Tarefa 9

Implementar um algoritmo genético para resolver um problema de minimização.

Autores

Renan Mateus Bernardo Nascimento
Vinícius Magalhães D'Assunção

Requisitos necessários

- python3
- numpy
- matplotlib
- python3-tk

Configuração do ambiente

Instale o python3, numpym, matplotlib e o python3-tk:

```
sudo apt-get install python3
pip3 install numpy
pip3 install matplotlib
sudo apt-get install python3-tk
```

Instrução para execução

É possível executar o código de duas maneira:

Jupyter notebook

- É necessário ter o Jupyter notebook
- Carregue o arquivo algoritmo_genetico.ipynb

Terminal

Execute o seguinte comando no terminal:

```
python3 main.py tam_populacao, inicio_alelo, fim_alelo, tax_cruzamento, tax_mutacao,
max_geracoes
```

Decisões de implementação

Foi utilizada a função de aptidão para a avaliação, o método da roleta para seleção, o cruzamento aritmético, sendo os valores de alfa gerados aleatoriamente no intervalo (0,1), a mutação CREEP, em que é somado um valor aleatório no intervalo [-2,2]. Todos os pais são substituídos pelos filhos ao final de cada iteração.

Classe Cromossomo

Define o TAD de cromossomo, que possui como atributos uma lista de alelos e um valor de aptidão.

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import random
```

In [2]:
```
class Cromossomo:

    def __init__(self, qtd_alelos):
        self.alelos = np.zeros(qtd_alelos)
        self.aptidao = 0.0
```

Classe Ag

Possui a implementação do algoritmo genético, possuindo os métodos para de população, cálculo de aptidão, seleção, cruzamento e mutação.

In [29]:
```
class Ag:

    def __init__(self, tam_populacao, inicio, fim, tax_cruzamento, tax_m
utacao, max_geracoes):
        self.tam_populacao = tam_populacao
        # valor inicial para alelos do cromossomo
        self.inicio = inicio
        # Valor final para alelos do cromossomo
        self.fim = fim
        self.tax_cruzamento = tax_cruzamento
        self.tax_mutacao = tax_mutacao
        self.max_geracoes = max_geracoes
        # offset da função de aptidão
        self.OFFSET = 1500
        self.MAX = 959.6407 + self.OFFSET
        self.QTD_ALELOS = 2
        self.ELITE = 5
        self.populacao = []

    '''
    ' Inicializa a lista de população
    '''
    def popular(self):
        self.populacao = []
        for c in range(0, self.tam_populacao):
            cromo = Cromossomo(self.QTD_ALELOS)
            for a in range(0, len(cromo.alelos)):
                cromo.alelos[a] = random.randrange(self.inicio, self.fim
+ 1)
            cromo.aptidao = self.calcAptidao(cromo.alelos[0], cromo.alel
os[1])
            self.populacao.append(cromo)

    '''
    ' Calcula a função de aptidão de toda a população
    '''
    def calcAptidao(self, x1, x2):
        return self.OFFSET + (x2 + 47) * np.sin(np.sqrt(abs(x2 + x1/2 +4
7))) + x1 * np.sin(np.sqrt(abs(x1 - (x2 + 47))))

    '''
    ' Faz a seleção da população utilizando o método da roleta
    ' @return selecao
    '''
    def selecionar(self):
        selecao = []
        roleta = {}
        total = 0

        # Adiciona índice do cromossomo à roleta
        for i, c in enumerate(self.populacao):
            # A roleta possui a tupla inicio_intervalo e fim_intervalo c
omo índice do dicionário
            # O valor corresponde ao índice do vetor de população
            roleta[(total, total + c.aptidao)] = i
            total += c.aptidao

        # Sorteia tam_populacao/2 números para selecionar na roleta
        for i in range(0, self.tam_populacao):
            # Sorteia o número entre 0(fechado) e a aptidão total(aberto
)
            num = random.uniform(0, total)
            for intervalo, c in roleta.items():
                if num >= intervalo[0] and num < intervalo[1]:
                    selecao.append(c)
        return selecao

