IA para jogar Flappy Bird utilizando o método NEAT

1st Halyson Ricardo Rabuski Junior
Instituto Federal Catarinense
Campus Videira
Videira, Brasil
rabuskihalyson@gmail.com

***Resumo*—Este artigo aborda a cosntrução de uma inteligência artificial com a finalidade de jogar o jogo Flappy Bird de forma perfeita, ou seja, não perder até que o jogo seja encerrado. São abordados conceitos como aprendizagem por reforço, algoritmos genéticos, redes neurais e neuroevolução.**

***Palavras-chave*—Inteligência Artificial, RNAs, AGs, NEAT, Flappy Birds**

I. Inteligência Artificial

O termo Inteligência Artificial (IA) já existe há algum tempo, mas vem ganhando força nas últimas décadas com os avanços da tecnologia. Em resumo, o conceito de IA refere-se à capacidade de máquinas analisarem dados e aprenderem por conta própria a tomar decisões a fim de realizar uma tarefa da melhor maneira possível. O uso dessa tecnologia vem ganhando força pois permite aumentar a produtividade, maximizar lucros e minimizar prejuízos em diversas áreas. Ela está presente em diversas tarefas do dia-a-dia mesmo que não de forma perceptível ao usuário, por exemplo, utilizamos IA para calcular a logística de entregas, ou quando fazemos uma pesquisa no Google e uma IA analisa os dados colhidos do usuário retornando resultados próximos do padrão de interesse do mesmo, ou no caso mais comum que seriam os sistemas de recomendação da Netflix e do YouTube por exemplo, que recomendam filmes e vídeos de acordo com as preferências do usuário [1].

II. Inteligência Artificial jogando jogos

A. Xadrez

Em 1996 aconteceu a primeira disputa entre a IA Deep Blue da IBM e Garry Kasparov, campeão mundial de xadrez desde 1985. A IA foi capaz de vencer a primeira partida, sendo a primeira vez em que uma máquina ganhava 1 partida jogando contra Kasparov. Contudo, apesar da vitória, a primeira disputa terminou 4 a 2 para Kasparov. No ano seguinte, os dois voltaram a se enfrentar, e desse vez Kasparov venceu a primeira partida. Apesar da derrota, na segunda partida a IA cometeu, um erro que chamou a atenção de Kasparov, pois estava em uma posição de vantagem e acabou levando o rei para o lado errado, o que futuramente foi explicado como um mecanismo para evitar uma sobrecarga de processamento que acabou entrando em ação. Após isso, os jogos 3, 4 e 5 terminaram empatados quando Kasparov adotou uma forma de jogar diferente da usual. Foi então na partida 6 que a IA sacrificou uma peça para se beneficiar a longo prazo e acabou vencendo a partida, terminando a disputa em 3,5 a 2,5 para a Deep Blue. O sacrifício da peça gerou polêmica e desconfiança, pois era uma jogada que se imaginava impossível para uma máquina, e foi futuramente explicado que poderia ter sido uma configuração prévia, o que não configurava trapaça, já que era uma posição teórica [2].

B. Go

Go é um jogo chinês jogado em um tabuleiro de 19x19, onde o objetivo é cercar todas as peças e obter mais território que o adversário [3].

Apesar de IAs já terem dominado jogos como dama, xadrez e até mesmo detetive anteriormente, dessa vez o feito era diferente, pois as IAs que venceram esses jogos eram baseadas em um código que analisava todas as jogadas possíveis e escolhia a melhor, ou seja, funcionava por força bruta. Porém, no Go é impossível analisar todas as jogadas possíveis, já que o jogo possui 10^{171} posições possíveis, o que é maior até que a estimativa do número de átomos no universo [3].

Em 2016 a inteligência artificial do laboratório DeepMind pertencente a Google, foi capaz de não só jogar, mas derrotar Lee Sedol, o melhor jogador de Go do mundo. Sedol joga Go desde os 12 anos de idade e até o momento da partida, havia sido campeão mundial de Go 18 vezes, sendo indiscutivelmente o melhor do mundo [4].

III. Jogos e Aprendizagem

O jogo é tido como uma forma de garantir um ambiente planejado, motivador, agradável e enriquecido, o que pode possibilitar a aprendizagem de diversas habilidades. O jogo pode ser utilizado como uma ferramenta a fim de melhorar o processo de aprendizagem do aluno, visto que é capaz de estimular o nível de interesse do mesmo [5].

Esse projeto mostra uma forma de aplicação de IA para realizar uma tarefa por meio de um jogo. Considerando dessa forma a possibilidade de ensino de IA por meio desse ambiente.

IV. Desenvolvimento do jogo

Para facilitar o processo de treinamento da IA, de modo a possibilitar uma população maior que 1, e também facilitar

as ações da mesma dentro do jogo, surgiu a necessidade de desenvolver o Flappy Bird.

Flappy Bird é um jogo onde o jogador controla um pássaro que voa para frente e vai caindo, a única ação possível é pular, e é utilizada para se manter no ar e passar por obstáculos que vem de encontro ao mesmo. A cada obstáculo passado, a pontuação é incrementada em 1, e quando o jogador colide com um obstáculo ou com o chão, ele perde.

Para o desenvolvimento do jogo foi decidido utilizar a biblioteca PyGame, que facilita o processo de renderização dos gráficos e utilizado para desenhar os objetos do jogo. A biblioteca disponibiliza também funções de colisão de objetos, que foram utilizadas para detectar quando o pássaro do jogo colidia com o mapa, além de outros recursos não explorados nesse projeto, como a inserção de efeitos sonoros por exemplo. As funções de controle por meio de interrupções de teclado e mouse não foram utilizados para esse projeto pois a entrada do jogo é feita pela própria IA, de modo que não há a necessidade de uso de teclado nem mouse.

Inicialmente foram definidas algumas constantes, como o tamanho da tela e imagens do pássaro, do chão, do plano de fundo e dos canos.

O desenvolvimento foi feito através de classes em python, onde foram criadas 3 classes ao todo para esse jogo em específico, sendo elas, o pássaro, o cano e o chão. A colisão dos objetos foi feita de modo que para um objeto colidir, os pixels devem estar aparecendo na tela realmente na mesma posição, o que evita por exemplo de punir um jogador por uma colisão que não ocorreu aos olhos dele. Para que essa colisão fosse feita, foi feito uma máscara do arquivo do pássaro e dos canos utilizando a função pygame.mask.from_surface(), e passando o arquivo como parâmetro, ao fazer isso e informar as posições de cada objeto na tela, é possível utilizar a função overlap() em uma máscara passando como parâmetro outra máscara, o que permite verificar se a máscara dos objetos estão colidindo, e se estiverem essa função retorna verdadeiro. É interessante utilizar as colisões dessa maneira pois ao comparar duas imagens sem a máscara, é feita um retângulo por fora da mesma, e se esses retângulos ocuparem o mesmo espaço, mas a imagem não, o algoritmo considera como uma colisão, mas ao comparar as máscaras, é possível verificar se além dos retângulos, os objetos propriamente ditos colidiram ou não, como demonstrado na figura 1.

Fig. 1. Exemplificação de pixel perfect collision

Fonte:https://lalarukhbutt.wordpress.com/tag/collision-detection-sprite-sheet/

V. Motivo de utilizar NEAT

A evolução de redes neurais artificiais (RNA) através de algoritmos genéticos (AG) têm se mostrado muito eficientes na tarefa de aprendizagem por reforço [6]. Uma aprendizagem por reforço, é uma forma de ensinar à IA qual ação tomar dependendo de uma situação em específico [7].

Pelos motivos citados acima foi decidido utilizar o método NEAT (*NeuroEvolution of Augmenting Topologies*), que é um algoritmo genético para a evolução de redes neurais artificiais ou então uma técnica de neuroevolução para a realização do projeto.

O NEAT altera tanto os pesos de cada parâmetro quanto as ligações da rede neural, na tentativa de se aproximar de um fit ideal através de suas evoluções e diversidades. Esse método aplica 3 técnicas chave, sendo a primeira que para performar o crossover, o sistema deve conseguir identificar quais genes se juntaram no meio de toda a população. O ponto principal de manter o histórico dos genes, é que dois genes que possuem o mesmo histórico de origem, possuem também as mesmas ligações, podendo mudar os pesos de cada um. A segunda técnica chave é de que para manter possíveis melhorias que necessitam de algumas gerações para serem aperfeiçoadas, a técnica separa cada inovação em uma espécie diferente, evitando assim descartar modificações em processo de melhoria. E o terceiro é de que ele começa a partir de uma estrutura mínima e cresce somente quando necessário, o que permite que as topologias sejam minimizadas através da evolução [6].

