Algoritmo Genético ou Pessoas: quem consegue terminar a fase primeiro?

Rafael Monteiro Zancanaro[1], **Ayslan Trevizan Possebom**[2]

[1]Departamento de Informática (DIN) - Universidade Estadual de Maringá (UEM)
Av. Colombo, 5790, UEM, Bloco C56. CEP 87020-900 – Maringá, PR – Brasil.

[2]Instituto Federal do Paraná Campus Paranavaí (IFPR)
Rua José F. Tequinha, 1400, Jardim das Nações. CEP 87703-536 – Paranavaí, PR - Brasil.

rafaelmonte46@gmail.com, ayslan.possebom@ifpr.edu.br

Abstract. *Genetic algorithms are part of Artificial Intelligence for learning using information search rules. This article deals with the use of one of them, called NEAT (NeuroEvolution of Augmenting Topologies), so that the algorithm can control the character of the game Super Mario World quickly and automatically to successfully complete a stage of the game. The results obtained were compared with the results of a group of human players, using the best times in the same stage in which the algorithm was executed. After collecting the data, they were compared and, finally, the results indicate a better time performance by the algorithm at the end of a game stage.*

Resumo. *Algoritmos genéticos fazem parte da Inteligência Artificial para o aprendizado utilizando regras de busca por informações. Este artigo trata do uso de um deles, chamado NEAT (NeuroEvolution of Augmenting Topologies), de forma que o algoritmo possa controlar o personagem do jogo Super Mario World de maneira rápida e automática para concluir com sucesso uma fase do jogo. Os resultados obtidos foram comparados com os de um grupo de jogadores humanos, utilizando dos melhores tempos na mesma fase em que o algoritmo foi executado. Após a coleta das informações, os dados foram comparados e, por fim, os resultados indicam um melhor desempenho de tempo por parte do algoritmo ao finalizar uma fase do jogo.*

1. Introdução

Com o avanço da tecnologia e dos sistemas de informação, surgem novas formas de se otimizar o trabalho que normalmente seria realizado por pessoas comuns. Uma dessas maneiras é a utilização de algoritmos de Inteligência Artificial (IA). O conceito de Inteligência Artificial surgiu há muito tempo, e possui muitos ramos na área de psicologia, filosofia e biologia, tendo como principal objetivo o estudo de "[...] sistemas que agem de um modo que a um observador qualquer pareça ser inteligente" [COPPIN 2015, p. 4]. Porém, a estrutura e as ações de uma IA dependem também do contexto que serão aplicadas, ou seja, os meios que ela utilizará para resolver um problema dependem da complexidade deste problema. Assim como os seres humanos, os conjuntos de meios para se resolver problemas leva a uma segunda definição de IA. Segundo [COPPIN 2015, p. 4]: "Inteligência Artificial envolve utilizar métodos baseados no comportamento inteligente de humanos e outros animais para solucionar problemas complexos".

Atualmente, a utilização desse tipo de tecnologia é visível em diversas áreas da nossa sociedade, como análises científica e financeira (aplicável na saúde e economia), em tarefas de comunicação entre máquina e humano, de inferência lógica e até em reconhecimento de ambientes para se movimentar (robótica) [ROSA 2011, p. 4-5].

Uma das formas de se realizar tarefas e fazer com que a máquina adquira "Inteligência" é por meio das metodologias de busca que visam encontrar um resultado da melhor forma possível, alterando ou descartando possíveis soluções de um problema até que uma solução ótima seja identificada [PETRY 2003, p. 22]. Dentre esses métodos de busca estão os Algoritmos Genéticos, que são considerados métodos de busca aleatória direcionada. Os Algoritmos Genéticos (AG) são modelos computacionais que têm como inspiração os modelos de evolução das espécies. Os cromossomos são representados por cadeias de bits aleatórios e são recombinados a partir de uma população inicial, gerando indivíduos cada vez mais adaptados, assim como no cruzamento das espécies [WHITLEY 1994, p. 65].

Tendo em mente os métodos de busca como os algoritmos genéticos e a utilização da IA nos jogos como um agente passivo, sendo necessária apenas para distrair o jogador, seria possível analisar a "inversão" desse papel da Inteligência Artificial na indústria dos jogos [KISHIMOTO 2004, p. 2]. Neste contexto, este trabalho tem como objetivo analisar o processo de "aprendizagem de máquinas" utilizando um AG e comparar os resultados obtidos pelo algoritmo com os resultados obtidos por pessoas comuns, marcando o tempo de cada pessoa para concluir uma fase de um jogo e decidir se a IA pode resolver problemas de maneira tão competente quanto um humano. Com isso, pretende-se: entender os conceitos relacionados aos Algoritmos Genéticos e sua aplicação em um jogo "Super Mario World" na fase "Yoshi's Island 2"; tabular os resultados obtidos pelo algoritmo; e comparar os resultados com o desempenho obtido por jogadores humanos. Desta forma, podemos responder às perguntas de pesquisas: "um algoritmo genético é capaz de aprender a conduzir um personagem de um determinado jogo para finalização de uma fase de forma semelhante a jogadores humanos?" e "O desempenho do algoritmo genético é equivalente ao desempenho de jogadores humanos?"

