ANTÔNIO CARLOS DURÃES DA SILVA
JOEL WILL BELMIRO

GitHub: https://github.com/duraes-antonio/IA_trab4_AG_real

INTELIGÊNCIA ARTIFICIAL – 4º TRABALHO :
ALGORÍTIMO GENÉTICO COM NÚMEROS REAIS

1 EXPLICAÇÃO TEÓRICA DO ALGORITMO

Inspirado em conceitos pertencentes à Teoria da Seleção Natural, registrada em nome de Charles Robert Darwin [1], o algorítimo genético visa tirar proveito de mecanismos semelhantes aos naturais para selecionar, modificar ou criar uma nova solução para um problema a partir de soluções pais.

O algorítimo genético herda da Biologia não só os mecanismos para encontrar a melhor solução mas também a modelagem utilizada para definir a estrutura de uma solução, ou melhor, de um indivíduo. Cada valor que representa uma solução pode ser modelado como um indivíduo, contendo seu código genético e sua aptidão.

É possível traçar uma analogia, em que uma fórmula matemática ou objetivo representa esse ambiente responsável por definir quem é mais adequado. Dessa forma, o código genético (simboliza um parâmetro) do indivíduo aliado ao ambiente (fórmula ou processamento que recebe a entrada) determina o quão apto (valor fitness) cada indivíduo é, sendo possível, selecionar tal indivíduo, reproduzi-lo e impulsionar pequenas alterações em sua genética.

O algoritmo favorece a descoberta de bons resultados para, principalmente, problemas em que sua solução final não é conhecida ou que não há uma solução ótima alcançável [2], sendo satisfatórias, soluções que oscilem entre uma faixa de aceitabilidade, graças à sua aleatoriedade, seleção e variação de soluções promissoras.

2 PROBLEMA PROPOSTO

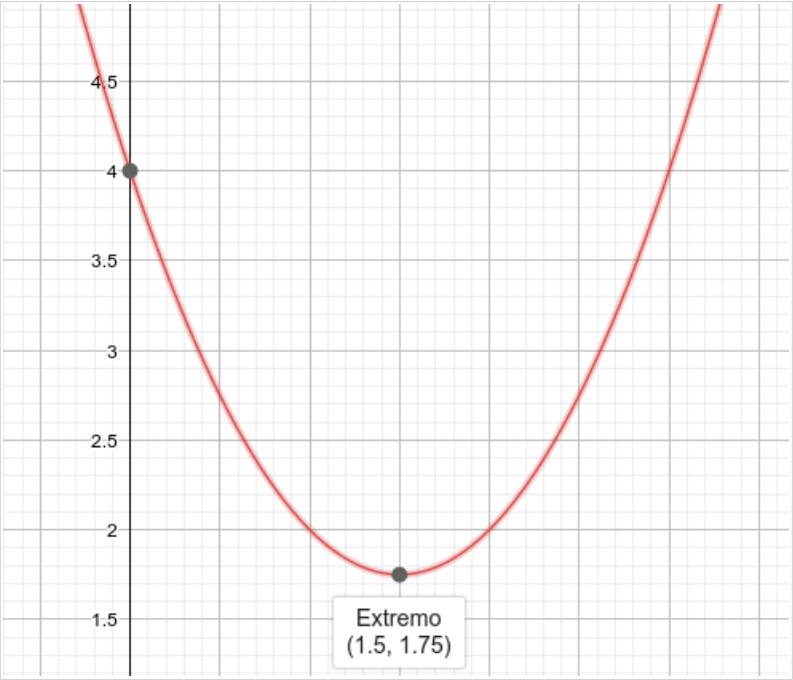
O algorítimo teve como tarefa encontrar o valor para o parâmetro X que minimizasse o resultado da função abaixo:

Fórmula 1: Função a ser minimizada

$$f(x) = x^2 - 3x + 4$$

Após plotar o gráfico 2D da função acima na ferramenta Geogebra, fica claro que seu ponto de mínimo é de 1.75, e é atingido para X com valor de 1.5.

