# Aplicação de Algoritmos de Colônia de Formigas no Jogo da Cobra

1st Gustavo Augusto
*Depto. Engenharia Elétrica - Engenharia de Sistemas*
*Universidade Federal de Minas Gerais*
Belo Horizonte, Brasil
gustavoaugustopires@ufmg.br

2nd Henrique Perim
*Depto. Engenharia Elétrica - Engenharia de Sistemas*
*Universidade Federal de Minas Gerais*
Belo Horizonte, Brasil
henriquedp@ufmg.br

3rd Marcelo Andrade
*Depto. Engenharia Elétrica - Engenharia de Sistemas*
*Universidade Federal de Minas Gerais*
Belo Horizonte, Brasil
marcelo428@ufmg.br

4th Vinícius Cardoso
*Depto. Engenharia Elétrica - Engenharia de Sistemas*
*Universidade Federal de Minas Gerais*
Belo Horizonte, Brasil
viniciusc@ufmg.br

***Abstract*—Algoritmos evolucionários podem ser amplamente aplicados no desenvolvimento de soluções para jogos, sistemas de realidade virtual e simulações. Nesse trabalho, eles são aplicados como forma de encontrar o melhor comportamento possível no jogo "Snake Game", de forma que seja possível alcançar a melhor pontuação. Para isso, foi aplicado o Algoritmo de Colônia de Formigas, que é um algoritmo evolucionário baseado na escolha de trajetos que formigas fazem pela quantidade de feromônios. No desenvolvimento do trabalho, diversos parâmetros foram testados com o objetivo de encontrar a melhor configuração para o modelo. Entre esses parâmetros estão a taxa de evaporação dos feromônios das arestas, o número de iterações e o número de formigas. Com isso, o projeto representa uma importante análise da influência dessas variáveis no desempenho de algoritmos evolucionários baseados no comportamento de enxames.**

***Index Terms*—component, formatting, style, styling, insert**

## I. Introduction

Nos últimos anos, a aplicação de algoritmos evolucionários tem se destacado na solução de diversos problemas complexos, abrangendo áreas como jogos, sistemas de realidade virtual e simulações. Esses algoritmos, inspirados em processos naturais como a seleção, recombinação e mutação, oferecem estratégias robustas para otimização. Dentro desse contexto, a Otimização por Colônia de Formigas (Ant Colony Optimization - ACO) tem se mostrado uma técnica eficaz, especialmente em problemas onde a busca por caminhos ótimos é essencial, como é o caso do clássico jogo da cobra (Snake Game).

O Snake Game oferece um cenário ideal para a aplicação de ACO. O objetivo principal no jogo é guiar uma cobra para consumir frutas espalhadas aleatoriamente pelo tabuleiro, maximizando a pontuação ao evitar colisões com obstáculos e com a própria cauda. A complexidade surge da necessidade de encontrar a trajetória mais eficiente para coletar as frutas no menor tempo possível.

Este trabalho visa aplicar o ACO no Snake Game para desenvolver um agente autônomo capaz de atingir a melhor pontuação possível, minimizando o tempo necessário para consumir todas as frutas no tabuleiro. Diversos parâmetros do ACO, como a taxa de evaporação de feromônios, o número de iterações e o número de formigas, foram testados para identificar a configuração que proporciona o melhor desempenho. Ao adaptar e implementar o algoritmo em um ambiente de simulação do jogo Snake, buscamos explorar a influência desses parâmetros e contribuir para a evolução das técnicas de otimização baseada em comportamento de enxames.

Este estudo se diferencia por redefinir a dinâmica tradicional do Snake Game, introduzindo um número variável de frutas e avaliando a eficiência do ACO na busca pelo melhor caminho para consumi-las. A implementação do algoritmo foi realizada em Python, utilizando a biblioteca Pygame para a interface gráfica e a biblioteca ACO para a execução dos algoritmos de otimização. Os resultados obtidos destacam a relevância dos parâmetros ajustados na performance do agente e fornecem bases para futuras aplicações de ACO em problemas similares.