Para tanto, serão escolhidas 10 pessoas de diferentes idades e experiências. Cada jogador poderá treinar o jogo por, no máximo, o tempo necessário ao treinamento do algoritmo genético. Os tempos de treinamento e de jogo para o jogador concluir a fase serão registrados para serem comparados com os tempos utilizados pelo algoritmo. Um emulador de jogos para computadores será utilizado para a execução do jogo. Os jogadores utilizarão o mesmo emulador e jogo em que o algoritmo genético será executado. Os resultados obtidos pelos jogadores envolverão: idade do jogador, nível de experiência com videogames, sistema operacional utilizado, horário de início e fim do treinamento e da jogada e pontuação obtida ao fim da fase.

Este trabalho está estruturado da seguinte forma. A seção 2 descreve a fundamentação teórica abordando os conceitos envolvidos nos algoritmos genéticos, NeuroEvolution of Augmenting Topologies (ou N.E.A.T) e ambiente para a execução do jogo. A seção 3 apresenta o experimento realizado. Os resultados são apresentados na seção 4. Trabalhos relacionados são tratados na seção 5. Por fim, a seção 6 apresenta as considerações finais e trabalhos futuros.

2. Fundamentação Teórica

Essa seção descreve o arcabouço teórico utilizado para a elaboração deste artigo. A base para o experimento utiliza um algoritmo genético, chamado de N.E.A.T (NeuroEvolution of Augmenting Topologies), que é responsável pelo aprendizado da evolução do personagem durante a fase. Além disso, um AG utiliza uma série de terminologias, que também serão apresentadas no decorrer desta seção.

2.1. Algoritmos Genéticos

Para que a máquina consiga movimentar o personagem do jogo, foi necessária uma abordagem que permitisse a ela entender o que deveria ser feito e como deveria ser feito. Para isso, utilizou-se um método de busca local chamado algoritmo genético (AG), para permitir que a máquina consiga "jogar" o *game*.

Métodos de busca são abordagens feitas no ramo da Inteligência Artificial que permitem a um agente atingir um objetivo por meio de um *espaço de busca*, que por sua vez é o nome dado ao conjunto de soluções possíveis para esse problema [COPPIN 2015, p. 62]. Portanto, de maneira genérica, um método de busca se baseia numa máquina procurando possíveis soluções para um problema, até que uma adequada seja encontrada. Os métodos de busca local, por sua vez, são soluções mais "sofisticadas", que geralmente partem de um ponto que é aleatório, e, por meio de ajustes na sua configuração inicial, vão testando as possíveis resoluções até encontrar uma que não pode ser melhorada, "peneirando" as soluções possíveis até que só reste a melhor [COPPIN 2015, p. 110].

O Algoritmo Genético busca utilizar nomenclaturas e conceitos da seleção natural e da genética para permitir que um agente consiga concluir sua tarefa, no caso, a máquina que irá concluir uma fase. A ideia por trás desse método de busca é, com base em cruzamentos de uma série inicial de soluções de um problema, gerar soluções melhores, que por sua vez vão cruzar entre si, até o momento em que exista uma solução que não pode ser melhorada, sendo assim a solução mais *apta* [MIRJALILI 2019, p. 66].

A execução de um algoritmo genético sempre começa com uma população inicial de membros, que são *strings* binárias (podem assumir apenas valores 0 ou 1) de um tamanho pré estabelecido. Usualmente essas strings, também chamadas de cromossomo ou genótipo, são geradas aleatoriamente e tem a elas associado um valor de aptidão (ou *fitness*). A aptidão de um indivíduo representa a sua propensão de reproduzir e gerar boas proles na biologia. Dentro do AG, essa característica é assumida por um valor que representa a probabilidade de um cromossomo em específico poder passar boas características para a próxima geração. Para [MIRJALILI 2019, p.ç 44], "Na natureza, os indivíduos mais aptos têm uma maior chance de conseguir comida e se acasalar. Isso implica em seus genes contribuírem mais na produção da próxima geração das mesmas espécies. Inspirado nessa simples ideia, o Algoritmo Genético emprega uma roleta para atribuir probabilidades para os indivíduos e selecioná-los para criar a próxima geração de maneira proporcional ao seu valor fitness (ou aptidão)".

Tendo essa população base estabelecida, o algoritmo inicia a criação de uma população intermediária, que é formada por um processo de *seleção*, que duplica partes específicas dos cromossomos e, após isso, esses genes são recombinados por meio do processo de *crossover*. A Figura 1 representa como os processos de duplicação e crossover acontecem, sendo os retângulos as gerações atual, intermediária e a próxima geração,

respectivamente, com o termo *offspring* representando as novas *strings* que surgirão após o *crossover*.

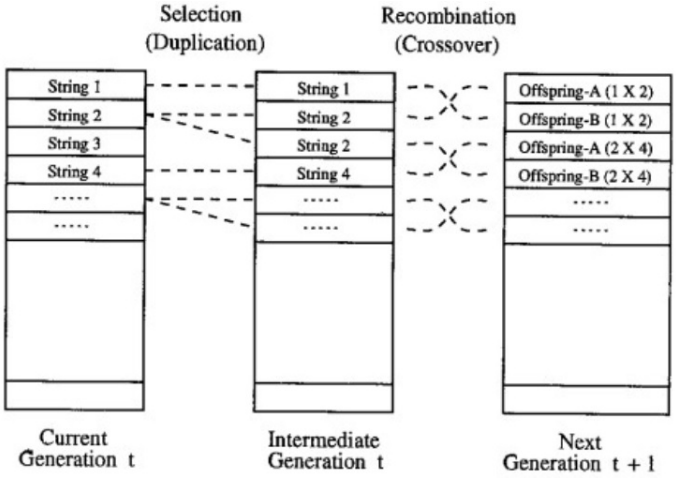

Figura 1. Funcionamento de um algoritmo genético básico [WHITLEY 1994, p. 67]

O processo de crossover propriamente dito não representa a recombinação de genes, mas na verdade a "quebra" dos vetores de strings que representam os cromossomos. Segundo [WANG 2003, p. 102-103], esse processo é o responsável pela criação de diversidade entre as populações pelas gerações. Esses vetores podem ser quebrados em lugares aleatórios e, em implementações reais, nem todos precisam sofrer esse processo, já que existe um valor pré-estabelecido que controla essas quebras de cromossomos, chamado de *crossover rate* (P_c), que está em um intervalo de 0 a 1, sendo que, quanto maior o valor de P_c, mais *crossovers* irão ocorrer na população.

Essa recombinação dos genes é feita por um mecanismo estocástico chamado roleta (do inglês *roulette wheel*). Nesse mecanismo, as escolhas que formam a próxima geração de indivíduos são baseadas nas suas aptidões. Portanto, de maneira semelhante à natureza, o indivíduo com o maior valor *fitness* (ou mais apto) possui maiores chances de seguir adiante, e, mesmo que algum indivíduo com baixa aptidão consiga avançar, a probabilidade que ele continue dentro das próximas gerações fica cada vez menor.

Também podem ser aplicados outros atores após a recombinação dos cromossomos, tais como as mutações nos genes que geralmente são equilibradas para ter de 0 a 1% de probabilidade de ocorrer. A recombinação de cromossomos tem a função de alterar ainda mais as características de um gene, seja por meio da criação de novos bits na *string* que o constitui, quanto pela inversão de alguns bits do gene, sendo o primeiro caso a maior parte (cerca de 50%) das causas das mutações [WHITLEY 1994, p. 67].

Após esse ciclo que envolve a geração de uma população, duplicação dos cromossomos, recombinação e aplicação de mutações, o ciclo se reinicia, até o momento onde a geração com indivíduos que não podem ser mais aptos seja alcançada [COPPIN 2015, p. 335-336].

2.2. N.E.A.T.

O NeuroEvolution of Augmenting Topologies (N.E.A.T) é um algoritmo genético que trabalha com o conceito de Redes Neurais Artificiais (RNA). As RNA são conjuntos de

neurônios artificiais que tentam simular as funções cognitivas humanas, ou seja, por meio de mapeamento dos componentes que formam o cérebro, se torna possível criar estruturas chamadas de neurônios artificiais, e com isso resolver determinados problemas cognitivos [RAUBER 2005, p. 3].

Um neurônio artificial é o menor componente de uma RNA, e, buscando simular o neurônio real e suas estruturas, como mostra a Figura 2a. Ele possui elementos com funções análogas ao neurônio biológico, como os dendritos (recebem informações em forma de pequenas cargas elétricas), núcleo (função que ativa o neurônio) e axônios (passam os pulsos elétricos adiante). No modelo de McCulloch e Pitts criado em 1943, os dados são recebidos nas entradas, em seguida somados e, por fim, dependendo do valor de um cálculo chamado de função de ativação, os dados passam adiante pela rede com valores 0 ou 1, conforme representado na Figura 2b.

