# Algoritmo Genético Com representação de Parâmetros por Números Reais

Vítor Rezende Silva

Abril 2024

## 1 Implementação

O algoritmo genético implementado usa dimensão n = 3, e recebe as seguintes variáveis como entrada:

- tc: taxa de cruzamento.
- tm: taxa de mutação.
- n: número de gerações.
- num_indivíduos: número de indivíduos por geração.
- cruzamento: forma como o algoritmo realizará o cruzamento, podendo ser "BLXa" para BLX$-\alpha$ ou "BLXab" para BLX$-\alpha\beta$.
- plot: variável para decidir se ao fim da execução o algoritmo vai ou não gerar um gráfico com as gerações formadas, podendo ser 1 para sim e 0 para não.

Foi implementado o elitismo sempre, ou seja, a cada iteração é salvo o indivíduo com o melhor fit e ele é passado para a próxima geração.

A seleção de pais é feita por meio de uma roleta, e a geração do conjunto de filhos é feita sorteando dois indivíduos aleatórios do conjunto de pais e realizando o cruzamento entre eles, isso até que o conjunto de filhos tenha o tamanho da população definida.

Ao fim da execução o algoritmo retornará os valores de fitness do melhor indíduo, e a média de todos os melhores fitness de cada geração. Além disso, como dito anteriormente, pode ser gerado um gráfico com os melhores fitness de cada geração, e todos esses valores são registrados num arquivo chamado "gen.txt".

## 2 Calibragem dos parâmetros

Para calibrar o algoritmo genético foram realizadas três etapas de testes, cada uma feita com a ajuda de um script em Python para automatizar os testes.

Na primeira etapa o algoritmo foi executado selecionando alguns valores para cada um dos parâmetros, e executando o AG para todas as combinações de parâmetros.

Abaixo está o script usado para essa etapa.

```
import AG
import csv

tm = [1, 5, 10]
tc = [50, 80, 100]
num_individuos = [25, 50, 100]
n = [25, 50, 100]
cruzamento = ["BLXa", "BLXab"]

for i in tc:
    for j in tm:
        for k in n:
            for l in num_individuos:
                for m in cruzamento:
                    fitness, media = AG.AG_real(i, j, k, l, m, 0)

                    with open("resultados.csv", "a") as file2:
                        csv_writer = csv.writer(file2)
                        csv_writer.writerow([i, j, k, l, m, fitness, media])
```

Os resultados foram salvos num arquivo csv, que foi importado para um gerenciador de planilhas, onde depois foram salvos os parâmetros utilizados para encontrar o menor valor entre eles, no caso, $4.440892098500626 * 10^{16}$.

Na segunda etapa, foram coletados os parâmetros salvos na etapa anterior, e usados novamente para a execução do AG. Cada um dos conjuntos de parâmetros foram executados 10 vezes usando o seguinte script:

```
import AG
import csv

entradas = [[80, 1, 100, 100, 1, 0], [80, 5, 50, 100, 1, 0],
            [80, 5, 100, 100, 1, 0], [80, 10, 100, 100, 1, 0],
            [100, 1, 50, 100, 1, 0], [100, 1, 50, 100, 0, 0],
            [100, 1, 100, 100, 1, 0], [100, 5, 50, 100, 1, 0],
            [100, 5, 100, 100, 0, 0], [100, 5, 100, 100, 1, 0],
            [100, 10, 100, 100, 0, 0], [100, 10, 100, 100, 1, 0]]
```

```
with open("resultados2.csv", "a") as file2:
    for i in entradas:
        for x in range(10):
            if i[4] == 0:
                y = 'BLXa'
            else:
                y = 'BLXab'

            fitness, media = AG.AG_real(i[0], i[1], i[2], i[3], y, i[5])


            csv_writer = csv.writer(file2)
            csv_writer.writerow([x, i[0], i[1], i[2], i[3], y, fitness, media])
```

Ao fim da execução, os parâmetros e o fitness de cada conjunto é salvo novamente em um outro arquivo csv, onde vamos filtrar novamente o conjunto de parâmetros, selecionando aqueles encontraram o menor valor mais vezes.

Com esses dados é realizada a terceira e ultima etapa de testes, onde executamos o algoritmo através do seguinte script:

```
import AG

entradas = [[80, 1, 100, 100, "BLXab", 1], [80, 5, 100, 100, "BLXab", 1],
            [80, 10, 100, 100, "BLXab", 1], [100, 1, 100, 100, "BLXab", 1],
            [100, 5, 100, 100, "BLXab", 1], [100, 10, 100, 100, "BLXab", 1]]

for i in entradas:
    fitness, media = AG.AG_real(i[0], i[1], i[2], i[3], i[4], i[5])
```

Ao contrário dos demais, nesse não salvamos os dados dos parâmetros, mas são gerados os gráficos das gerações de cada execução do AG. Em todas as execuções do AG nessa etapa o menor valor foi encontrado.