Gráfico 1: Plotagem da fórmula 1



Desta vez, haverá um foco maior na comparação dos resultados entre os operadores genéricos que foram escolhidos como melhores para se trabalhar neste problema (com números reais) e o resultado do trabalho anterior. Foram usados diversas formas de crossover e mutação, mas para comparação serão usados o Crossover Linear e a Mutação Uniforme, pois foram os que obtiveram os melhores resultados.

3 MUDANÇAS NA IMPLEMENTAÇÃO

3.1 INDIVÍDUO

Modelagem: Não houve grandes mudanças na implementação, tirando o fato de trocar a string de bits que representava os genes/cromossomos por um número real, retirando a necessidade de normalização.

3.2 POPULAÇÃO

Modelagem: A única mudança significativa de modelagem é a respeito da possibilidade de se passar por parâmetro qual método de crossover e mutação deve ser usada.

Crossover: Nesta versão, o crossover só gera um filho quando a taxa de crossover é “aceitavel”. Logo, pode ser que a quantidade de filhos gerados seja maior que a quantidade de individuos inicial, sendo necessário eliminar o pior indivíduo da população.

Mutação: Também não sofreu grandes alterações, tendo somente a assinatura alterada. Recebe 2 parâmetros: g_atual e g_max, número da geração atual e o valor máximo de gerações respectivamente. Esses parâmetros podem ser necessários para algumas classes que aplicam de fato a mutações.

3.3 CLASSES DE CROSSOVER E MUTAÇÃO

Modelagem: Foram criadas classes abstratas para estes operadores genéricos para padronizar as assinaturas de funções, e para cada tipo destes operadores foram feitas classes concretas que implementam os métodos da classe pai abstrata. São essas as classes que devem ser passadas para a classe de população, alterando assim a forma como são feitos os crossovers e mutações daquela população.

4 EXEMPLO DE USO

4.1 EXEMPLO DE CHAMADA E ENTRADAS

O algoritmo foi estruturado sob uma aplicação de linha de comando (CLI), portanto, os parâmetros e taxas envolvidas são livres para entrada do usuário. Abaixo está um exemplo de chamada num terminal Linux com 4 indivíduos na população, 1% de taxa de mutação, e 60% de taxa de crossover respectivamente:

```
$ python3 -m src.main -i 4 -m 1 -c 60
```

Em caso de dúvidas sobre cada parâmetro, basta chamar a aplicação passando o parâmetro '-h':

```
$ python3 -m src.main -h
```

Observação: Para sistemas Windows, a chamada é a mesma, porém em vez de utilizar “python3” para chamar a linguagem, usa-se apenas “python”. O Terminal deve estar aberto na pasta logo acima da pasta “src”, senão ter um erro de importação de módulo (ModuleNotFoundError: No module named 'src')

Explicação sobre parâmetros de entrada:

- -i / --individuo: Número de indivíduos;
- -m / --mutacao: Porcentagem de chance de ocorrer mutação por bit;
- -c / --crossover: Porcentagem de chance de ocorrer crossover entre 2 indivíduos;

4.2 SAÍDA

A saída após a execução do programa é um arquivo .CSV para cada iteração e configuração. Por exemplo, para 5 gerações, 10 execuções, serão gerados 10 arquivos, um para cada execução.

O nome do arquivo segue o padrão:

{número_indivíduos}i_{número_gerações}g_{número_execução_atual}exec.csv

Exemplos: 4i_5g_1exec.csv, 4i_5g_2exec.csv, 4i_5g_3exec.csv, ...

