UM ALGORITMO EVOLUTIVO PARA JOGAR ASTEROIDS

Jacqueline Vianna Alves[1]
Kleber de Oliveira Andrade[2]

DOI: 10.47283/244670492020080237

Resumo
Este trabalho apresenta um Algoritmo Evolutivo (AE) para controlar de forma autônoma a nave do jogo Asteroids. Para elaboração do AE e do clone do jogo Asteroids, foi utilizado a *game engine* Unity 3D e a linguagem C\#. Os resultados obtidos foram satisfatórios, mostrando que os AEs satisfazem as necessidades de um agente inteligente. Sendo assim, o presente trabalho vem a contribuir com a área, onde são poucas as publicações de AEs aplicados a jogos digitais.
Palavras-chave: Inteligência Artificial. Algoritmos Genéticos. Jogos.

Abstract
This project presents a development of an Evolutionary Algorithm (EA) to autonomously control the Asteroids game ship. The Unity 3D game engine and the C\# language were used to elaborate the AEs and the Asteroids game clone. The results obtained were satisfactory, showing that EAs meet the needs of an intelligent agent. Thus, the present work contributes to the area, where there are few EA publications applied to digital games.
Keywords: *Artificial intelligence. Genetic Algorithms. Games.*

Introdução

Os jogos digitais têm evoluído constantemente, principalmente em realismo gráfico. Para melhorar esta realidade, é necessário que os personagens virtuais estejam aptos a responder de forma adequada as ações do jogador. Então, para que isso se torne realidade, tanto os pesquisadores quanto as empresas, recorrem as técnicas de Inteligência Artificial (IA).

A Inteligência Artificial abrange vários campos, como ciências da computação, robótica, aprendizado de máquina, processamento de linguagem natural, engenharia e visão computacional. Dentre as diversas técnicas envolvidas em IA, os Algoritmos Evolutivos (AEs) tem sido amplamente utilizado em problemas de otimização e pouco utilizado na indústria dos jogos.

Por meio de AEs, o agente pode navegar pelo ambiente (MARTIN, 2011), pode adaptar estratégias dos inimigos (ANDRADE et al., 2009), pode balancear a dificuldade do jogo (ANDRADE et al., 2018; GARCIA et al., 2018) e até mesmo jogar sozinho (KUHN e FOLEY, 2015; SODER, 2018; ARORA, 2019) - geralmente usando algoritmos híbridos.

O presente trabalho tem por objetivo contribuir com a aplicação dos AEs no controle da nave no jogo Asteroids, pois existem poucos trabalhos de AEs em jogos encontrados até o momento. Vale ressaltar, que, muitos trabalhos utilizam-se do uso de AEs combinados com Redes Neurais para controle dos agentes em jogos.

[1] Discente da Fatec Americana. E-mail: j.taquetto@gmail.com
[2] Docente da Fatec Americana. E-mail: kleber.andrade@fatec.sp.gov.br

Este trabalho está estruturado em 3 seções: a Seção 1 apresenta os conceitos fundamentais para entendimento deste trabalho. Seção 2 detalha os materiais e métodos utilizados para cumprimento do objetivo proposto. Os experimentos e resultados são discutidos na Seção 3. Por fim, é apresenta as conclusões do trabalho e o que será possível fazer no futuro.

1.1 Algoritmo evolutivo

Algoritmo Evolutivo (AE) é considerado uma técnica computacional de busca e otimização inspirada na evolução natural das espécies. Este algoritmo foi inicialmente desenvolvido por Holland (1975). Segundo Eiben e Smith (2015) o princípio básico de todo AE, é o mesmo:

- Geração aleatória de indivíduos (somente na primeira geração). Cada indivíduo ou solução também é conhecido como cromossomo ou genoma;
- Avaliação da população seguindo algum critério determinado por uma função que calcula a qualidade individual de cada cromossomo (aptidão ou *fitness*);
- Seleção de indivíduos para realizar reprodução (crossover) por meio do seu *fitness*;
- Imposição de variações aleatórias nos cromossomos dos indivíduos resultantes (mutações);
- Reinício do processo com os novos indivíduos (nova geração).
- Estes passos são repetidos até que um critério de parada preestabelecido seja atingido.