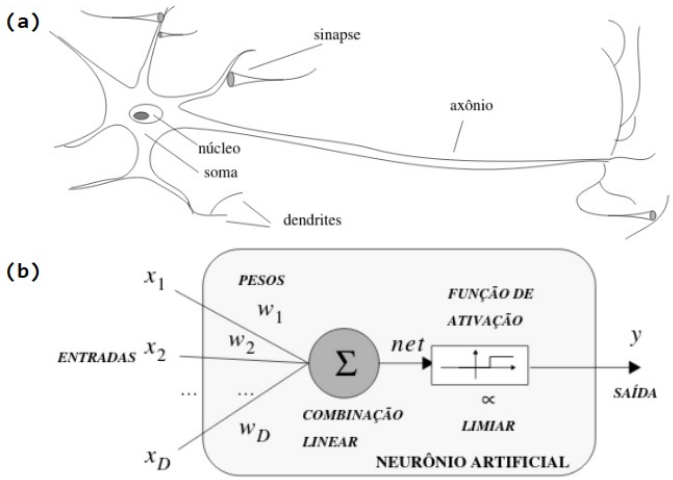

Figura 2. (a)Representação de neurônio biológico. (b)Representação de neurônio artificial [RAUBER 2005, p. 04]

O funcionamento básico de uma RNA é dividido em 3 camadas (ou nós) de neurônios artificiais, a primeira chamada de *camada de entrada*, onde as informações chegam e são repassadas para toda rede, em seguida passam pelas *camadas ocultas*, que podem ser de uma até três camadas de neurônios, e por fim a *camada de saída* que é onde as informações processadas pela rede saem [WANG 2003, p. 81]. Cada uma das ligações entre os neurônios das camadas são associadas a um valor numérico chamado *peso*. A Figura 3 mostra uma rede neural com suas camadas e conexões (representadas por uma linha reta), sendo os círculos inferiores, no meio e superiores os neurônios da camada de entrada, oculta e de saída, respectivamente.

O NEAT utiliza as redes neurais juntamente aos AGs vistos na seção anterior, de maneira que, ao invés de cruzar cadeias de bits, ele cruza estruturas de redes neurais para solucionar um problema, com suas conexões e camadas, até que uma topologia ideal para a resolução daquele problema seja resolvida. A Figura 4 apresenta um exemplo de como o NEAT é representado em uma rede neural. Os pais são representados na parte superior (chamados de *Parent1* e *Parent2*) e que cruzam suas estruturas, gerando uma nova topologia (chamada de *offspring*, representada na região inferior da imagem), com cada círculo representando um neurônio e cada linha representando uma conexão.

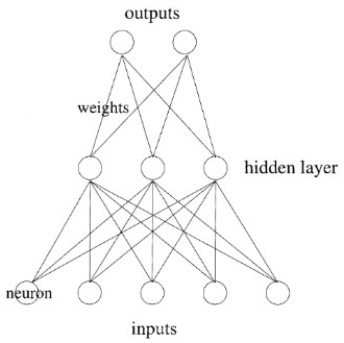

Figura 3. Representação de uma RNA [WANG 2003, p. 82]

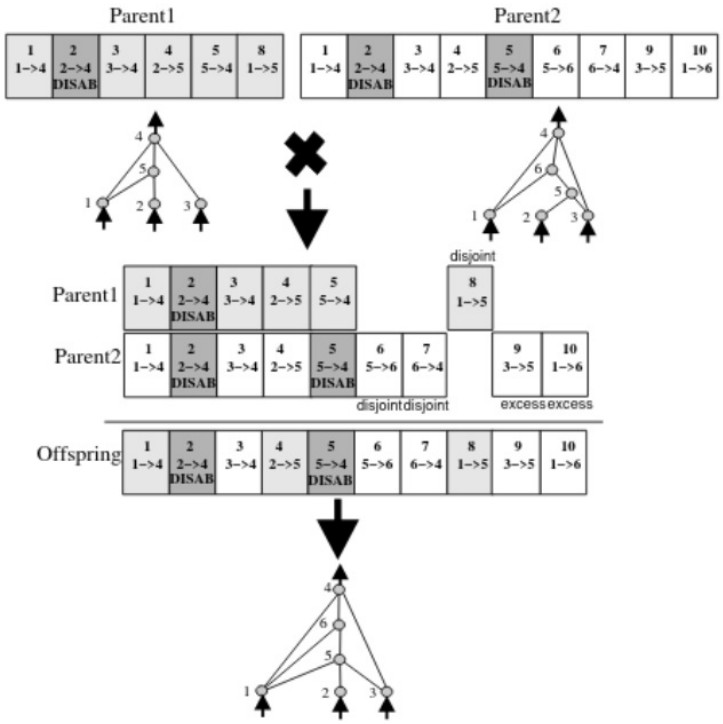

Figura 4. Exemplo de NEAT em Rede Neural.

Cada um dos retângulos acima do desenho das topologias representa uma conexão, por exemplo: a conexão 1-4 representa a conexão entre os neurônios 1 e 4 da topologia, e as conexões que estão em cinza com o texto "DISAB" são as conexões que foram desabilitadas, portanto, não tem função na rede. Ao duas topologias cruzarem (representado pelo meio da imagem), existe a chance de ocorrerem mutações na *Offspring*: como os *disjoints*, que são conexões novas criadas a partir do cruzamento, e os *excesses*, que são nós em excesso, já que as redes cruzadas podem ter diferenças no número de conexões e neurônios.

2.3. Emulador de Jogos

O ambiente utilizado para a máquina jogar o game foi um emulador de jogos, que é um software que busca simular uma plataforma em específico, com as suas funções e rotinas e executando os seus jogos, que são arquivos salvos em formato ROM. De acordo

com [FileInfo 2022], os arquivos *.rom* (*Read Only Memory Image*) carregam os exatos dados somente-leitura de um chip ou dispositivo, com o fim de poder emulá-los. No caso desse projeto, esse arquivo busca simular os dados que seriam encontrados dentro do cartucho do jogo *Super Mario World*[1] da plataforma SNES (Super Nintendo Entertainment System), que por sua vez é simulada pelo emulador BizHawk[2].

O emulador de jogos BizHawk, além de poder simular um SNES, permite utilizar *scripts* da linguagem de programação Lua dentro da ROM em tempo de execução, o que é essencial para que o NEAT funcionasse. A linguagem de programação Lua, desenvolvida por brasileiros, é tipada, de marcação dinâmica e baseada em arrays associativos e pode ser tanto orientada ao objeto como procedural, sendo assim bastante flexível e permitindo-a ser utilizada em softwares mais sofisticados, como games [PUC-Rio 2022].

3. Experimentação

Esta seção trata dos procedimentos adotados para a aplicação do experimento feito com o AG e com os participantes, ou seja, apresenta os materiais separados para o ambiente de execução do algoritmo, a maneira com que ele foi aplicado e também as regras delimitadas para os participantes para que a coleta dos dados fosse realizada e comparada. Para a execução do experimento, foi utilizado um computador notebook com as seguintes características de hardware e software:

- Notebook Acer Aspire 8, contendo 250 GB de espaço de armazenamento em Solid State Drive (SSD)
- Memória Random Access Memory (RAM) de 12 GB DDR4;
- Processador Intel Core i5-7200U
- Windows 10 Home Edition;
- Emulador BizHawk;
- ROM do jogo *Super Mario World* (USA);
- Algoritmo Genético[3].

Para executar o ambiente, precisamos do emulador e da implementação do algoritmo genético. Para configurar o algoritmo genético no emulador, primeiramente é necessário abrir a ROM por meio do menu File ¿ Open ROM. Com o jogo aberto, é necessário inserir o script Lua. Para tanto, isso deve ser feito a partir do menu Tools ¿ Lua Console ¿ Open script, e inserir o arquivo com o algoritmo genético.

Durante a execução do jogo com o algoritmo, é possível observar algumas informações sobre o aprendizado de máquinas, como mostrado na Figura 5. Essas informações são: o *max fitness* (que é a maior aptidão alcançada pelas gerações até o momento), o *fitness* (que é a aptidão atual), a geração (Gen), o genoma (*genome*) e a espécie (*species*). Após a execução do algoritmo genético e da finalização com sucesso da fase *Yoshi's Island 2*, pôde-se obter o tempo necessário gasto para que a fase fosse concluída de forma automática por meio do algoritmo genético e, a partir deste momento, pode-se comparar o desempenho com os humanos.