    '''
```

Valor mínimo da função eggholder

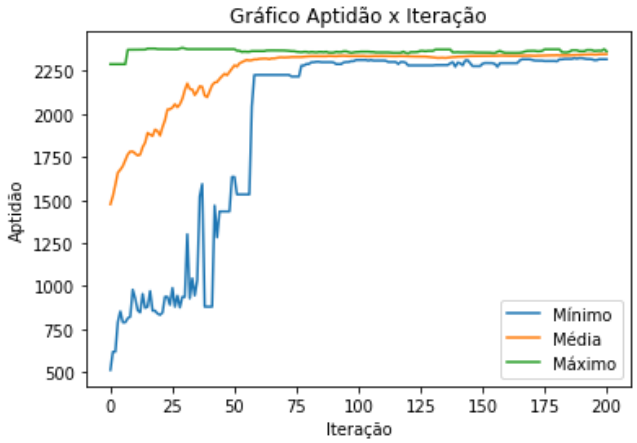
Com a inversão de sinal da função e a soma de um OFFSET = 1500, o mínimo da função agora corresponde ao máximo **2459.6407**.

Variando a taxa de cruzamento

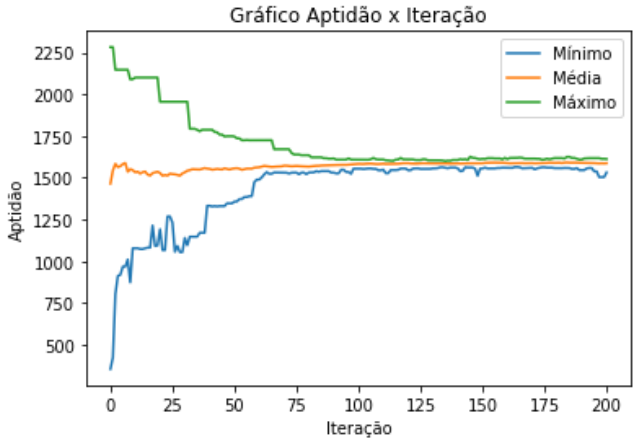
É possível observar que uma menor taxa de cruzamento (0,2) faz com que o máximo, o mínimo e a média convirjam para o valor máximo. Um aumento na taxa de cruzamento faz com que o valor convirja para uma aptidão de 1500.

In [27]:
```
tam_populacao, inicio, fim, tax_cruzamento, tax_mutacao, max_geracoes =
(200, -600, 600, 0.2, 0.1, 200)

for i in range(1,4):
    tax_cruzamento = 2*i/10
    ag = Ag(tam_populacao, inicio, fim, tax_cruzamento, tax_mutacao, max
_geracoes)
    print(ag.MAX)
    print("Taxa de cruzamento: " + str(tax_cruzamento))
    ag.executar()
    print('-----------------\n')
```

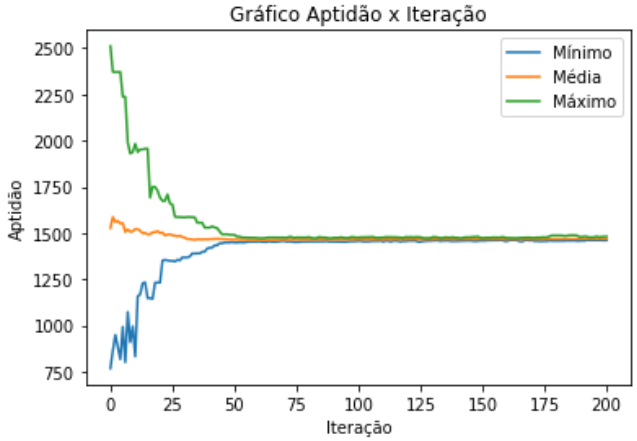
```
2459.6407
Taxa de cruzamento: 0.2
```




```
Melhores cromossomos:
[-456.89949334  375.95446835]   =       2364.5763751
[-453.11297173  377.17484424]   =       2363.27276665
[-453.11297173  377.17484424]   =       2363.27276665
-----------------

2459.6407
Taxa de cruzamento: 0.4
```




```
Melhores cromossomos:
[ 141.08799663   41.28557022]   =       1612.26041938
[ 140.0164139    41.26648949]   =       1611.85846742
[ 139.85303868   41.39246878]   =       1610.13707837
-----------------

