UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA ELÉTRICA

GUSTAVO ALVES PACHECO

PROBLEMA DO TOUR DO CAVALO

Uberlândia-MG

2019

GUSTAVO ALVES PACHECO

PROBLEMA DO TOUR DO CAVALO

Trabalho apresentado ao curso de Engenharia de Computação da Universidade Federal de Uberlândia, como requisito parcial de avaliação da disciplina de Algoritmos Genéticos.

Prof.: Keiji Yamanaka

Uberlândia-MG

2019

SUMÁRIO

1. INTRODUÇÃO

No projeto atual, a problemática veio através de um desafio bastante comum no meio do xadrez: O problema do tour do cavalo (do inglês *Knight's Tour*).

Este problema consiste em posicionar o cavalo em um tabuleiro de xadrez, numa posição aleatória e, a partir daí, percorrer todo o tabuleiro, apenas com movimentos válidos, e sem que se passe em uma mesma casa da matriz mais de uma vez.

Assim como anteriormente, torna-se necessário fazer o design dos cromossomos, bem como das funções para maximização e exibição dos resultados.

2. OBJETIVOS

- Desenvolver um algoritmo genético para resolver o problema do tour do cavalo
- Desenvolver novo design para os cromossomos
- Formular a equação a ser trabalhada
- Desenvolver forma de visualização dos resultados
- Verificar impacto dos parâmetros do algoritmo genético, no processo de *tuning*

3. MATERIAIS E MÉTODOS

A linguagem utilizada foi o Racket, uma linguagem de uso geral, baseada em Scheme e Lisp. Como interface de desenvolvimento, foi utilizado o DrRacket.

Para confecção da interface gráfica, utilizou-se a linguagem *racket/gui*, um recurso nativo da própria linguagem Racket. O processo de plotagem foi realizado através das funções do pacote *plot*. Para animação do resultado final, utilizou-se o pacote *2htdp.*

Como métodos, utilizou-se uma abordagem *top-down*, com auxílio de um framework SCRUM para desenvolvimento ágil. Em algumas situações, uma visão *bottom-up* foi adicionada ao projeto, na parte de incremento da própria linguagem, ao criar novos operadores e funcionalidades. O código produzido segue majoritariamente uma abordagem funcional, mas possui elementos procedurais, para facilitar a leitura de algumas funções.

Durante o desenvolvimento, a documentação de funções se mostrou fundamental. Principalmente, para que algumas estruturas de dados fossem melhor especificadas, e a execução do código fosse correta.

4. RESULTADOS E DISCUSSÕES

Inicialmente, foi implementada uma função para animação do resultado do algoritmo genético, que exibisse a movimentação do cavalo por todo o tabuleiro. Tal implementação foi feita a partir dos pacotes *2htdp/image* e *2htdp/universe*. Para cada posição do tabuleiro, é gerada uma nova imagem, na qual o cavalo aparece nessa nova casa e, em seguida, essas imagens são animadas de acordo com a velocidade escolhida na interface gráfica.

Em seguida, a própria interface foi adaptada, removendo o número máximo de gerações, já que o processo deveria acabar quando o caminho fosse encontrado, não sendo necessário ir além. Também foi removido o quadro de exibição das funções, restando apenas o painel de configuração e a tela de performance.

Em seguida, iniciou-se o processo de construção do algoritmo em si. Primeiramente, optou-se por maximizar os caminhos válidos que passam por todas as casas. Nesse caso, o algoritmo utilizaria a mesma estratégia do TSP. O gene seria uma lista de coluna e linha, enquanto o indivíduo seria uma lista de genes.

Entretanto, ao rodar esse algoritmo, não ocorria a convergência desejada. Ao chegar em torno de 57, 58 movimentos válidos, a performance ficava assintótica, sendo necessário um número exorbitante de gerações para avançar o *fitness*.

