UNIVERSIDADE FEDERAL DO SUL E SUDESTE DO PARÁ

INSTITUTO DE GEOCIÊNCIAS E ENGENHARIAS

FACULDADE DE COMPUTAÇÃO E ENGENHARIA ELÉTRICA

WARLEY RABELO GALVÃO

KAYRO SANTOS COSTA

IAGO COSTA DAS FLORES

GUSTAVO OLIVEIRA LACERDA

MARABÁ, PA

2022

WARLEY RABELO GALVÃO

KAYRO SANTOS COSTA

IAGO COSTA DAS FLORES

GUSTAVO OLIVEIRA LACERDA

Relatório elaborado com o intuito de obtenção parcial do conceito da disciplina de Inteligência Artificial do curso de Bacharelado em Engenharia da Computação da Universidade Federal do Sul e Sudeste do Pará ministrada pelo Docente Elton Rafael Alves.

MARABÁ, PA

2022

**Sumário**

[1. Introdução 4](#_Toc105014384)

[2. Problema Proposto 4](#_Toc105014385)

[3. Implementação 4](#_Toc105014386)

[4. Resultados 11](#_Toc105014387)

[5. Conclusão 13](#_Toc105014388)

[6. Referências 13](#_Toc105014389)

# 1. Introdução

Na disciplina de Inteligência Artificial foi apresentado uma problemática de transporte de produtos para os discentes do curso de Engenharia da Computação, no qual o objetivo é realizar um Algoritmo Genético, com o intuito de maximizar o valor do lucro da empresa no setor de transporte de produtos, levando em consideração o volume de cada produto e a capacidade total suportada por cada caminhão, será descrito a seguir como ocorreu a construção desse Algoritmo Genético e serão comparados os resultados obtidos com elitismo e sem elitismo.

# 2. Problema Proposto

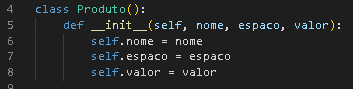
Diante das atuais circunstâncias econômicas do petróleo, reduzir os gastos em combustíveis é uma ótima saída para economizar, principalmente em uma empresa que depende ativamente de derivados de petróleo, como uma empresa de entregas. Assim é nítida a necessidade de buscar formas de otimizar é mitigar o uso de combustíveis.

Neste relatório iremos abordar o uso do algoritmo genético para obter combinações ideais de transporte de produtos baseando-se no preço e no volume de cada produto, afim de além de diminuir na conta do fornecedor de combustível, maximizar os lucros em cada transporte de mercadorias até seus clientes.

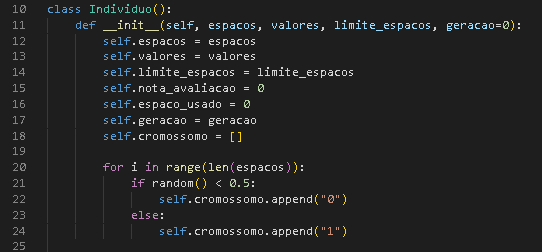
Na empresa fictícia que iremos simular nesta problemática, teremos em posse um caminhão de capacidade de 3m³, e disponível a venda a variedade de 16 produtos de preços e dimensões distintas, totalizando o total de 4.79m³. Será o objetivo do algoritmo genético, propor a melhor combinação possível de transporte.

# 3. Implementação

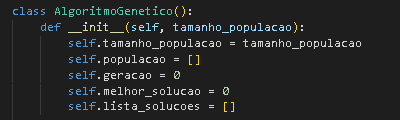
Na implementação fizemos uso de 3 classes Produto na figura 01, Indivíduo na figura 02 e AlgoritmoGenético na figura 03.



