propriedade. Essa decisão era tomada aplicando o vetor de pesos à Equação (5), e depois por meio da Equação (6) a seguir:

$$a = \{1, \text{se jogador tem as demais de mesma cor}, 0 \text{ do contrário}\} \quad \text{Equação (1)}$$

$$b = \{1, \text{se saldo maior que o definido}, 0 \text{ do contrário}\} \quad \text{Equação (2)}$$

$$c = \{1, \text{se aluguel maior que o definido}, 0 \text{ do contrário}\} \quad \text{Equação (3)}$$

$$d = 1 \quad \text{Equação (4)}$$

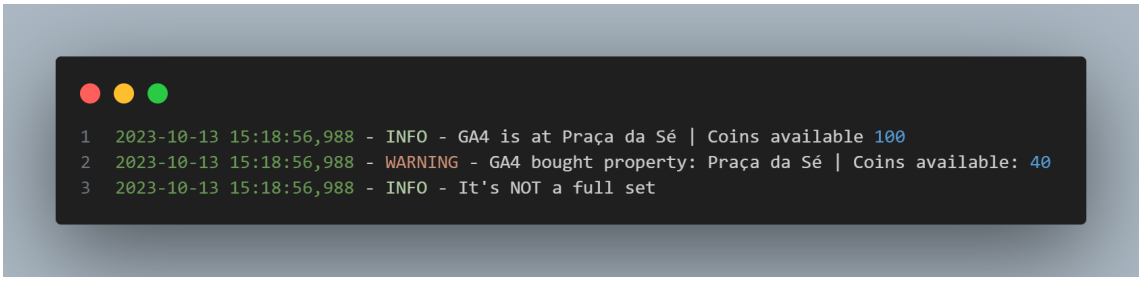
$$decisão = p_a a + p_b b + p_c c + p_d d \quad \text{Equação (5)}$$

$$comprar = \{True, se\ decisão > 0, Falsedo\ contrário\} \quad \text{Equação (6)}$$

Uma vez que a Equação(6) retorne True, o jogador compra a propriedade. Caso ele já tenha propriedades da mesma cor, é avaliado se ele tem um conjunto completo, o que torna o aluguel mais caro. Se ele não comprar, nada acontece e seu turno se encerra.

Abaixo, na Figura 3.7, é apresentado um exemplo de compra de uma propriedade. Nessa simulação todos os jogadores foram gerados pelo AG. O indivíduo identificado como GA4 rolou os dados e parou na propriedade “Praça da Sé”. O GA4 possuía 100 moedas e adquiriu a propriedade por 60 moedas. Como o GA4 não possuía as demais propriedades de mesma cor, ele não formou um conjunto completo.

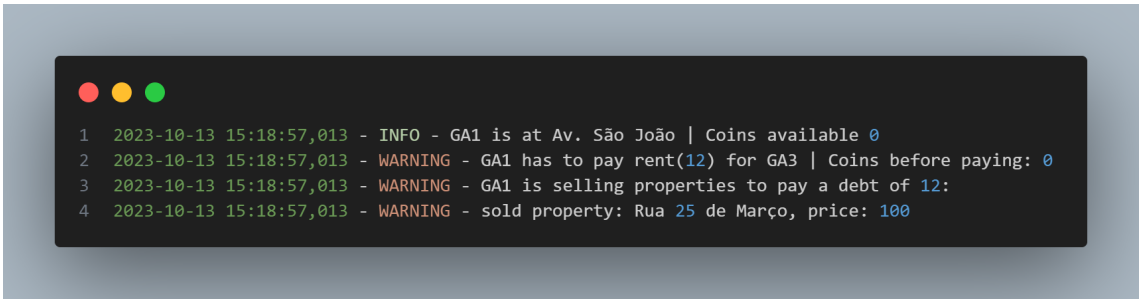
Figura 3.7 - AG compra uma propriedade



Fonte: Autora

Ao parar em uma propriedade identificada como “Income Tax”, o jogador tinha de pagar o valor do imposto de forma compulsória. Não tendo saldo suficiente para o pagamento, suas propriedades eram vendidas até que ele tivesse o suficiente para pagar, ou fosse à falência. O mesmo acontecia no caso de pagamento de aluguel. Na Figura 3.8 pode-se verificar um exemplo onde o jogador GA1 precisou vender suas propriedades para pagar aluguel ao jogador GA3:

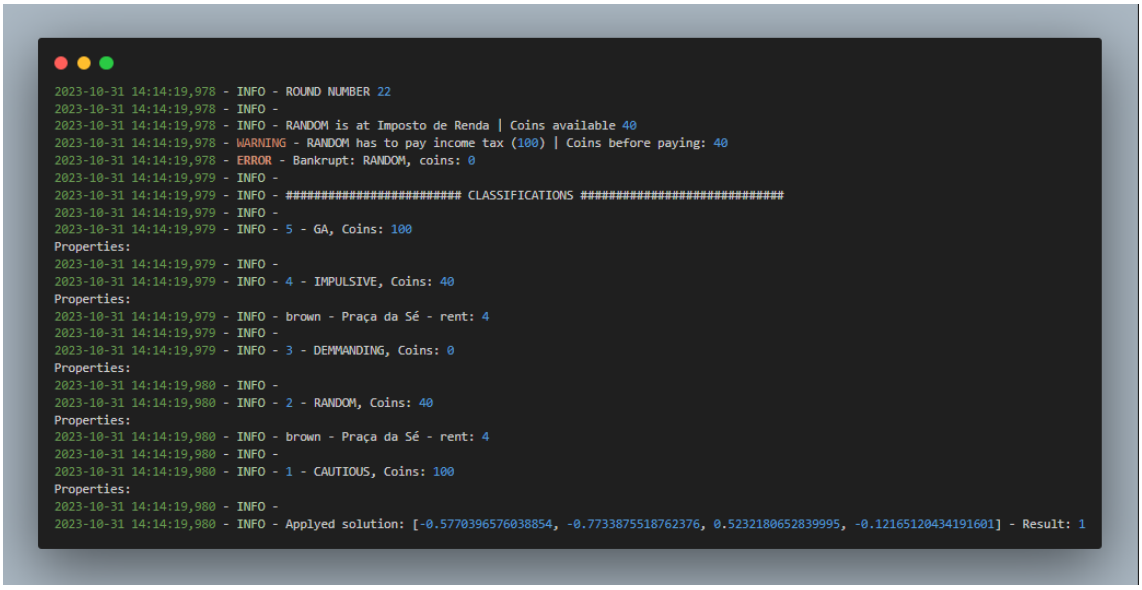
Figura 3.8 - AG vende uma propriedade



Fonte: Autora

A simulação continuava até que se obtivesse um vencedor, fosse porque todos os outros foram à falência, ou sendo o melhor classificado ao final da quantidade de rodadas estabelecidas. Na Figura 3.9 vemos o resultado final de uma partida onde todos os jogadores foram à falência e o jogo terminou na rodada de número 22 tendo como vencedor o jogador gerado pelo AG. Após ser determinado um vencedor a classificação de todos os jogadores era exibida. No caso do jogador ter ido à falência mesmo possuindo propriedades (caso o saldo após a venda não fosse suficiente), a lista de suas propriedades é exibida junto de sua classificação. Na Figura 3.9 podemos ver que o jogador com perfil Impulsivo ficou em quarto lugar e possuía a “Praça da Sé” ao falir. Nessas condições as propriedades passavam a não ter mais um proprietário, podendo ser adquiridas pelos demais jogadores. Como se pode notar, após a falência do Impulsivo, o jogador Aleatório adquiriu a mesma propriedade.