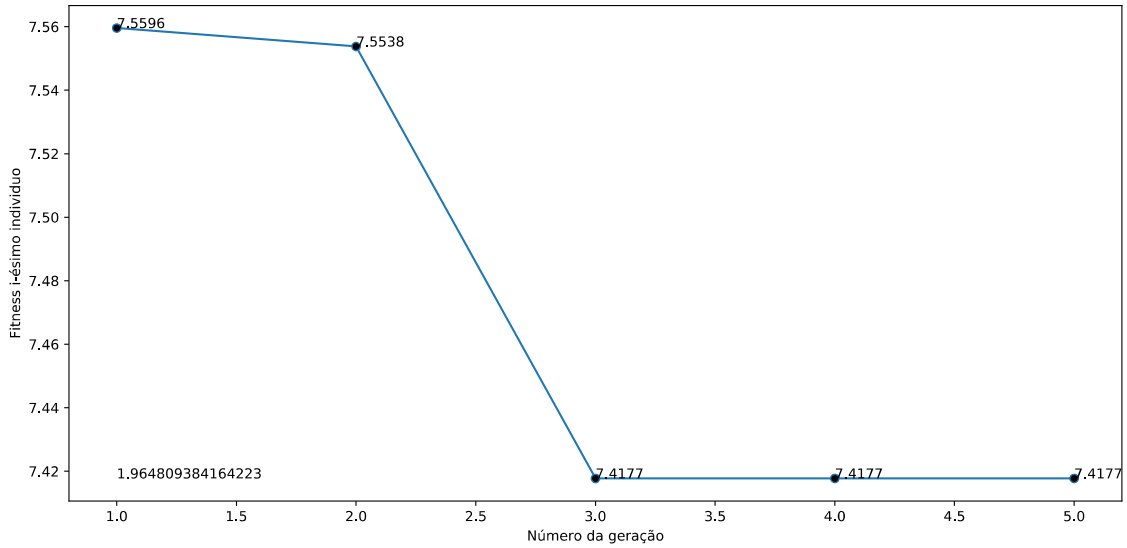
Cada linha do arquivo contém o número da geração, os bits do indivíduo mais apto, seu valor de X, o valor normalizado de X e valor obtido a partir da função fitness:

1;Bits = 1001010110;X = 598;X_normalizado = 1.691104594330401;Fitness = 1.7865209659741

5 RESULTADOS E OBSERVAÇÕES

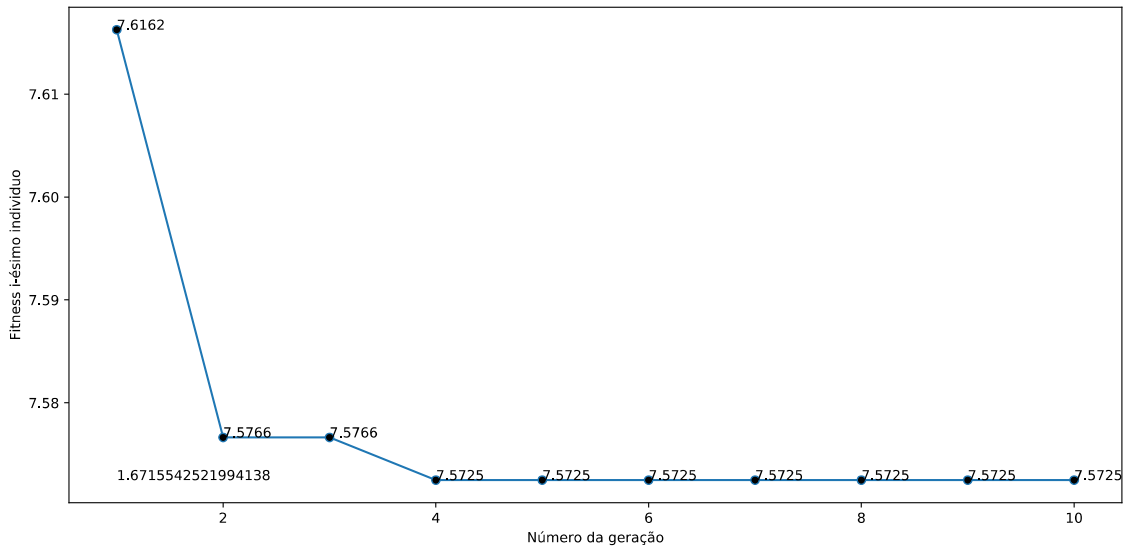
Abaixo no gráfico 2 é possível ver o resultado de uma chamada com 10 execuções e 5 gerações na versão anterior (usando bits).