Os jogadores participantes do experimento utilizaram o mesmo emulador e ROM, entretanto, sem a presença do algoritmo genético. Foi feito a anotação da hora atual antes

[1] A ROM utilizada foi obtida em: `http://encurtador.com.br/jqC67`.
[2] O emulador pode ser baixado em: `https://github.com/TASVideos/BizHawk`
[3] O Algoritmo Genético pode ser encontrado em: `https://pastebin.com/ZZmSNaHX`

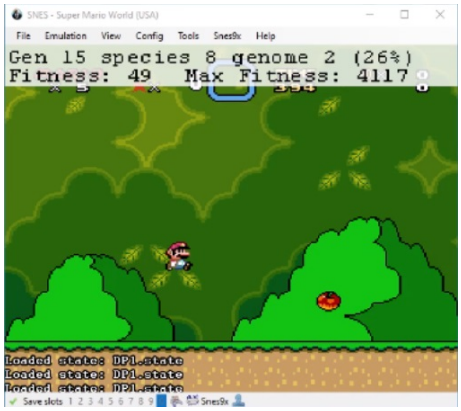

Figura 5. Algoritmo Genético em execução.

do início do jogo e a hora atual após a conclusão da fase para poder medir o tempo que foi necessário para se jogar a fase do jogo, bem como a pontuação obtida. Os participantes foram determinados por meio do preenchimento de um formulário de inscrição, onde foram inseridos os seguintes dados: Endereço de Email; Idade; Conhecimento sobre vídeo-games, sendo um conjunto de opções de 0 à 5, com 0 significando "Desconheço" e 5 "Conheço"; Já jogou algum jogo de plataforma 2D (sendo as respostas "sim" ou "não"); Familiaridade com jogos de plataforma 2D sendo um conjunto de opções de 0 à 5, com 0 significando "Não conheço" e 5 "Familiarizado"; Sistema operacional utilizado. Após preencherem o formulário, os participantes receberam um email com as orientações sobre suas tentativas, onde os participantes foram instruídos a fazer um treino antes de enviar a tentativa final, na qual os dados são registrados e comparados com o AG. O treino deveria ter uma extensão de, no máximo, 9 horas e o tempo inicial e final de treino deveriam ser mandados no email, juntamente ao tempo final das tentativas.

4. Resultados

Os resultados dos participantes foram separados de acordo com as idades. A Figura 6 mostra um gráfico das idades, que possibilita visualizar como elas variaram no experimento. Foram, ao total, 8 participantes, sendo um para as idades de 13, 14, 15, 19, 40 e 45 anos, e dois com 18 anos.

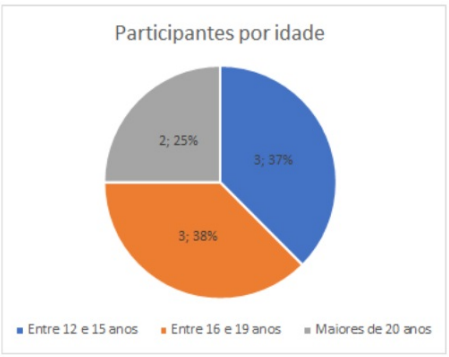

Figura 6. Dispersão das idades dos participantes.

Os gráficos de linhas apresentados na Figuras 7 mostram, respectivamente, o conhecimento dos participantes sobre vídeo-games (a) e a familiaridade deles sobre jogos

de plataforma 2D (b). As perguntas deveriam ter respostas que variam de 0 à 5, sendo 0 nenhum conhecimento sobre, e 5 completamente familiarizado.

Figura 7. (a)Conhecimento dos participantes sobre vídeo-games. (b) Familiaridade dos participantes com jogos em 2D.

Os dados sobre as melhores tentativas de cada participante também foram registradas e diagramadas. A Figura 8 mostra um gráfico com o melhor tempo, em segundos, de cada participante para passar a fase.

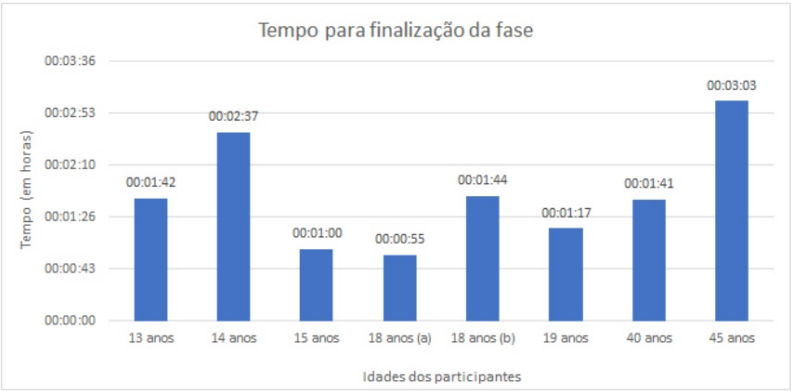

Figura 8. Tempo de cada participante para conclusão da fase (em segundos).

