Algoritmos genéticos aplicados ao Tetris

André Almeida
RA: 164047

Igor Torrente
RA: 169820

Lucas Cunha
RA: 172655

João Spuri
RA: 155943

I. Resumo

Nesse projeto, foi proposta e experimentada uma abordagem para a criação de um algoritmo que consiga fazer jogadas no jogo eletrônico *Tetris*, obtendo uma pontuação compatível com a esperada por humanos. Para tal, foram utilizadas técnicas de algoritmos genéticos aplicadas em redes neurais artificiais.

II. Introdução

A. Tetris

1) O jogo: Tetris é um jogo eletrônico criado em 1984 pelo matemático soviético Alexey Pazhitnov, tendo obtido grande popularidade principalmente nas plataformas *Atari ST* e no *Nintendo Entertainment System* [1]. Até hoje, já foram vendidas mais de 50 milhões de cópias mundialmente. O jogo é do gênero *puzzle*, onde o jogador precisa resolver algum tipo de quebra-cabeça.
No Tetris, o ”tabuleiro” do jogo é formado por uma malha de $22x10$ quadrados (com as duas linhas do topo ocultas ao jogador), onde o jogador deve ir encaixando as peças (os ”Tetraminós”) que caem verticalmente no tabuleiro em uma sequência aleatória. Existem 7 tipos de Tetraminós, cada um formato distinto. O objetivo do jogador é manipular essas peças, movendo-as horizontalmente e girando-as de forma a criar uma linha horizontal no tabuleiro sem espaços vazios. Quando uma linha assim é completa, ela é destruída, as peças acima dela ”caem” uma linha para baixo e o jogador pontua.

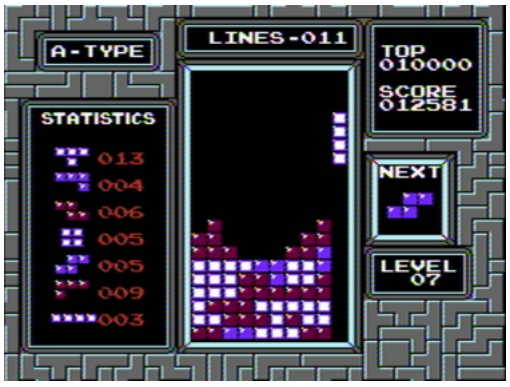

Figura 1: *captura de tela da versão* NES *do jogo*

2) Complexidade computacional: Foi provado [2] que em uma versão de Tetris onde o jogador já conhece toda a sequência de peças que virão, os seguinte objetivos são problemas NP-Completos:

- Maximizar o número de linhas limpas enquanto joga com a sequência dada;
- Maximizar o número de peças colocadas antes de completar uma linha;
- Maximizar o número de pontuações simultâneas de quatro linhas;
- Minimizar a altura da última peça colocada em uma sequência.

Com exceção do terceiro, todos esses objetivos também são difíceis de serem aproximados.

B. Algoritmos genéticos

Um algoritmo genético é uma técnica para encontrar uma solução ideal ou próxima à ideal para um problema computacional, com inspirações nas teorias darwinistas de evolução dos seres vivos. As interações do algoritmo são sobre as gerações, onde uma geração é um conjuto de indivíduos e indivíduos são funções/modelos. Em síntese, um algoritmo genético funciona da seguinte maneira [3]:

1) Os parâmetros dos indivíduos da primeira geração são gerados randomicamente;
2) Algum tipo de função de custo é aplicada para avaliar cada indivíduo. Essa função é usada para determinar o sucesso dos indivíduos;
3) Alguma porcentagem dos melhores indivíduos (segundo seus resultados do item anterior) é escolhida.
4) Os indivíduos escolhidos no item anterior serão os ”pais” da nova geração. A partir deles, combinações e mutações irão gerar os outros indivíduos da nova geração.
5) Repita os passos 2-4 até alguma condição de parada for atingida. Quando isso acontecer retorna o melhor indivíduo da última geração.

