Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP)
Departamento de Estatística, Matemática Aplicada e Computação (DEMAC)
Curso: Pós-Graduação em Ciência da Computação
Disciplina: Computação Inspirada pela Natureza
Prof.: Fabricio Breve – Trabalho 1

1 – Algoritmo Genético para reconhecimento de padrões, reconhecer o número 0, representado pelo bitstring [1 1 1 1 0 1 1 0 1 1 1 1]

1	1	1
1	0	1
1	0	1
1	1	1

Para conseguir reproduzir os passos desse exercício será nexessário ter instalado os seguintes componentes:

Instalar o python 2.7
Instalar o python-tk
Instalar o python-pip
Instalar o modulo numpy com pip
Instalar o modulo matplotlib

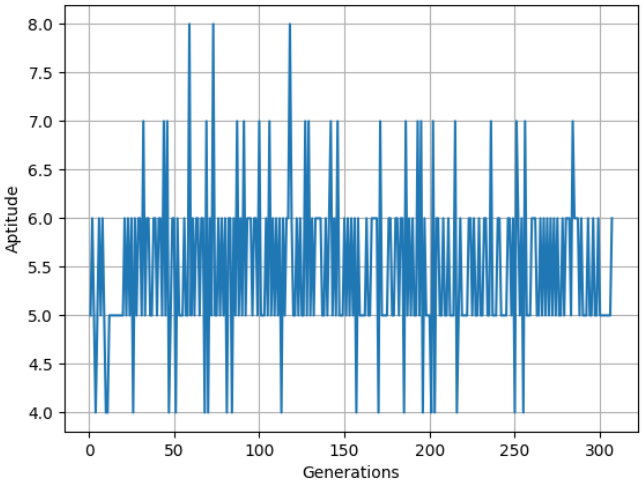
Testes executados

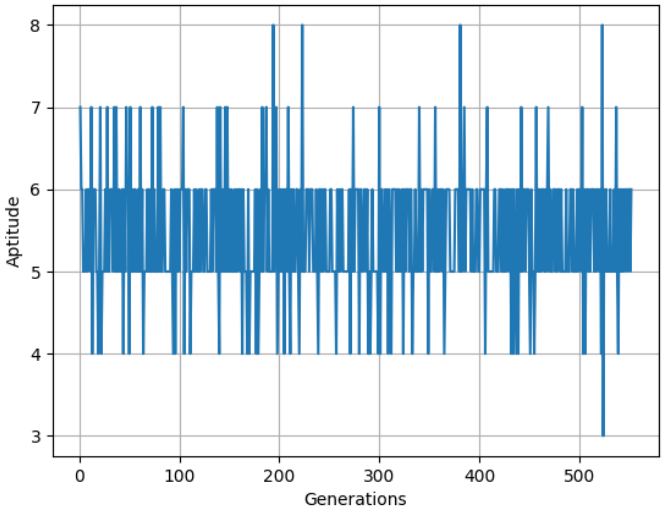
1 - Taxa de crossover = 1 e taxa de mutação = 1

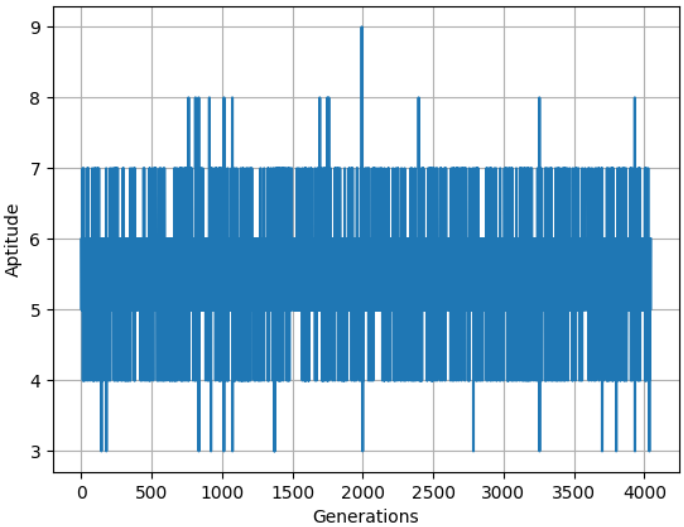
Passou das 100.000 gerações e não terminou de executar. Nesse caso sempre estamos executando crossover e mutação, ou seja, existe muita mudança de uma geração para outra e não conseguimos convergir para o alvo.

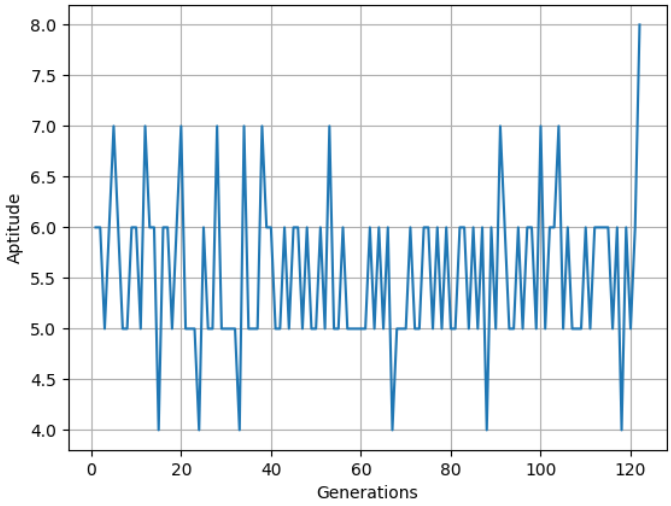
2 - Taxa de crossover = 0.75 e taxa de mutação = 0.75

Nesse caso o algoritmo consegue encontrar o alvo porém a aptidão de uma geração para outra oscila muito, executei o algoritmo algumas vezes e os resultados são sempre parecidos, abaixo temos alguns exemplos de execução









Depois de ter executado esses dois testes com taxas altas decidi partir para o caminho contrário, ou seja, zerei as duas taxas.

3 - Taxa de crossover = 0.0 e taxa de mutação = 0.0

Quando zeramos as duas taxas, o algoritmo não encontra o alvo. Isso ocorre porque quando zeramos as duas taxas não existe variação de uma geração para outra e com isso não conseguimos encontrar o alvo.

4 - Taxa de crossover = 0.25 e taxa de mutação = 0.25

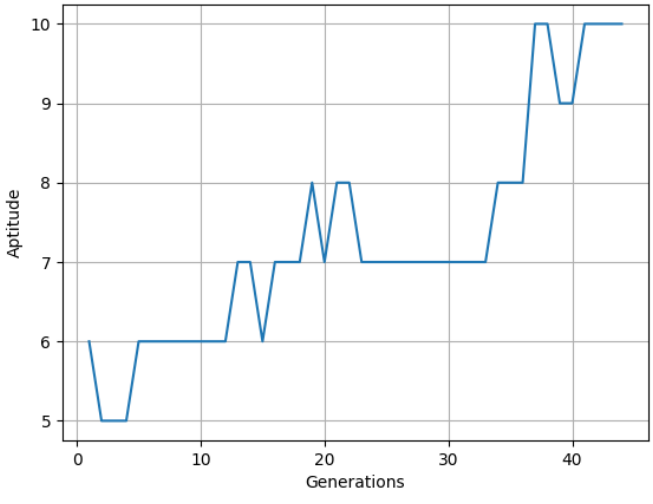
Nessa configuração, os resultados obtidos foram bem parecidos com o cenário 2.