## II. Revisão da Literatura

### A. Algoritmos Evolucionarios

Os Algoritmos Evolucionários são técnicas computacionais inspiradas na evolução natural que buscam otimizar a solução de problemas complexos. Eles são baseados em três princípios fundamentais: seleção natural, recombinação e mutação. Por meio da geração aleatória de uma população inicial de soluções, os algoritmos evolucionários procuram encontrar a melhor solução em termos de adaptação ao ambiente, tendo em vista critérios definidos pelo usuário. São comuns na otimização de problemas mono-objetivo (em que há apenas um objetivo a ser alcançado) e multi-objetivo (em que há vários objetivos a serem considerados). Alguns exemplos de algoritmos evolucionários citados no texto são o algoritmo genético, o algoritmo de seleção clonal e o NSGA-II [1].

### B. Otimização por Colônia de Formigas

O Ant Colony Optimization (ACO), ou Otimização por Colônia de Formigas é um algoritmo de otimização baseado em uma metáfora biológica de comportamento de formigas. O objetivo é encontrar o caminho mais curto entre duas posições em um espaço de busca explorado por várias formigas artificiais. As formigas artificiais constroem soluções viajando em um grafo e modificam a trilha com base em um princípio de retroalimentação positiva, conhecido como feromônio. A ideia é que as formigas artificiais deixem feromônios em sua trilha enquanto exploram diferentes caminhos e a trilha com uma quantidade maior de feromônios seja mais provável de ser seguida por outras formigas [2].

O ACO consiste em três fases principais que são executadas até que o método atinja a condição de parada, que pode ser o número de iterações ou um valor mínimo desejado. Essas estapas são:

1) **Construção de soluções por formigas:** Um conjunto de formigas artificiais constrói soluções a partir de componentes de solução disponíveis. Cada formiga tem uma memória que armazena os vértices que ela visitou, construindo, assim, a solução. Um caminho de solução parcial é criado por cada formiga através de um grafo de construção que é um subconjunto do grafo total de componentes de solução viáveis. A escolha do próximo componente de solução é guiada por um mecanismo estocástico que é influenciado pelas quantidades de feromônio depositadas nas arestas do grafo. Cada formiga, em cada iteração, decide qual vértice visitar a seguir baseando-se em uma regra de probabilidade. A probabilidade de uma formiga visitar uma cidade específica é influenciada por pelos feromônios e pela heurística. As formigas depositam feromônios nas arestas entre os vértices que visitam. A quantidade de feromônio depositada é inversamente proporcional ao custo da aresta. A heurística é uma medida da aptidão de um vértice com base em informações adicionais, como a distância entre eles.
2) **Melhoria local das soluções:** Cada solução parcial é melhorada individualmente, utilizando um método de busca local, que varia de acordo com o problema específico. Isso é feito para aumentar a qualidade da solução.
3) **Atualização do feromônio:** As trilhas de feromônios são atualizadas para refletir a qualidade das soluções encontradas. Mais especificamente, as trilhas deixadas pelas formigas artificiais na etapa 1 são reforçadas após a construção de boas soluções e enfraquecidas após a construção de soluções ruins. A atualização do feromônio pode ser realizada de várias maneiras, mas em geral envolve uma combinação de evaporação de feromônios antigos, determinada por uma taxa de evaporação, e deposição de feromônios novos e frescos.

Essas três etapas são executadas iterativamente em cada ciclo do algoritmo até que um critério de parada seja alcançado, como um número fixo de iterações ou a convergência para uma solução satisfatória.

### C. Trabalhos Correlatos

Dado a eficiência do uso de algoritmos evolutivos em diversos tipos de problemas de otimização, eles tem sido amplamente empregados em diversos tipos de aplicações. Entre essas aplicações está o seu uso em jogos, simulações e realidade virtual. Nesses contextos, o algoritmo fica a cargo de encontrar um melhor conjunto de operações, entre as possíveis, para que a solução tenha o melhor desempenho possível. Além do desempenho, é possível adicionar restrições e demais funções de custo para evitar comportamentos indesejados.

