Atividade Prática 03
Algoritmos Genéticos

Universidade Tecnológica Federal do Paraná (UTFPR), campus Apucarana
Curso de Engenharia de Computação
Sistemas Inteligentes 1 - SICO7A
Prof. Dr. Rafael Gomes Mantovani

1 Descrição da atividade

Jogar *Snake* sozinho, por si só é fácil. Mas e fazer uma IA para jogar para nós? Como seria? O objetivo deste trabalho é modelar e implementar um Algoritmo Genético (AG) capaz de controlar o jogo da cobrinha (*Snake*). Para facilitar o desenvolvimento, uma estrutura de jogo implementada com programação orientada a objetos (POO) em Python e usando a biblioteca *PyGame*[1] é disponibilizada. Esses arquivos devem ser manipulados para que diferentes indivíduos sejam gerados pelo AG, avaliados, recombinados, e o melhor(es) selecionado(s) ao final da execução. A Figura 1 apresenta uma tela de execução de uma instância do *Snake* feito com a biblioteca *PyGame*, ao qual o agente ainda não conseguiu capturar nenhuma fonte de alimento (por isso o Score = 0). O código necessário para implementação e execução do jogo está disponível no Moodle (e github da disciplina) com o nome `Snake.zip`. Na Tabela 1 estão descritos todos os arquivos contidos neste zip.

Tabela 1: Arquivos contidos no pacote de implementação do *Snake* em Python.

Arquivos que devem ser editados	
`GeneticAlgorithm.py`	Classe que define um Algoritmo Genético (AG) e seus operadores
`Snake.py`	Classe de implementação do jogo *Snake* usando PyGames
`main.py`	O arquivo principal que instancia e roda jogos de *Snake*.
Arquivos que podem ser ignorados	
`Colors.py`	Arquivo com a definição de algumas constantes para valores de cores (interface gráfica)

[1] https://www.pygame.org/news

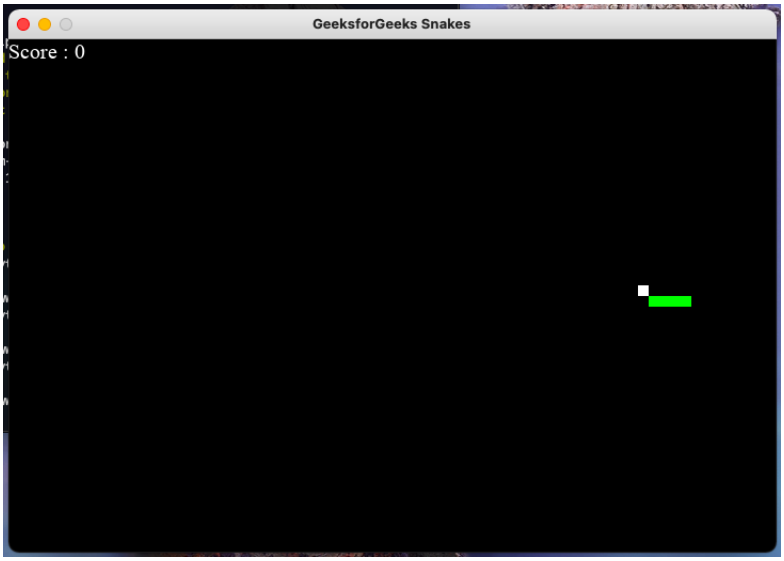

Figura 1: Snake em ação: o clássico jogo da cobrinha implementado em Python, via PyGame.

2 Instruções Gerais

Implemente um Algoritmo Genético (AG) para jogar o Snake, maximizando o tamanho da cobrinha o quanto puder. O projeto e modelagem do problema é livre e sem restrições. Logo, na modelagem em si, é preciso que vocês (equipe) definam:

- O que é um indivíduo no AG? Qual a sua codificação?
- Como implementar uma função de *fitness* que avalie bons indivíduos? Pode ser necessário combinar uma métrica que avalia diferentes questões como: tamanho da cobrinha, tempo que sobrevive durante o jogo, quantidade de alimentos coletados, etc;
- Como selecionar os indivíduos mais aptos em cada geração? Roleta? Torneio? Qual o tamanho do torneio?
- Como realizar a reprodução (*crossover*) dos indivíduos em uma geração?
- Como realizar mutação na codificação escolhida?

Façam a implementação do programa principal no arquivo `main.py`, de maneira que seja possível executar a aplicação com o seguinte comando:

```
python main.py
```

Algumas perguntas e dicas:

1. lembrem-se que a definição dos operadores é diretamente afetada pela forma que os indivíduos são codificados. Justifique todas as escolhas da modelagem adotada pela equipe no relatório;

2. se for o caso, façam execuções que possam mostrar se elitismo é vantajoso ou não;

3. usem um *seed* fixo no arquivo `main.py` para que todos os alimentos sejam gerados em uma mesma ordem durante a execução dos jogos, caso contrário não será possível otimizar nada;

4. além disso, como o AG é aleatório, façam diferentes execuções com diferentes *seeds* para ver como o algoritmo se comporta para diferentes problemas;

5. façam experimentos com diferentes operadores e configurações iniciais considerando tamanhos de população e gerações. Discorra sobre a convergência ou não das soluções.

3 O que entregar?

Vocês (equipe) devem entregar um único arquivo compactado contendo os seguintes itens:

- os arquivos fonte do projeto codificado;
- um relatório (em PDF) apresentando análise e comentários/explicações sobre os resultados obtidos. Descreva e explique também partes relevantes do código implementado.
- O trabalho deve ser submetido pelo Moodle até o prazo final estabelecido na página do curso.

4 Links

- https://github.com/rgmantovani/intelligentSystems1/tree/main/activities/03_genetic_algorithm
- https://pygad.readthedocs.io/en/latest/
- https://pypi.org/project/geneticalgorithm/
- https://towardsdatascience.com/complete-step-by-step-genetic-algorithm-from-scratch-for-global-optimization-6fee5c55dd3b
- https://medium.com/datacat/simple-genetic-algorithm-in-python-from-scratch-d87cd88626c5

Referências

[1] LUGER, George F. Inteligência artificial. 6. ed. São Paulo, SP: Pearson Education do Brasil, 2013. xvii, 614 p. ISBN 9788581435503.

[2] RUSSELL, Stuart J.; NORVIG, Peter. Inteligência artificial. Rio de Janeiro, RJ: Elsevier, 2013. 988 p. ISBN 9788535237016.

[3] LUKE, Sean. Essentials of Metaheuristics. Second edition. Lulu: 2013. Available for free at http://cs.gmu.edu/$\sim$sean/book/metaheuristics/

DE JONG, Kenneth A. Evoluationary Computation: A Unified Approach. The MIT Press: 2006.