

## Breve Tutorial sobre a Biblioteca Python para GA

Além do Python com a biblioteca numpy, matplotlib e math é necessário instalar a biblioteca Python para GA: `pip install geneticalgorithm2` (<https://pypi.org/project/geneticalgorithm2/> )

Informações sobre a biblioteca:

Na biblioteca a função `f` é a função objetivo que se deseja minimizar, onde a entrada é o conjunto de `X` (variáveis de decisão).

Caso exista alguma restrição ela pode ser incluída na função objetivo. No exemplo abaixo o exemplo indica como a avaliação para o caso a soma das duas primeiras variáveis `X0` e `X1` fossem inferior a 2.

```
def f(X):  
    penalidade=0  
    if X[0]+X[1]<2:  
        penalidade =500+1000*(2-X[0]-X[1]) # quanto mais próximo de 2 menor será a punição.  
    return np.sum(X)+ penalidade
```

O algoritmo genético é projetado para minimizar a função dada. Para problemas de maximização é possível multiplicar a função objetivo por um sinal negativo. Assim, o valor absoluto da saída é o máximo da função. Exemplo:

```
def f(X):  
    return -np.sum(X)
```

As variáveis precisam ser identificadas:

- Os limites das variáveis devem ser definidos como um array numpy e, para cada variável, precisa-se de um limite identificado:
- Para três variáveis, todas tendo os mesmos limites: `varbound=np.array([[0,10]]*3)`
- Para variáveis reais (contínuas), usa-se a string 'real' para notificar o tipo real para as variáveis: `variable_type='real'`
- Para variáveis inteiras, usa-se a string 'int' para notificar o tipo inteiro para as variáveis: `variable_type='int'`
- Aceitando também Boolean, usa-se `variable_type` é igual a 'bool', não há necessidade de `variable_boundaries` ser definida.
- Caso de os limites sejam diferentes, veja o exemplo com variáveis mista:  
Se as variáveis têm diferentes tipos, por exemplo:  
x1 seja uma variável real (contínua) em [0.5,1,5],  
x2 seja uma variável inteira em [1,100] e  
x3 seja uma variável booleana (pode ser zero ou um).

Já sabemos que a resposta é `X = (0.5,1,0)` onde `f(X) = 1,5`

```
varbound=np.array([[0.5,1.5],[1,100],[0,1]])  
variable_type =np.array(['real'],['int'],['int'])  
model=ga(function=f,dimension=3, variable_type_mixed= variable_type, variable_boundaries=varbound)
```

Observe que o comprimento de `variable_boundaries` deve ser igual a `dimension=3`.

A saída com a melhor resposta do problema de otimização definido e encontrada pelo algoritmo genético é do tipo dicionário e um relatório do progresso do algoritmo genético:

- **convergence=model.report:** é uma lista que inclui a convergência do algoritmo sobre as iterações
- **solution=model.output\_dict:** é um dicionário incluindo o melhor conjunto de variáveis encontradas e o valor da função dada associada a ele ({'variável':, 'função':}).

Os parâmetros do algoritmo genético podem ser definidos em:

```
algorithm_param = {
    'max_num_iteration': 300,\
    'population_size':100,\
    'mutation_probability':0.08,\
    'elit_ratio': 0.01,\
    'crossover_probability': 0.6,\
    'parents_portion': 0.3,\
    'crossover_type':'uniform',\
    'max_iteration_without_improv':None
    'selection_type': 'roulette'
}
```

```
model=ga(function=f,\
    dimension=3,\
    variable_type='real',\
    variable_boundaries=varbound,\
    algorithm_parameters=algorithm_param)
```

**max\_num\_iteration** número máximo de iterações;

é o parâmetro para o critério de finalização do algoritmo genético.

Se o valor deste parâmetro for **None**, o algoritmo define o número máximo de iterações automaticamente como uma função da dimensão, limites e tamanho da população.

O usuário pode inserir qualquer número de iterações que desejar. É altamente recomendável que o próprio usuário determine o **max\_num\_iteration**.

**população\_size:** determina o número de soluções em cada iteração. O valor padrão é 100;

**mutation\_probability:** determina a chance de cada gene em cada solução individual ser substituído por um valor aleatório. O padrão é 0,1 (ou seja, 10 por cento);

**elit\_ratio:** determina o número de membros que serão preservados na próxima população. O valor padrão é 0,01 (ou seja, 1 por cento). Por exemplo, quando o tamanho da população é 100 e elit\_ratio é 0,01, então há um indivíduo na população (o melhor), que será mantido na próxima geração. Se este parâmetro for definido como zero, o algoritmo genético implementa um algoritmo genético padrão em vez do GA elitista;

**crossover\_probability:** determina a chance de uma solução existente passar seu genoma para os filhos (também conhecido como descendência); o valor padrão é 0,5 (ou seja, 50 por cento);

**parent\_portion:** a porção da população preenchida pelos membros da geração anterior (também conhecidos como pais); o padrão é 0,3 (ou seja, 30 por cento da população). Definida como zero, significa que toda a população é preenchida com as soluções recém-geradas. Por outro lado, ter este parâmetro igual a 1 (ou seja, 100 por cento) significa que nenhuma nova solução é gerada e o algoritmo iria apenas repetir os valores anteriores sem qualquer alteração que não seja significativa e eficaz, obviamente. O valor ideal depende do problema.

**crossover\_type:** existem três opções incluindo **one\_point**; **two\_point** e **uniform**; o padrão é crossover uniform;

**max\_iteration\_without\_improv:** se os algoritmos não melhorarem a função objetivo sobre o número de iterações sucessivas determinado por este parâmetro, o algoritmo genético finaliza e relata a melhor solução encontrada antes de **max\_num\_iterations** a serem atendidas. O valor padrão é **None**.

**selection\_type:** há as seguintes opções para tipos de seleção de indivíduos: **fully\_random**, **roulette**, **tournament**; entre outras. O `linear_ranking(selection_pressure = 1.5)` indica uma normalização linear com taxa 1.5. O padrão é **roulette**.

**remove\_duplicates\_generation\_step:** intervalo para remover os duplicados; se 1 remove a cada geração;

[1] Rastrigin, L. A. "Systems of extremal control." Mir, Moscow (1974).