No trabalho desenvolvido em [3], é aplicado a técnica NEAT (NeuroEvolution through Augmenting Topologies), onde uma rede neural é utilizada para a tomada de decisões, com informações relevantes do ambiente e dando decisões em tempo real, enquanto um algoritmo genético é aplicado para a etapa de aprendizado dos pesos, camadas e parâmetros das redes neurais. Isso significa que a técnica NEAT utiliza simultaneamente tanto algoritmos genéticos quanto redes neurais. Com isso, o algoritmo genético é utilizado para a otimização dos parâmetros das redes neurais por meio de reprodução e seleção dos indivíduos que apresentam melhores resultados e uma espécie de evolução artificial. Dessa forma, a cada geração os indivíduos com mutações e melhores resultados são selecionados para passarem seus parâmetros e estruturas adiante, e ao longo dessas gerações, o resultado esperado é a obtenção de uma rede neural que produza comportamentos mais eficientes e adaptativos. O principal objetivo da implementação é gerar, com treinamento, um indivíduo capaz de conseguir uma pontuação dita boa no jogo Snake.

Já o método proposto em [4] desenvolve um sistema de direção autônoma capaz de aprender a tomar decisões com base em informações sensoriais do ambiente utilizando redes neurais profundas e algoritmos genéticos. A abordagem para capacitar o carro autônomo a trafegar em ambientes com obstáculos envolve uma de RNA treinada por meio de um Algoritmo Genético. Além disso, um ambiente virtual foi criado para treinar carros autônomos usando a simulação de cenários com obstáculos. Desse modo, esse trabalho mostra um paralelo na resolução do problema de condução de um agente, como o apresentado por este desenvolvimento. Contudo, se difere na aplicação de uma rede neural para encontrar os pesos da rede e não para realizar o processo em si.

Por fim, no trabalho desenvolvido em [5] analisa várias técnicas de aprendizagem de máquina e otimização no contexto da busca por melhores resultados em jogos. Entre os jogos analisados está o jogo da cobra. A otimização apresentada utiliza a técnica de busca heurística, que é uma extensão da busca em profundidade que utiliza uma heurística para estimar a melhor escolha de caminho em direção ao objetivo. A técnica utiliza uma função de custo que considera tanto o custo atual do caminho percorrido quanto a estimativa heurística do custo mínimo para atingir o objetivo. Isso garante que a busca se

mova na direção correta, priorizando caminhos que levam mais perto do objetivo.

### D. Jogo da Cobra

## III. Metodologia

Para otimizar o jogo Snake (também conhecido como ”jogo da cobrinha”) utilizando o algoritmo de Otimização por Colônia de Formigas (ACO - Ant Colony Optimization). Inicialmente, a ideia era começar o jogo com uma única fruta, e após várias gerações do algoritmo, a cobra encontraria o melhor caminho, que frequentemente seria uma “diagonal”. No entanto, após uma análise crítica, concluiu-se que uma abordagem diferente para o problema seria mais interessante para a otimização.

Dessa forma, o problema foi redefinido da seguinte maneira: inicialmente, um número X de frutas é colocado aleatoriamente no tabuleiro. O objetivo da cobra é encontrar o melhor caminho para consumir todas as frutas, sendo que o ”melhor caminho” é definido como aquele que minimiza o tempo total necessário para coletar todas as frutas. Assim, a eficiência do algoritmo é avaliada com base na capacidade da cobra de encontrar o caminho mais curto e rápido para consumir todas as frutas no menor tempo possível.

### A. Implementação

Para a implementação do nosso trabalho, utilizamos uma base de código do jogo Snake disponível no *GitHub*, adaptando-o para se adequar ao nosso problema específico. Todo o desenvolvimento foi realizado em *Python*, aproveitando suas bibliotecas e *frameworks* para manipulação de gráficos, algoritmos de otimização e gerenciamento de estados do jogo.