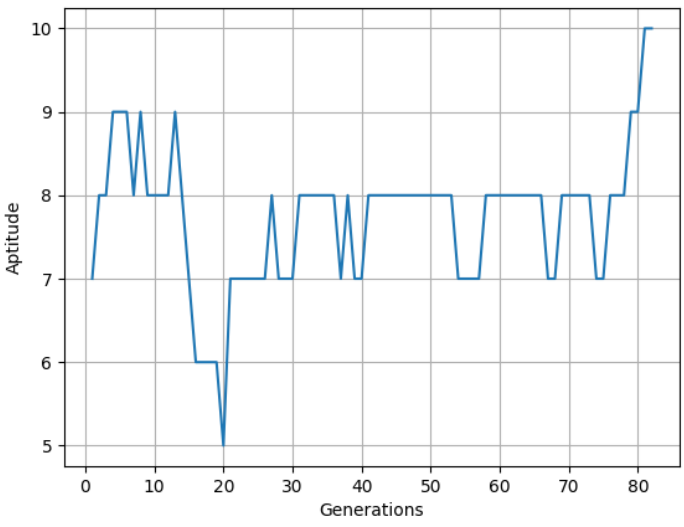
5 - Taxa de crossover = 0.5 e taxa de mutação = 0.5

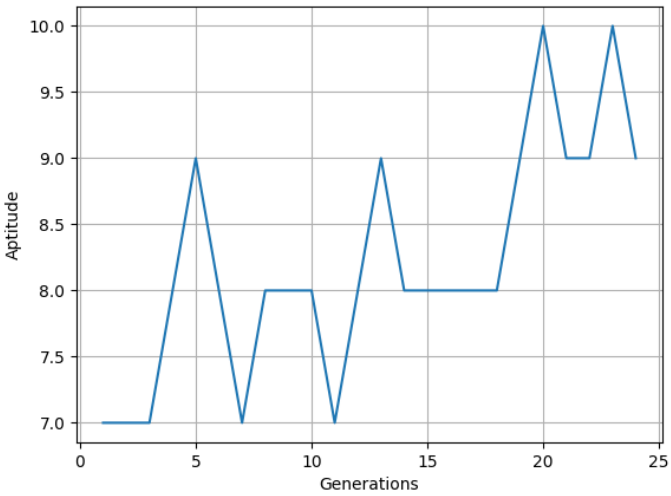
Nessa configuração, os resultados obtidos foram bem parecidos com o cenário 2.

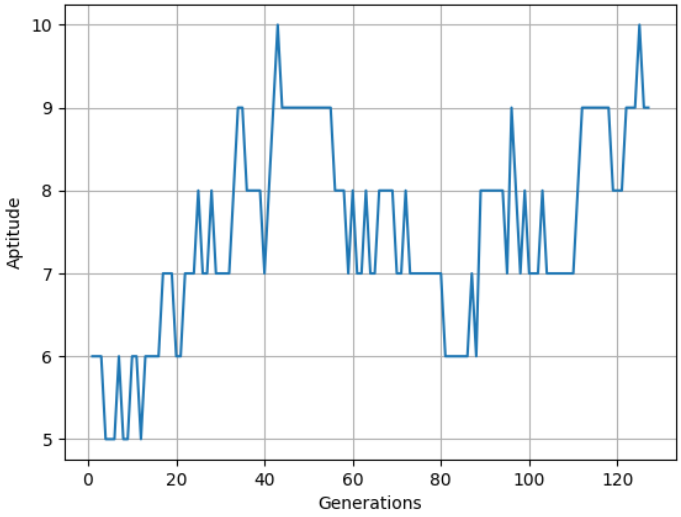
6 - Taxa de crossover = 0.6 e taxa de mutação = 0.02

Baseado nos experimentos anteriores, vimos que não parece uma boa configuração utilizarmos taxas iguais. Então decidi realizar o teste com uma taxa média de crossover e bem baixa para mutação, conforme apresentado no exemplo da aula. Nesse caso o algoritmo se comportou melhor, melhorando a média da aptidão dos indivíduos de geração em geração, e inclusive, encontrando o alvo mais rápido. Abaixo alguns exemplos de execução nessa configuração.









7 - Taxa de crossover = 0.02 e taxa de mutação = 0.6

Invertendo a situação do cenário anterior, ou seja, deixando a taxa de crossover baixa e a taxa de mutação média alta, o algoritmo consegue encontrar o alvo porém leva entre 1000 e 3500 gerações para encontra-lo e as gerações oscilam bastante.

8 - Taxa de crossover = 0.5 e sem mutação

O algoritmo não termina

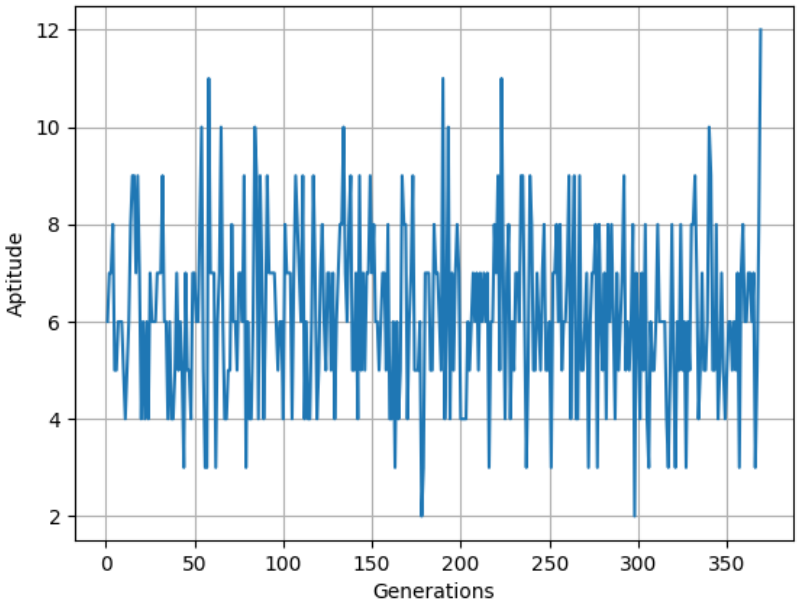
8 – Sem crossover e taxa de mutação = 0.5

Ao realizar esse teste descobri um defeito no código, pois em algumas execuções a aptidão de todos os indivíduos da população resultava em 0, o que gerava uma divisão por zero no momento de selecionar a faixa de valores da roleta (360 * aptidão do indivíduo / aptidão total).

