Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mestrado Integrado em Engenharia Informática
Computação Natural
2018/2019

Paulo Novais, Cesar Analide, Filipe Gonçalves

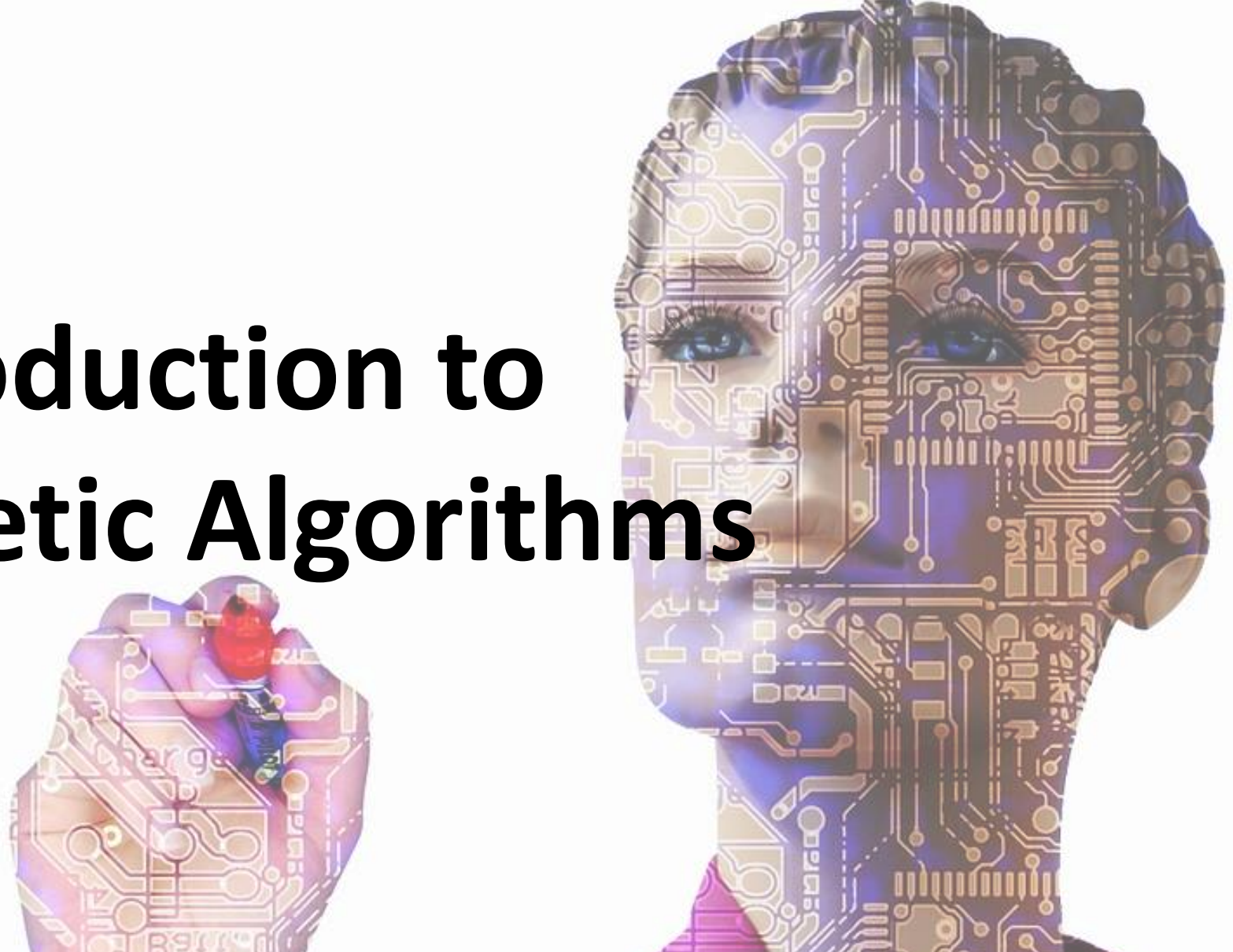
ISLab
Synthetic Intelligence Lab

Computação Natural@ 2018/2019

- Paulo Novais – pjon@di.uminho.pt
- Cesar Analide – cesar.analide@di.uminho.pt
- Filipe Gonçalves – fgoncalves@algoritmi.uminho.pt