### B. Estrutura geral do código

O código foi desenvolvido em *Python* com a utilização de orientação a objetos e utiliza a biblioteca *Pygame* para a interface gráfica do jogo, além de bibliotecas adicionais como *Matplotlib* para visualização dos resultados. O algoritmo de otimização por colônia de formigas (ACO) foi implementado utilizando a biblioteca *aco* do *Python* e integrado ao jogo Snake.

*1) Configurações básicas:* Inicialmente, algumas configurações básicas do jogo são definidas, como a velocidade da cobra, a posição inicial, o tamanho da janela, e as cores utilizadas. Essas configurações são essenciais para definir o ambiente do jogo e a mecânica básica de movimento da cobra.

*2) Classes principais:*

- **Classe Snake:** Responsável pela criação e movimentação da cobra. Contém métodos para mudar a direção, atualizar a posição, verificar se a cobra comeu uma fruta e calcular a pontuação. Os métodos `change_direction` e `update_direction` permitem que a cobra mude de direção conforme a entrada do usuário. O método `move` atualiza a posição da cobra e verifica se ela comeu uma fruta.
- **Classe Fruit:** Responsável pela criação de frutas em posições aleatórias na tela.
- **Classe Game:** Integra todos os componentes e controla a lógica do jogo. Inicializa o *Pygame*, cria a cobra e as frutas, e gerencia a lógica do jogo, incluindo a detecção de colisões e a exibição da pontuação.

*3) Algoritmo de Otimização por Colônia de Formigas (ACO):* Utilizando a biblioteca *aco*, o método `ant_colony_optimization` aplica o algoritmo ACO para encontrar o melhor caminho entre as frutas. O resultado é uma lista de coordenadas que representa o caminho ótimo que a cobra deve seguir.

### C. Lógica do jogo com ACO

*1) Inicialização:* Ao iniciar o jogo, são geradas aleatoriamente as coordenadas das frutas. Para este experimento, são posicionadas 10 frutas dentro da grade do jogo. As posições são determinadas por um gerador de números aleatórios, garantindo que cada fruta ocupe uma célula única na grade.

*2) Otimização:* O algoritmo ACO é utilizado para determinar o melhor caminho que a cobra deve seguir para coletar todas as frutas, similar ao problema clássico do Caixeiro Viajante (TSP - Traveling Salesman Problem). O ACO é um algoritmo probabilístico inspirado no comportamento de forrageamento das formigas, onde soluções potenciais são exploradas e a melhor solução é reforçada por meio de um ”feromônio” virtual.

Com os caminhos otimizados gerados pelo ACO, que consistem em uma lista de coordenadas XY, é necessário identificar a fruta mais próxima da posição inicial da cobra. A cobra começa no canto superior esquerdo da grade. Uma função específica calcula a distância entre a posição inicial da cobra e cada fruta, determinando qual fruta está mais próxima à direita e abaixo da posição inicial.

*3) Movimentação:* Após identificar a fruta mais próxima, o caminho entre cada fruta é gerado. Este caminho é então convertido de uma linha contínua em um caminho discreto que só permite movimentos nas quatro direções principais: esquerda, direita, cima e baixo, eliminando qualquer possibilidade de movimentos diagonais.

Com o conjunto de caminhos discretos estabelecidos, que são basicamente as coordenadas XY que a cobra deve percorrer, são gerados os comandos correspondentes que a cobra precisa seguir para percorrer o caminho. Os comandos possíveis são: up, down, right e left.

*4) Execução:* Finalmente, os comandos são executados sequencialmente, movendo a cobra conforme o caminho determinado. O resultado final pode ser exibido ao usuário, mostrando o desempenho da cobra na coleta das frutas seguindo o caminho otimizado pelo algoritmo ACO.

