GenAlgNB

December 10, 2018

The code below is example of how GenAlgorithmString class works. That was simple exercise, the aim was to reate an algorithm that finds the given sentence.

First thing to do is importing the whole algorithm and setting parameters, in the example below those are: n_population - number of individuals for each generation desired_fitness - the condition, when algorithm should stop, when set to zero- it works until it finds the sentence mutation_probability - the probability of mutation threshold

```
In [1]: #importing class GenAlgorithmString
        from genalgorithm import GenAlgorithmString as GA

        #Setting parameters.
        ga = GA(n_population=10,desired_fitness=0,mutation_probability=0.02)

        aim = input('Provide the sentence, that is supposed to be found \n')
```

```
Provide the sentence, that is supposed to be found
Programming is awesome!
```

Method fit() is fitting the given sentence and creating population of individuals.

```
In [2]: ga.fit(aim)
        print('First population:\n')
        print(ga.population)
        print('First population\'s fitnessL \n')
        print(ga._population_fitness(ga.population))
```

```
First population:

[['a' '.' 'h' '}' '?' ')' '<' '_' '/' 'C' '"' '[' 'j' '?' '/' '-' '&' '@'
  'f' 'u' 'v' 'X' 'm']
 ['-' 'y' 'J' ')' '_' 'G' '?' 'L' '=' 'v' 'W' '+' 'y' 'D' 'g' 'c' '%' 'C'
  ':' 'm' '<' '?' 'H']
 ['g' 'q' '\\' 'w' '}' 'n' 'U' 'g' 'i' 'x' 'Q' '`' 'k' 'A' '`' 'u' 'y'
  'b' 'K' ';' '@' '#' '.']
 ['(' '^' '/' 'P' '&' 'A' 'l' 'h' 'X' 'U' 'E' '>' '<' ',' 'H' 'v' '^' '.'
  'P' 'q' '}' 'y' '~']
 ['Q' '?' 'U' ']' 'y' '&' 'X' 'O' ']' 'M' ':' 'e' '.' 'N' '=' 'l' ';' '<'
```

```
  't' '\\' 'w' 'r' ':']
 ['?' '"' '"' 'Q' '*' 't' 'P' 'F' 'q' ')' 'z' 'N' 'Q' 'Z' 'f' '\\' 'A'
  'a' '?' '\\' 'q' 'S' '?']
 ['@' 'P' 'R' 'k' 'U' 'i' '`' 'S' 'w' 'I' 'I' '#' ',' '*' 'D' 'c' 'j' 'd'
  'g' 'K' 'O' '=' '~']
 ['A' '!' '>' '=' 'x' '~' 'K' 'o' 'P' 'A' 'U' '"' 'k' 'Q' 'A' 'J' ':' ']'
  'I' 'P' ']' '%' 'I']
 ['@' '~' 'W' 'k' 't' '>' 'e' 'V' 'p' 'b' '=' '\\' 'd' 'T' '=' '?' 'a'
  '!' "'" 'O' 'b' 'R' '-']
 ['V' 'n' 'r' '[' 'I' '<' 'o' 'C' 'u' 'W' 's' 'i' '$' ',' 'Z' 'b' '~' 'F'
  '?' 'G' '@' 'o' '}']]
First population's fitnessL

[868 781 566 790 669 802 640 702 584 743]
```

Now, when the first generation is created, we can start execution of our algorithm, with transform() method.