Ao final da aplicação do evolutivo, obtêm um ou mais indivíduos evoluídos, ou seja, soluções mais aptas ao ambiente.

Diversos elementos desse processo evolutivo são estocásticos: a seleção favorece indivíduos mais bem adaptados, mas existe também a possibilidade de outras soluções serem selecionados. A recombinação dos indivíduos também é aleatória, assim como a mutação. Destaca-se que é possível adotar também a técnica de substituição de indivíduos baseadas no conceito do elitismo, evitando a perda de soluções de alta aptidão (EIBEN e SMITH, 2015; DE JONG, 2012).

1.2 Algoritmo evolutivo em jogos

Os algoritmos evolutivos estão sendo utilizados para o desenvolvimento de alguns jogos. Martin (2011) criou o jogo "InvAIders" – uma versão moderna do jogo "Space Invaders" - para controlar as naves inimigas contra o jogador. O AE funciona dividindo o cromossomo em duas partes: i) posicionamento das naves (eixo x) na parte superior; ii) comportamento da nave, que determina como o inimigo se moverá e atirará. O *fitness* leva em consideração a quantidade de vezes que o inimigo mata o jogador e por quanto tempo ele sobrevive. Após cada geração, pode-se ver um aumento imediato no desempenho da nova geração.

"O Último Sobrevivente" foi um jogo criado por Andrade *et al*., (2009), na qual, o AE é responsável por adaptar as estratégias de quatro NPCs (personagens não controlados pelo jogador). Neste jogo, o usuário controla um cavaleiro que deve destruir seus oponentes em um curto espaço de tempo. O cromossomo criado leva em consideração a vida atual do NPC e a quantidade de aliados próximos. No final de cada batalha, o *fitness* dos indivíduos é calculado pela quantidade de ataques bem-sucedidos (+5 por ataque) e por tempo de vida (+2 / 10 segundos vivos). Como resultado, além do AE conseguir adaptar as estratégias para derrotar o jogador, os NPCs também começaram atacar em grupos.

Garcia *et al*., (2018) criam um AE capaz de adaptar a dificuldade do jogo "Whac-A-Mole" para reabilitação motora de pessoas pós-AVC, baseado no jogo "The Catcher" de Andrade et al. (2018). Em ambos os trabalhos, os autores criam um jogador virtual (movimento do punho que será reabilitado) para calcular a dificuldade do jogo e a habilidade do jogador. Em "Whac-A-Mole'" o AE adapta as posições e tempo de aparição da toupeira. No jogo "The Catcher" é adaptado a posição e velocidade de queda das castanhas. Ambos os trabalhos, o AE consegue equilibrar a dificuldade do jogo em relação a habilidade do jogador, baseado na teoria de Fluxo.

Soder (2018) utilizou o jogo "Gradius" do Nintendo Entertainment System (NES), que funciona dentro de um emulador, interagindo e sendo controlado por uma rede neural dinâmica (NEAT). NEAT é uma RNA que evolui através de mudança em sua topologia e alteração de pesos das suas conexões, seguindo a abordagem de AEs. Então, partindo de zero conhecimento, a NEAT foi capaz de jogar sozinha um trecho do jogo, efetuando ações que normalmente seriam realizadas por seres humanos. Kuhn e Foley (2015) comparam técnicas hibridas como NEAT e EBT (*Evolving Behavior Trees* - cromossomo que evolui uma arvore de comportamento) para jogar Super Mario World.

Por fim, Arora (2019) utiliza um código híbrido de Redes Neurais Artificiais (RNA) e Algoritmos Genéticos, para jogar, o clássico jogo de rolagem lateral do dinossauro disponível no Google Chrome ("Chrome Dinosaur Game'"). O AE codifica todos os pesos sinápticos e funções de transferências que definem o comportamento de uma RNA. Sendo assim, a RNA tem suas entradas definidas como a distância até o alvo, tamanho do alvo e velocidade do dinossauro. Essas entradas são processadas pela RNA através da configuração do cromossomo e como saída obtém-se o controle de pular e abaixar o dinossauro. O resultado foi eficiente e eficaz, na qual em poucas gerações o dinossauro aprendeu a jogar.