### D. Sistema de pontuação

O sistema de pontuação é desenvolvido para incentivar a eficiência no consumo das frutas. A cada fruta consumida, a cobra ganha uma pontuação padrão. Adicionalmente, se o intervalo de tempo entre consumir uma fruta e a próxima for

inferior a 5 segundos, um bônus de pontuação é concedido, incentivando movimentos rápidos e precisos. Se o intervalo for superior a 5 segundos, apenas a pontuação padrão é adicionada.

## IV. Resultados

Ao executar o algoritmo foi feita uma análise de como a alteração dos parâmetros do processo de otimização influenciava na qualidade geral do sistemas. Para isso foi alterado o número de formigas, de iterações e a taxa de evaporação. Primeiramente, foi definido os seguintes valores padrões:

- Número de Formigas Padrão: 10
- Número de Iterações Padrão: 100
- Taxa de Evaporação Padrão: 0.5

No primeiro teste o número de formigas foi modificado, mas mantendo os demais valores padrões. O resultado encontrado está representado na figura 1. Nesse resultado é possível verificar que o número de 10 formigas foi o melhor para esse problema. O aumento desse parâmetro implicou em uma piora do desempenho do modelo.

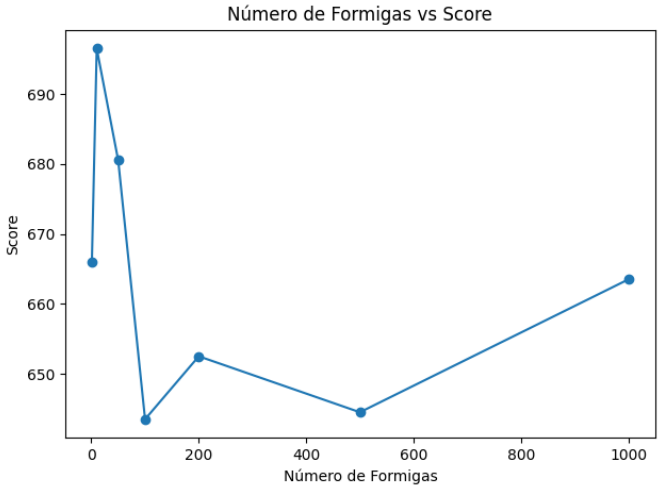


Fig. 1. Score médio no jogo por número de formigas.

No segundo teste o número de iterações foi modificado e os demais valores padrões foram mantidos. O resultado dessa alteração está representado na figura 2. Nessa demonstração é possível verificar que, pelo fato do valor padrão de 10 formigas ser o melhor encontrado nos testes, o número de iterações influencias pouco no resultado geral. O melhor valor foi obtido com 10 iterações. Desse modo, é possível concluir que, para esse problema, o número de iterações teve baixa influencia no score final do algoritmo.

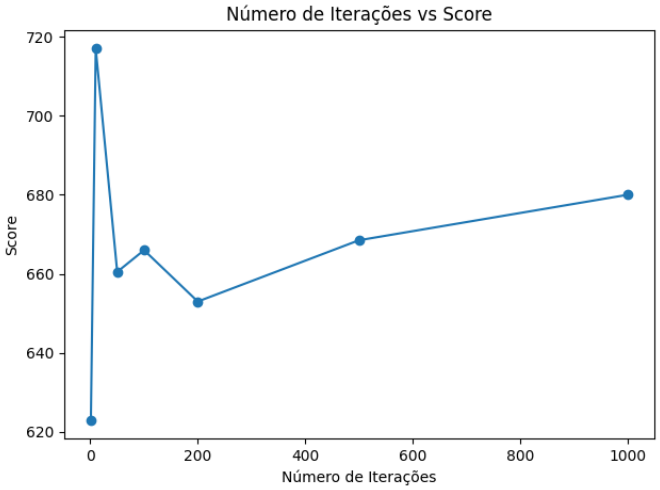


Fig. 2. Score médio no jogo por número de iterações.