- Departamento de Informática
  Escola de Engenharia
  Universidade do Minho
- Grupo ISLab – (Synthetic Intelligence Lab)
- Centro ALGORITMI
  Universidade do Minho

Introduction to Genetic Algorithms



- Training a Machine Learning Model
  - Data:

x1	x2	x3	x4	x5	x6	y
4	-2	7	5	11	1	44.1

$$Y = w1.x1 + w2.x2 + w3.x3 + w4.x4 + w5.x5 + w6.x6$$

  - Goal is to find the set of parameters (w1,w2,w3,w4,w5,w6) that maps the following input to its output:

$$Y' = 4.w1 + (-2).w2 + 7.w3 + 5.w4 + 11.w5 + 1.w6$$

- Solution 1:

w1	w2	w3	w4	w5	w6
2,4	0,7	8	-2	5	1,1

  - $Y' = 4.w1 - 2.w2 + 7.w3 + 5.w4 + 11.w5 + w6 \Leftrightarrow Y' = 110,3$

- Absolute Error:

  Error = $| Y - Y' |$

  Error = $| 44,1 - 110,3 |$

  Error = 66,2

- Solution 2:

w1	w2	w3	w4	w5	w6
-0,4	2,7	5	-1	7	0,1

  - Y' = -0,4.w1 + 2,7.w2 + 5.w3 - 1.w4 + 7.w5 + 0,1.w6 ⇔ Y' = 100,1

- Absolute Error:

  Error = | Y – Y' |

  Error = | 44,1 – 100,1 |

  Error = 56

- Solution 3:

w1	w2	w3	w4	w5	w6
-1	2	2	-3	2	0,9

  - Y' = -1.w1 + 2.w2 + 2.w3 - 3.w4 + 2.w5 + 0,9.w6 ⇔ Y' = 13,9

- Absolute Error:

  Error = | Y – Y' |

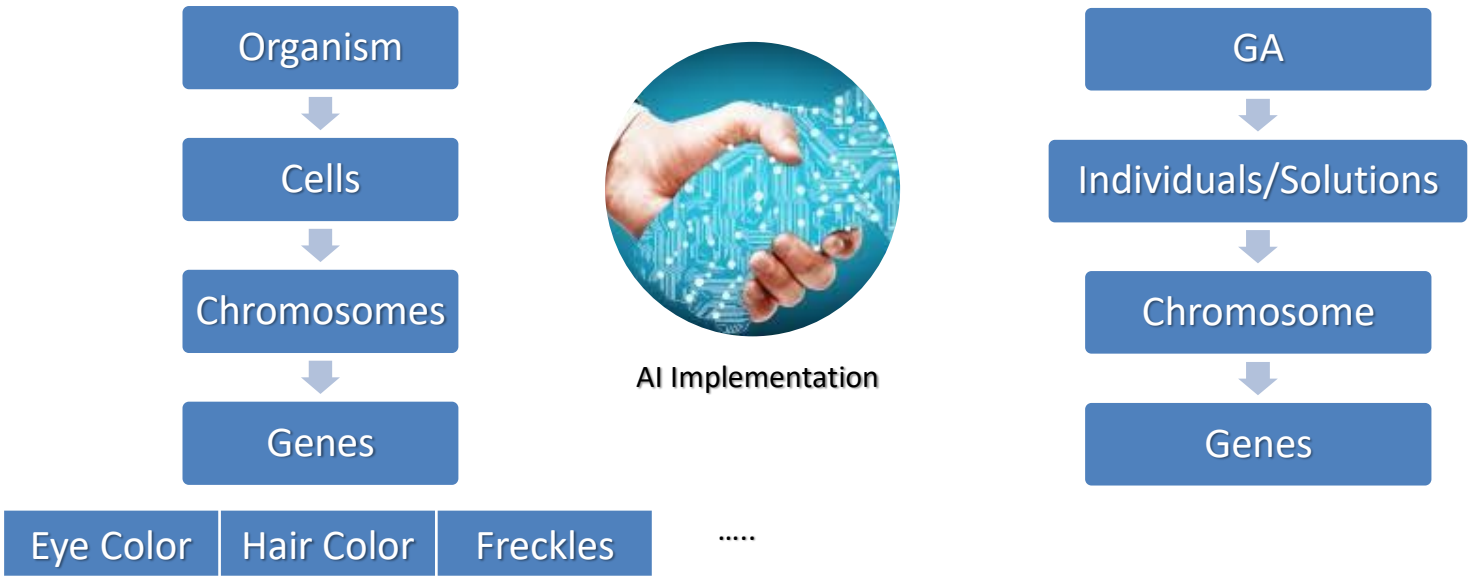
  Error = | 44,1 – 13,9 |

  Error = 30,2

Difficult to find the best solution manually!

Use optimization Techniques such as **Genetic Algorithms (GA)**

- Genetic Algorithm (GA)
  - Based on Natural Evolution of organisms
  - A brief biological background is helpful to better understand GA



- What are Genes?

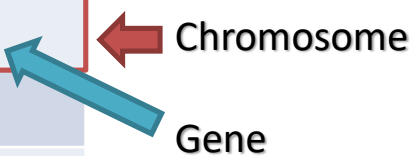
  Y = w1.x1 + w2.x2 + w3.x3 + w4.x4 + w5.x5 + w6.x6

  - Gene is anything that is able to enhance the results when changed
  - By exploring the following model, the 6 weights are able to enhance the results
    - Thus each weight will represent a gene in GA

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
w1	w2	w3	w4	w5	w6

- Initial Population of Solutions (Generation 0)

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
2,4	0,7	8	-2	5	1,1
-0,4	2,7	5	-1	7	0,1
-1	2	2	-3	2	0,9
4	7	12	6,1	1,4	-4
3,1	4	0	2,4	4,8	0
-2	3	-7	6	3	3



- Initial Population of Solutions (Generation 0)

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
2,4	0,7	8	-2	5	1,1
-0,4	2,7	5	-1	7	0,1
-1	2	2	-3	2	0,9
4	7	12	6,1	1,4	-4
3,1	4	0	2,4	4,8	0
-2	3	-7	6	3	3

Survival of the Fittest

↓

Fitness Function

↓

Fitness Value

↓

Higher Value => Better Solution

- Initial Population of Solutions (Generation 0)

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Y'	F(C)
2,4	0,7	8	-2	5	1,1	110,3	
-0,4	2,7	5	-1	7	0,1		
-1	2	2	-3	2	0,9		
4	7	12	6,1	1,4	-4		
3,1	4	0	2,4	4,8	0		
-2	3	-7	6	3	3		

$Y' = 4.w1 - 2.w2 + 7.w3 + 5.w4 + 11.w5 + w6$

$Y' = 4 \times 2,4 - 2 \times 0,7 + 7 \times 8 + 5 \times (-2) + 11 \times 5 + 1 \times 1,1$

$Y' = 110,3$

- Initial Population of Solutions (Generation 0)

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Y'	F(C)
2,4	0,7	8	-2	5	1,1	110,3	0,015
-0,4	2,7	5	-1	7	0,1		
-1	2	2	-3	2	0,9		
4	7	12	6,1	1,4	-4		
3,1	4	0	2,4	4,8	0		
-2	3	-7	6	3	3		

$$F(c) = \frac{1}{error} = \frac{1}{|44.1 - 110.3|} = \frac{1}{66.2} = 0.015$$

- Initial Population of Solutions (Generation 0)