2. O Jogo Asteroid

Asteroids (Figura 2) é um jogo de arcade multidirecional com tema espacial, projetado por Lyle Rains, Ed Logg e Dominic Walsh e lançado em novembro de 1979 pela Atari, Inc. O jogador controla uma única nave espacial em um campo de asteroides que é periodicamente atravessado por discos voadores. O objetivo do jogo é atirar e destruir os asteroides e discos voadores, sem colidir com nenhum deles, nem ser atingido pelo contra fogo dos discos voadores. O jogo fica mais difícil à medida que o número de asteroides aumenta (GAMER, 2009).

Para desenvolvimento deste trabalho foi utilizado a *game engine* Unity 3D para desenvolver um clone do jogo, não sendo codificado os discos voadores, somente os asteroides. Esse clone foi composto pelas seguintes classes:

- Asteroid: determina o comportamento dos asteroides e sua destruição criando asteroides menores;
- Bullet: gerencia a direção e o tempo de vida do projétil disparado;
- ShipController: controla a nave através da entrada de teclado e por meio do AE.
- EuclideanTorus: altera a posição dos objetos que alcançam o limite da tela;
- Extension: determina o tamanho da tela;
- GameManager: gerencia o estado do jogo, desde o início da partida, quantidade de asteroides e término do jogo.

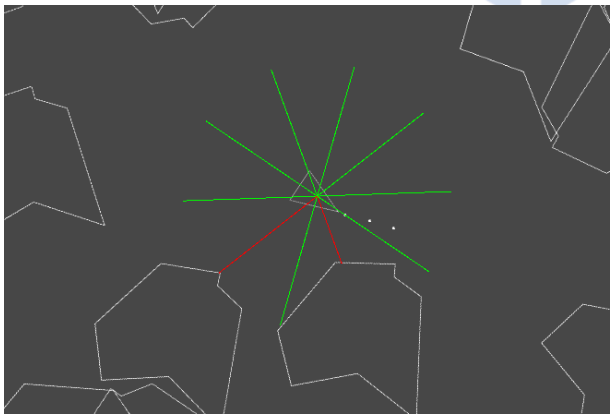
Figura 2- Captura de tela do jogo Asteroids original.

Fonte: Jogo Asteroids.

2. 1 Cromossomo e População inicial

O cromossomo projetado leva em consideração que a nave que deve ser controlada, tem 10 sensores igualmente espaçados (ângulo de 36º graus). Cada sensor é representado por uma linha de 2 unidades de distância na Unity 3D (Figura 3). Quando o sensor não detecta nenhum asteroide, seu valor é zero (0), quando ele detecta, seu valor é um (1). Sendo assim, pode-se representar essa combinação como se fosse um número binário (2^{10} = 1024).

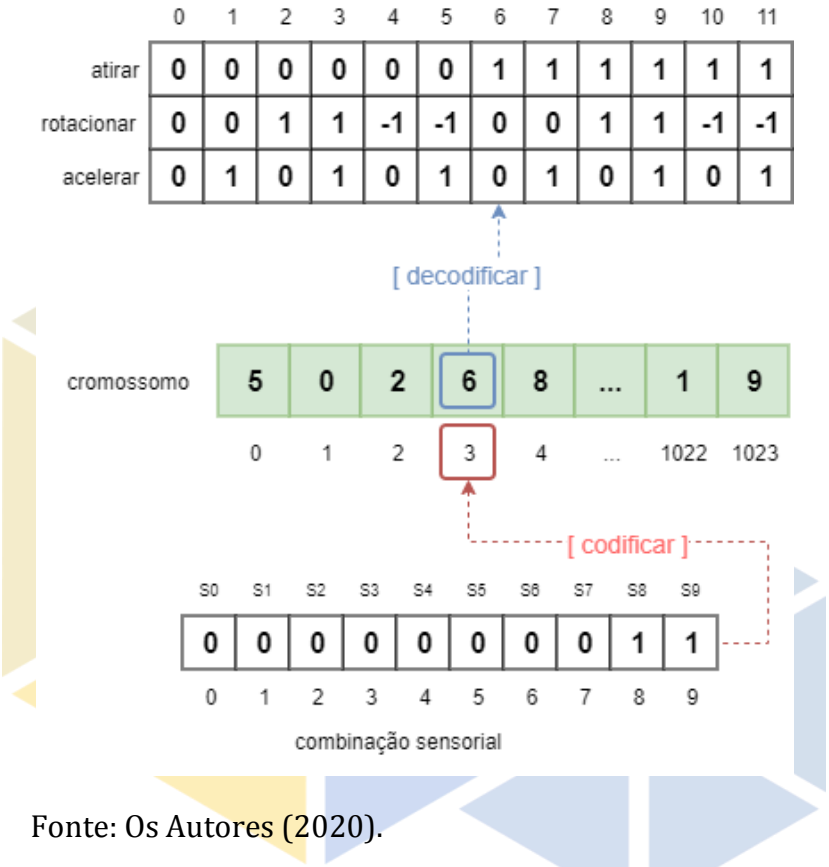
Figura 3- Posicionamento dos sensores na nave.