Logo, uma nova estratégia foi abordada, que tentava maximizar o número de casas inéditas visitadas apenas com movimentos válidos. Nessa estratégia, quase tudo foi mantido da anterior, exceto pelos processos de cruzamento e mutação, que foram feitos como no algoritmo básico.

Novamente, quando aproximadamente 57 casas eram visitadas, o algoritmo travava nessa marca, sem evoluções. Lembrando que com 64 movimentos, o resultado desejado era encontrado.

Começou-se a achar que apenas o algoritmo genético não encontraria os valores esperados. Procurou-se implementar uma forma de adaptação dos indivíduos, durante o cálculo do fitness, que aumentava o tamanho de cada cromossomo, realizando apenas movimentos válidos. Essa estratégia também foi rapidamente descartada, já que apresentou muitos problemas acerca dos operadores genéticos.

Foi feita uma pesquisa e, segundo o artigo Genetic Algorithms with Heuristic (Al-Gharaibeh, Qawagneh, & Al-Zahawi, 2015), realmente não era possível a

convergência dessa maneira. Inspirado pela proposta do artigo, foram feitas adaptações no indivíduo para que fosse representado como uma lista contendo uma posição xy inicial, e uma string binária. Nessa string, cada três bits representavam um dos oito movimentos possíveis para o cavalo, relativos à posição atual que ele se encontra.

No artigo, o processo chamado de adaptação é feito juntamente com o cálculo da aptidão. Entretanto, para que o código fique mais organizado, optou-se por manter esse processo ao final da mutação, adicionando um parâmetro adicional ao AG: a Probabilidade de Adaptação, que permite verificar o impacto desse processo na convergência.

Em linhas mais abstratas, esse processo de adaptação seguiria uma linha de visão mais lamarckiana em respeito à evolução. Nesse caso, as melhorias adaptativas que um indivíduo obtém em vida são passadas para as próximas gerações.

Aplicado ao algoritmo genético, a adaptação funciona da seguinte forma. A string de movimentos é percorrida, sendo que os movimentos válidos são apenas passados adiante. Quando uma posição inválida é encontrada, faz-se necessário mudar o trecho na string de movimento que levou àquela posição. Para realizar essa mudança, pega-se a casa anterior à inválida, calcula-se todas as posições válidas para as quais o cavalo poderia ter pulado e, para cada uma dessas casas, faz-se uma contagem do número de casas válidas a partir delas. Com isso, há uma certa previsão da dificuldade de acesso posterior à cada uma dessas casas. Portanto, a casa mais difícil de ser acessada (com menos movimentos válidos possíveis) é escolhida. O trecho da string é, então, trocado, para que o movimento caia nessa casa escolhida.

A adaptação para assim que não é possível realizar movimentos válidos. Esse mesmo processo ocorre para calcular o fitness. Há a contagem de movimentos válidos só até o algoritmo encontrar um movimento inválido. A partir desse movimento, nenhum outro entra na contagem de aptidão desse indivíduo.

Realizando essa adição, o processo melhorou imensamente, chegando a encontrar a solução em duas gerações, enquanto que anteriormente eram necessárias em torno de 2500 para alcançar a marca de 57 movimentos válidos. Abaixo estão representadas as telas da interface, com o gráfico de performance, exibindo a influência da heurística na otimização da função.

