# NEAT Snake: a both evolutionary and neural network adaptation approach

Alisson Steffens Henrique
ali.steffens@gmail.com
Universidade do Vale de Itajaí
Itajaí, Santa Catarina

Vinicius almeida dos Santos
vinicius@rudinei.cnt.br
Universidade do Vale de Itajaí
Itajaí, Santa Catarina

Rodrigo Lyra
rodrily@gmail.com
Universidade do Vale de Itajaí
Itajaí, Santa Catarina

## ABSTRACT

There are several challenges when modeling artificial intelligence methods for autonomous players on games (bots). NEAT is one of the models that, combining genetic algorithms and neural networks, seek to describe a bot behavior more intelligently. In NEAT, a neural network is used for decision making, taking relevant inputs from the environment and giving real-time decisions. In a more abstract way, a genetic algorithm is applied for the learning step of the neural networks' weights, layers, and parameters. This paper proposes the use of relative position as the input of the neural network, based on the hypothesis that the bot profit will be improved.

## KEYWORDS

Genetic, Game, Artificial Intelligence, NEAT, Neural Network

## 1 INTRODUÇÃO

Um jogo é considerado divertido quando consegue manter o engajamento dos jogadores [9]. Este estado de imersão é apenas alcançado quando o jogo entrega ao jogador uma experiência com dificuldade equilibrada: trazendo desafios, porém sem ser frustrante para o jogador [5].

Para auxiliar os desenvolvedores de jogos nessa tarefa, são utilizadas técnicas de inteligência artificial para descrever o comportamento dos NPCs (*Non Playable Characters*), que são personagens controlados pelo computador. Para trazer uma dificuldade equilibrada, os NPCs devem adaptar-se ao jogador [2].

Diversas técnicas são utilizadas na literatura para esse fim, entre elas, redes neurais e algoritmos genéticos. Embora alguns alcancem seus objetivos, não são incomuns exemplos onde estas técnicas isoladas não sejam suficientes para a aplicação em jogos [1].

Este artigo utiliza de uma técnica denominada NEAT (NeuroEvolution through Augmenting Topologies), que usufrui tanto de redes neurais quanto de algoritmos genéticos para gerar indivíduos mais aptos [8].

Enquanto a rede neural descreve o comportamento dos indivíduos, os algoritmos genéticos buscam otimizar os parâmetros da rede. O principal objetivo da implementação é gerar, com treinamento, um indivíduo capaz de conseguir uma pontuação dita boa no jogo *Snake*.

## 2 TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL

Os jogos são comumente utilizados como ferramentas para o desenvolvimento de novas técnicas de Inteligência Artificial, com o uso de NPCs. Esses costumam assumir o papel por tratarem de problemas semelhantes aos encontrados no mundo real, porém com níveis de abstração e complexidade diferentes [6]. Mesmo assim, a maioria das inteligências em jogos é determinística, não adaptando-se ao jogador [10].

Para uma abordagem mais adaptativa dos NPCs, é comum encontrar exemplos de aplicações que utilizam de algoritmos genéticos [3]. Um problema com essas abordagens, entretanto, é o fato de não serem suficientes para a descrição do comportamento de NPCs, visto que, de modo geral, apenas otimizam os parâmetros do mesmo.

Redes neurais também podem ser utilizadas para isso. Porém, pelo fato de exigirem treinamento supervisionado, acabam assimilando somente ações baseadas no treinamento.

Pensando nesse problema, foi desenvolvida uma variante do algoritmo evolutivo denominada NEAT [8]. Esse método consiste em uma rede neural cujos parâmetros são definidos por meio de um algoritmo genético. Dessa maneira o algoritmo genético otimiza a rede, fazendo com que os indivíduos tenham comportamentos mais adaptáveis.