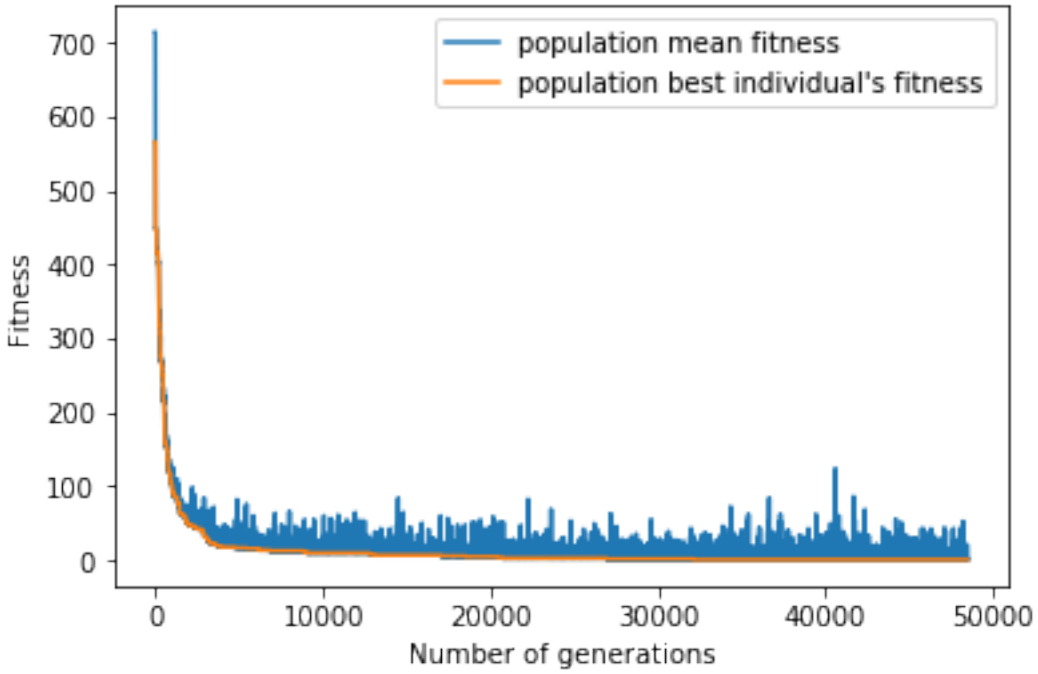
```
In [3]: #Executing an algorithm
        ga.transform()
        print('Last population:')
        print(ga.population)
        print('best_individual:')
        print(ga.best_individual(ga.population))
```

```
Last population:
[['P' 'r' 'o' 'f' 'r' 'a' 'm' 'm' 'i' 'n' 'g' ' ' 'i' 's' ' ' 'a' 'w' 'e'
  's' 'o' 'm' 'e' '!']
 ['P' 'r' 'o' 'g' 'r' 'a' 'm' 'm' 'i' 'n' 'g' ' ' 'i' 's' ' ' 'a' 'w' 'e'
  's' 'o' 'm' 'e' '!']
 ['P' 'r' 'o' 'f' 'r' 'a' 'm' 'm' 'i' 'n' 'g' ' ' 'i' 's' ' ' 'a' 'w' 'e'
  's' 'o' 'm' 'e' '!']
 ['P' 'r' 'o' 'f' 'r' 'a' 'm' 'm' 'i' 'n' 'g' ' ' 'i' 's' ' ' 'a' 'w' 'e'
  's' 'o' 'm' 'e' '!']
 ['P' 'r' 'o' 'f' 'r' 'a' 'm' 'm' 'i' 'n' 'g' ' ' 'i' 's' ' ' 'a' 'w' 'e'
  's' 'o' 'm' 'e' '!']
 ['P' 'r' 'o' 'f' 'r' 'a' 'm' 'm' 'i' 'n' 'g' ' ' 'i' 's' ' ' 'a' 'w' 'e'
  's' 'o' 'm' 'e' '!']
 ['P' 'r' 'o' 'f' 'r' 'a' 'm' 'm' 'i' 'n' 'g' ' ' 'i' 's' ' ' 'a' 'w' 'e'
  's' 'o' 'm' 'e' '!']
 ['P' 'r' 'o' 'f' 'r' 'a' 'm' 'm' 'i' 'n' 'g' ' ' 'i' 's' ' ' 'a' 'w' 'e'
  's' 'o' 'm' 'e' '!']
 ['P' 'r' 'o' 'f' 'r' 'a' 'm' 'm' 'i' 'n' 'g' ' ' 'i' 's' ' ' 'a' 'w' 'e'
  's' 'o' 'm' 'e' '!']
 ['P' 'r' 'o' 'f' 'r' 'a' 'm' 'm' 'i' 'n' 'g' ' ' 'i' 's' ' ' 'a' 'w' 'e'
  's' 'o' 'm' 'e' '!']]
best_individual:
```

```
['P' 'r' 'o' 'g' 'r' 'a' 'm' 'm' 'i' 'n' 'g' ' ' 'i' 's' ' ' 'a' 'w' 'e'
 's' 'o' 'm' 'e' '!']
```

Once we achieved our goal, we can plot to see how many generations did it take and how the algorithm is performing.

```
In [5]: ga.plot_fitness()
```



Algorithm was performing accurately, it took a lot of generations actually to find the best individual.