**Figura 1: Classe Produto**

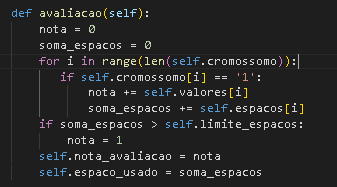


**Figura 2: Classe Indivíduo**

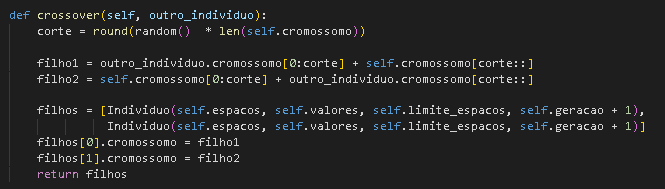


**Figura 3: Classe AlgoritmoGenético**

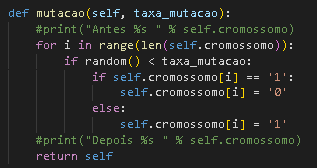
A classe produto é usada para carregar os dados de cada produto como nome, volume e valor. A classe Indivíduo serve para usarmos as funções de crossover, mutação e avaliação que são inerentes ao indivíduo em uma população de um algoritmo genético. Cada indivíduo vai ser equivalente a uma lista de bits 0 ou 1 de 14 posições, onde cada posição se refere a um dos 14 produtos que podemos colocar dentro do caminhão.



**Figura 4: Função avaliação da classe indivíduo**



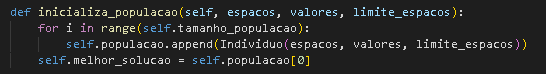
**Figura 5: Função Crossover da classe indivíduo**



**Figura 6: Função mutação da classe indivíduo**

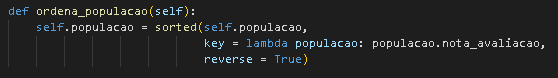
Na classe AlgoritmoGenetico temos a função inicializa\_populacao na figura 07, ordena\_populacao na figura 08, melhor\_individuo na figura 09, soma\_avaliacoes na figura 10, seleciona\_pai\_roleta\_viciada na figura 12, visualiza\_geracao na figura 11 e resolver na figura 12. As funções da classe AlgoritmoGenetico são auxiliares da função principal que é a função resolver.

A função inicializa\_populacao é responsável por inicializar a geração 0 (zero) do algoritmo genético.



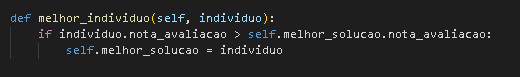
**Figura 7: Função inicializa\_populacao**

A função ordena\_população serve para ordenar os indivíduos, baseado na sua nota de avaliação da maior nota para a menor como pode ser visto na figura 08.



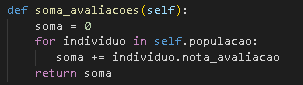
**Figura 8: Função oderna\_populacao**

A função melhor\_individuo serve para salvar o melhor indivíduo de cada geração, ela compara a nota do indivíduo na avaliação e se essa nota for maior do que a nota da melhor solução da avaliação a gente salva esse melhor indivíduo.



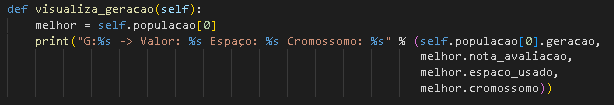
**Figura 9: Função melhor\_individuo**

A função soma\_avaliações serve para retornar o valor total de todas as avaliações dos indivíduos da geração atual ela é usada na função da roleta viciada para gerar as probabilidades de cada indivíduo para ser selecionado como pai/mãe da próxima geração.



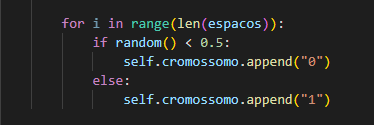
**Figura 10: Função soma\_avaliacoes**

A função visualiza\_geracao serve apenas para mostrar qual o melhor indivíduo encontrado em cada geração.