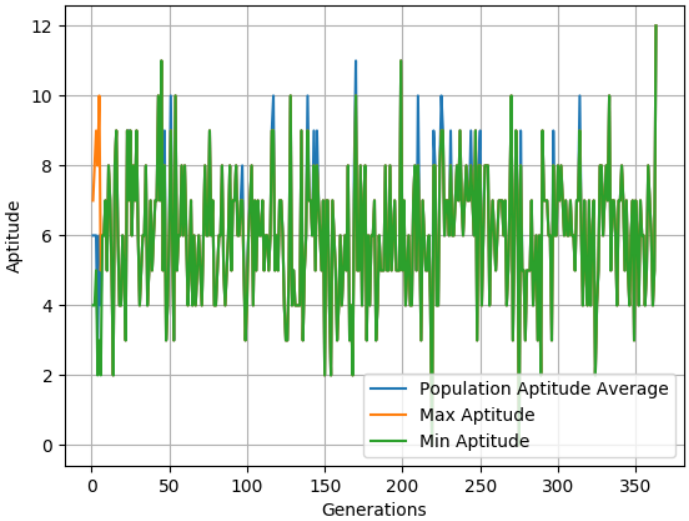
Corrigi esse defeito fazendo uma verificação da aptidão da total, que é a soma das aptidões de todos os indivíduos, quando ela resulta em zero, ou seja, todos os indivíduos da população tem aptidão 0, o método responsável por separar as faixas de valores da roleta para cada indivíduo retorna faixas de tamanhos iguais para cada um deles (tamanho da faixa = 360/ tamanho da população).

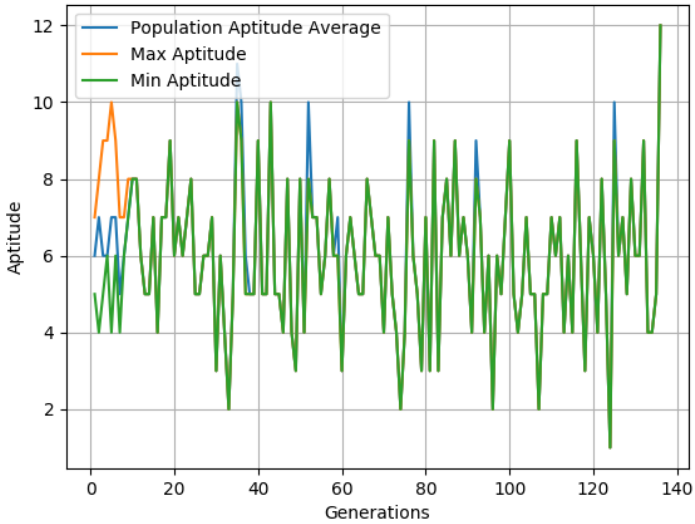
Com essa correção rodei o algoritmo algumas vezes nessa configuração e vi que ele sempre finalizava com todos os indivíduos iguais ao alvo, ou seja, com aptidão 12.

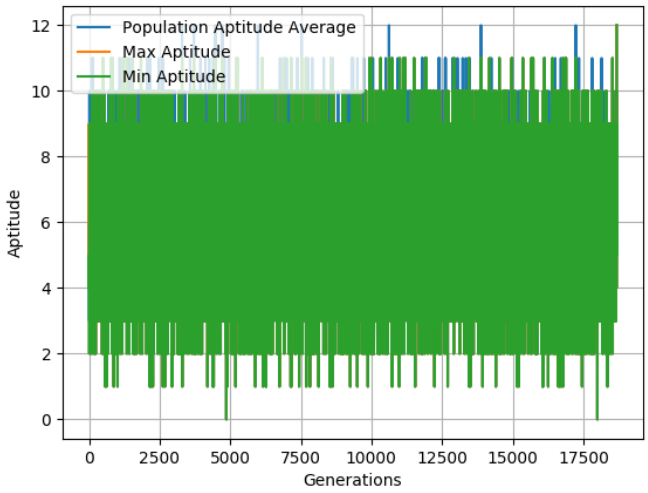
```
======== Geracao 370  ========

Populacao

1 - [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
2 - [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
3 - [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
4 - [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
5 - [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
6 - [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
7 - [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
8 - [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
None

Aptidoes = [12, 12, 12, 12, 12, 12, 12, 12]

Avaliacao media da geracao = 12.0
```

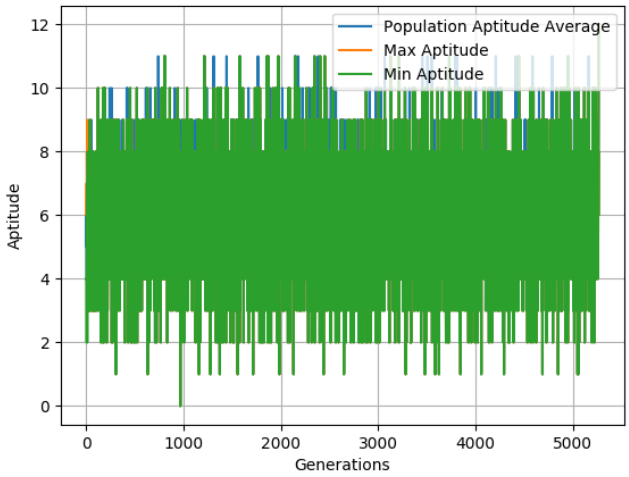


Para analisar melhor o que estava acontecendo, adicionei duas novas séries ao gráfico, uma com o maior valor de aptidão da população em cada geração e outra com o menor valor. E obtive os seguintes resultados :









O que os resultados nos mostram é que a população inicia diversificada, porém rapidamente todos os indivíduos ficam iguais, visto que todos os valores (máximo, mínimo e média) ficam iguais rapidamente.

Achei esses resultados estranhos e para conseguir avaliar melhor o que estava acontecendo coloquei alguns logs no código e obtive o seguinte resultado do log:

```
Reconhecer o 0 representado por [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
Verbose level 2
Tamanho da populacao : 8
Taxa de crossover : 0
Taxa de mutacao : 0.5

=== EVALUATION STEP ===
```

```
Target [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]

Population

1 - [1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0]
2 - [1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1]
3 - [1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1]
4 - [0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0]
5 - [1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0]
6 - [0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0]
7 - [0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1]
8 - [0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1]
Target size 12
Hamming Array [7, 3, 7, 6, 6, 6, 6, 6]
Aptitudes result [5, 9, 5, 6, 6, 6, 6, 6]

=== SELECTION STEP ===
Target [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]

Population

1 - [1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0]
2 - [1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1]
3 - [1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1]
4 - [0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0]
5 - [1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0]
6 - [0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0]
7 - [0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1]
8 - [0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1]
Aptitudes [5, 9, 5, 6, 6, 6, 6, 6]

Selection Ranges

1 - [0, 36.734693877551024]
2 - [36.734693877551024, 102.85714285714286]
3 - [102.85714285714286, 139.59183673469389]
4 - [139.59183673469389, 183.67346938775512]
5 - [183.67346938775512, 227.75510204081635]
6 - [227.75510204081635, 271.8367346938776]
7 - [271.8367346938776, 315.9183673469388]
8 - [315.9183673469388, 360.0]
None
Random numbers [ 20 14 318 87 211 308 231 132]

New Generation Selected

1 - [1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0]
2 - [1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0]
3 - [1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0]
4 - [1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0]
```

```
5 - [1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0]
6 - [1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0]
7 - [1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0]
8 - [1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0]
```

Ou seja, existe um defeito no passo de seleção de uma nova geração.