Fonte: Os Autores (2020).

A combinação sensorial foi utilizada para identificar o índice do cromossomo. Cada posição do cromossomo contém um número inteiro que varia de 0 a 11 - cada número representa uma posição da tabela de controle da nave: atirar (0 – não atira, 1 atira); rotacionar (-1 rotação para esquerda, 0 sem rotação, 1 rotação para direita); acelerar (0 – não acelera; 1 acelera). A Figura 4 ilustra de uma forma simples a explicação de como o cromossomo foi representado.

Figura 4 Cromossomo desenvolvido.



Fonte: Os Autores (2020).

Após definir o formato do cromossomo, é possível criar uma população inicial. Essa população será composta por indivíduos, com seus genes aleatórios dentro dos limites permitidos.

2.2 Função de avaliação

Um dos pontos importantes para o AE convergir, é criar uma boa função de avaliação. Como visto anteriormente, essa função avalia a proximidade de uma determinada solução com a solução ideal do problema desejado. Neste caso, o ideal para este projeto, pode ser, a nave que destruir mais asteroides.

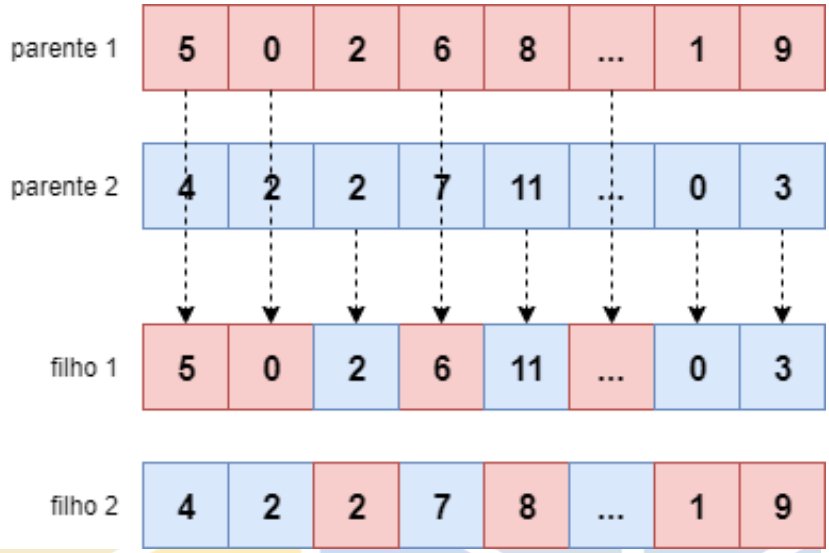
Por isso a pontuação é calculada pela quantidade de asteroides destruídos. Sendo assim, quanto maior a pontuação da nave, mais asteroides foram destruídos, então, melhor será a solução.

2.3 Operadores Codificados

O método de seleção utilizado é o por Torneio (EIBEN e SMITH, 2015), no qual, 2 indivíduos são escolhidos aleatoriamente e o que obtiver o melhor *fitness* é selecionado para reproduzir. Esse processo é realizado para selecionar 2 pais, e assim, recombinar suas informações.