VI. Configuração e parâmetros utilizados

Como o NEAT evoluí uma rede neural, precisamos passar alguns parâmetros para fazer a configuração inicial da IA. Os valores de cada parâmetro vão depender do problema a ser resolvido, sendo no caso desse artigo fazer com que a IA jogue Flappy Bird de modo a não morrer nunca. A primeira coisa a ser feita, foi definir alguns parâmetros básicos que o NEAT iria receber, sendo eles as entradas, a saída (ou saídas caso fosse mais do que uma), função de ativação, função fitness e quantidade máxima de gerações.

Como primeira entrada, foi decidido passar a posição do pássaro no eixo Y (altura onde se encontra), e foi passado somente a posição Y pois o pássaro não se movimenta no eixo X no jogo Flappy Bird. Como segundo e terceiro parâmetros de entrada respectivamente, foram passados, a diferença no eixo Y entre o pássaro e o próximo cano de cima (obstáculo superior), e a diferença no eixo Y entre o pássaro e o próximo cano de baixo (obstáculo inferior), tendo assim 3 parâmetros de entrada.

Como no jogo o único movimento possível é pular, o número de saídas escolhido foi de 1, que vai determinar com base nos parâmetros de entrada se o pássaro pula ou não. A função de ativação escolhida foi a tanh, que retorna um valor entre -1 e 1, e futuramente no tratamento dessa saída vai ser verificado se o valor é maior que 0,5 e caso for o pássaro pula, mas caso não seja, o pássaro não faz ação nenhuma, ou seja, deixa de pular. Poderia ter sido utilizada outra função de ativação como a sigmoid por exemplo, que retorna um valor entre 0 e 1, mas a escolhida para esse projeto foi a tanh devido ao fato de possuir um maior alcance de valores, sendo de -1 a 1 quando compara os de 0 a 1 da sigmoid.

Flappy Bird é um jogo extremamente simples de aprender, pois os obstáculos são sempre iguais e a forma de passar por um é sempre a mesma. Foi então decidido utilizar uma população de 20 para que fosse possível acompanhar um pouco do processo de evolução da IA. Porém é importante verificar que esse número pode ser aumentado e se o que se busca é alcançar um ótimo o mais rápido possível é aconselhável que ele seja maior.

Para a função de fitness, é considerado o quão longe um pássaro conseguiu ir, ou seja, quanto mais longe um pássaro chegar, melhor vai ser o fit do mesmo. Para incentivar uma melhor evolução, alguns bônus e penalidades são aplicados em execução, com por exemplo, quando um pássaro passa por um obstáculo é adicionado 5 de fit como recompensa, o que torna o fit do mesmo muito melhor do que o de um pássaro que foi até o objetivo mas acabou colidindo com o mesmo. O próprio ato de colidir com um cano também é penalizado, de forma que ao colidir com um cano, o indivíduo ”morre” e perde 1 de fit. durante o tempo de execução, para cada frame que o pássaro permanece vivo, ele ganha 0,1 de fit como recompensa, e para entender a conta, é importante verificar que o jogo roda a 30fps.

O número máximo de gerações escolhido para esse problema é o de 30 gerações, e isso indica que se em 30 gerações um fit máximo ainda não tiver sido alcançado, o programa não cria mais gerações e encerra. Esse número foi escolhido pois como Flappy Bird é relativamente simples de aprender e possui somente 1 ação, 30 gerações devem ser mais do que o suficiente para se obter um bom resultado.