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Y'	F(C)
2,4	0,7	8	-2	5	1,1	110,3	0,015
-0,4	2,7	5	-1	7	0,1	100,1	0,018
-1	2	2	-3	2	0,9	13,9	0,033
4	7	12	6,1	1,4	-4	127,9	0,012
3,1	4	0	2,4	4,8	0	69,2	0,0398
-2	3	-7	6	3	3	3	0,024

- Mating Pool

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Y'	F(C)
2,4	0,7	8	-2	5	1,1	110,3	0,015
-0,4	2,7	5	-1	7	0,1	100,1	0,018
-1	2	2	-3	2	0,9	13,9	0,033
4	7	12	6,1	1,4	-4	127,9	0,012
3,1	4	0	2,4	4,8	0	69,2	0,0398
-2	3	-7	6	3	3	3	0,024

Select best individuals as parents for mating to generate new individuals.

- Mating Pool

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Y'	F(C)
2,4	0,7	8	-2	5	1,1	110,3	0,015
-0,4	2,7	5	-1	7	0,1	100,1	0,018
-1	2	2	-3	2	0,9	13,9	0,033
4	7	12	6,1	1,4	-4	127,9	0,012
3,1	4	0	2,4	4,8	0	69,2	0,0398
-2	3	-7	6	3	3	3	0,024

Add top 3 individuals to the mating pool for producing the next generation of solutions

- Mating Pool

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
-1	2	2	-3	2	0,9
3,1	4	0	2,4	4,8	0
-2	3	-7	6	3	3

- Mating Pool

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
-1	2	2	-3	2	0,9
3,1	4	0	2,4	4,8	0

- Mating Pool

Crossover

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
-1	2	2	-3	2	0,9
3,1	4	0	2,4	4,8	0

Offspring

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
-1	2	2	2,4	4,8	0

= 4,8 / 2 = 2,4

Mutation

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
-1	2	2	2,4	2,4	0

- Mating Pool

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
3,1	4	0	2,4	4,8	0
-2	3	-7	6	3	3

- Mating Pool

Crossover

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
3,1	4	0	2,4	4,8	0
-1	8	-7	6	3	3

Offspring

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
3,1	4	0	6	3	3

= 3 / 2 = 1,5

Mutation

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
3,1	4	0	6	1,5	3

- Mating Pool

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
-2	3	-7	6	3	3
-1	2	2	-3	2	0,9

- Mating Pool

Crossover

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
-2	3	-7	6	3	3
-			-3	2	0,9

Offspring

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
-2	3	-7	-3	2	0,9

= 2 / 2 = 1

Mutation

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
-2	3	-7	-3	1	0,9

- New Population (Generation 1)

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	
-1	2	2	-3	2	0,9	Old Individuals
3,1	4	0	2,4	4,8	0	
-2	3	-7	6	3	3	
-1	2	2	2,4	2,4	0	New Individuals
3,1	4	0	6	1,5	3	
-2	3	-7	-3	1	0,9	

Since there is no garantee that the new individuals will be better than the previous individuals, keeping old individuals saves the results from getting worse

- New Population (Generation 1)

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	
-1	2	2	-3	2	0,9	Old Individuals
3,1	4	0	2,4	4,8	0	
-2	3	-7	6	3	3	
-1	2	2	2,4	2,4	0	New Individuals
3,1	4	0	6	1,5	3	
-2	3	-7	-3	1	0,9	

- New Population (Generation 1)

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Y'	F(C)
-1	2	2	-3	2	0,9	13,9	0,033
3,1	4	0	2,4	4,8	0	69,2	0,04
-2	3	-7	6	3	3	3	0,024
-1	2	2	2,4	2,4	0	44,4	3,333
3,1	4	0	6	1,5	3	53,9	0,102
-2	3	-7	-3	1	0,9	-66,1	0,009

Note: New individuals presented worse results than the old individuals.