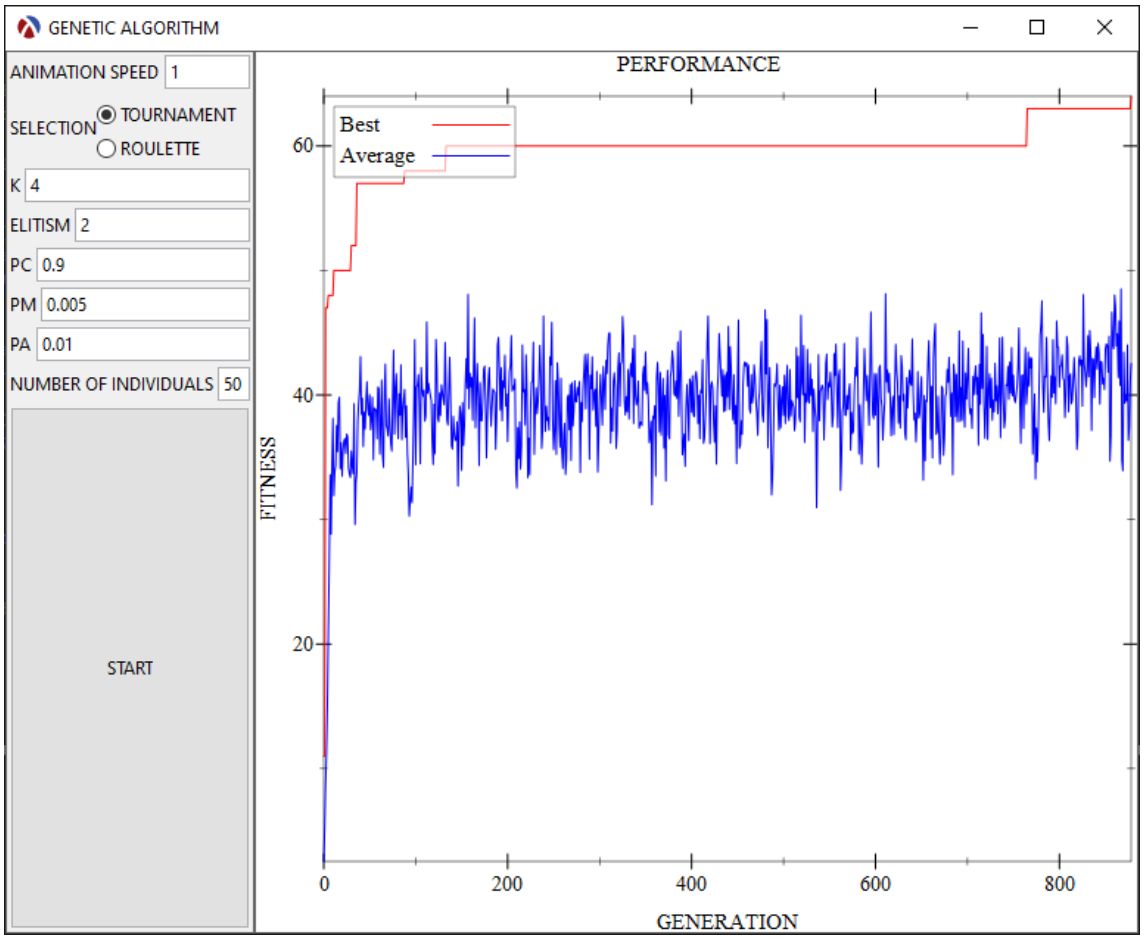

Figura 1: Adaptação baixa, foram necessárias 900 gerações para encontrar o resultado em uma população de 50 indivíduos

