

Relatório: Algoritmo Genético

Lucas Ribeiro (87703) e Willian Lemos (87708)

Algoritmo Genético

Segundo o Wikipedia, um algoritmo genético consiste da utilização de técnicas inspiradas na biologia evolutiva, como: *hereditariedade*, *mutação*, *seleção natural* e *recombinação*, com o objetivo de encontrar soluções aproximadas em problemas de otimização e busca.

Parâmetros

Neste exercício foi utilizado um solução pré-apresentada pela professora Silvia, que lida com os seguintes parâmetros:

1. **target**: valor de fitness ideal que buscamos.
 - a. **Fitness**: média de todos os valores dentro do vetor de um indivíduo.
2. **i_length**: número de valores por cada indivíduo.
3. **i_max**: máximo valor possível de um valor dentro do vetor de valores de um indivíduo.
4. **i_min**: mínimo valor possível de um valor dentro do vetor de valores de um indivíduo.
5. **p_count**: número de indivíduos de uma população.
6. **epochs**: número de iterações que serão feitas até encontrar a população ideal.

Geração de População

Inicialmente, com a função **population** é feita a geração de uma população, ou seja, um conjunto de **p_count** indivíduos com um vetor de **i_length** valores variando aleatoriamente entre **i_min** e **i_max**.

Fitness: Médias dos Indivíduos e da População

Em seguida, utilizando a função **media_fitness** é calculada a média de Fitness de toda a população: uma média aritmética do Fitness de todos os indivíduos naquela população.

O Fitness é calculado pela função **fitness** e é basicamente a média de todos os valores presentes em seu vetor de estrutura (genes) subtraindo-se o valor **target**. Assim, o Fitness ideal de uma população, é aquele mais próximo de zero.

Elitismo

Nesse exercício, a **Elite** é composta por 20% dos indivíduos com o melhor Fitness, ou seja, o 20% das pessoas com valor de fitness mais próximo de zero. Essa porcentagem pode ser alterada alterando-se o parâmetro **retain** presente na função **evolve**.

Seleção Original

Após o elitismo garantir a permanência de 20% da população como indivíduos reprodutores, o restante da população entram um processo de escolha randômica complementada por um teste de “cara ou corôa” com 5% de chance de ser “escolhido”. Todos os indivíduos selecionados (20% + Sorteados) são adicionados ao vetor **parents**.

Mutação

Em consequente, é feito o processo de mutação com os indivíduos presentes na estrutura **parents**. Inicialmente, descobre-se o número de filhos que precisam ser gerados através da fórmula: $\text{length}(\text{pop}) - \text{length}(\text{parents})$. Em seguida, metade da população é definido como **male** e a outra metade como **female**, e, aplicando-se crossover (metade dos genes de cada parent), cria-se um novo indivíduo que estará na próxima iteração.

Nossa Seleção: Lógica

Substituímos o algoritmo de seleção aleatória de pais (os 80%) por um simples algoritmo de **Torneio**:

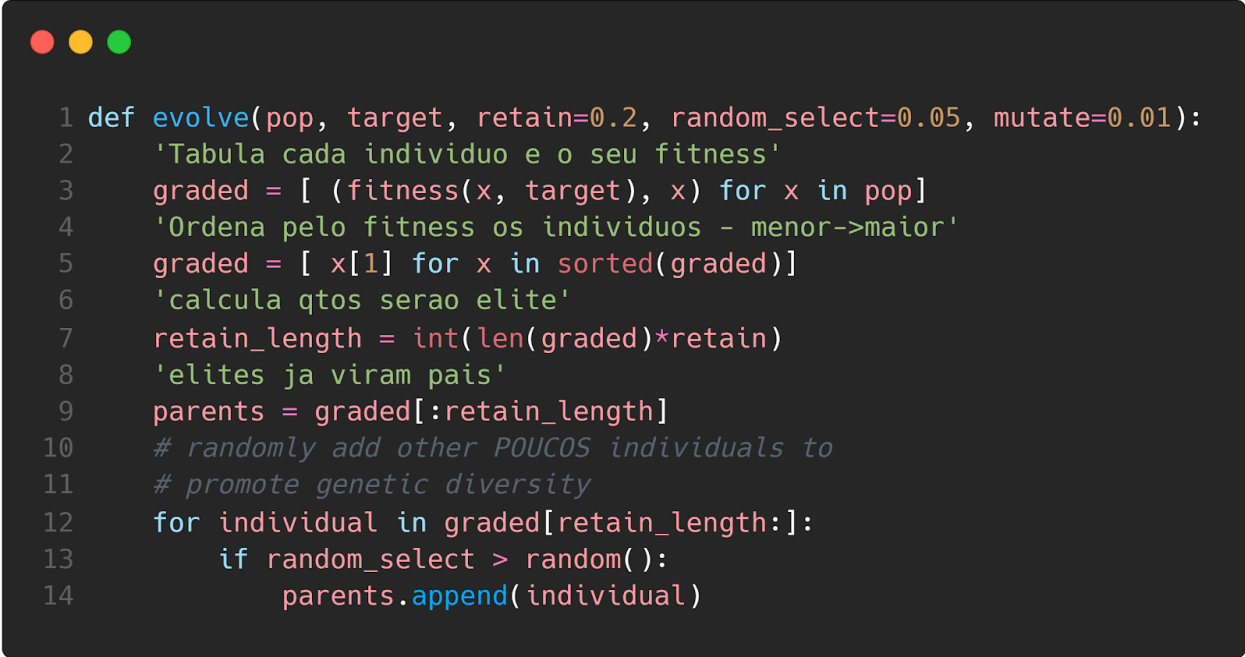
1. Cria-se um loop que irá selecionar aleatoriamente — de dentro dos 80% de indivíduos com pior fitness — 3 lutadores (range variável) que irão se “enfrentar” em um torneio.
2. Os três lutadores são adicionados a uma estrutura denominada **sorteio**.
3. Em seguida, ordenamos essa estrutura de acordo com o fitness dos três lutadores, em uma lista chamada **chosen**. Em seguida, colocamos um “limitador” de indivíduos vencedores (5%) para evitar que a lista de **parents** seja muito parecida com a população inicial.
4. Os indivíduos com melhor fitness, que passaram por esse limitador de 5%, são adicionados a estrutura **parents** para participarem de reprodução da nova geração.
5. Lutadores que venceram o torneio uma vez, não são sorteados novamente.

Problemas Enfrentados

Nossa solução, rapidamente chega ao valor de saída ideal: zero (ou o mais próximo possível), entretanto, devido a continuação do processo de mutação, esse valor volta a aumentar nas iterações médias, e, por fim retorna a algo próximo do ideal. Aparentemente, não estamos lidando corretamente com a “condição de parada” da mutação, porém, esse não era um dos requisitos do problema apresentado.

Demonstração das Alterações Feitas

Nas imagens abaixo, podemos observar as implementações iniciais e finais do exercícios.



```
1 def evolve(pop, target, retain=0.2, random_select=0.05, mutate=0.01):
2     'Tabula cada individuo e o seu fitness'
3     graded = [ (fitness(x, target), x) for x in pop]
4     'Ordena pelo fitness os individuos - menor->maior'
5     graded = [ x[1] for x in sorted(graded)]
6     'calcula qtos serao elite'
7     retain_length = int(len(graded)*retain)
8     'elites ja viram pais'
9     parents = graded[:retain_length]
10    # randomly add other POUCOS individuals to
11    # promote genetic diversity
12    for individual in graded[retain_length:]:
13        if random_select > random():
14            parents.append(individual)
```

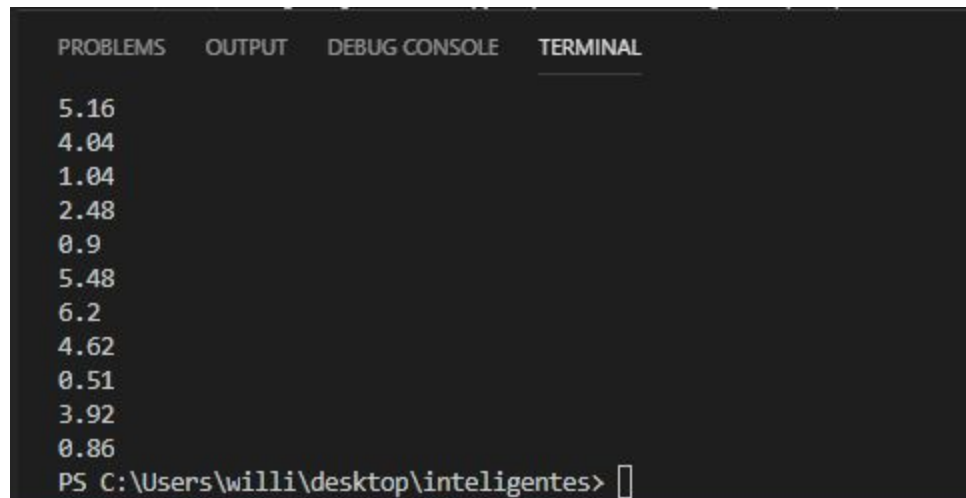
Figura 1. Código Original com Seleção Randômica.