Gráfico 2: Resultado com 4 indivíduos, 1% de chance de mutação, 60% de crossover, 5 gerações e 10 execuções.



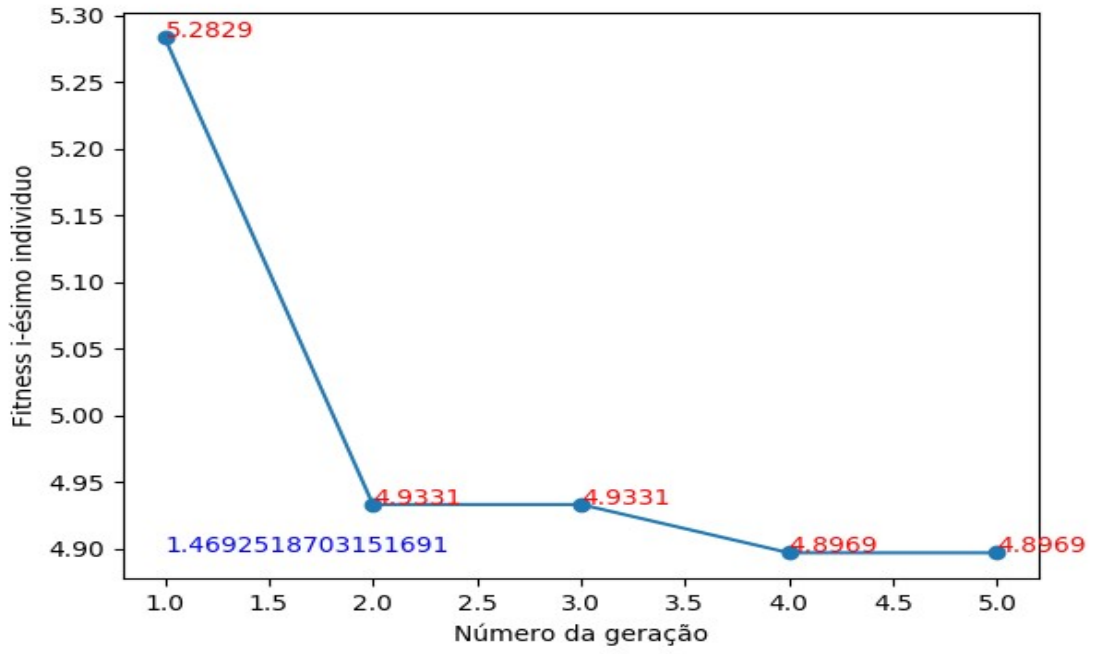
Já no gráfico 3 é possível ver o resultado de uma chamada com 10 execuções e 10 gerações também na versão anterior.

Gráfico 3: Resultado com 4 indivíduos, 1% de chance de mutação, 60% de crossover, 10 gerações e 10 execuções.