Para configurar esses e mais parâmetros é utilizado um arquivo de configuração, cuja estrutura está sendo demonstrada na figura 2. Também na figura 2 é possível ver o fitness_criterion que está sendo passado como max, ou seja, procura-se maximizar o valor, ou obter o valor mais alto possível e não o contrário. O segundo parâmetro fitness_threshold determina quanto é o fitness máximo, nesse caso foi passado o valor 500 pois se o fitness atinigr 500, um resultado satisfatório já deve ter sido alcançado. O terceiro parâmetro é o tamanho da população, que como citado anteriormente está em 20. O parâmetro reset_on_extinction está setado como False e determina se quando todas as espécies se tornarem extintas devido a uma estagnação, novas espécies sejam criadas ou não, no caso de False elas não são criadas. O parâmetro activation_default, determina que tipo de função de ativação está sendo utilizada, sendo nesse caso a tanh. Activation_mutate_rate é a chance de que quando criado um novo membro da população ele receba uma outra função aleatória de ativação sendo ela passada no parâmetro logo abaixo, activation_options, e está setado como 0, já que não é julgado interessante nesse caso. Outros parâmetros como número de entradas e saídas já citados anteriormente por exemplo, também são passados através desse arquivo que é um arquivo de texto.

Fig. 2. Estrutura do arquivo de configuração de parâmetros

```
[NEAT]
fitness_criterion     = max
fitness_threshold     = 500
pop_size              = 20
reset_on_extinction   = False

[DefaultGenome]
# node activation options
activation_default      = tanh
activation_mutate_rate  = 0.0
activation_options      = tanh
```

VII. Aplicação da IA no jogo

Para esse projeto foi utilizada a biblioteca neat-pyton, que basicamente auxilia na utilização de NEAT na linguagem de programação python, utilizada nesse projeto.

O primeiro passo foi fazer a importação do arquivo de configuração citado na seção acima e com ele configurar e inicializar a chamada da função de fitness, como demonstrado na figura 3. O comando winner = p.run(eval_fitness, 30), es’ta responsável por chamar a função eval_fitness no código por no máximo 30 gerações, e salvar o vencedor na variável winner.

Fig. 3. importando configurações e chamando a função de fitness

```
def run(config_path):
    config = neat.config.Config(neat.DefaultGenome, neat.DefaultReproduction,
                neat.DefaultSpeciesSet, neat.DefaultStagnation,
                config_path)

    p = neat.Population(config)

    p.add_reporter(neat.StdOutReporter(True))
    stats = neat.StatisticsReporter()
    p.add_reporter(stats)

    winner = p.run(eval_fitness , 30)

if __name__ == "__main__":
    local_dir = os.path.dirname(__file__)
    config_path = os.path.join(local_dir, "config-feedforward.txt")
    run(config_path)
```

A função eval_fitness é a main do código e é lá que os valores de fitness são calculados, e onde fica o looping principal do jogo, para que a IA possa interagir com o mesmo. Na figura 4 está sendo demonstrado o looping que roda para cada indivíduo, sendo nesse caso cada pássaro, e é possível observar que a cada execução é chamada a função

de movimentação do pássaro e em termos de IA, a função de fitness é recompensada com 0.1, pois isso quer dizer que o pássaro está vivo nesse frame, e serve para aumentar o fit a cada frame que o pássaro passa vivo.

Fig. 4. Importando configurações e chamando a função de fitness

```
for x, bird in enumerate(birds):
    bird.move()
    ge[x].fitness += 0.1
```

Na figura 5 é demonstrado a chamada da função de ativação da rede passando como parâmetros a altura do pássaro, diferença de altura do pássaro com o cano de cima e a diferença de altura com o cano de baixo. Como está sendo utilizado tanh como função de ativação, a saída será entre -1 e 1, e pode-se observar que se a saída for maior do que 0,5, é chamada a função que faz o pássaro pular.

Fig. 5. Função de output e acão do pássaro

```
    output = nets[x].activate((bird.y, abs(

    if output[0] > 0.5:
        bird.jump()
# bird.move()
```

Na figura 6 é verificado se cada pássaro está colidindo com um cano, e caso esteja, esse indivíduo é penalizado com -1 de fit, para reforçar a ideia para a IA de que a colisão é algo negativo. Além de retirar o fit, é retirado também a animação desse pássaro da tela, pois o indivíduo perdeu o jogo segundo suas regras e ele é retirado das listas de redes neurais e genomas, parando seu fit no valor atual, pois na próxima execução, como ele não se encontra nas listas, não seguirá adiante e não será recompensado.