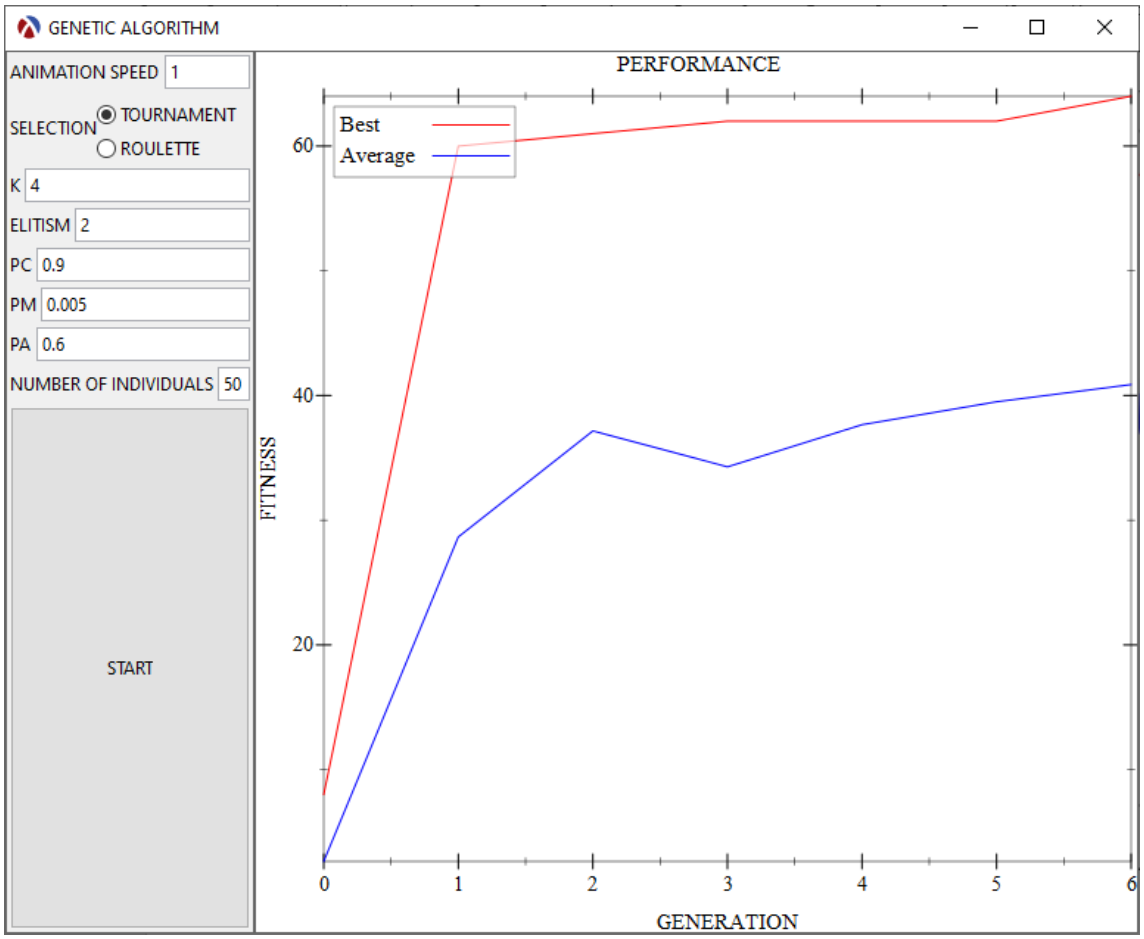

Figura 2: Adaptação habilitada, foram necessárias apenas 6 gerações para o mesmo número de indivíduos