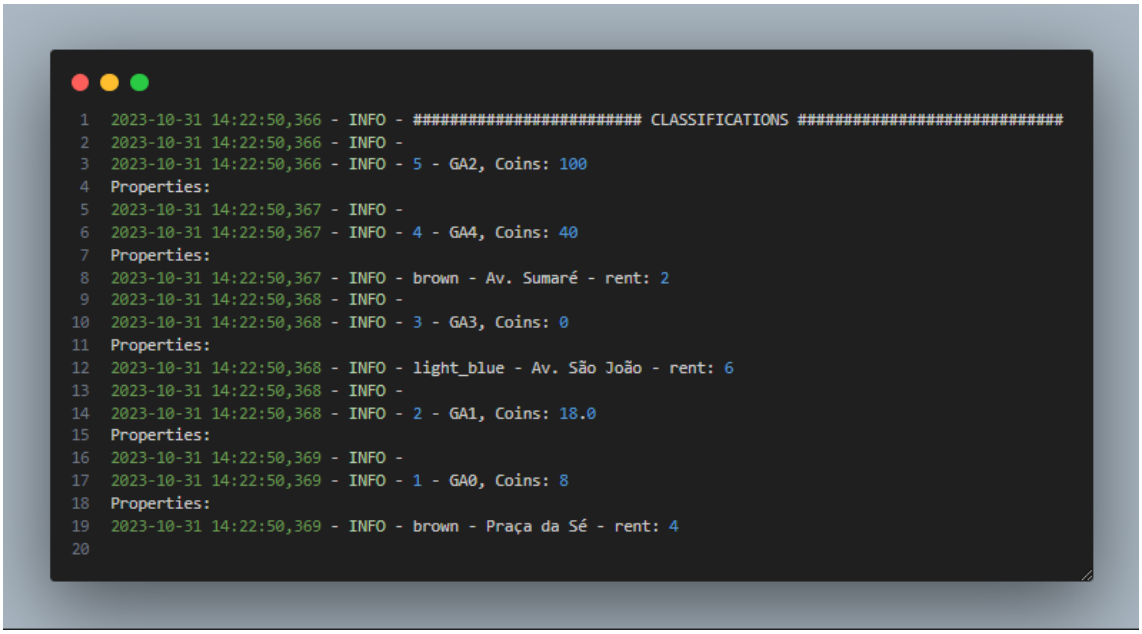
Figura 3.9 - AG vence jogo contra perfis pré-determinados



Fonte: Autora

Já na Figura 3.10 podemos ver a classificação da partida de uma outra simulação, onde os jogadores eram todos indivíduos gerados pelo AG.

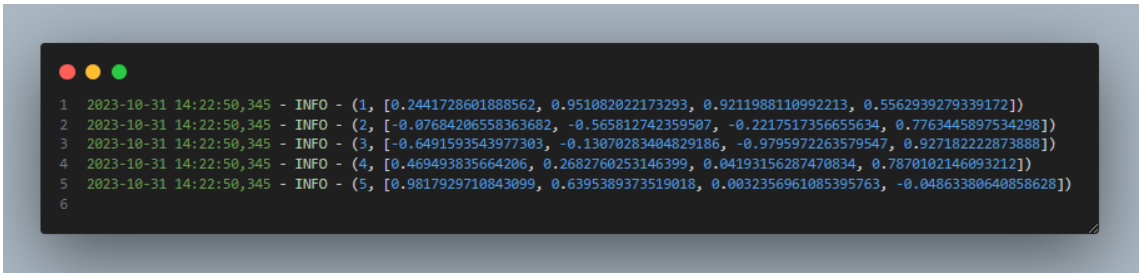
Figura 3.10 - Classificação dos indivíduos de uma geração



Fonte: Autora

Depois disso as soluções (indivíduos) eram ranqueadas. Nas simulações onde cada indivíduo disputava contra perfis pré-estabelecidos, essa classificação acontecia depois de todos os jogos. Nas simulações onde os indivíduos de uma mesma geração do AG disputavam a mesma partida, essa classificação acontecia ao final da partida. Na Figura 3.11 pode-se ver as soluções (indivíduos) classificadas conforme seu desempenho no jogo. É importante observar que nesse momento cada indivíduo era pontuado conforme seu desempenho, de modo que o primeiro colocado recebia 5 pontos, o segundo colocado, 4 pontos, e assim por diante.

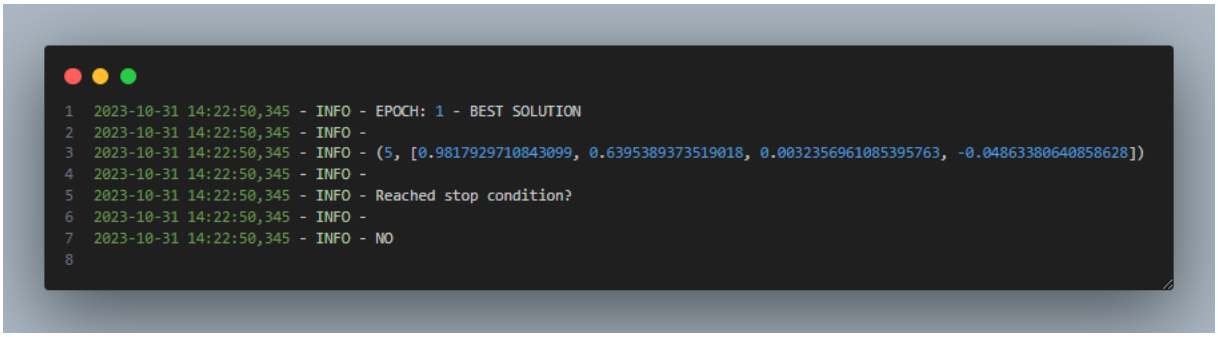
Figura 3.11 - Classificação dos indivíduos de uma geração após término da partida