2459.6407
Taxa de cruzamento: 0.6
```



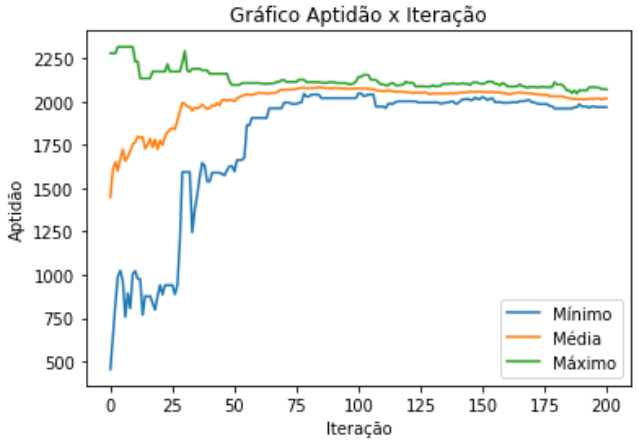

```
Melhores cromossomos:
[ 43.94990003  14.30787912]     =       1480.4917747
[ 44.7251695   14.67038456]     =       1478.71039926
[ 43.88264275  14.73829872]     =       1478.10574277
-----------------
```

Variando a taxa de mutação

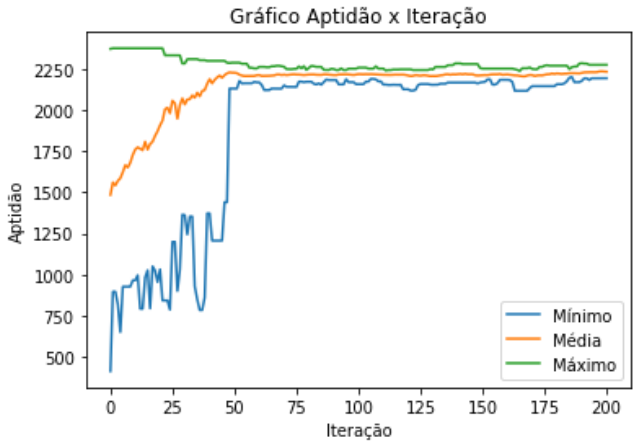
Uma taxa de mutação de 0.1 fez com que tempo de subida da curva fosse menor. Ao aumentar a taxa de mutação observou-se que o tempo de subida aumentou, ou seja, foram necessárias mais iterações. Para um valor de mutação muito grande, os valores de aptidão começam a convergir para um valor menor do que o desejado.

In [25]:
```
tam_populacao, inicio, fim, tax_cruzamento, tax_mutacao, max_geracoes =
(100, -600, 600, 0.2, 0.1, 200)

for i in range(1,4):
    ag = Ag(tam_populacao, inicio, fim, tax_cruzamento, tax_mutacao, max
_geracoes)
    print("Taxa de mutação: " + str(round(tax_mutacao, 2)))
    ag.executar()
    print('-----------------\n')
    tax_mutacao += 0.2
```

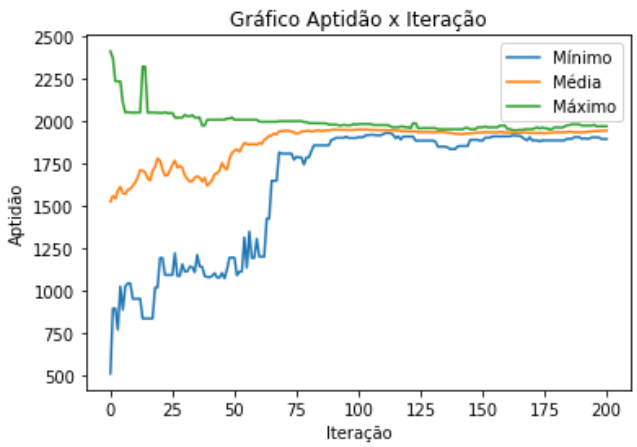
```
Taxa de mutação: 0.1
```




```
Melhores cromossomos:
[ 188.46164484  532.40326383]   =       2069.72125688
[ 188.46164484  532.40326383]   =       2069.72125688
[ 188.46164484  532.40326383]   =       2069.72125688
-----------------

Taxa de mutação: 0.3
```




```
Melhores cromossomos:
[-435.84186849  371.93523928]   =       2277.03782406
[-435.84186849  371.93523928]   =       2277.03782406
[-435.11913658  368.14071961]   =       2269.15872975
-----------------

Taxa de mutação: 0.5
```



```
Melhores cromossomos:
[ -54.02216205  399.05366375]   =       1967.69451562
[ -48.76773826  396.01471814]   =       1967.23477481
[ -48.76773826  396.01471814]   =       1967.23477481
-----------------
```