O terceiro teste foi realizado modificando a taxa de evaporação e mantendo os demais valores padrões. O resultado do teste está apresentado na figura 3. Com os resultados é possível verificar que o melhor desempenho acontece com uma taxa de 0,6. Com taxa de evaporação igual a 1, o algoritmo perde completamente sua eficiência porque ele vai eliminar completamente os feromônios, tornando o modelo puramente aleatório.

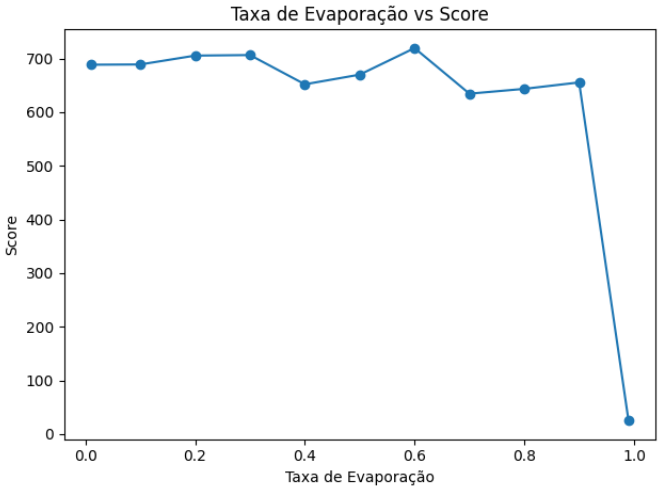


Fig. 3. Score médio no jogo por taxa de evaporação.

## V. Conclusão

Nesse trabalho foi realizado a otimição do jogo Snake com o algoritmo de Otimização por Colônia de Formigas (ACO). Nele redefinimos o problema para incluir várias frutas no tabuleiro. A cobra deve encontrar o melhor caminho para consumir todas as frutas, minimizando o tempo total. O código foi implementado em Python, usando Pygame para a interface gráfica e a biblioteca aco para o algoritmo. O ACO encontra o melhor caminho entre as frutas, que é convertido

em movimentos discretos. O sistema de pontuação incentiva a eficiência, premiando a rapidez no consumo das frutas.

Diversos testes foram realizados. No primeiro teste, ao variar o número de formigas, verificou-se que 10 formigas proporcionaram o melhor desempenho. No segundo teste, alterando o número de iterações, constatou-se que 10 iterações foram suficientes, com pouca influência no resultado geral. No terceiro teste, ao modificar a taxa de evaporação, o melhor desempenho ocorreu com uma taxa de 0,6. Taxas de evaporação muito altas reduziram significativamente a eficiência do algoritmo, tornando-o aleatório.

O desenvolvimento desse trabalho implicou em uma contribuição para o desenvolvimento de algoritmos evolucionários, dado sua aplicação na otimização do comportamento do jogo Snake, anteriormente pouco explorada na bibliografia. Além disso, a análise de parâmetros do algoritmo produziu experimentos interessantes com relação ao método de otimização por colônia de formigas.

**Código disponível em [6].**

## REFERENCES


[1] E. G. Carrano, "Algoritmos evolucionários eficientes para otimizaçao de redes," 2007.

[2] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, p. 28–39, Nov. 2006. [Online]. Available: http://dx.doi.org/10.1109/MCI.2006.329691

[3] A. S. Henrique, V. A. dos Santos, and R. Lyra, "Neat snake: a both evolutionary and neural network adaptation approach," *Anais do Computer on the Beach*, vol. 11, pp. 052–053, 2020.

[4] M. Menezes *et al.*, "Redes neurais profundas e algoritmos genéticos: Estudo de caso: Treinamento de carros autonomos," 2023.

[5] M. D. S. BERETTA, "Técnicas de heurística como agentes inteligentes para o projeto de jogos," 2013.

[6] V. C. et al., "genetic_snake_game," 2024, código fonte disponível no GitHub. [Online]. Available: https://github.com/vinicius-cardoso/genetic_snake_game