Corrigi o problema, porém os resultados continuam os mesmos, ou seja, os indivíduos da população não ficam todos iguais logo na primeira geração, porém com o tempo o algoritmo converge para isso nesse cenário. Vide arquivo nocrossover_with_mutation.log

Testes após as correções

População = 8, Taxa de crossover = 0 e Taxa de mutação = 0

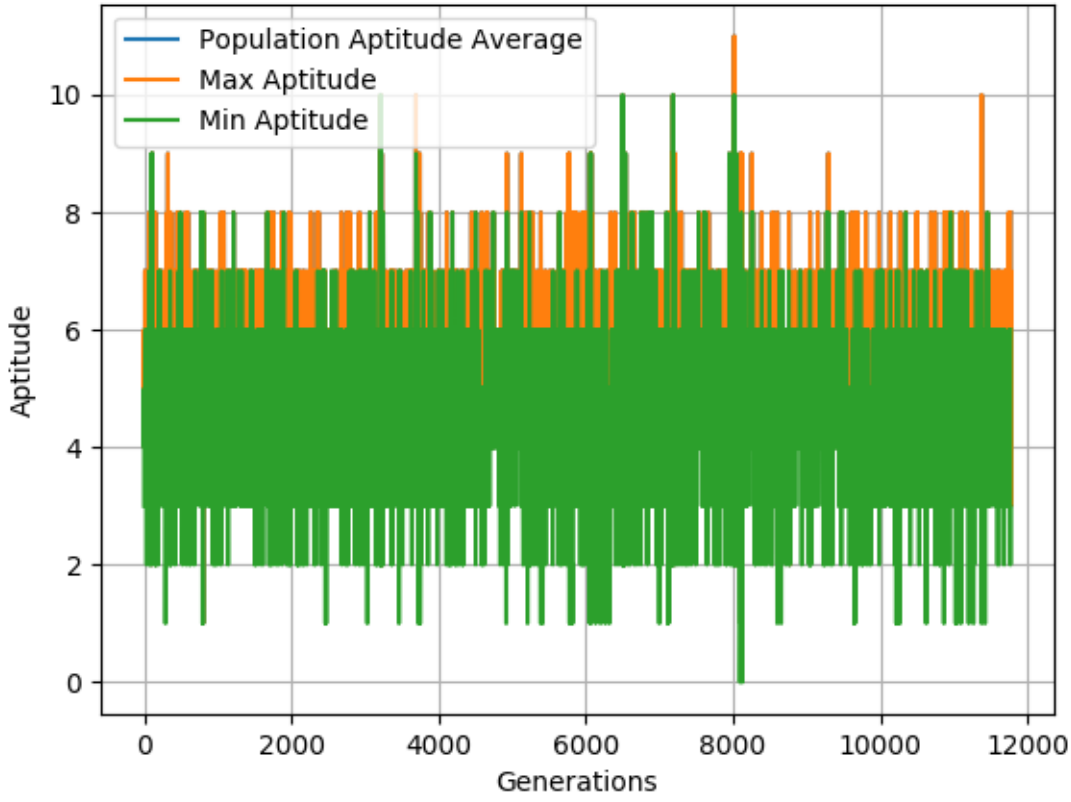
Nesse cenário a população nunca muda de geração para geração e com isso, se o alvo não estiver na população inicial, ele nunca é encontrado.

População = 8, Taxa de crossover = 1 e Taxa de mutação = 1

Nessa coonfiguração a população muda muito de uma geração para outra e faz com que o algoritmo precise de muitas gerações para encontrar o alvo. Conseguimos analisar isso claramente no gráfico da primeira execução a seguir.

Execução 1

```
vpfeifer@vpfeifer-Vostro-5470:~/unesp/cin/src/evlutionary-computing-algorithms/trabalho01/ex01$ python ex01.py
Reconhecer o 0 representado por [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
Verbose level 0
Tamanho da populacao : 8
Taxa de crossover : 1
Taxa de mutacao : 1

======== Geracao 11770  ========

Population

1 -    [0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0]
2 -    [1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
3 -    [0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1]
4 -    [1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1]
5 -    [0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1]
6 -    [1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0]
7 -    [1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1]
8 -    [0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1]
None

Aptidoes = [6, 12, 9, 7, 9, 9, 7, 9]

Aptidao media da geracao = 8.0
```



Execução 2

Encontrou o alvo na geração 11.723 com gráfico muito semelhante ao da execução anterior.

Execução 3

Encontrou o alvo na geração 5.285 com gráfico muito semelhante ao da execução 1.