## 3 IMPLEMENTAÇÃO

Visando experimentar o NEAT em jogos como o *Snake*, foi desenvolvido um jogo em Python capaz de autotreinar e armazenar informações relacionadas às gerações. Em relação ao desenvolvimento, foram utilizadas as bibliotecas pygame [7] e neat-python [4], facilitando o desenvolvimento de jogos e o uso de NEAT, respectivamente.

O mapa do jogo tem 16 por 16 unidades, e a cobra inicia com tamanho 4. Alimentos são posicionados aleatoriamente no mesmo ambiente. Antes de cada *frame*, a rede neural recebe 21 entradas, e suas 3 saídas definem a movimentação da cobra. Consequentemente, o tempo que a rede demora para responder não é relevante para a tomada de decisão.

A cada geração são criados 200 genomas, cada qual descrevendo a topologia da rede neural do personagem. Outros parâmetros são definidos para a execução do algoritmo genético, estes envolvem a probabilidade de diversos eventos relacionados à população inicial e mutação da rede neural. É possível acessá-los no repositório público em *https://github.com/lia-univali/neat-snake*.

A rede neural é iniciada sem nodos na camada escondida, sem conexões, com 21 entradas e 3 saídas. Estas 21 entradas (demonstradas na Figura 1), representam a distância do personagem para a parede, para comida e para o seu corpo, nas direções: frente, esquerda, direita e as quatro diagonais. As saídas representam a decisão do personagem de seguir em frente, virar à esquerda ou à direita.

Nas primeiras gerações, a rede é simples e o algoritmo genético ainda está trabalhando nos pesos das conexões. Sendo assim, a rede apresenta uma complexidade pequena. Com o avanço das gerações, poucas conexões são feitas e dificilmente nodos da rede são criados.

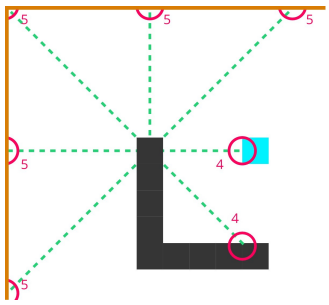


**Figure 1: Entradas da rede**

Conforme soluções com maior qualidade vão surgindo, características boas se propagam aos poucos, melhorando a qualidade da população como um todo.

Estes indivíduos são colocados a prova, sendo pontuados de acordo com a quantidade de comidas coletadas, com bonificação de proximidade ao alimento, e punição por movimento em *loop*. Com o tempo, o algoritmo consegue ir modificando a topologia das redes e, com cerca de 14 gerações, já começam a surgir alguns nodos escondidos.

A quantidade de conexões - bem como seus pesos - e nodos escondidos vai mudando conforme a população evolui. Por volta da geração 60, alguns indivíduos já apresentam redes mais complexas, com até cinco nodos escondidos.

Nem sempre, entretanto, as redes precisam ser complexas para terem bons resultados. A rede da Figura 2, por exemplo, não tem nodos escondidos porém teve um bons resultados quando comparada as outras da mesma geração.

**Figure 2: Rede com pesos - geração 82**

A evolução dos indivíduos, demonstrada na Figura 3, leva à interpretação de que, para grandes avanços ocorrerem, o aprendizado depende de mutações boas. E essas mutações dificilmente acontecem.

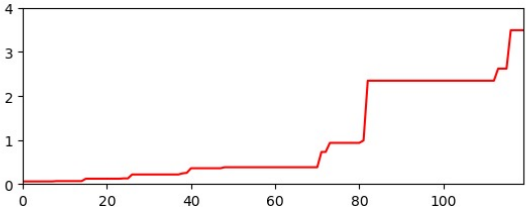


**Figure 3: Curva de aprendizado**

Mutações que apresentam boa melhora dificilmente acontecem, possivelmente, pelo fato que a complexidade do problema aumenta proporcionalmente ao tamanho do indivíduo. A partir de certo tamanho a cobra deve criar estratégias para, além de evitar paredes e ir atrás de comida, não se prender com o próprio corpo.