- Mating Pool

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
3,1	4	0	2,4	4,8	0
-1	2	2	2,4	2,4	0
3,1	4	0	6	1,5	3

- Repeat the process to generate new Population (Generation 2)

- New Population (Generation 2)

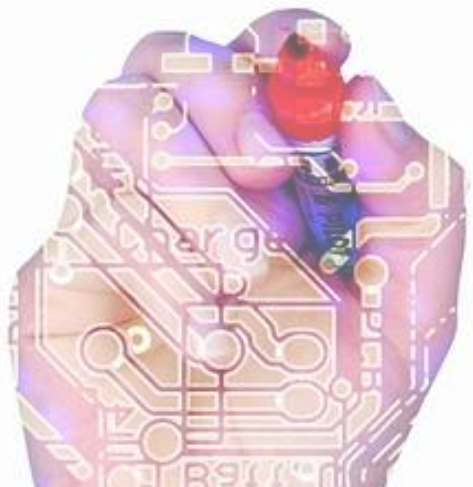
Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	
3,1	4	0	2,4	4,8	0	Old Individuals
-1	2	2	2,4	2,4	0	
3,1	4	0	6	1,5	3	
3,1	4	0	2,4	1,2	0	New Individuals
-1	2	2	6	0,75	3	
3,1	4	0	2,4	2,4	0	

- New Population (Generation 2)

Gene 0	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Y'	F(C)
3,1	4	0	2,4	4,8	0	69,2	0,04
-1	2	2	2,4	2,4	0	44,4	3,333
3,1	4	0	6	1,5	3	53,9	0,102
3,1	4	0	2,4	1,2	0	29,6	0,069
-1	2	2	6	0,75	3	47,25	0,318
3,1	4	0	2,4	2,4	0	42,8	0,77

- Continue the process!

GA – ML Optimization Implementation



- Code:

```
import numpy


# Population size
sol_per_pop = 6
# Mating pool size
num_parents_mating = 3

# Creating an initial population randomly.
new_population = numpy.zeros((6, 6))
new_population[0, :] = [2.4, 0.7, 8, -2, 5, 1.1]
new_population[1, :] = [-0.4, 2.7, 5, -1, 7, 0.1]
new_population[2, :] = [-1, 2, 2, -3, 2, 0.9]
new_population[3, :] = [4, 7, 12, 6.1, 1.4, -4]
new_population[4, :] = [3.1, 4, 0, 2.4, 4.8, 0]
new_population[5, :] = [-2, 3, -7, 6, 3, 3]

for iteration in range(10):
    # Measing the fitness of each chromosome in the population.
    qualities = GARI.pop_fitness(new_population)

    # Selecting the best parents in the population for mating.
    parents = GARI.select_mating_pool(new_population, qualities,
                                      num_parents_mating)
    # Generating next generation using crossover.
    new_population = GARI.crossover(parents,
                                    n_individuals=sol_per_pop)

    new_population = GARI.mutation(new_population)

in_sample = [4, -2, 7, 5, 11, 1]
for k in range(6):
    print(numpy.sum(new_population[k, :] * in_sample))
```

```
def fitness_fun(indiv_chrom):
    in_sample = [4,-2,7,5,11,1]
    quality = 1/numpy.abs(44.1-numpy.sum(indiv_chrom*in_sample))
    return quality

def pop_fitness(pop):
    qualities = numpy.zeros(pop.shape[0])
     for indv_num in range(pop.shape[0]):
         qualities[indv_num] = fitness_fun(pop[indv_num, :])
     return qualities
def select_mating_pool(pop, qualities, num_parents):
    parents = numpy.empty((num_parents, pop.shape[1]))
    for parent_num in range(num_parents):
        max_qual_idx = numpy.where(qualities == numpy.max(qualities))
        max_qual_idx = max_qual_idx[0][0]
        parents[parent_num, :] = pop[max_qual_idx, :]
        qualities[max_qual_idx] = -1
    return parents
```

```
def crossover(parents, n_individuals=6):
    new_population = numpy.empty((6,6))

    #Previous parents (best elements).
    new_population[0:parents.shape[0], :] = parents
    new_population[3, 0:3] = parents[0, 0:3]
    new_population[3, 3:] = parents[1, 3:]
    new_population[4, 0:3] = parents[1, 0:3]
    new_population[4, 3:] = parents[2, 3:]
    new_population[5, 0:3] = parents[2, 0:3]
    new_population[5, 3:] = parents[0, 3:]

    return new_population

def mutation(population):
    for idx in range(population.shape[0]):
        population[idx, 4] = population[idx, 4]/3
    return population
```