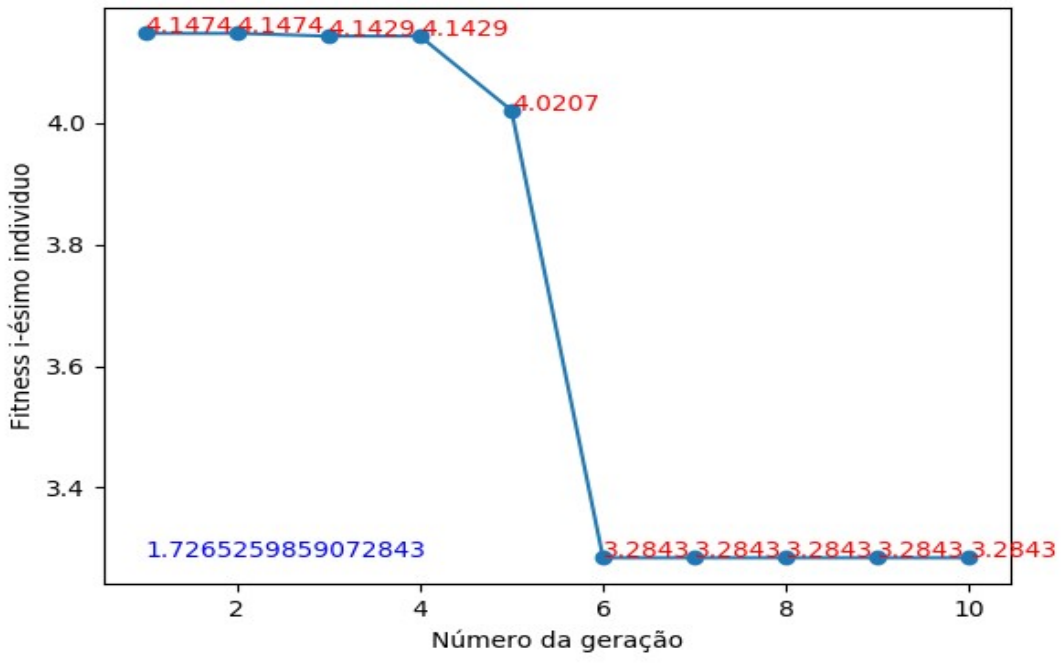
A partir daqui, os gráficos estarão demostrado os resultados obtidos usando Crossover Linear e Mutação Uniforme, como já foi dito anteriormente. Abaixo no gráfico 4 é possível ver o resultado de uma chamada com 10 execuções e 5 gerações.

Gráfico 4: Resultado com 4 indivíduos, 1% de chance de mutação, 60% de crossover, 5 gerações e 10 execuções.



Já no gráfico 5 é possível ver o resultado de uma chamada com 10 execuções e 10 gerações.

Gráfico 5: Resultado com 4 indivíduos, 1% de chance de mutação, 60% de crossover, 10 gerações e 10 execuções.



5.1 COMPARAÇÃO DOS RESULTADOS:

Pode-se observar claramente uma melhoria no valor do fitness em média em relação a versão anterior, mas com base em testes feitos, percebe-se que muitas das vezes o fitness também costuma ficas “estagnado” e o seu valor não melhora depois de certo ponto, criando em certos momentos apenas uma linha.

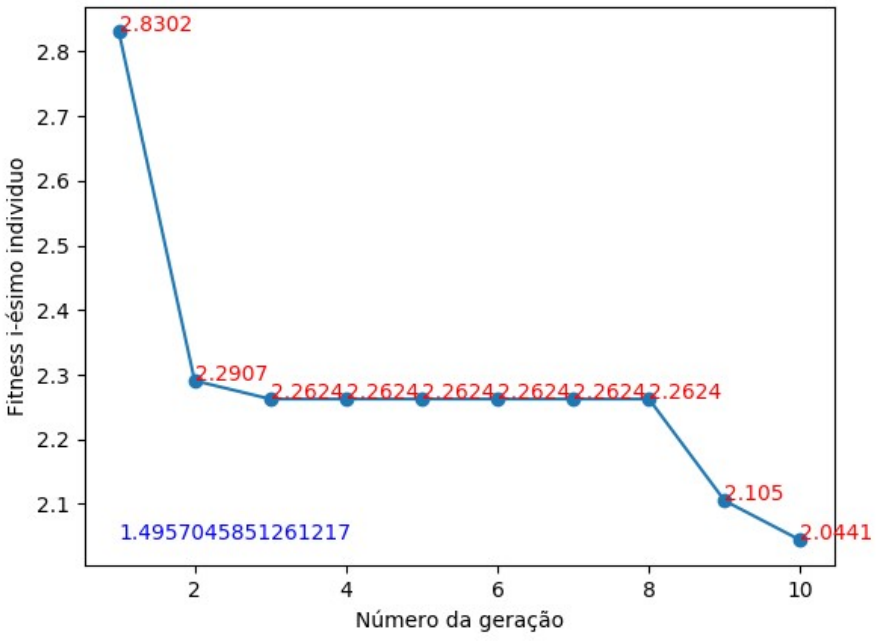
Por outro lado, foi possível observar que a versão atual (com números reais como gene), é muito mais sensível a quantidade de indivíduos da população do que a versão anterior. Tendo convergido bem mais e também mais rápido, como será explicado a seguir.

5.2 RELAÇÃO: TAMANHO DA POPULAÇÃO E PRECISÃO

O gráfico 6 indica que além começar mais próximo do fitness final (1.75), a melhoria absoluta vista entre a primeira geração e a última é de 0,7861 (2,8302 – 2,0441).

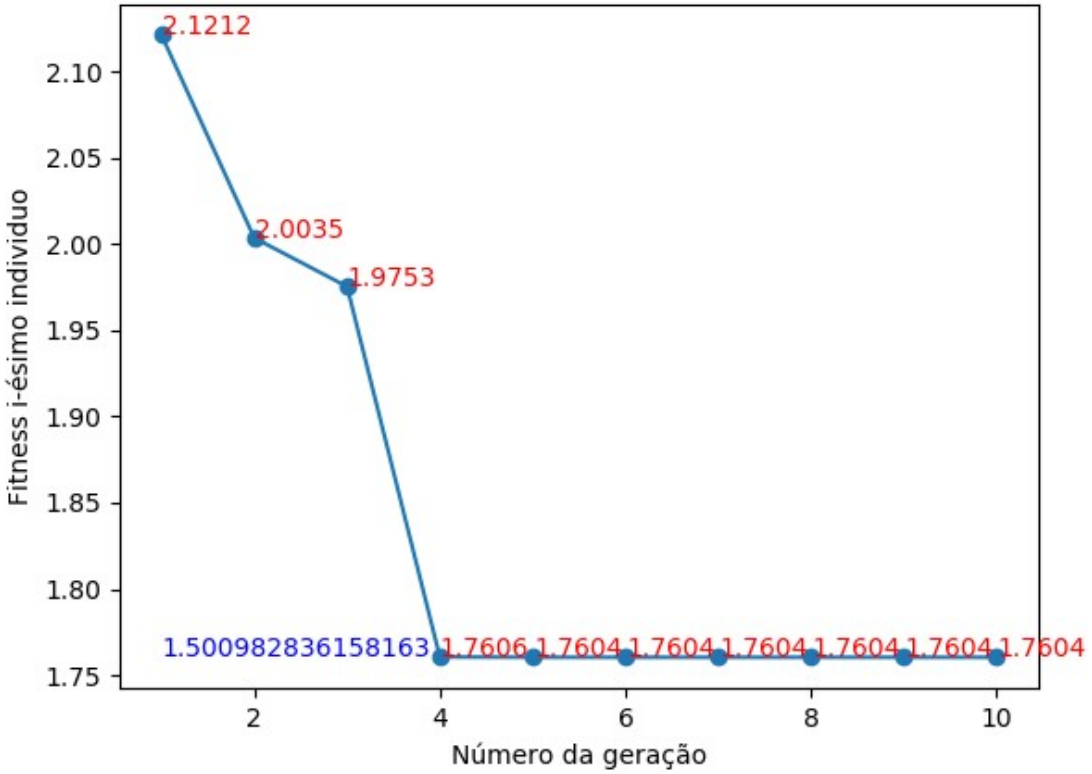
Isso acontece pelo mesmo motivo que acontecia também na versão anterior, mas com o adicional de o crossover linear gerar 3 indivíduos novos e escolher o melhor deles, fazendo uma elitização no momento do crossover. Por conta disso, o resultado já é claramente muito superior a versão anterior e também a versão atual com apenas 4 indivíduos.

Gráfico 6: Resultado com 8 indivíduos, 1% de chance de mutação, 60% de crossover, 10 gerações e 10 execuções.



O gráfico 7 mostra a melhoria que se tem quando existem 12 indivíduos na população.

Gráfico 7: Resultado com 12 indivíduos, 1% de chance de mutação, 60% de crossover, 10 gerações e 10 execuções.