Ao longo de seu treinamento, o algoritmo genético também teve os dados de treinamento anotados. A Figura 9 apresenta um gráfico que mostra o *Max Fitness* (ou maior aptidão) adquirida nas datas em que foi treinado. No primeiro dia de treinamento, foi obtido um *Max Fitness* de 1645. Nos dias seguintes de treinamento, o algoritmo foi restaurado no ponto em que havia sido interrompido para continuar seu aprendizado. Quando

o algoritmo atingiu a aptidão máxima, que é o menor valor necessário para o personagem concluir a fase, o processo de treinamento levou um total de 8,5 horas (510 minutos).

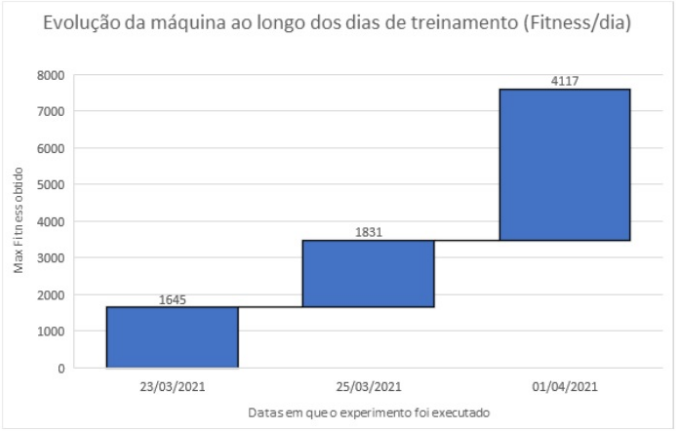

Figura 9. Maior aptidão do AG em suas datas de treinamento.

Após coletados todos os dados, foi possível comparar os desempenhos dos participantes em relação ao algoritmo genético. A Figura 10 mostra um gráfico comparando a média dos tempos dos participantes com o tempo do AG para terminar a fase, sendo que, no final de seu treinamento, ele completou-a em 47 segundos.

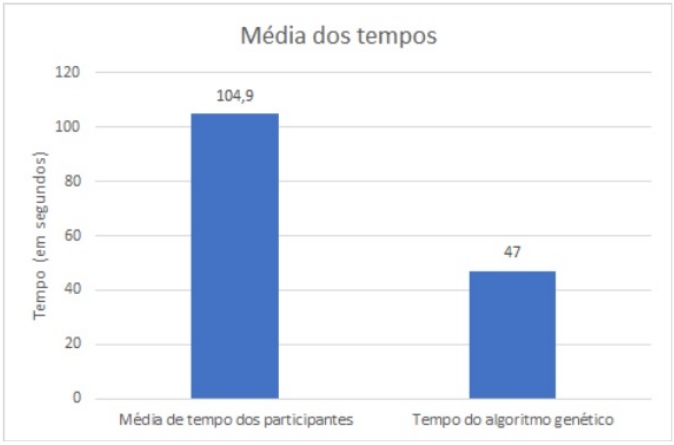

Figura 10. Comparação da média dos tempos dos participantes e do AG.

Por meio do experimento, pôde-se observar que jogadores com idades mais avançadas tendem a ter menos conhecimentos sobre video-games. Isto pode ser observado na Figura 7a. Ainda neste mesmo sentido, a Figura 7b demonstra que pessoas mais novas possuem menos conhecimentos sobre jogos que utilizam a plataforma 2D. Um dos motivos para isso se deve ao fato de que este perfil de jogador já utiliza em seu cotidiano jogos em smartphones e videogames mais atuais, onde jogos 3D são mais populares e permitem melhores movimentos aos personagens.

Na Figura 8 observa-se que adolescentes e jovens tendem a dominar melhor a movimentação do personagem durante os jogos. Isso reflete em um menor intervalo de tempo para a conclusão da fase do jogo em estudo.

Embora o treinamento do algoritmo genético tenha utilizado um tempo grande de oito horas e trinta minutos (510 minutos), a partir do treinamento, o personagem consegue concluir a fase em um tempo estimado de 47 segundos, ou seja, um valor bem inferior do que a média dos tempos dos participantes (105 segundos) e ainda um tempo inferior do que o tempo utilizado pelo melhor jogador (55 segundos). Desta forma, apesar do algoritmo levar um tempo maior de aprendizagem, após o aprendizado ser efetuado, ele consegue desempenhar sua tarefa de forma mais satisfatória.

5. Trabalhos Correlatos

Existem outros trabalhos que utilizam e tratam de algoritmos genéticos. [ARAÚJO and SILVA 2021] utiliza o algoritmo genético NEAT no jogo Super Mario World, porém não tem como objetivo comparar a performance de um grupo de participantes com a do algoritmo genético, mas sim apresentar os conceitos do AG no jogo e mostrar o seu desempenho ao longo das gerações.