Para recombinar o cromossomo, foi codificado o *crossover* uniforme (EIBEN e SMITH, 2015) com probabilidade de 50% de selecionar os genes de cada pai. A Figura 5 ilustra como essa recombinação pode gerar 2 novos cromossomos para a nova geração.

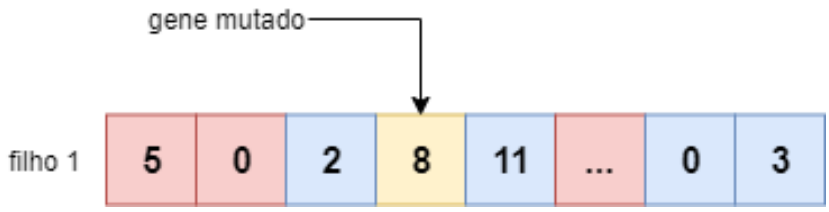
Figura 5 - Recombinação uniforme aplicada a um cromossomo.



Fonte: Os Autores (2020).

Logo após a recombinação, cada filho gerado tem 5% de chance de ter a informação de nenhum ou muitos genes excluídos, gerando um novo valor aleatório dentro dos limites permitidos (EIBEN e SMITH, 2015). A Figura 6 utiliza o filho 1 gerado no exemplo da Figura 7, para ilustrar a mutação no quarto gene.

Figura 6 Mutação aleatória aplicada a um cromossomo.



Fonte: Os Autores (2020).

3. Resultados

Esta seção apresenta os experimentos realizados para comprovar que o AE desenvolvido, consegue evoluir. Para isso, foi projetado inicialmente dois experimentos com três configurações de taxa de mutação, sendo elas 2%, 5% e 10%. Porém, ao final, realizou-se um terceiro experimento com as melhores configurações (mutação e elitismo) e com uma população quatro vezes maior.

3.1 Experimento I

O primeiro experimento foi configurado com uma população de 50 indivíduos, sendo que nenhum indivíduo foi copiado para a nova população (elite = 0). A taxa de cruzamento uniforme é de 50% para ambos os pais. O tempo de jogo é de 30 segundos para cada cromossomo, com critério de parada de 50 gerações, sendo assim, se todos indivíduos ficarem vivos os 30 segundos de jogo, o teste terá aproximadamente 20:50:00 horas. A semente aleatória é 17 - número usado para iniciar o algoritmo gerador de números pseudoaleatórios. A Tabela 1 exibe um resumo das configurações do primeiro experimento.

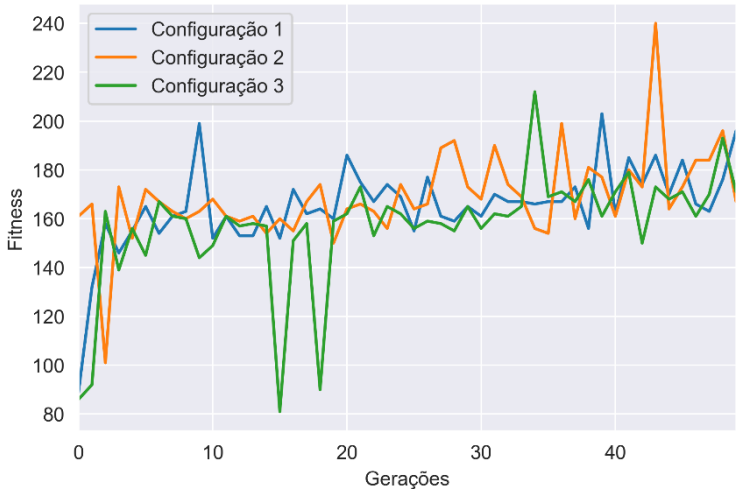
Tabela 2 Configuração inicial do primeiro experimento.

Variável	Valores
Semente aleatório	17
Tamanho da população	50 cromossomos
Elitismo	0%
Taxa de cruzamento	50%
Taxa de mutação	2%, 5% e 10%
Número máximo de gerações	50
Quantidade de torneio	2
Tempo máximo de jogo	30 segundos
Número inicial de asteroides	30
Aumento de asteroides	10

Fonte: Os Autores (2020).