**Figura 11: Função visualiza\_geracao**

Na função resolver é onde acontece propriamente a resolução do problema através das gerações de populações. Explicando mais a fundo essa função, primeiro ela começa chamando a função inicializa\_populacao para carregar a primeira população da geração 0, transformando os dados reais dos produtos em uma lista binária, quando o produto recebe 1, significa que o produto vai ser levado e quando recebe 0 significa que ele não será levado no caminhão. É válido lembrar que essa primeira geração será gerada totalmente aleatória com 50% de chance de um produto ser levado ou não. Realizando essa operação nessa parte do código que fica dentro da função individuo

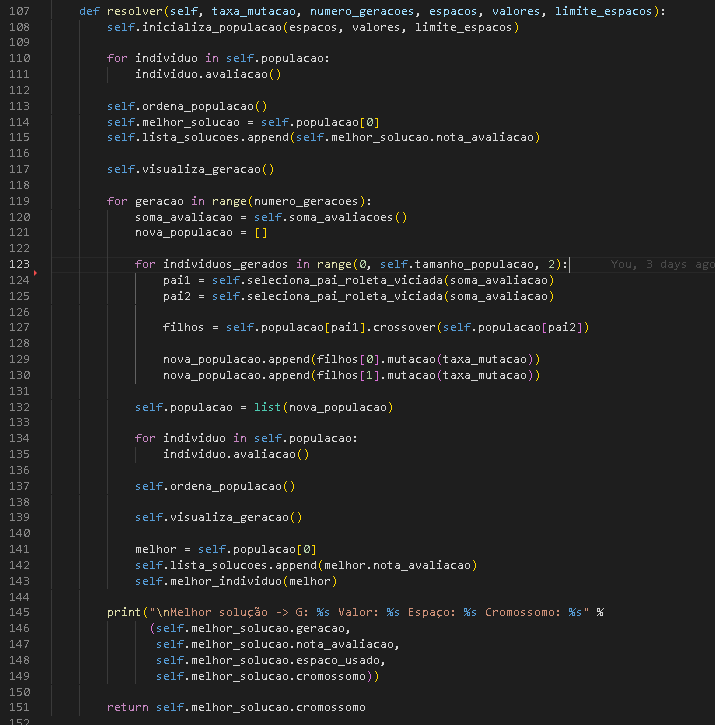


**Figura 12: Probabilidade dos produtos**

Após inicializado a geração 0 (zero) aleatoriamente. Cria-se um looping para avaliar cada um dos indivíduos e salvar sua nota. Com isso a função avaliação descrita na figura 04 da classe Indivíduo é chamada e ela calcula o fitness que será o valor máximo que o indivíduo consegue chegar sem ultrapassar o espaço/volume de 3 (três) metros cúbicos. Caso ultrapasse o volume permitido a nota é declarada como 1 que é a pior possível o que quase descarta por completo a chance desse indivíduo de continuar como pai da próxima geração.

Prosseguindo, na linha 117 temos a visualização do melhor indivíduo com a melhor nota de avaliação ou fitness da geração atual. Logo após entra-se no looping que irá calcular geração por geração, até quantas gerações forem programadas para ser realizadas. Desse modo, ele executará no loop a operação de crossover com metade da população de pares de pais escolhidos aleatoriamente com a função da roleta viciada, que pode ser visto na figura 14.

Cada par de pais irá gerar dois novos filhos, o que retorna à população novamente apenas com os filhos gerados. Feito o crossover, é realizado e ativado a função de mutação descrita na figura 06, em cada filho com 1% de chance de acontecer em cada cromossomo de cada filho. Com isso o loop continuará até esgotar todas as gerações.

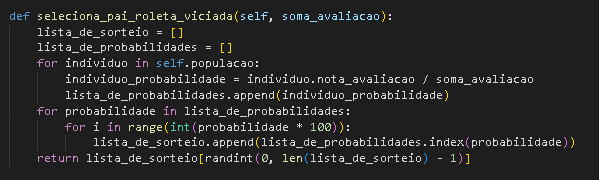


**Figura 13: Função resolver sem elitismo**

**Seleção via roleta viciada**

Na função da roleta viciada temos dois loopings, o primeiro é para calcular a probabilidade de cada indivíduo de ser escolhido como pai/mãe dos novos indivíduos da próxima geração, o segundo é onde temos um loop interno no qual é criado uma lista de 100 posições. A partir da probabilidade de cada indivíduo é obtido a quantidade posições que ele tem direito na lista de sorteio.