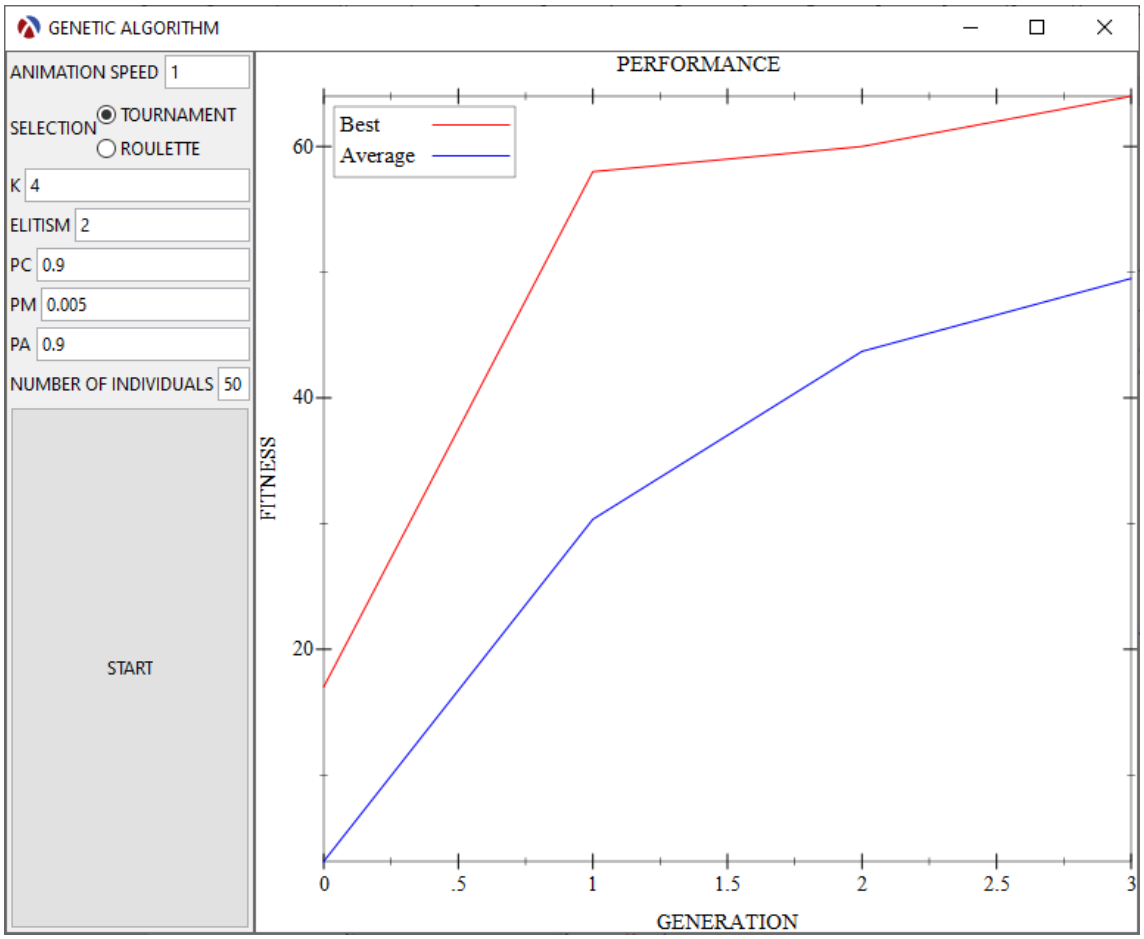

Figura 3: Resultados ainda melhores para valores mais altos de PA.