Fonte: Autora

Uma vez classificados os indivíduos da geração atual, era verificada a condição de parada. Nesse caso, optou-se por definir os critérios como sendo a classificação do indivíduo em primeiro lugar, 5 pontos, e também que ele fosse o melhor indivíduo por 5 gerações. Como se pode ver na Figura 3.12, a condição de parada não foi atingida, neste caso, por ainda não ter vencido 5 partidas consecutivas.

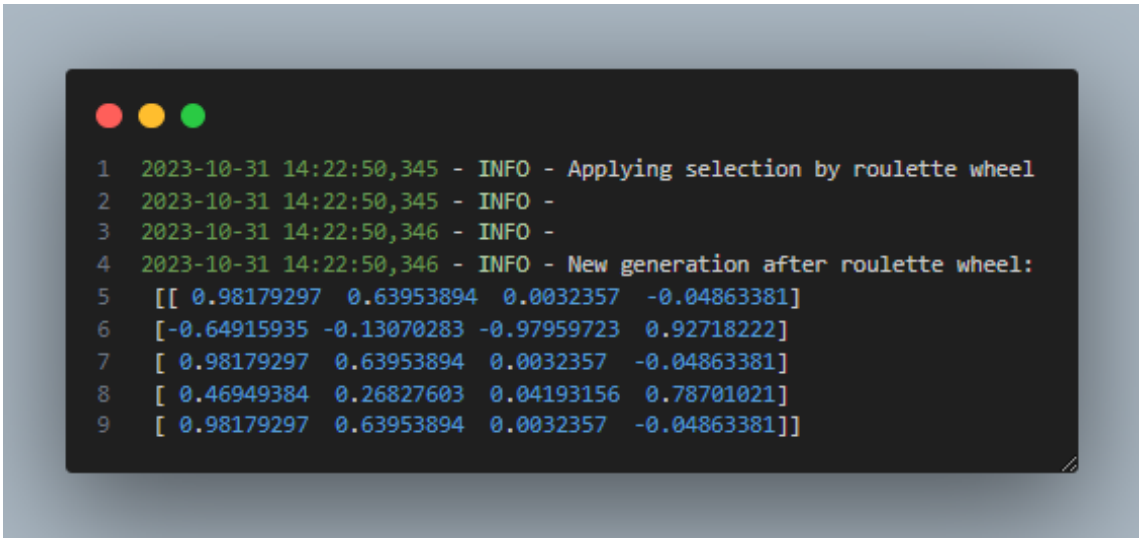
Figura 3.12 - Critério de parada



Fonte: Autora

Uma vez que o critério de parada não foi atingido, o algoritmo prosseguia com os operadores genéticos. Na Figura 3.13 pode-se verificar a nova geração após seleção por roleta. Como esperado, por ter uma maior probabilidade, o indivíduo melhor classificado foi escolhido.