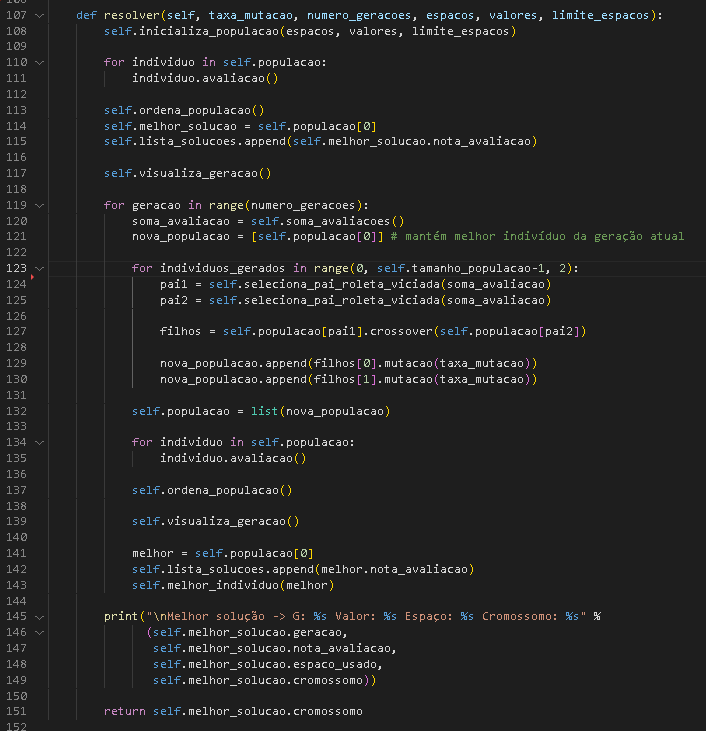
Exemplo para um indivíduo de 54% de chance de ser escolhido ele terá direito a 54 posições na lista de sorteio, assim sucessivamente até a lista ser preenchida completamente.



**Figura 14: roleta\_viciada**

**Elitismo**

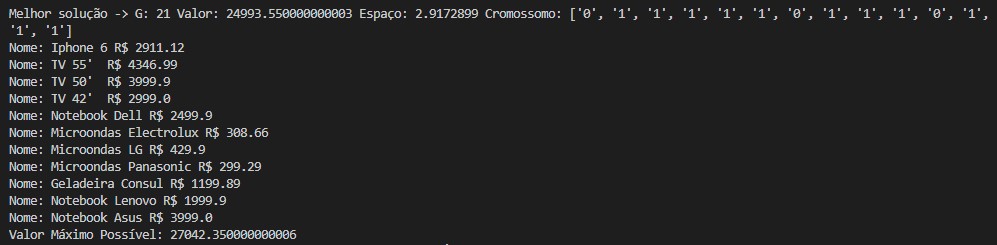
Para ativarmos o elitismo no AlgoritmoGenetico basta mantermos o melhor indivíduo da geração atual para a próxima geração de indivíduos. O que pode ser observado na linha 121 da figura 14.



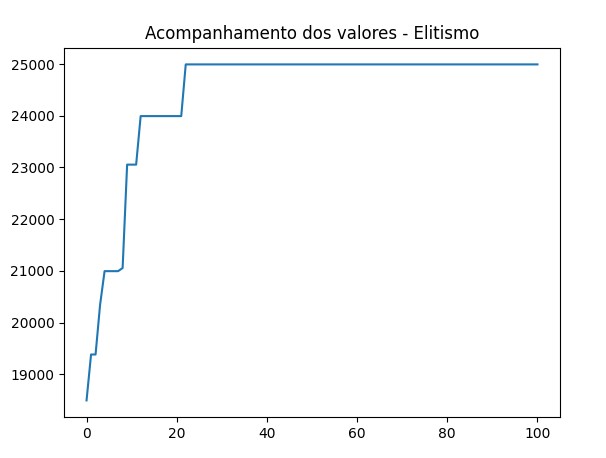
**Figura 14: resolver com elitismo**

# 4. Resultados

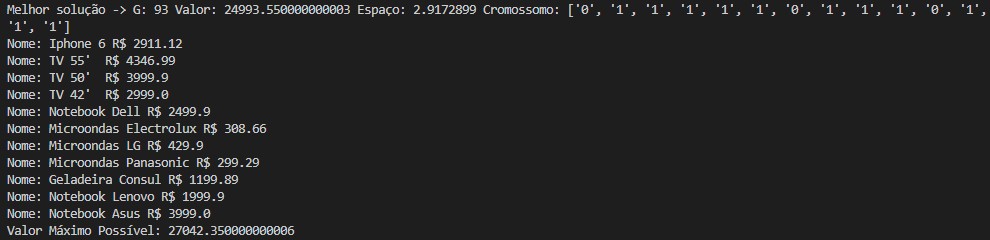
* Comparação com e sem elitismo



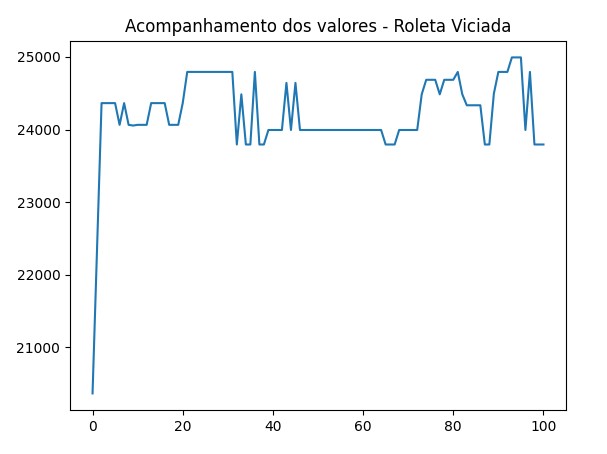
**Figura 15: Resultado com elitismo no cmd terminal**



**Figura 16: Resultado com elitismo gráfico**



**Figura 17: Resultado sem elitismo no cmd terminal**



**Figura 18: Resultado sem elitismo no cmd terminal**



**Figura 19: init\_main sem elitismo**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sem elitismo** | | | **Com elitismo** | | |
| **Geração (melhor valor)** | **Espaço ocupado** | **Valor de mercadorias** | **Espaço ocupado** | **Valor de mercadorias** | **Geração (melhor valor)** |
| 93 |  | R$ 24.993,55 |  |  |  |

**Tabela 1: Comparação de resultados**

**5. Conclusão**

. Para verificar os resultados de forma mais resumida, clara e objetiva basta verificar a tabela 1 e ao analisar os resultados obtidos é possível perceber que os resultados foram iguais ocupando o mesmo espaço e o mesmo valor de soma de mercadorias que serão transportados pelo veículo da empresa, porém podemos perceber que usando o método com elitismo foi possível alcançar os resultados satisfatórios bem mais cedo na geração 21 em quanto que o método sem elitismo alcançou o mesmo resultado somente na geração 93, portanto podemos afirmar que o método com elitismo nessa ocasião obteve uma otimização de tempo maior para alcançar os resultados esperados. Pata maiores esclarecimentos os códigos fontes serão enviados via sigaa para correção e análise juntamente com este relatório.

**6. Referências**

1.MUNDIM,A. Algoritmo Genético na prática com Python! Disponível em: <https://alexandremundim.medium.com/algoritmo-gen%C3%A9tico-na pr%C3%A1tica-com-python-e24d2b1254d8>.Acesso em: 2 jun.2022

2.HEISE,L.M. Algoritmos Genéticos. Disponível em: <https://medium.com/turing-talks/turing-talks-8-algoritmos-gen%C3%A9ticos-a791c25bd7ba>.Acesso em: 2 jun.2022.