- GA Example – Crack Account Password

```
import random
import datetime

def get_fitness(guess):
    return sum(1 for expected, actual in zip(target, guess)
                 if expected == actual)


def display(guess):
    timeDiff = datetime.datetime.now() - startTime
    fitness = get_fitness(guess)
    print("{0}\t{1}\t{2}".format(guess, fitness, str(timeDiff)))


def generate_parent(length):
    genes = []
    while len(genes) < length:
        sampleSize = min(length - len(genes), len(geneSet))
        genes.extend(random.sample(geneSet, sampleSize))
    return ''.join(genes)
```

```
def mutate(parent):
    index = random.randrange(0, len(parent))
    childGenes = list(parent)
    newGene, alternate = random.sample(geneSet, 2)
    childGenes[index] = alternate \
        if newGene == childGenes[index] \
        else newGene
    return ''.join(childGenes)


random.seed()
geneSet = " abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!."
target = "PasswordDesconhecida"
startTime = datetime.datetime.now()
bestParent = generate_parent(len(target))
bestFitness = get_fitness(bestParent)
display(bestParent)
while True:
    child = mutate(bestParent)
    childFitness = get_fitness(child)

    if bestFitness >= childFitness:
        continue
    display(child)
    if childFitness >= len(bestParent):
        break
    bestFitness = childFitness
    bestParent = child
```

- GA Example - Result:

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
kD ZSYJQoxC!wXcVqPiA    0   0:00:00.000196
kD ZSYJQoxC!wXcVcPiA    1   0:00:00.001691
kD ZSYJQoxCcwXcVcPiA    2   0:00:00.002332
kD sSYJQoxCcwXcVcPiA    3   0:00:00.002494
kD sSYJQoxCcwXcVcPdA    4   0:00:00.002837
kD sSYJQoxCcwXcVcPda    5   0:00:00.003333
kD sSYJQoxCcwXcVcida    6   0:00:00.004080
kD swYJQoxCcwXcVcida    7   0:00:00.008330
kD swYJQoeCcwXcVcida    8   0:00:00.011033
kD swYrQoeCcwXcVcida    9   0:00:00.012536
kD swYrQoeCcoXcVcida    10  0:00:00.012997
kD swYrQoescoXcVcida    11  0:00:00.019064
PD swYrQoescoXcVcida    12  0:00:00.019218
PD sworQoescoXcVcida    13  0:00:00.021669
Pa sworQoescoXcVcida    14  0:00:00.022342
Pa sworQoescoXcecida    15  0:00:00.024002
Pa sworQoescoXhecida    16  0:00:00.024711
PassworQoescoXhecida    17  0:00:00.025256
PasswordoescoXhecida    18  0:00:00.094835
Passwordoesconhecida    19  0:00:00.097151
PasswordDesconhecida    20  0:00:00.109849
```

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mestrado Integrado em Engenharia Informática
Computação Natural
2018/2019

Paulo Novais, Cesar Analide, Filipe Gonçalves