O gráfico ilustrado na Figura 7, apresenta a evolução do melhor *fitness* para cada configuração do primeiro experimento. O tempo de teste para cada configuração respectivamente: 4:18:02 horas, 4:43:47 horas e 3:47:58 horas. Percebe-se que houve muitas variações nos *fitness* dos indivíduos, porém existe uma tendência de adaptação.

Figura 7 Gráfico da evolução do *fitness* da população ao longo das gerações no experimento I

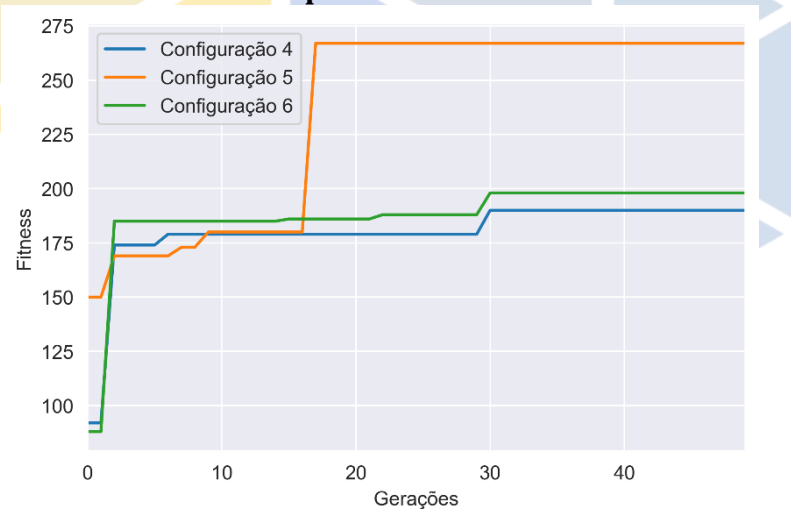


Fonte: Os Autores.

3.2 Experimento II

O segundo experimento consiste na mesma ideia do primeiro experimento, porém houve uma modificação no número de elite, passando para 20% de indivíduos copiados para a nova geração (os 10 melhores cromossomos). O gráfico ilustrado na Figura 8, apresenta a evolução do melhor *fitness* para cada configuração do segundo experimento.

Figura 8 Gráfico da evolução do *fitness* da população ao longo das gerações no experimento II



Fonte: Os Autores.

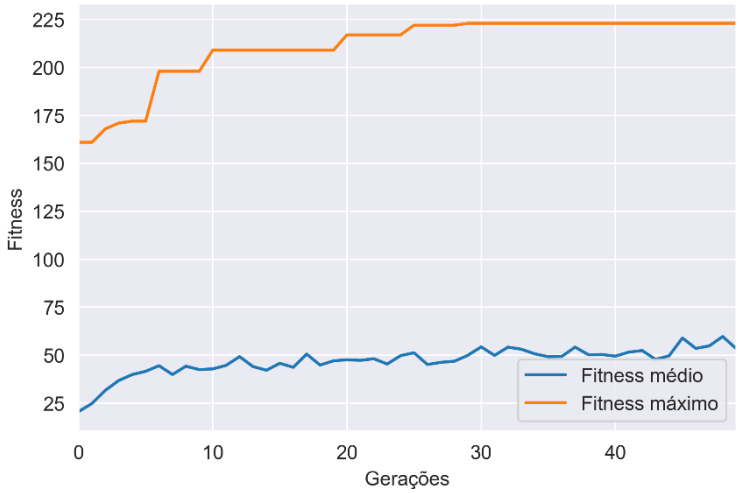
O tempo de teste para cada configuração foi de: 3:04:17 horas, 3:49:26 horas e 3:36:29 horas. Pode-se observar uma menor variação em relação ao Experimento I, porém, durante várias gerações não houve mudança no *fitness* máximo.

3.3 Experimento III

O terceiro experimento, consiste em utilizar 5% na taxa de mutação – valor que se mostrou melhor nos experimentos 1 e 2. Também será copiado os 10 melhores indivíduos para as novas gerações (elite = 5%), pois adicionando o elitismo comprovou-se que a população atinge melhores notas. Por fim, aumentou-se o tamanho da população para 200 indivíduos, pois a baixa variação inicial pode estar levando os cromossomos para alguns máximos locais.