Observando os gráficos abaixo pode-se perceber que em todos os casos o algoritmo achou o menor valor por volta da vigésima iteração.

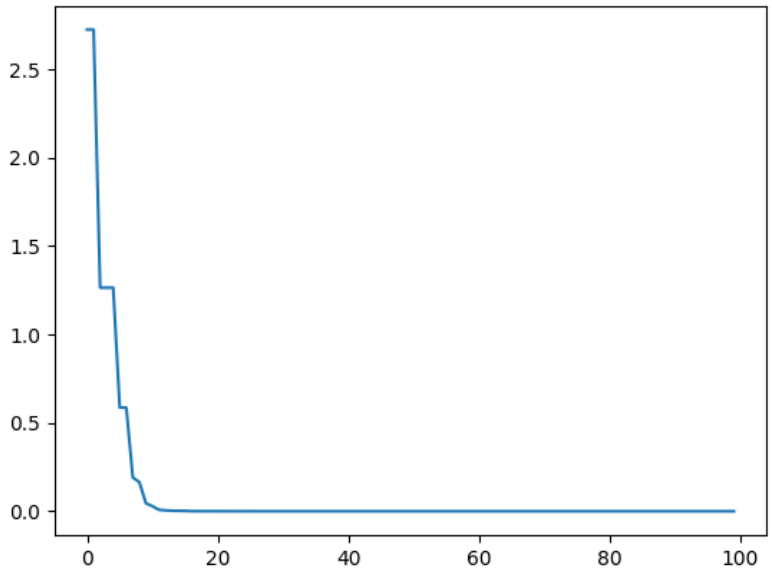


Figure 1: $tc = 80; tm = 1, n = 100, num\_individuos = 100, cruzamento = BLXab$

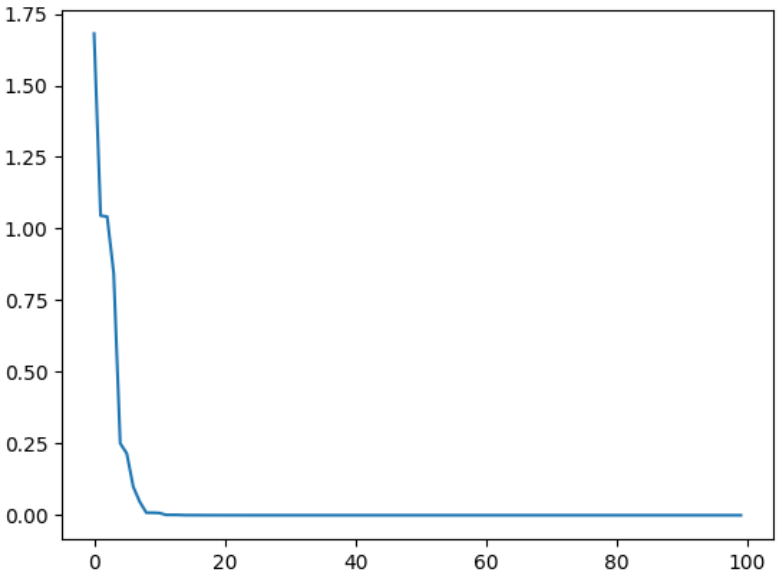


Figure 2: $tc = 80; tm = 5, n = 100, num\_individuos = 100, cruzamento = BLXab$

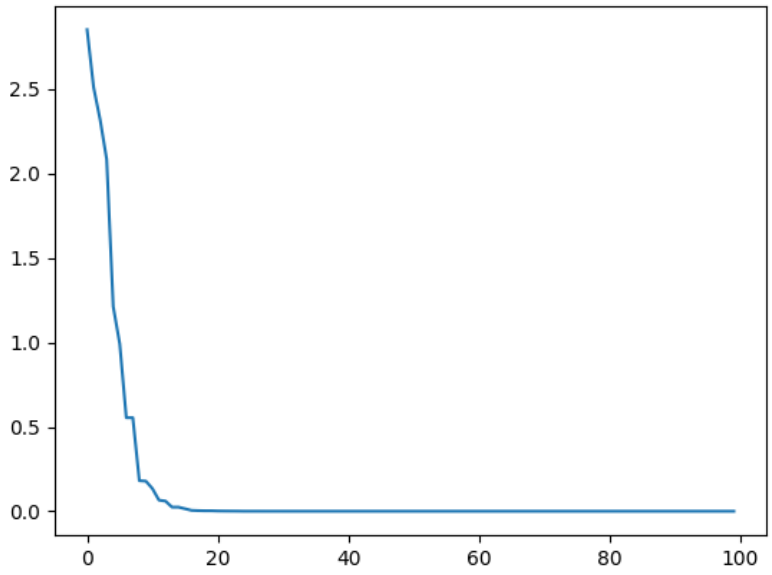