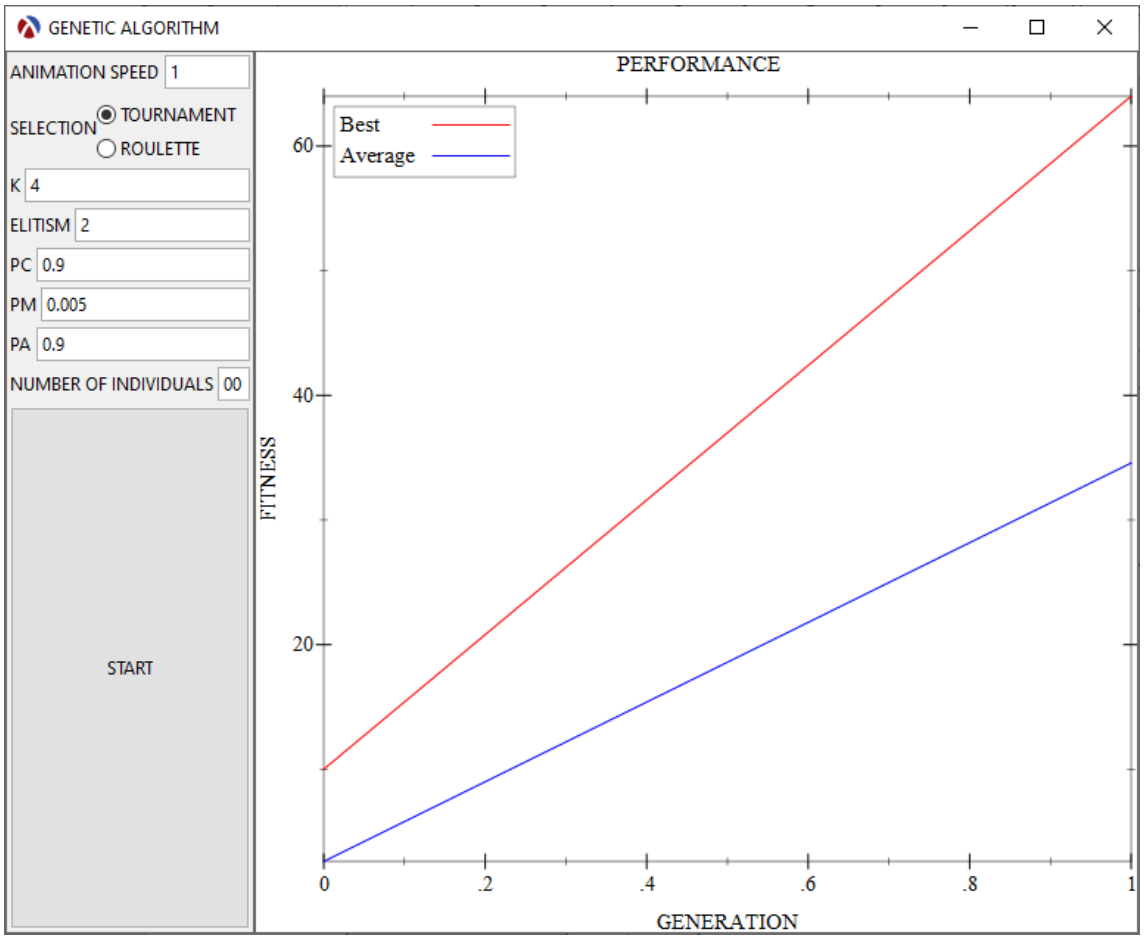

Figura 4: Convergência em duas gerações, utilizando 200 indivíduos e PA alto.