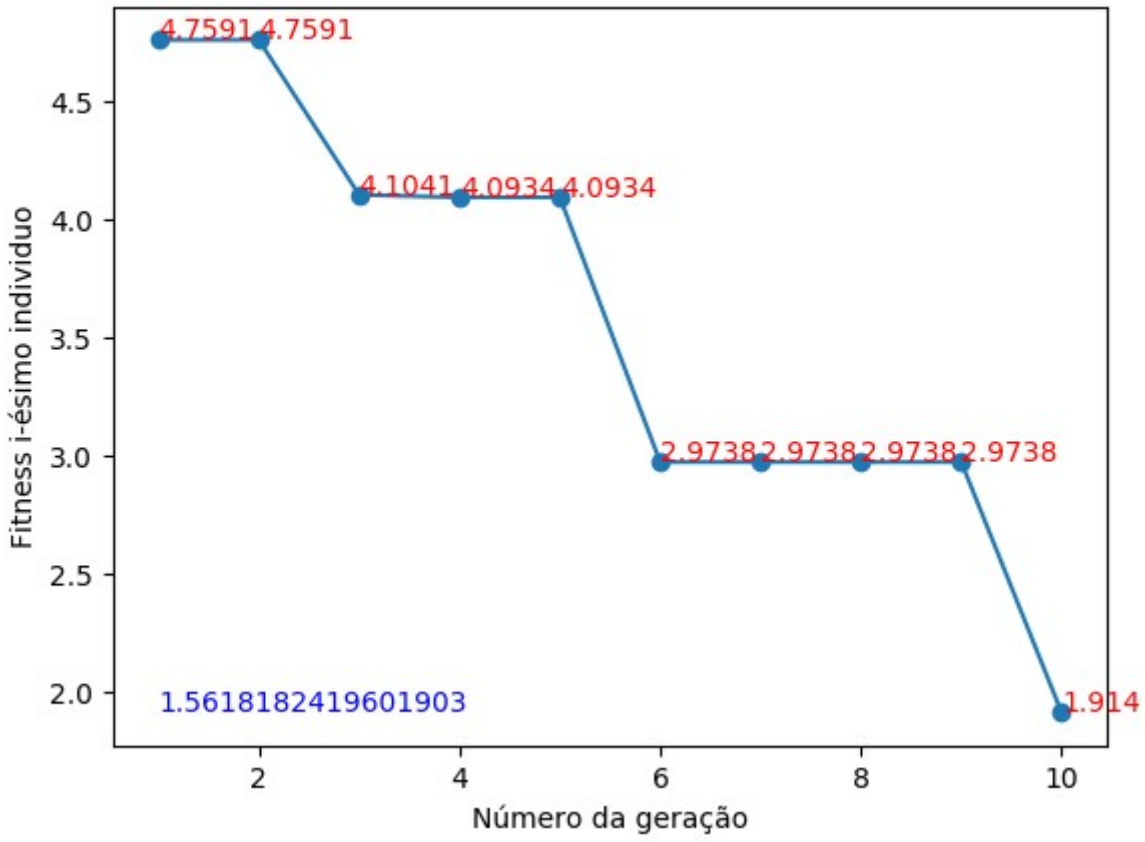


Pode-se ver que o resultado está muito próximo do valor ótimo.

5.3 RELAÇÃO: TAXA DE MUTAÇÃO E PRECISÃO

Com os resultados obtidos com os teste, não foi possível notar grande diferença no quanto a mutação uniforme influência a convergência em comparação com a versão anterior, já que ambas as mutações podem fazer o indivíduo se tornar outro completamente diferente do que era. O fato que não mudou também (entre as duas versões), é que quanto maior a mutação mais rápido se converge para o valor ótimo, como pode ser visto no gráfico a seguir:

Gráfico 8: Resultado com 4 indivíduos, 10% de chance de mutação, 60% de crossover, 10 gerações e 10 execuções.



6 REFERÊNCIAS E OUTROS MATERIAIS BASE

[1]. Página 43. Acesso em 25/05/2019. Disponível em: http://revista.pgsskroton.com.br/index.php/rcext/article/view/2394/2298

[2]. Página 2. Acesso em 25/05/2019. Disponível em: http://www.fsma.edu.br/si/edicao3/aplicacoes_de_alg_geneticos.pdf