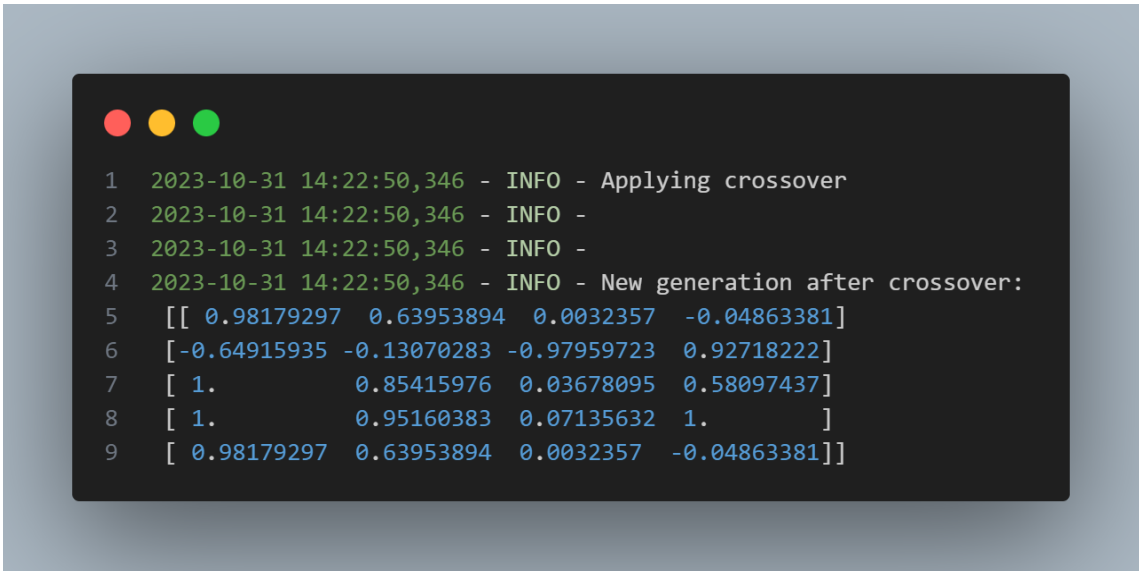
Figura 3.13 - Seleção por roleta



Fonte: Autora

O próximo operador genético a ser aplicado era o crossover, como exemplificado na figura 3.14.

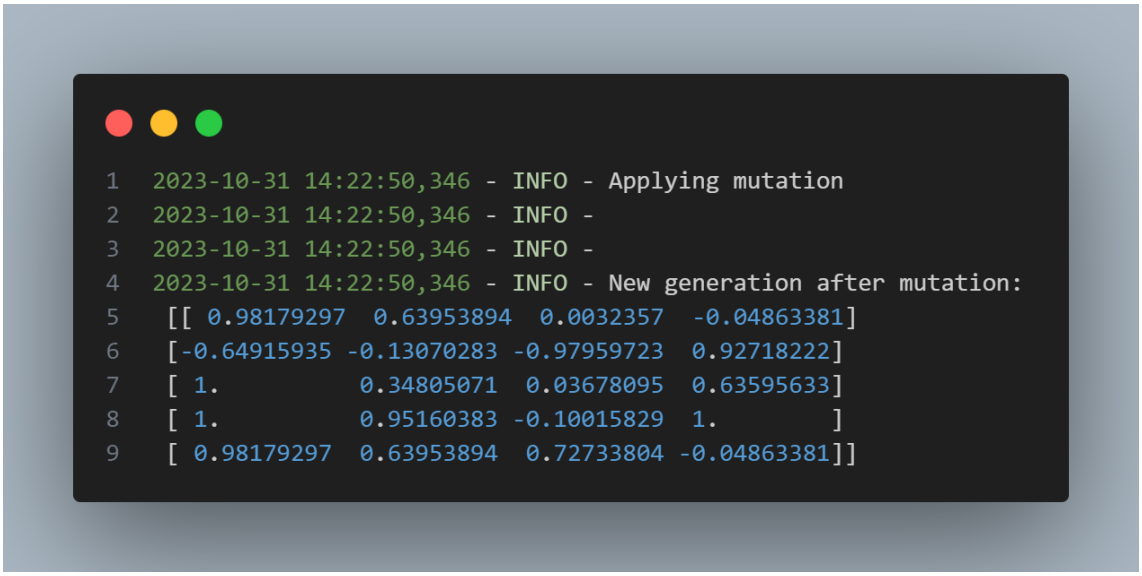
Figura 3.14 - Crossover



Fonte: Autora

Depois de se aplicar o crossover, os operadores genéticos seguintes eram a mutação e o elitismo. Na Figura 3.15 verifica-se a nova geração após mutação dos indivíduos.

Figura 3.15 - Mutação

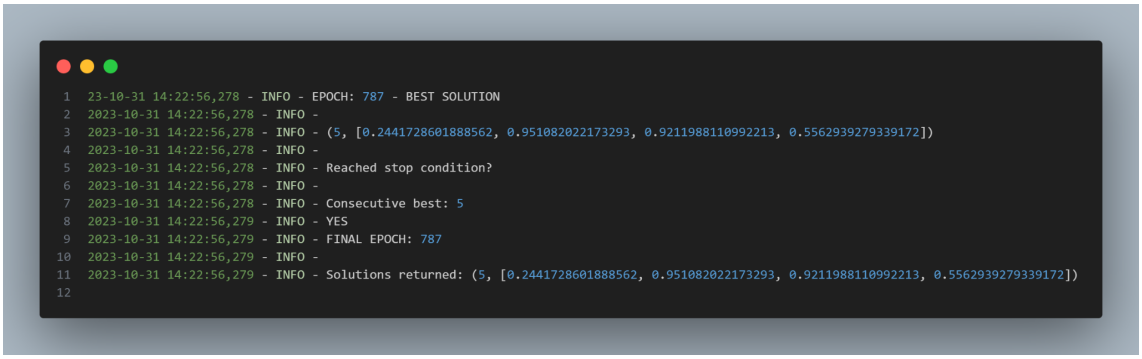


Fonte: Autora

Por meio do elitismo, o melhor indivíduo era selecionado para compor a nova geração a ser avaliada por meio da FF.