Fig. 6. Importando configurações e chamando a função de fitness

```
for x, bird in enumerate(birds):
    if pipe.collide(bird):
        ge[x].fitness -= 1
        birds.pop(x)
        nets.pop(x)
        ge.pop(x)
```

Quando um indivíduo passa por um obstáculo, ele é recompensado com 5 de fit, para incentivar a IA a retornar uma saída sempre propícia a passar por eles, e isso está sendo demonstrado na figura 7.

Fig. 7. Recompensa ao passar de um cano

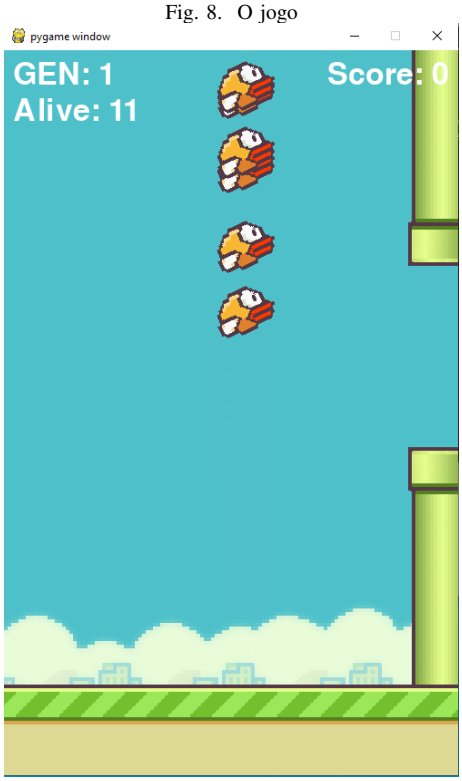
```
if add_pipe:
    score += 1
    for g in ge:
        g.fitness +=5
```

É possível observar que a função de fitness e o jogo rodam no mesmo loop, o que permite a interação da IA com o mesmo, realizando ações como pular dependendo de sua saída e permitindo receber como entradas informações como a altura do pássaro e a diferença de altura com os próximos obstáculos.

VIII. Resultados

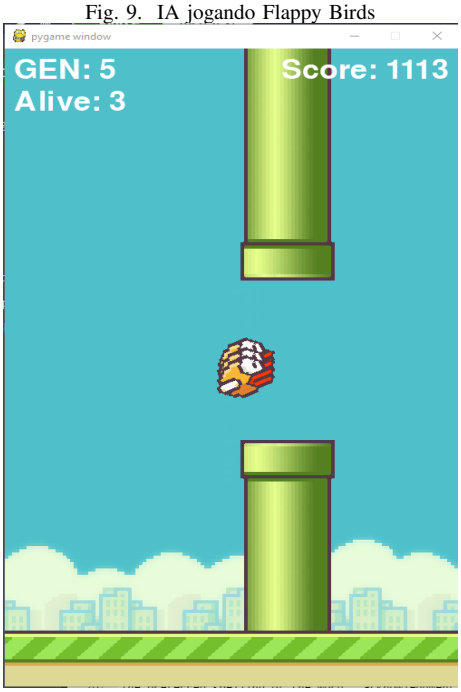
Ao executar o código obtemos um Flappy Bird funcional com vários indivíduos tentando e aprendendo a jogar, quando todos os indivíduos de uma geração perdem, o jogo é reiniciado com a próxima geração. A figura 8 demonstra

Fig. 8. O jogo



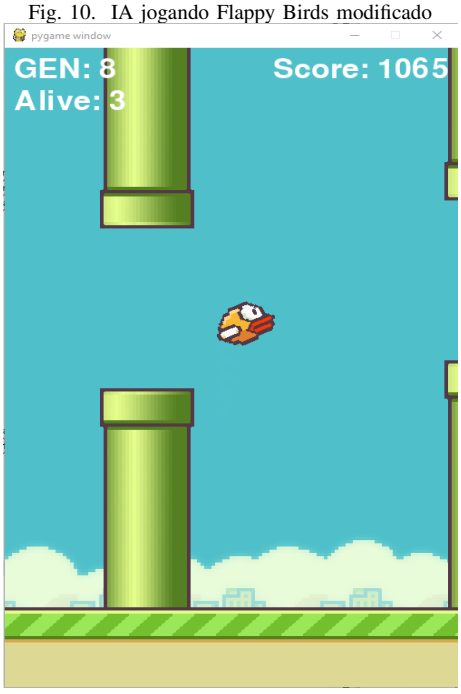
Como citado anteriormente Flappy Bird é um jogo muito simples, e como possui somente uma saída e os obstáculos são sempre iguais, mudando somente a posição, acaba sendo algo simples para uma IA aprender e obter um bom resultado. A figura 9 demonstra que após 5 gerações já tinham 3 indivíduos que aprenderam a jogar o jogo de forma que nunca perdem.