C. Redes neurais artificiais

Rede neural artificial é uma coleção de unidades chamadas neurônios que são interligados entre si com uma ordem (camadas), com inspiração biológica no funcionamento do cérebro dos animais. Cada neurônio de uma camada se liga a todos os neurônios da anterior (se não for a camada de entrada) e da posterior (se não for a camada de saída). Cada neurônio é ativado de acordo com uma função de ativação, os parâmetros dessa função de ativação são as saídas dos neurônios anteriores multiplicados por um peso nas arestas que conectam estes neurônios. Nos neurônios de entrada são inseridos metadados e nas funções de saída algum tipo de resposta ou classificação [4].

D. Aprendizado de máquina para jogos eletrônicos

Aprendizado de máquinas aplicado em jogos eletrônicos vem sendo estudado pela academia como forma de avaliar a capacidade da máquina de executar tão bem quantas tarefas complexas, como é o caso de jogar videogames. Um estudo em destaque é o caso do AlphaGo, uma inteligência artificial que usa técnicas de aprendizado profundo e conseguiu vencer o campeão mundial do jogo Go [5]. Alguns desses estudos utilizam a técnica de algortimos genéticos para encontrar um modelo computacional adequado para o jogo, e acabaram gerando modelos que são compatíveis com jogadores reais, em jogos como Snake [6] e Counter-Strike [7]. Nesses jogos, parâmetros como pontuação do jogador e desempenho estratégico são usados como função de custo para treinar o algoritmo.

E. Trabalhos relacionados

Em [8], o autor cria uma rede neural para jogar Tetris, usando algoritmos genéticos para otimizar os pesos dos neurônios (substituindo o *backpropagation*). O jogador pontua ao descer peças no tabuleiro e completar linhas. A função de custo é composta por: número de linhas completadas, altura máxima, altura acumulada, altura relativa, buracos das linhas e completude do tabuleiro. Durante o jogo, a cada nova peça, todas as jogadas possíveis são testadas e a que retornar a melhor pontuação (seguindo as heurísticas acima) é executada.

No trabalho [9], o autor utiliza uma função com as heurísticas: altura agregada, linhas completadas, buracos e diferença de alturas vizinhas. Novamente, todas as jogadas possíveis são testadas e a que retornar o maior valor é executada. A função utilizada é a soma do produto das heurísticas com uma variável de valor inicialmente desconhecido. Os pesos de cada uma das heurísticas foi determinado usando algoritmos genéticos.

No artigo [10], as autoras utilizam uma metodologia semelhante à vista em [9], só que com as seguintes heurísticas: maior pilha, buracos, buracos conectados, linhas removidas, diferença de altitude, profundidade máxima, soma dos vales, altura do último tetraminó, número de células ocupadas, número de células ocupadas por coluna, soma de transições ocupadas/desocupadas horizontalmente e verticalmente. A função de custo é dado por $\sum_{i=1}^{n} w_i|r_i(b)-d_i|^{e_i}$, onde $r_i(b)$ é o valor retornado por cada uma das heurísticas e os valores de w_i, d_i e e_i foram descobertos utilizando algoritmos genéticos.

III. Metodologia

IV. Resultados

V. Conclusão

VI. Estudos futuros

References

[1] http://www.atarihq.com/tsr/special/tetrishist.html

[2] Demaine, E. D., Hohenberger, S., & Liben-Nowell, D. (2003, July). Tetris is hard, even to approximate. In COCOON (pp. 351-363).

[3] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning.

[4] Geron, A. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.*

[5] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Chen, Y. (2017). Mastering the game of go without human knowledge. Nature, 550(7676), 354-359.

[6] Yeh, J. F., Su, P. H., Huang, S. H., & Chiang, T. C. (2016, November). Snake game AI: Movement rating functions and evolutionary algorithm-based optimization. In Technologies and Applications of Artificial Intelligence (TAAI), 2016 Conference on (pp. 256-261). IEEE.

[7] Cole, N., Louis, S. J., & Miles, C. (2004, June). Using a genetic algorithm to tune first-person shooter bots. In Evolutionary Computation, 2004. CEC2004. Congress on (Vol. 1, pp. 139-145). IEEE.

[8] https://github.com/llSourcell/How_to_make_an_evolutionary_tetris_bot

[9] https://codemyroad.wordpress.com/2013/04/14/tetris-ai-the-near-perfect-player/

[10] Böhm, N., Kókai, G., & Mandl, S. (2005). An evolutionary approach to tetris. In The Sixth Metaheuristics International Conference (MIC2005) (p. 5).