Ao ser atingida a condição de parada, o melhor indivíduo era exibido com os valores de seus cromossomos, permitindo avaliar a solução encontrada. Além disso, é possível verificar quantas gerações foram necessárias até que o AG convergisse para uma solução que atendesse aos critérios de parada.

Figura 3.16 - Solução otimizada



Fonte: Autora

CAPÍTULO IV

4.1. RESULTADOS E DISCUSSÕES

Foram feitas 156 simulações ao todo em busca de uma solução otimizada para o jogo implementado. A princípio, várias simulações foram interrompidas antes da condição de parada pela necessidade de ajustes na FF ou mesmo no AG. Ao todo foram feitas 18 simulações do tipo GAxProfiles, sendo que dessas, 8 convergiram para uma solução. As simulações do tipo GAxGA totalizaram 50 experimentos, onde 36 convergiram para uma solução.

Nas simulações iniciais vários ajustes na implementação do jogo se mostraram necessários. Um exemplo de ajuste das mecânicas do jogo é a forma como os pagamentos eram realizados. A princípio o excedente de um pagamento por meio de venda de propriedades não era devolvido, o que fazia com que os jogadores fossem à falência mais rápido do que se esperava.

A quantidade de dados jogados também se mostrou relevante pelo tamanho do tabuleiro implementado. O uso de dois dados não era ideal, uma vez que ele parava em poucas propriedades antes de completar uma volta.

O valor dos aluguéis também precisou ser reajustado. Uma vez que não há possibilidade de se adicionar casas e hotéis, os valores não aumentavam tanto quanto no jogo original. Para ajustar esses valores, foi adicionado um parâmetro para modificar os valores base, ou mesmo para definir um novo valor quando o jogador era dono de um conjunto completo.

Outra análise feita foi quanto à proporção entre os valores recebidos pelo jogador e os valores pagos por ele ao longo do jogo. A princípio havia apenas uma propriedade de pagamento de impostos, o que é pouco comparado ao jogo original que possui duas propriedades onde se paga impostos, duas companhias de serviços (sanitária e elétrica), além de cartas de sorte que podem implicar no pagamento de mais taxas (consultas médicas, multas, presentes para os outros jogadores, dentre outros).

Além desses valores, também se observou a necessidade de aumentar o valor inicial de moedas recebido pelo jogador. Dado que ele começava o jogo com 100 moedas e logo poderia

ter que pagar um imposto de 200 moedas, ele vinha à falência antes mesmo de ter a chance de comprar uma propriedade.

Depois de ajustar os parâmetros do jogo, foi possível perceber a necessidade de ajustar alguns valores do AG. A quantidade máxima de gerações precisou ser aumentada a fim de que o AG atingisse o critério de parada mais tardiamente.

Quanto aos parâmetros do AG, foi possível perceber como esses influenciavam ou não na convergência para uma solução. Observou-se que aumentar a *Tm* faz com que um *Ng* muito maior seja necessário para encontrar uma solução. Por outro lado, não foram observadas mudanças significativas no *Ng* necessárias para convergência ao alterar a *Tc*.

Quanto ao critério de parada, observa-se que quanto mais vitórias consecutivas, maior o *Ng* necessário para encontrar uma solução. A maioria das simulações tinha como critério de parada 5 vitórias consecutivas ou *Ng* igual a 1000 gerações. Para esse critério, houve uma taxa de convergência de 65%. Ao alterar o critério de parada para 10 vitórias consecutivas, mesmo considerando um *Ng* igual a 20.000, o AG não encontrou uma solução.

Quanto às soluções encontradas, observa-se um escopo de soluções possíveis. As mais interessantes serão discutidas a seguir. Os valores dos parâmetros utilizados no jogo podem ser verificados na Tabela 4.1 e os resultados se encontram na Tabela 4.2.