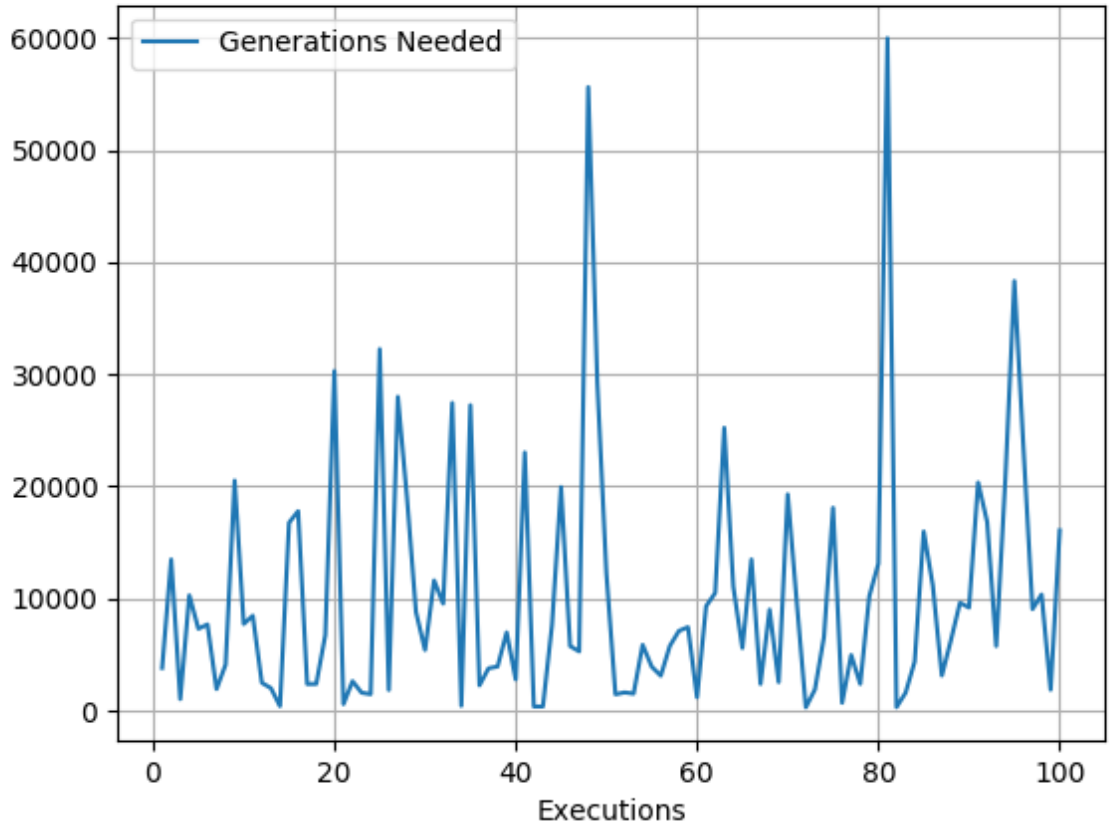
Execução 4

Encontrou o alvo na geração 12.590 com gráfico muito semelhante ao da execução 1.

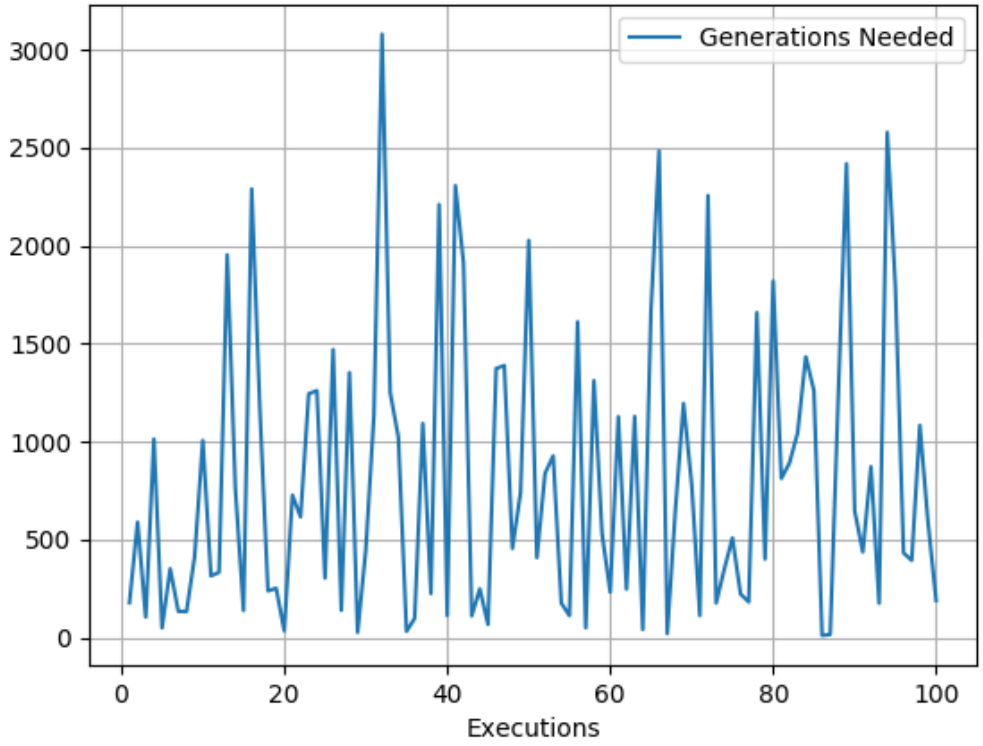
Execução 5

Encontrou o alvo na geração 2.892 com gráfico muito semelhante ao da execução 1.

Para não ter que executar multiplas vezes manualmente para todos os cenários, adicionei um laço de repetição para executar o algoritmo multiplas vezes e conseguir avaliar o numero de gerações necessárias para encontrar o alvo em cada execução