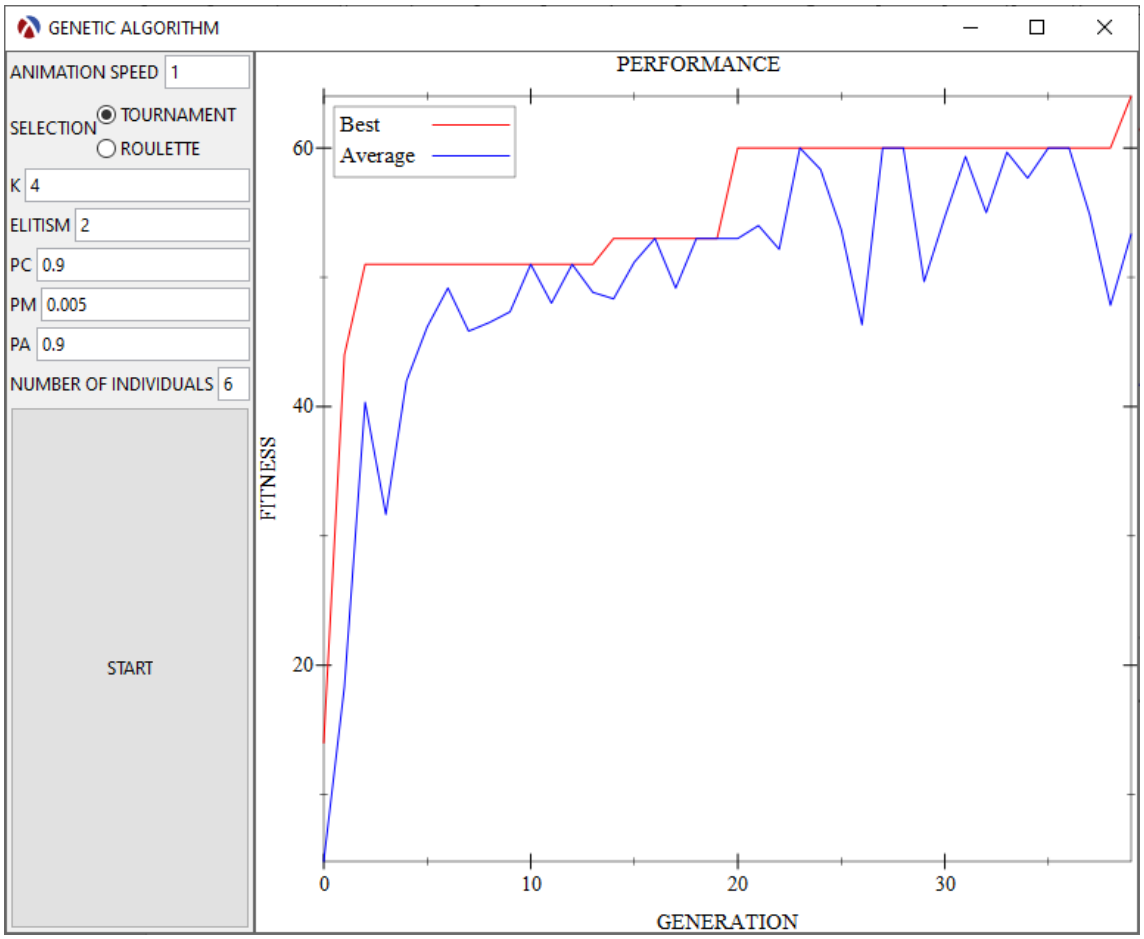

Figura 5: Apesar da presença da adaptação, com poucos indivíduos é possível ver as curvas características encontradas até o momento.