Tabela 4.1: Parâmetros das simulações

Simulação	Tipo	Saldo Inicial	Recompensa Por Volta Completa	Saldo Após Compra	Valor Aluguel Para Perfil Exigente	Condição de parada (vitórias consecutivas)
33	GAxProfiles	200	50	50	16	5
51	GAxProfiles	200	50	50	16	5
121	GAxGA	100	0	100	50	5
131	GAxGA	100	0	100	50	7
141	GAxGA	500	0	100	50	7
154	GAxGA	500	100	100	50	5

Fonte: Autora

Tabela 4.2: Resultados das simulações

Simulação	Soluções			
	Conjunto Completo	Saldo Após Compra	Valor Mínimo de Aluguel	Impulsividade
33	0.44	-0.85	-0.85	-0.80
51	-0.40	-0.38	-0.20	-0.92
121	-0.35	-0.97	0.17	0.82
131	0.96	-0.98	0.63	-0.99
141	0.83	-0.85	0.53	0.72
154	-0.55	-0.81	-0.25	-0.47

Fonte: Autora

Pode-se perceber que ao disputar contra perfis pré-estabelecidos, a solução encontrada pelo AG (simulações 33 e 51) não tem interesse em comprar propriedades, dado que as soluções encontradas aplicam pesos negativos aos critérios de avaliação de compra de uma propriedade. Esses pesos, quando aplicados à Equação (5), resultarão sempre em um valor menor que 0, o que fará com que a Equação (6) retorne sempre False para a decisão de comprar uma propriedade.

Uma possível explicação é o fato dos adversários serem previsíveis, exigindo pouco esforço por parte do AG. Um outro fator a se considerar foi o fato de que os jogadores recebiam, nessas simulações, mais recursos (maior saldo inicial, recompensa por volta completa), o que dava ao AG a possibilidade de permanecer no jogo sem a necessidade de realizar compras para obter recursos por meio do recebimento de aluguéis. Ao observar os logs das simulações 33 e 51, é possível ver que os jogadores representados pelas soluções vencedoras nunca compraram uma propriedade.

De forma semelhante, a simulação 154 também convergiu para uma solução onde o jogador nunca compra. Note que nessa simulação os jogadores recebiam até mais recursos do que nas simulações 22 e 51.

Por outro lado, as soluções das simulações 121, 131 e 141 têm estratégias bem diferentes. Enquanto a solução da simulação 121 considera a impulsividade a melhor estratégia, a solução da simulação 131 evita ser impulsiva e não se importa com o saldo após a compra. Já a estratégia adotada pela solução da simulação 141 mostra que o mais importante é considerar se a propriedade forma um conjunto completo com as demais em posse do jogador.

CAPÍTULO V

5.1. CONSIDERAÇÕES FINAIS

Com o desenvolvimento deste trabalho foi possível perceber quão amplo é o ambiente oferecido pelo setor de jogos para aplicações de IA. Não apenas por oferecer soluções para esse mercado em si, mas também pela possibilidade de simular ambientes mais complexos que se assemelham às situações onde decisões devem ser tomadas com base em um grande número de informações.

Pode-se notar que a utilização de IA em testes de jogos em desenvolvimento é uma importante ferramenta para evitar que comportamentos indesejados sejam adotados pelos jogadores. Isso permite aos criadores direcionar os jogadores para determinadas ações.

Também pode-se observar que dado o problema em questão, é muito difícil convergir para uma solução única, pois várias estratégias podem ser bem-sucedidas no jogo de Monopoly, ressaltando-se a compra de conjuntos completos e a impulsividade.

5.2. ESTUDOS FUTUROS

Há cada vez mais espaço para a utilização de técnicas de IA no desenvolvimento e análise de jogos. Técnicas que outrora encontravam grande resistência por parte de programadores, muito pelo custo computacional, agora são bem-vindas na intenção de validar mecânicas de jogos, testar caminhos e até mesmo para desenvolvimento de NPCs que proporcionem interações mais realistas. Espera-se que a IA torne-se mais amplamente utilizada no desenvolvimento de jogos possibilitando melhores implementações.

Para trabalhos futuros, pode-se considerar:

- A análise do comportamento de jogadores adversários que tenham estratégias bem consolidadas, identificando padrões em suas tomadas de decisão;
- Adicionar novas mecânicas à implementação do jogo e analisar se as melhores estratégias classificadas se mantêm.