[BORIS and GORAN 2016] também utilizam o NEAT, só que ao invés de jogar Super Mario World, o AG joga o game “2048”. Também são apresentados os dados da evolução do algoritmo genético pelas gerações, porém não busca comparar a rapidez, mas a quantidade de pontos de que ele pode fazer no jogo.

Já [SILVEIRA and BARONE 1998] busca outra abordagem para os AGs, diferentemente deste artigo, que utiliza o AG em um âmbito comparativo com a performance de uma série de participantes, o autor visa mostrar os algoritmos genéticos como uma forma de auxiliar no desenvolvimento da informática na área da educação, sendo o AG utilizado para jogar o jogo do pinguim e da abelha e por fim, é proposto um jogo de labirinto onde o usuário criaria os mapas e o algoritmo deveria encontrar a saída.

6. Conclusão

Este trabalho apresentou um estudo comparativo das capacidades de um algoritmo genético para aprender e passar uma fase de um jogo eletrônico em comparação com um grupo de participantes com idades e conhecimentos diversos sobre video-games e jogos de plataforma 2D, onde ambos deveriam treinar por um tempo de no máximo 9 horas e marcar os menores tempos, com a finalidade de apresentar quem consegue passar a fase de maneira mais rápida.

Dessa forma, com os dados coletados e o resultado do experimento, é possível concluir que o algoritmo genético, mesmo demorando muito mais tempo para aprender a jogar o game, consegue realizar com muito mais êxito a tarefa de passar a fase, já que o seu tempo ficou abaixo do melhor participante do experimento (sendo 47 segundos o tempo do AG e 55 segundos o tempo do melhor participante). Também é possível concluir que pessoas mais velhas tendem a ter menos conhecimento sobre video-games, assim como pessoas mais novas tendem a dominar menos os jogos de plataforma 2D.

Como trabalho futuro, pretende-se fazer uma avaliação com uma quantia maior de jogadores para medir com maior exatidão as aptidões e tempos necessários por cada faixa etária. Além disso, pode-se utilizar um tempo maior de treinamento para o algoritmo genético na tentativa de redução do tempo de jogo ao finalizar uma fase em específico. Por fim, é interessante fazer uma avaliação quanto à pontuação obtida pelos jogadores

e pelo algoritmo genético que, quando aplicado alguma métrica de eficiência, pode-se utilizar o tempo necessário para conclusão da fase juntamente com a pontuação obtida ao determinar quem obteve o melhor resultado.

Referências

ARAÚJO, A. d. C. and SILVA, A. A. (2021). Usando neuroevolution of augmenting topologies (neat) para jogar super mario world. `https://nca.ufma.br/~geraldo/vc/20171/trab/How_to_train_your_Mario.pdf`. Acessado em: 2022-10-09.

BORIS, T. and GORAN, Š. (2016). Evolving neural network to play game 2048. In *2016 24th Telecommunications Forum (TELFOR)*, pages 1–3. IEEE.

COPPIN, B. (2015). *Inteligência artificial*. Grupo Gen-LTC.

FileInfo (2022). .rom file extension. `https://fileinfo.com/extension/rom`. Acessado em: 2022-10-09.

KISHIMOTO, A. (2004). Inteligência artificial em jogos eletrônicos. *Academic research about Artificial Intelligence for games*.

MIRJALILI, S. (2019). Evolutionary algorithms and neural networks. In *Studies in computational intelligence*, volume 780. Springer.

PETRY, C. D. (2003). *Auxiliando o aprendizado de métodos de busca utilizando simulaÃ§ão*. PhD thesis, Dissertação (Mestrado). UFSC, Programa de Pós-Graduação em Ciência da Computação.

PUC-Rio (2022). The programming language lua. `http://www.lua.org/`. Acessado em: 2022-10-09.

RAUBER, T. W. (2005). Redes neurais artificiais. Universidade Federal do Espírito Santo. `http://encurtador.com.br/bvKNV`. Acessado em: 2022-10-09.

ROSA, J. L. G. (2011). Fundamentos da inteligência artificial. *Rio de Janeiro: LTC*, 1.

SILVEIRA, R. S. and BARONE, D. A. C. (1998). Jogos educativos computadorizados utilizando a abordagem de algoritmos genéticos. *Universidade Federal do Rio Grande do Sul. Instituto de Informática. Curso de Pós-Graduação em Ciências da Computação*.

WANG, S.-C. (2003). Genetic algorithm. In *Interdisciplinary computing in Java programming*, volume 743 of *The Springer International Series in Engineering and Computer Science*. Springer Science & Business Media.

WHITLEY, D. (1994). A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85.