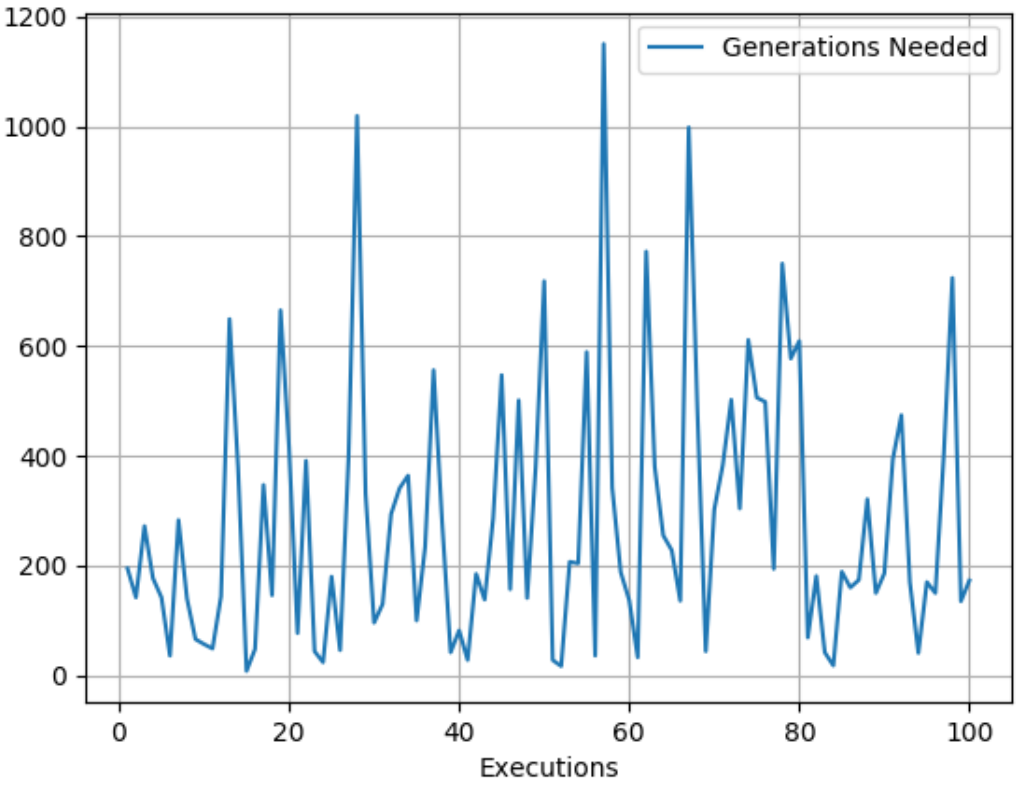


População = 8, Taxa de crossover = 0.5 e Taxa de mutação = 0.5



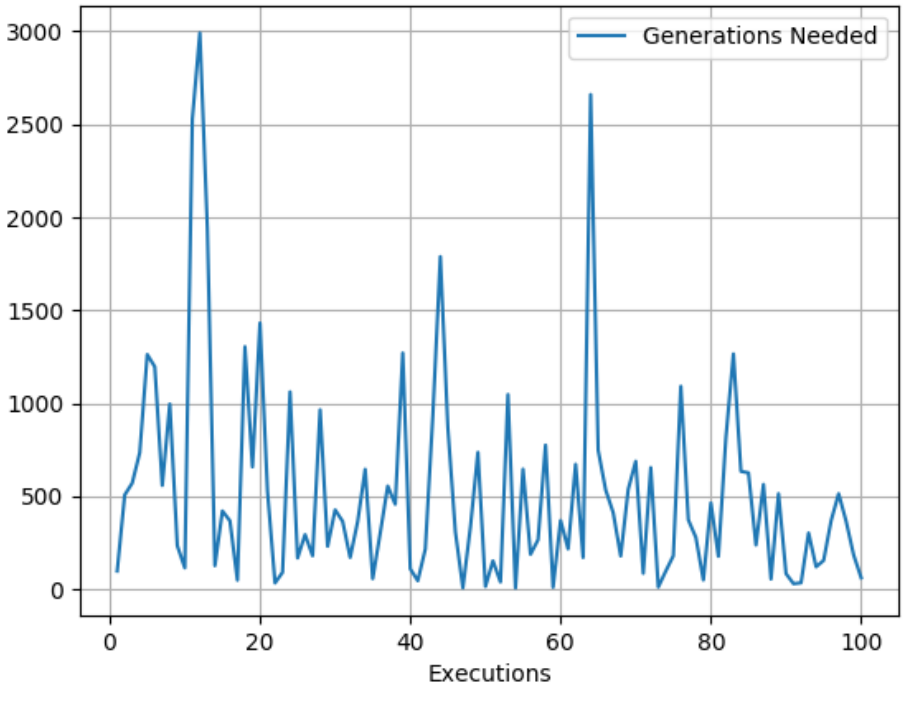
População = 8, Taxa de crossover = 0.6 e Taxa de mutação = 0.02

Nessa configuração, que foi a do exemplo em aula, o número de gerações necessárias para encontrar o alvo é bem menor do que as anteriores

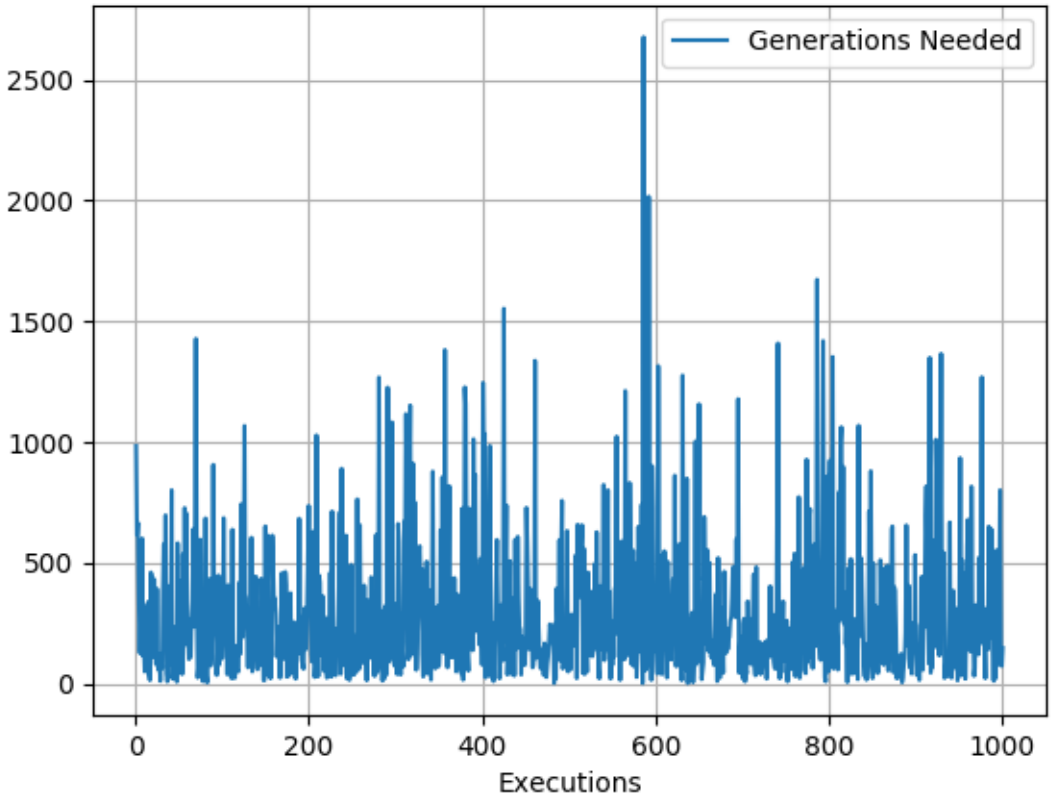


População = 8, Taxa de crossover = 0.6 e Taxa de mutação = 0.2

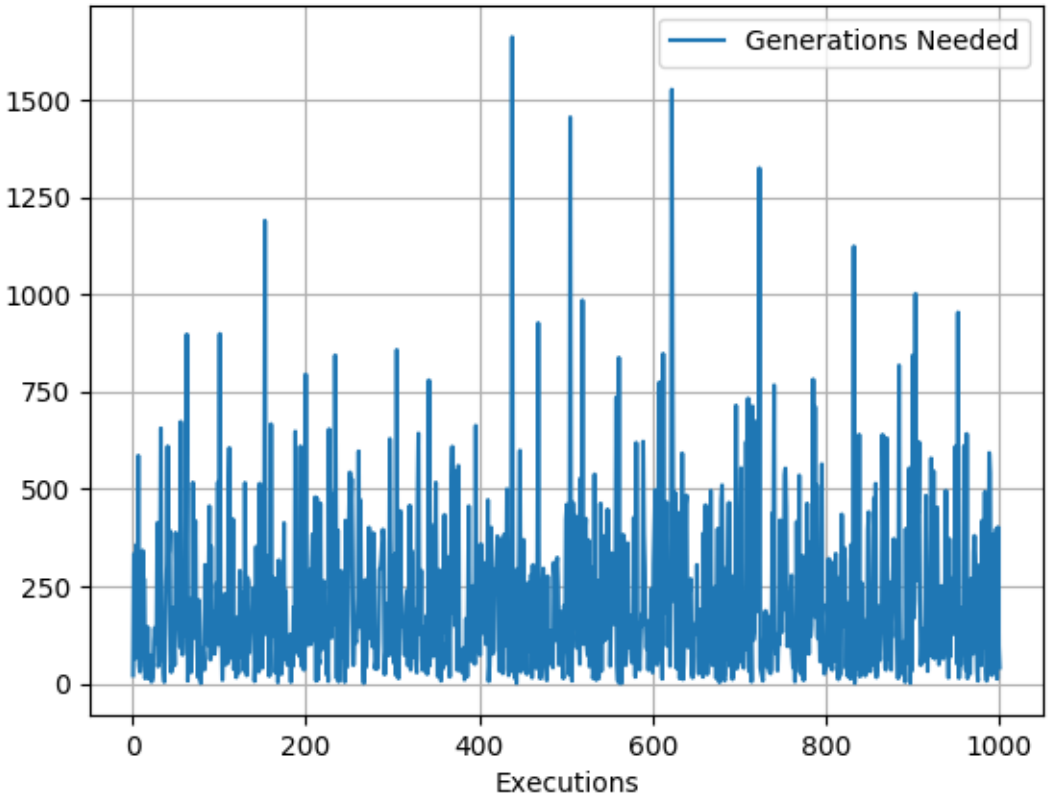
Aumentando a taxa de mutação em 10x faz com que em algumas execuções necessitem mais do que o dobro de execuções para encontrar o alvo.