Algoritmo de Torneio Resultante

```
1 def evolve(pop, target, retain=0.2, random_select=0.05, mutate=0.2):
2     'Tabula cada individuo e o seu fitness'
3     graded = [ (fitness(x, target), x) for x in pop]
4     aux = sorted(graded) #salvando informação
5     # print("\n\n *****GERAÇÃO:" + str(aux))
6     'Ordena pelo fitness os individuos - menor->maior'
7     graded = [ x[1] for x in sorted(graded)]
8     'calcula qtos serao elite'
9     retain_length = int(len(graded)*retain)
10    'elites ja viram pais'
11
12    parents = graded[:retain_length] #elites 20%
13
14    aux = aux[retain_length:] #separa os 80% restantes
15
16    for individual in aux: #torneio
17        if random_select > random():
18            sorteio = []
19            chosen = []
20            for x in range(2):
21                lutador = choice(aux)
22                sorteio.append(lutador)
23                # print("\nLUTADORES:" + str(sorteio))
24            chosen = [ x[1] for x in sorted(sorteio) ]
25            parents.append(chosen[0])
26            # print("\nVENCEDOR:" + str(chosen[0]))
27            chosen.clear()
28            sorteio.clear()
29            # print("\n *****PARENTS: *****"+str(parents))
```

Figura 2. Alterações no Código: Função **evolve** com Torneio.

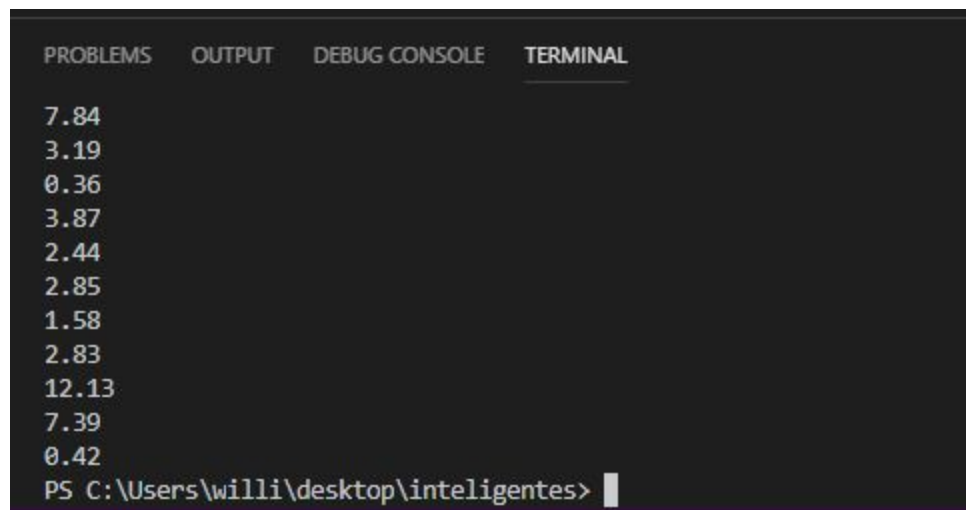
Saídas com as Matrículas dos Alunos.



A terminal window with a dark background and light gray text. At the top, there are four tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is selected and underlined. Below the tabs, a list of 13 numerical values is displayed, representing fitness values for student Willian. The values are: 5.16, 4.04, 1.04, 2.48, 0.9, 5.48, 6.2, 4.62, 0.51, 3.92, 0.86, and a final value of 0. At the bottom of the terminal, the command prompt shows the path 'PS C:\Users\willi\desktop\inteligentes>' followed by a cursor.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  
5.16  
4.04  
1.04  
2.48  
0.9  
5.48  
6.2  
4.62  
0.51  
3.92  
0.86  
0  
PS C:\Users\willi\desktop\inteligentes>
```

Figura 3. Saídas com **target** de Matrícula 708: Willian.



A terminal window with a dark background and light gray text. At the top, there are four tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is selected and underlined. Below the tabs, a list of 13 numerical values is displayed, representing fitness values for student Lucas. The values are: 7.84, 3.19, 0.36, 3.87, 2.44, 2.85, 1.58, 2.83, 12.13, 7.39, 0.42, and a final value of 0. At the bottom of the terminal, the command prompt shows the path 'PS C:\Users\willi\desktop\inteligentes>' followed by a cursor.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  
7.84  
3.19  
0.36  
3.87  
2.44  
2.85  
1.58  
2.83  
12.13  
7.39  
0.42  
0  
PS C:\Users\willi\desktop\inteligentes>
```

Figura 4. Saídas com **target** de Matrícula 703: Lucas.

Observação: O algoritmo encontrou o valor de saída **zero** muito rapidamente, porém, não foi capaz de transmitir esse valor para as gerações seguinte, e teve um leve aumento nas gerações intermediárias, devido a mutação.