Fig. 9. IA jogando Flappy Birds



Devido a essa facilidade, foi feito uma versão em que os obstáculos se movimentam no eixo Y, ou seja, de cima para baixo. Nessa versão demorou um pouco mais para obter um bom resultado, mas ainda assim a IA foi capaz de conseguir jogar o jogo de forma que nunca iria perder, sendo demonstrado o resultado na figura 10.

Fig. 10. IA jogando Flappy Birds modificado



Em ambas as figuras 9 e 10, o valor de score, cujo o qual corresponde a 1113 na primeira e 1065 na segunda, é incrementado toda vez que o pássaro passa por um obstáculo. Com base nisso e no fato de que a forma de passar os obstáculos é sempre a mesma, pode-se dizer que esses indivíduos vão ficar jogando até o programa ser encerrado, pois não existe score máximo.

No caso da IA aplicada no Flappy Birds padrão, foram necessárias 5 gerações com 20 indivíduos para obter o resultado desejado. Apesar do jogo ficar um pouco mais difícil com os canos se movimentando, a IA ainda consegue aprender com certa facilidade a jogar esse jogo, como demonstrado na figura 10, onde foram necessárias 8 gerações de 20 indivíduos em cada para alcançar o mesmo resultado do que no jogo normal.

Conclusão

Levando-se em conta o que foi observado, pode-se dizer que o método de aprendizagem por reforço pode ser utilizado para ensinar inteligências artificiais a realizar tarefas como jogar jogos, e o método NEAT de neuroevolução obteve um bom desempenho nesse aspecto.

É possível observar que foram abordados diversos conceitos de inteligência artificial, como aprendizagem por reforço, algoritmos genéticos, redes neurais e neuroevolução por exemplo. Esses mesmos conceitos foram colocados em prática em um projeto potencialmente divertido, o que evidencia a possibilidade de um ensino por meio de aplicação em jogos, com o objetivo de despertar interesse e prender a atenção dos alunos.

Referências

[1] BONA, André. Inteligência Artificial: 8 coisas que confirmam a presença da I.A no cotidiano. 2019. Disponível em: https://andrebona.com.br/inteligencia-artificial-8-coisas-que-mostram-a-presenca-da-i-a-no-cotidiano/. Acesso em: 05 ago. 2020.

[2] LEITAO, Rafael. O Homem E A Máquina: O Match Kasparov X Deep Blue. 2015. Disponível em: https://rafaelleitao.com/o-homem-e-a-mquina-o-match-kasparov-x-deep-blue/. Acesso em: 05 ago. 2020.

[3] NUNES, Rodrigo. Regras do jogo GO. 2019. Disponível em: https://www.bananaquantica.com.br/regras-do-jogo-go/. Acesso em: 06 ago. 2020.

[4] PRADO, Jean. Um computador do Google venceu um campeão mundial neste jogo chinês. 2016. Disponível em: https://tecnoblog.net/192604/computador-google-vence-campeao-go/. Acesso em: 06 ago. 2020.

[5] ALVES, Luciana; BIANCHIN, Maysa Alahmar. O jogo como recurso de aprendizagem. 2010. Disponível em: http://pepsic.bvsalud.org/scielo.php?script=sci_arttextpid=S0103-84862010000200013. Acesso em: 06 ago. 2020.

[6] K. O. Stanley and R. Miikkulainen, "Efficient evolution of neural network topologies," Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), Honolulu, HI, USA, 2002, pp. 1757-1762 vol.2, doi: 10.1109/CEC.2002.1004508.

[7] TANAKA, Marcos. 3 tipos de aprendizado caracterizam o Machine Learning. 2018. Disponível em: https://cio.com.br/3-tipos-de-aprendizado-caracterizam-o-machine-learning/. Acesso em: 08 ago. 2020.