População = 8, Taxa de crossover = 0.6 e Taxa de mutação = 0.02 com 1000 execuções



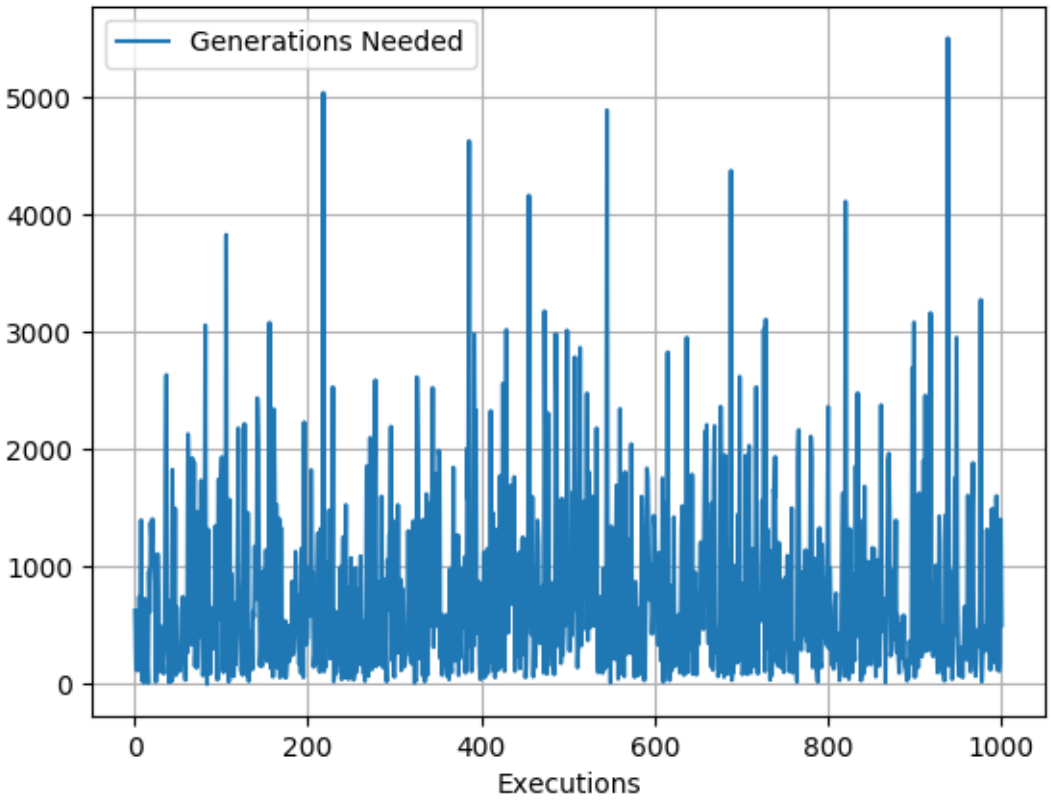
População = 16, Taxa de crossover = 0.6 e Taxa de mutação = 0.02 com 1000 execuções



Aumentando a população de 8 para 16 vemos que o algoritmo reduz o número de gerações para encontrar o alvo

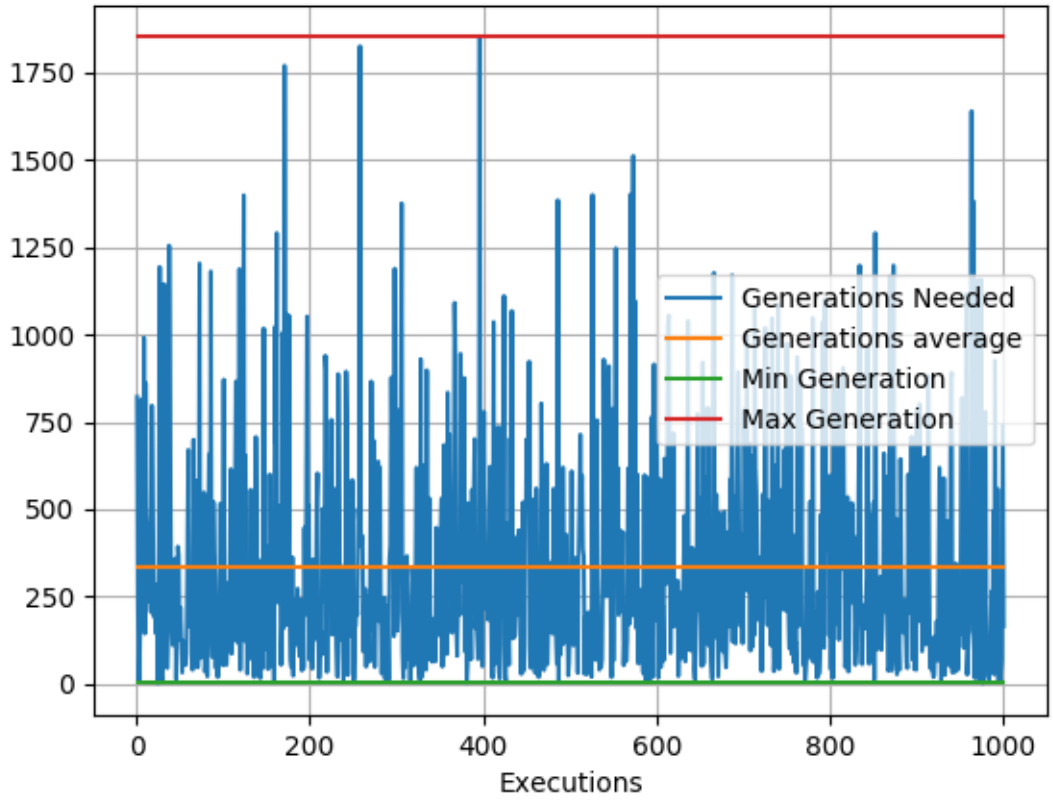
População = 4, Taxa de crossover = 0.6 e Taxa de mutação = 0.02 com 1000 execuções

Reduzindo a população o número de gerações necessárias para encontrar o alvo aumenta