REFERÊNCIA BIBLIOGRÁFICA

[1] KOTRIK, ROBERT. **Searching for a strategy of Monopoly game using cognitive and artificial intelligence approach**. (Master's thesis) - Comenius University In Bratislava, Bratislava, 2012.

[2] Aarseth, E. Playing Research: **Methodological approaches to game analysis**. Digital Arts Culture, 2004.

[3] Konzack, L. Computer Game Criticism: **A Method for Computer Game Analysis**. Tampere University Press, 2002.

[4] Silva, Heitor Ferreira Camargos. **Sistema Automatizado de Tomada de Decisão em Um Jogo de Poker**. Universidade Federal de Uberlândia, 2018.

[5] PyData Amsterdam Meetup May 2016. Disponível em: <https://godatadriven.com/blog/pydata-amsterdam-meetup-may-2016/>. Acesso em: 03/10/2021

[6] Shree Raj Shrestha; Daniel S. Myers; Richard A. Lewin; Rollins College. Optimizing Strategies for Monopoly: The Mega Edition Using Genetic Algorithms and Simulations. **Academy of Economics and Finance Journal**, Volume 7, página 87.

[7] Dodson, Edward J. How Henry George's Principles Were Corrupted Into The Game Called Monopoly. **The True History of the Monopoly Game**, 2011. Disponível em: <https://www.henrygeorge.org/dodson_on_monopoly.htm>. Acesso em 2 de out. 2021.

[8] Disponível em: <https://www.gqportugal.pt/monopoly-historia-jogo>. Acesso em: 03/10/2021.

[9] Goldberg, 1989. Disponível em: <http://www.leg.ufpr.br/~leonardo/artigos/tese_mahfoud.pdf>. Acessado em: 29/09/2021.

[10] CASTRO NETO, HENRIQUE DE., **Uma nova abordagem de aprendizagem de máquina combinando elicitação automática de casos, aprendizagem por reforço e mineração de padrões sequenciais para agentes jogadores de damas**. Tese (Doutorado) – Universidade Federal de Uberlândia, Minas Gerais, 2016.

[11] WONG, T.-T. **Performance evaluation of classification algorithms by k-fold and**

leave-one-out cross validation. Pattern Recognition, Elsevier, v. 48, n. 9, p. 2839–2846,

2015.

[12] MIRALLES-PECHUÁN, L.; PONCE, H.; MARTÍNEZ-VILLASEÑOR, L. **A novel methodology for optimizing display advertising campaigns using genetic algorithms**. Electronic Commerce Research and Applications, Elsevier, v. 27, p. 39–51, 2018.

[13] BIRCH, CHAD. 2010. Disponível em: <http://gameinternals.com/understanding-pac-man-ghost-behavior> Acessado em 13/10/2023.

[14] FURNKRANZ, 2004. Disponível em: <https://repositorio.ufu.br/bitstream/123456789/22184/3/MetodosInteligenciaArtificial.pdf>. Acessado em: 29/09/2021.

[15] MILLINGTON; FUNGE, 2009. Disponível em: <https://repositorio.ufu.br/bitstream/123456789/22184/3/MetodosInteligenciaArtificial.pdf>. Acessado em: 29/09/2021.

[16] Nogueira M. L., Saavedra O. R.. Estratégias Evolutivas Aplicadas à Resolução de Otimização Multimodal, Simpósio Brasileiro de Automação Inteligente, 1999.

[17] Holland J. H., **Adaptation in natural and artificial systems**. The University of Michigan Press, 1975.

[18] LINDEN, R.. Algoritmos Genéticos: **Uma Importante Ferramenta Da Inteligência Computacional**. 2a edição. Rio de Janeiro: Editora Brasport, 2006.

[19] MICHALEWICZ, Z.; SCHOENAUER, M.. **Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation**, Vol. 4, No. 1, pp. 1-32, 1996.

[20] MITCHELL, M. **An introduction to genetic algorithms**. [S.l.]: MIT press, 1998.

[21] LIPOWSKI, A.; LIPOWSKA, D. Roulette-wheel selection via stochastic acceptance. **Physica A: Statistical Mechanics and its Applications**, Elsevier, v. 391, n. 6, p. 2193–2196, 2012.

[22] HOLLAND, J. H. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence**. MIT Press, 1992.

[23] PYTHON SOFTWARE FOUNDATION. Python Language Site: Documentation, 2023. Página de documentação. Disponível em: <https://www.python.org/doc/>. Acesso em: 13 de outubro de 2023.