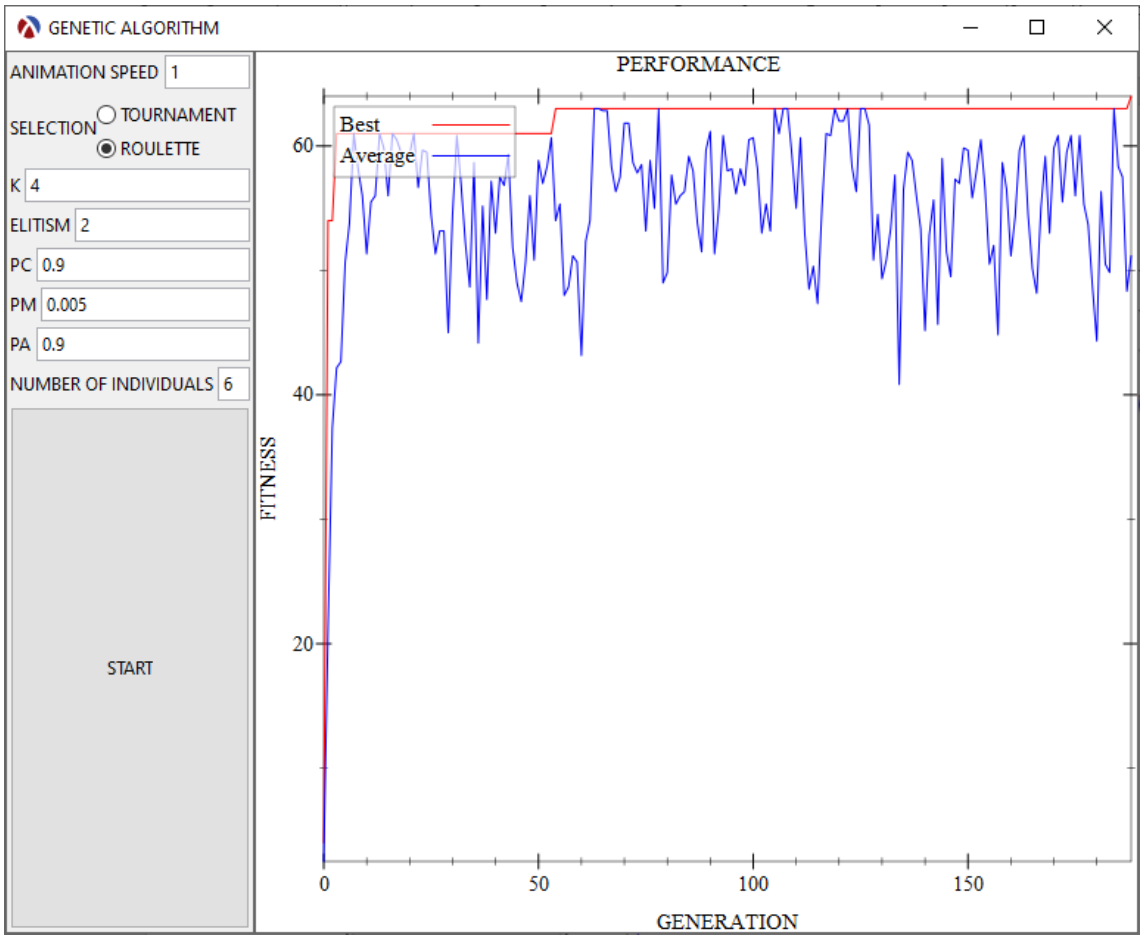

Figura 6: Mesmas configurações da anterior, apenas trocando de torneio para roleta. É evidente a característica evolucionária na performance.

Figura 7: Execução e término da animação dos movimentos.

5. CONCLUSÃO

A adição da adaptação foi fundamental para o funcionamento, além de trazer resultados extremamente satisfatórios, chegando a encontrar o resultado na segunda geração.

Por outro lado, é possível questionar se tal algoritmo ainda se enquadra como evolucionário (ou genético), já que existe outro facilitador para a convergência.

Entretanto, apesar de que tenham sido feitas modificações substanciais à estrutura do AG, implementando um operador genético que não existe na evolução real, ainda é garantida a natureza evolucionária.

Pelos testes, é possível ver as mesmas curvas de performance encontradas anteriormente, além de que há o uso de crossover, mutação, fitness e seleção. Tudo isso continua a enquadrar esse tipo de código como genético.

O projeto se mostrou bastante desafiador, sendo necessária a adição de novas técnicas e estratégias, mas foi essencial para a evolução do conhecimento na área.

6. REFERÊNCIAS BIBLIOGRÁFICAS

Al-Gharaibeh, J., Qawagneh, Z., & Al-Zahawi, H. (29 de Outubro de 2015). Genetic Algorithms with Heuristic. *Research Gate*.

Mallawaarachchi, V. (7 de Julho de 2017). *Introduction to Genetic Algorithms*. Fonte: Towards Data Science: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

Montesanti, J. d. (s.d.). *Seleção natural*. Fonte: InfoEscola: https://www.infoescola.com/evolucao/selecao-natural/

Tomassini, M. (s.d.). A Survey of Genetic Algorithms. *Annual Reviews of Computational Physics, Volume III*.

Yamanaka, P. K. (Agosto de 2017). *ALGORITMOS GENÉTICOS: Fundamentos e Aplicações*.

Zini, É. d., Neto, A. B., & Garbelini, E. (18 de Novembro de 2014). ALGORITMO MULTIOBJETIVO PARA OTIMIZAÇÃO DE PROBLEMAS RESTRITOS APLICADOS A INDÚSTRIA. *Congresso Nacional de Matemática Aplicada à Indústria*.