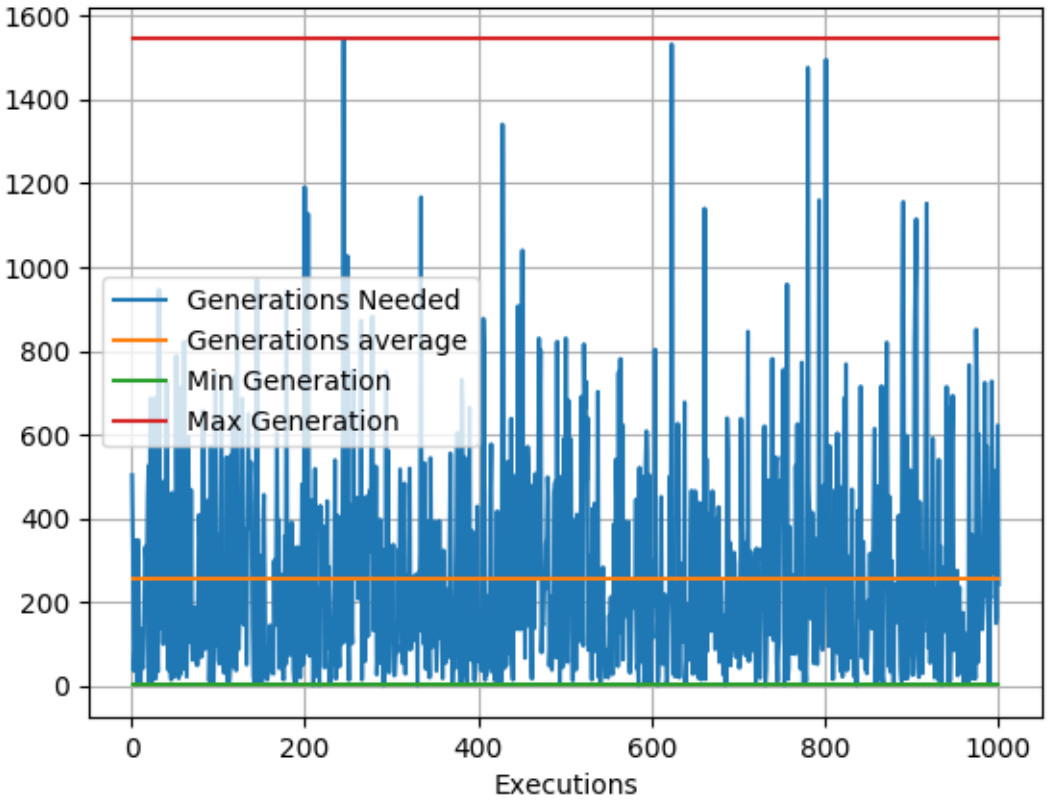


População = 8, Taxa de crossover = 0.4 e Taxa de mutação = 0.02 com 1000 execuções

Reduzindo um pouco a taxa de crossover o número de gerações aumenta um pouco



População = 8, Taxa de crossover = 0.5 e Taxa de mutação = 0.02 com 1000 execuções



População = 32, Taxa de crossover = 0.6 e Taxa de mutação = 0.02 com 1000 execuções

Aumentando bastante a população, o tempo de execução também aumentou, porém não temos grandes ganhos na média, apenas nos máximos que reduziram um pouco.

Conclusão

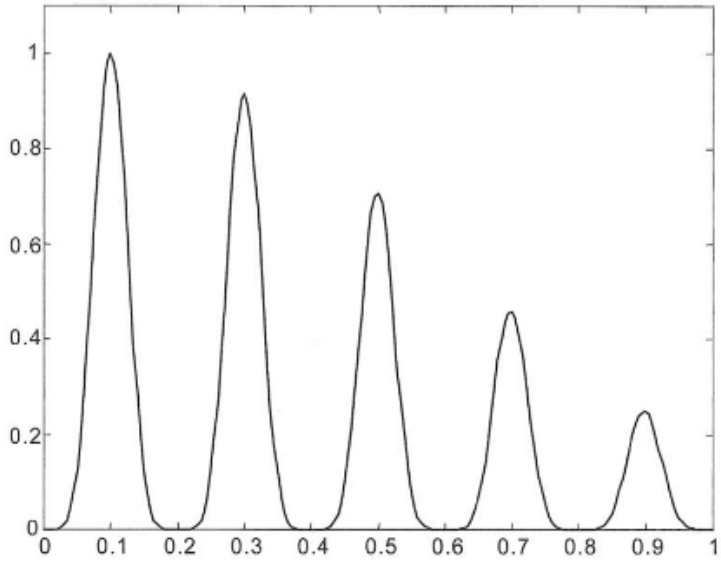
Baseado nos testes apresentados percebi que os parâmetros de entrada para o algoritmo variam da seguinte maneira:

Tamanho da população : Quanto maior a população menor a quantidade de gerações necessárias para encontrar o alvo, porém o tempo para executar cada geração aumenta. O ideal é manter entre 8 e 16.

Taxa de crossover : O ideal é manter em uma taxa média, entre 0,4 e 0,6. Quando muito baixa, os indivíduos variam pouco entre uma geração e outra, fazendo com que o número de gerações necessárias para encontrar o alvo aumentem. Quando muito alta, os indivíduos variam muito e com isso a quantidade de gerações para encontrar o alvo aumente.

Taxa de mutação : No caso desse parâmetro, o ideal é uma taxa baixa, abaixo de 0,1, pois se ela for muito alta os idividuos vão variar muito e o número de gerações para encontrar o alvo aumenta.

2) Implemente um Algoritmo Genético para maximizar a função $g(x) = 2^{-2((x-0.1)/0.9)^2}(\sin(5\pi x))^6$, já utilizada nos exercícios feitos em aula. Utilize uma representação de *bitstring*. Compare o resultado obtido com os resultados que você obteve com os algoritmos Subida da Colina e Recozimento Simulado aplicados a esta mesma função nos exercícios feitos em sala de aula.



Representação

Para representar um indivíduo utilizarei um bitstring com 10 bits, onde consigo representar números entre 0 e 1023.

O bitstring será convertido para decimal e dividido por 1000, o que resultará em um numero entre 0 e 1,024 esse numero será utilizado na função de avaliação.

Aptidão

A aptidão de um indivíduo será medida de acordo com o resultado da função avaliação.

Seleção

Seleção será feita por roleta.

Reprodução

Utilizando taxa de crossover de 0,5 baseado nos testes do exercício 1.

Mutação

Taxa de 0,02 conforme análise do exercício 1.

Resultado

Comparando o algoritmo genético com subida da colina e recozimento simulado, me pareceu que o primeiro é menos dependente dos parâmetros de entrada do que os outros, por exemplo, no subida da colina se eu inicio o algoritmo próximo aos pontos de máximo local o resultado é um máximo local.

Mesmo que eu comece o subida da colinas de vários pontos, pode ser que eu inicie em vários máximos locais e nunca atinja o máximo global.

3) Utilize um Algoritmo Genético para minimizar a seguinte função no intervalo contínuo $\begin{bmatrix} -5 & +5 \\ -5 & +5 \end{bmatrix}$:

$$f(x,y) = (1-x)^2 + 100(y-x^2)^2$$

Representação em bitstring de 4 bits onde o primeiro é sinal

População = 30

Aptidão por ranking

Seleção por roleta