ga

Release 1.0.0

Jhonat Heberson

Oct 28, 2020

CONTENTS:

class ga.**Genetic**(*goal=1.0*, *bounds=None*, *mutation_probability=0.5*, *selection_probability=0.5*, *sigma=0.5*, *num_parents=2*, *num_elitism=2*, *maxiter=None*, *selection_method='elitism'*, *cross_over='uniform'*, *mutation='gaussian'*, *submit_to_cluster=False*)

Genetic(maxiter=1000, goal=0, cross_over='one_point', mutation_probability=0.01, mutation='uniform', selection_method='elitism',num_parents=2, num_elitism=10, bounds=np.array((np.ones(2) * 10 * -1, np.ones(2) * 10)))

É a classe que é responsavel por realizar a criar a população, selecionar os parentes da população, realizar os cruzamentos e mutação.

goal [float, opcional] Define o valor a qual queremos nos aproximar

bounds [numpy.ndarray] Define até onde pode ir os valores dos individuo, como inferio e superior

mutation_probability [float] Define a probabilidade de ocorrer a mutação

selection_probability [float] Define a probabilidade de ocorrer a seleção de pais

sigma [float] Define a probabilidade do individuo ter mutação

num_parents [integer] Define o numero de parentes que será esolhido no metodo da seleção

num_elitism [integer] Define o numero de pais que será preservado entre as gerações

maxiter [integer,opcional] Define o numero de interações maximo que o algoritmo irá fazer para encontrar o resultado

selection_method [str, opcional] Define o metodo de seleção que será usado para escolher os pais da proxima geração

cross_over [str, opcional] Define o metodo de cruzamento que será usado

mutation [str, opcional] Define o metodo de mutação que será usado

submit_to_cluster [bool, opcional] Define se a meta-hurística será executada no cluster

```
>>> from fffit import ga
>>> bounds = np.array((np.ones(2) * 10 * -1, np.ones(2) * 10))
>>> ga.Genetic(maxiter=1000, goal=0, cross_over='one_point',
                   mutation_probability=0.01, mutation='uniform',
                   selection_method='elitism',num_parents=2,
                   num_elitism=10, bounds=bounds)
```

```
>>>
```

calculate_avg_fitness()

Calcula a aptidão de media entre todas as populações e salva na lista da classe genética.

Returns:

return void

calculate_best_fitness()

Calcula a melhor aptidão entre todas as populações e salva na lista da classe genética.

Returns:

return void

calculate_pop_fitness(*func*)

calcula a aptidão da população e retorna em lista da classe genética.

Returns:

return void

cross_over_one_point(*population*)
Esta função realiza o cruzamento de um ponto no gene.

Args: :param population:(`list`): Lists one containing a population. :param parents :(`float`): parents gene to carry out the mutation.

Returns:

cross_over_two_points(*population*)
Esta função realiza o cruzamento de dois pontos no gene.

Args: :param population:(`list`): Lists one containing a population. :param parents :(`float`): parents gene to carry out the mutation.

Returns:

cross_over_uniform(*population*)
Esta função realiza o cruzamento uniforme no gene.

Args: :param population:(`list`): Lists one containing a population. :param parents :(`float`): parents gene to carry out the mutation.

Returns:

do_full_step(*func*, ***kwargs*)
Execute uma etapa completa de GA.

Este método passa por todos os outros métodos para realizar uma completa Etapa GA, para que possa ser chamada a partir de um loop no método run ()..

elitism_selection()
Função que realiza seleção elitista.

Args: :param population:(`list`): Lists one containing a population.

Returns: :return:population(`list`): An element of the population list select.

evaluate_single_fitness_test(*func*, *enum_particles=False*, *add_step_num=False*, ***kwargs*)
Execute a função fornecida como o teste de aptidão para todas as partículas.

fun [callable] The fitness test function to be minimized:

```
func(individual.ichromosome, **kwargs) -> float.
```

enum_particles [boolean] If *True*, the population will be enumerated and the individual index will be passed to *func* as keyword *part_idx*, added to *kwargs*

add_step_num [boolean] If *True*, the current step number will be passed to *func* as keyword *step_num*, added to *kwargs*

**kwargs: Other keywords to the fitness function, will be passed as is.

fitness_variation(*fitness_evaluation*)
Essa função realiza a verificação da variação do fitness entre as populações

Args: fitness_evaluation ([type]): [description]

mutation_binary(*population*)
A função realiza a mutação binária e retorna a população com a modificação, vale ressaltar que esta mutação só é válida para população binária.