O gráfico ilustrado na Figura 9, apresenta a evolução do melhor *fitness* e a média dos *fitness* ao longo das 50 gerações. O tempo de jogo foi de 16:24:25 horas e a nota se estabilizou na geração 30 – o mesmo aconteceu no experimento II.

Figura 9 Gráfico da evolução do *fitness* da população ao longo das gerações no experimento III.



Fonte: Os Autores

Conclusão

A proposta deste projeto, foi verificar a possibilidade de desenvolver um AE para controlar de forma autônoma a nave do jogo Asteroids. Para isso, inicialmente foi necessário desenvolver um clone do jogo na Unity 3D com a linguagem C#.

Logo em seguida, foi projetado um cromossomo para satisfazer as entradas e saídas de controle da nave. Assim, como todo AE, um outro ponto importante foi criar a função de avaliação (*fitness*) que define a nota do quão boa é a solução.

Por fim, submeteu-se o AE para controlar a nave, definindo de forma empírica uma configuração dos atributos. Ao final dos testes, foram obtidos resultados satisfatórios de controle, mostrando que o AE projetado consegue resolver o problema.

Como possíveis trabalhos futuros, pode ser citado: i) codificação e testes de novos operadores de *crossover*, mutação e seleção; ii) criação e testes de outras funções de avaliação, levando em consideração a precisão dos tiros, distância percorrida e etc.; iii) definição de uma bateria de testes utilizando uma combinação dos parâmetros e operadores; iv) refinar a modelagem, a fim de, obter uma evolução maior.

Referências

ANDRADE, K. O.; SILVA, A. E. A.; CROCOMO, M. **Um Algoritmo Evolutivo para a Adaptação de NPCs em um Jogo de Ação**. In: I Simpósio Santa Catarina Games, 2009, Florianópolis. Anais do I Simpósio Santa Catarina Games - SCGAMES, 2009.

ANDRADE, K. O.; JOAQUIM, R. C.; CAURIN, G. A.P.; CROCOMO, M. K. **Evolutionary algorithms for a better gaming experience in rehabilitation robotics**. Computers in Entertainment, vol. 16, no. 2, 2018.

ARORA A. **Using Genetic Algorithms to Automate the Chrome Dinosaur Game (Parte 2).** 2019. Disponível em <https://heartbeat.fritz.ai/using-genetic-algorithms-to-automate-the-chrome-dinosaur-game-part-2-1c0007334297>

DE JONG, K. **Evolutionary computation: A unified approach**. 2012. GECCO'12 - Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Companion [...]. [*S. l.*: *s. n.*], 2012. p. 737–750.

EIBEN, A E; SMITH, J. E. **Introduction to Evolutionary Computing**. 2nd ed. [*S. l.*]: Springer Publishing Company, Incorporated, 2015.

GAMER, R. **The Making of Asteroids**. Imagine Publishing, p. 24–29. 2009.

GARCIA, B. E.R.; CROCOMO, M. K.; ANDRADE, K. O. **Dynamic Difficulty Adjustment in a Whac-a-Mole like Game**. Brazilian Symposium on Games and Digital Entertainment, SBGAMES, vol. 2018-November, p. 88–96, 2018.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems: {A}n Introductory Analysis with Applications to Biology, Control and Artificial Intelligence**. Ann Arbor, MI: University of Michigan Press, 1975.

KUHN, K. J.; FOLEY, R. P. **A Comparison of Genetic Algorithms using Super Mario Bros**. 2015. Disponível em <https://digitalcommons.wpi.edu/mqp-all/3737>

MARTIN, M. **Using a Genetic algorithm to Create Adaptive Enemy AI**. Gamasutra – The Art & Business of Making Game. 2011. Disponível em <https://www.gamasutra.com/blogs/MichaelMartin/20110830/90109/Using_a_Genetic_Algorithm_to_Creat>.

SODER, A. **Avaliação da estratégia de aprendizado NEAT aplicada a um jogo eletrônico de console 8 Bits**. Monografia (Graduação em Engenharia da Computação) – Universidade do Vale do Taquari - Univates, Lajeado, 10 jul. 2018. Disponível em: <http://hdl.handle.net/10737/2183>.