Por conta dessa situação, o aumento da complexidade da rede se torna necessário. Na Figura 4, é demonstrado um representante da geração 116. Ele já apresenta nodos escondidos, e obtém resultados superiores aos anteriores, pois sabe lidar com o próprio corpo.

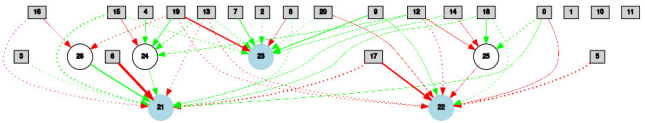

**Figure 4: Rede do melhor indivíduo da geração 116**

## 4 CONSIDERAÇÕES FINAIS

Um problema deste tipo de treinamento no jogo *Snake*, é proveniente do fato de, inicialmente, o tamanho da cobra ser pequeno. Desta forma, as redes que vão sendo selecionadas, não necessariamente sabem desviar do próprio corpo, o que faz com que, em gerações mais altas, a maior parte das mortes seja proveniente deste tipo de colisão.

Além disso, os indivíduos costumam aprender a "virar" sempre para o mesmo lado antes de alimentar-se, isso é causado pelo fato de as saídas de virar para esquerda ou direita serem diferentes. Desta forma, a equação de uma das saídas pode acabar se sobressaindo e determinar sozinha a avaliação de toda a rede.

Também foram feitos testes com diferentes quantidades de entradas, que conseguiram resultados interessantes, porém tiveram problemas na convergência do algoritmo genético.

O trabalho ainda demanda melhorias em relação a parâmetros do NEAT, mas acredita-se que os resultados atuais já possam ser considerados razoáveis. Possíveis trabalhos futuros podem envolver: melhorar o cálculo score do jogo, fazendo com que a melhora seja ainda mais gradual; treinar o NEAT durante mais tempo; paralelizar a avaliação da rede para acelerar o treinamento; experimentar outras entradas para a rede; e testar parâmetros diferentes.

## REFERENCES

[1] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. 2019. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528* (2019).
[2] Maria Cutumisu, Duane Szafron, Michael H Bowling, and Richard S Sutton. 2008. Agent Learning using Action-Dependent Learning Rates in Computer Role-Playing Games.. In *AIIDE*.
[3] Chang-Shing Lee, Mei-Hui Wang, Li-Chuang Chen, Yusuke Nojima, Tzong-Xiang Huang, Jinseok Woo, Naoyuki Kubota, Eri Sato-Shimokawara, and Toru Yamaguchi. 2019. A GFML-based Robot Agent for Human and Machine Cooperative Learning on Game of Go. *arXiv preprint arXiv:1901.07191* (2019).
[4] Alan McIntyre, Matt Kallada, Cesar G. Miguel, and Carolina Feher da Silva. [n.d.]. neat-python. https://github.com/CodeReclaimers/neat-python.
[5] Dave Novak. 2003. Solder man. Video. In *ACM SIGGRAPH 2003 Video Review on Animation theater Program: Part I - Vol. 145 (July 27–27, 2003)*. ACM Press, New York, NY, 4. https://doi.org/99.9999/woot07-S422
[6] Jeff Orkin. 2006. Three states and a plan: the AI of FEAR. In *Game Developers Conference*, Vol. 2006. 4.
[7] Pete Shinners et al. 2011. Pygame. *Dostupné z: http://pygame. org/[Online* (2011).
[8] Kenneth O Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10, 2 (2002), 99–127.
[9] Penelope Sweetser and Peta Wyeth. 2005. GameFlow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)* 3, 3 (2005), 3–3.
[10] YR Tsoy and VG Spitsyn. 2005. Using genetic algorithm with adaptive mutation mechanism for neural networks design and training. In *Proceedings. The 9th Russian-Korean International Symposium on Science and Technology, 2005. KORUS 2005.* IEEE, 709–714.