Figure 3: $tc = 80; tm = 10, n = 100, num\_individuos = 100, cruzamento = BLXab$

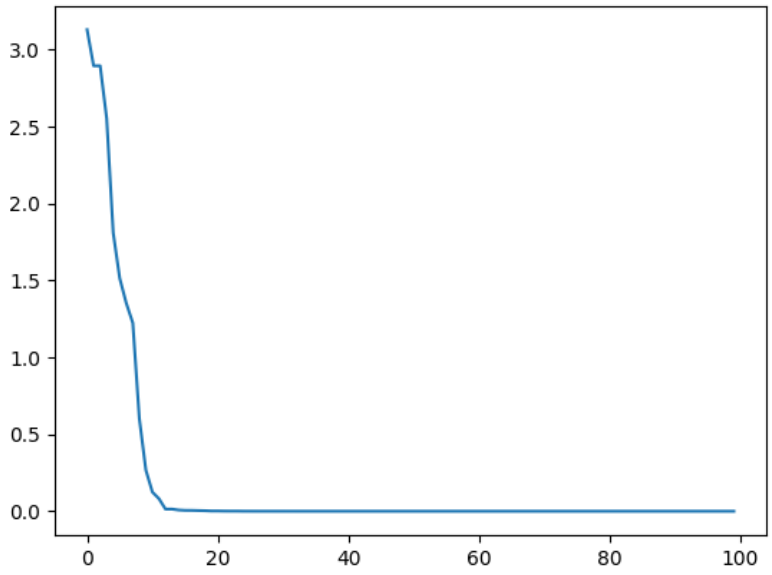


Figure 4: $tc = 100; tm = 1, n = 100, num\_individuos = 100, cruzamento = BLXab$

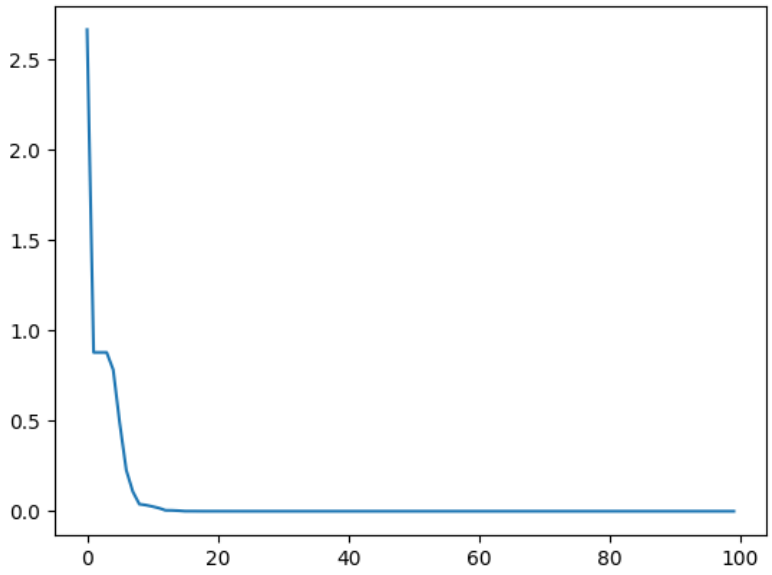


Figure 5: $tc = 100; tm = 5, n = 100, num\_individuos = 100, cruzamento = BLXab$

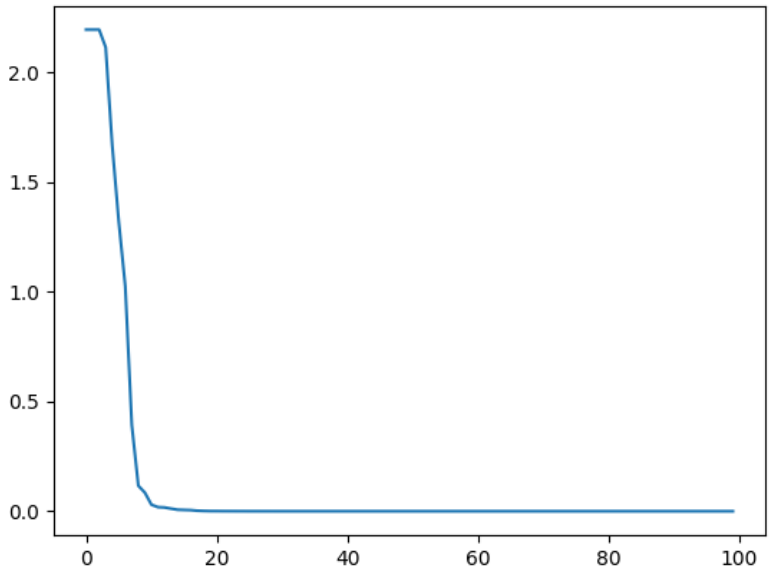


Figure 6: $tc = 100; tm = 10, n = 100, num\_individuos = 100, cruzamento = BLXab$