Args: :param population:(`list`): Lists one containing a population.

Returns:

mutation_gaussian(*population*)
A função realiza a mutação de Gausiana e retorna a população com a modificada.

Args: :param population:(`list`): Lists one containing a population.

Returns:

mutation_uniform(*population*)
A função realiza a mutação uniforme e retorna a população modificada.

Args: :param population:(`list`): Lists one containing a population.

Returns:

populate(*size_population*, *bounds=None*, *x0=None*, *ndim=None*, *sigma=None*, *type_create='uniform'*)
Retorna uma lista consistindo de vários indivíduos que formam a população.

Return:

return void

random_selection()
Esta função realiza uma seleção do torneio e retorno gene vencedor.

Args: :param population:(`list`): Lists one containing a population.

Returns:

return sub_population(`list`): An element of the sub_population list select.

roulette_selection()
Esta função realiza a seleção por releta do gene a ser realizado a mutação.

Args:

param population population:(`list`): Lists one containing a population.

Returns:

return selected(`list`): An element of the population list select.

run(*func*, *DEBUG=None*, ***kwargs*)
Execute uma execução de otimização completa.

Faz a otimização com a execução da atualização das velocidades e as coordenadas também verifica o critério interrompido para encontrar fitnnes.

func [callable] Function that calculates fitnnes.

The dictionary that stores the optimization results.

selection()
[summary]

Raises: ValueError: [description]

Returns: [type]: [description]

static sorted_population(*population*)
Selecione o gene a ser realizado a mutação.

Args: :param population:(`list`): Lists one containing a population.

Returns:

return score(`list`): An element of the population list select.

update_swarm()
Atualize a população realizando cruzamento, mutação

Returns: :return population(`list` of `Particles`): returns a list of swarms.

class ga.**Individual**(*x0=np.zeros(2)*, *ndim=2*, *bounds=np.array(np.ones(2) * 10 * - 1, np.ones(2) * 10)*, *type_create='uniform'*)
Cria os individuos que compoem a população da classe Genetic

fitness [None or flout] é o melhor valor do individuo

size_individual [int] é o tamanho do indiviuo

create_individual [class 'fffit.ga.Individual'] cria o individuo com suas caracteristicas

type_create [str, optional] Define qual será o tipo de criação da população inicial

x0 [np.ndarray, optional] Define qual o ponto inicial até os limites de onde vai criar o valor do indviduo

bounds [numpy.ndarray] Define até onde pode ir os valores dos individuo, como inferio e superior

ndim [integer] Define quantos dimensões tem o indiviuo ou seja o tamanho do array

sigma :float, opcional Define a probabilidade do individuo ter mutação

```
>>> from fffit import ga
>>> import numpy as np
>>> ranges = 2
>>> ndim = 2
>>> bounds = np.array((np.ones(2) * 10 * -1, np.ones(2) * 10))
>>> individual = ga.Individual(x0=np.zeros(2), ndim=2, bounds=bounds, sigma=None,␣
↪type_create='uniform')
<fffit.ga.Individual object at 0x7f8968797c18>
>>> individual.chromosome
array([ 5.7287427 , -0.54066483])
```

create_individual(*type_create=uniform*, *x0=np.zeros(2)*, *np.array((np.ones(2) * 10 * -1, np.ones(2) * 10))*, *sigma=None*)
Cria os chromosome que pertence a classe Individual

type_create [str, optional] Define qual será o tipo de criação da população inicial

x0 [np.ndarray, optional] Define qual o ponto inicial até os limites de onde vai criar o valor do indviduo

bounds [numpy.ndarray] Define até onde pode ir os valores dos individuo, como inferio e superior

sigma :float, opcional Define a probabilidade do individuo ter mutação

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX