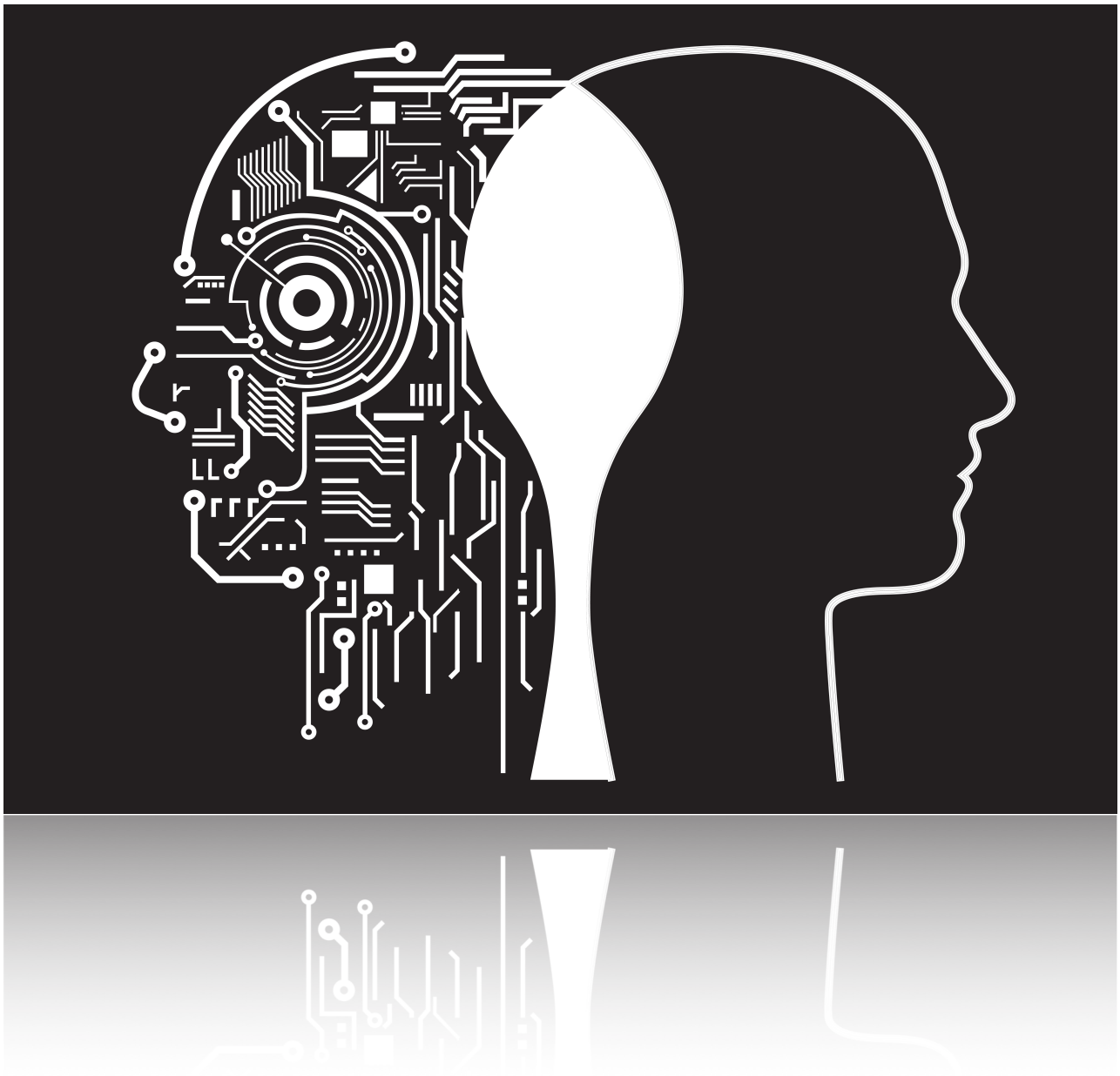
Genetic Algorithm Optimization

Elements of Artificial Intelligence

Coursework II

Caetán Tojeiro Carpente

7 de diciembre de 2016



INDEX

Introduction

1 - What is Genetic Algorithm (*GA*)?

Genetic Algorithm (*GA*) is a metaheuristic based on the process of natural selection. The idea is so close with the principles of Charles Darwin. Commonly this kind of algorithms are used to generate high-quality solutions to problems of optimization.

2 - Origin

The father of the original GA was *John Henry Holland* (1929-2015) who developed it in the early 1970's. This kind of algorithms were born with the target of simulated processes that we can observe in natural evolution. The idea is use this power of evolution to solve some optimization problems.

Coursework

1 - Algorithm to implement

The algorithm which will be implemented is GA. We have to minimize the length of the journey between N[1] cities without repeat visits to any cities. This kind of problem is known as "Traveling salesman problem" (*TSP*).

[1] For this optimization problem N=10

2 - Python code

The programming language that is used for this task is *Python*. The choice of this language is motivated because of the easiness that *Python* gives us to this kind of matters.

```
import numpy as np
import random,time

#Function to calculate the cost of the route given

def funcion_calcular_custo(ruta):
      custo=0
      ruta1=ruta
      for i in range(9):
            custo=custo+(a.item(ruta1[i],ruta1[i+1]))
      return custo


#Mutate between 2 random cities with a 15% of probability

def funcion_mutar(ruta):
      rutaMUTADA=ruta
      if random.randint(0,100)<15:
            A=random.randint(0,9)
            B=random.randint(0,9)
            rutaMUTADA[A],rutaMUTADA[B]=rutaMUTADA[B],rutaMUTADA[A]
      return rutaMUTADA


A=0
B=0
custoMIN=1000
route1=[0,1,2,3,4,5,6,7,8,9]
a = np.matrix(  ' 0 5 4 7 6 5 7 4 2 9;' +
                        ' 6 0 5 6 4 8 5 4 3 8;' +
                        ' 3 5 0 3 5 6 9 8 7 6;' +
                        ' 7 5 4 0 3 5 7 9 8 3;' +
                        ' 5 4 5 3 0 4 6 7 8 7;' +
                        ' 5 6 5 5 4 0 5 4 3 2;' +
                        ' 6 7 9 7 6 6 0 5 7 9;' +
                        ' 5 4 8 7 6 4 4 0 6 5;' +
                        ' 2 3 6 9 8 3 7 5 0 7;' +
                        ' 9 7 5 4 8 3 9 5 7 0')

class Ruta:
            def __init__(self, path=[0,1,2,3,4,5,6,7,8,9]):
                  self.path=path

            def __str__(self):
                  return str(["ABCDEFGHIJ"[i] for i in self.path])
```

```
for x in range(0,1000):

      #Generate a pseudo-random father route


routeRANDOM=[random.randint(0,9),random.randint(0,9),random.randint(0,9),random
.randint(0,9),random.randint(0,9),random.randint(0,9),random.randint(0,9),rando
m.randint(0,9),random.randint(0,9),random.randint(0,9)]
      np.unique(routeRANDOM)
      routeRANDOMAUX=routeRANDOM + route1
      indexes = np.unique(routeRANDOMAUX, return_index=True)[1]
      route4=[routeRANDOMAUX[index] for index in sorted(indexes)]
      #print(Ruta(route4))


routeRANDOM=[random.randint(0,9),random.randint(0,9),random.randint(0,9),random
.randint(0,9),random.randint(0,9),random.randint(0,9),random.randint(0,9),rando
m.randint(0,9),random.randint(0,9),random.randint(0,9)]
      np.unique(routeRANDOM)
      routeRANDOMAUX=routeRANDOM + route1
      indexes = np.unique(routeRANDOMAUX, return_index=True)[1]
      route5=[routeRANDOMAUX[index] for index in sorted(indexes)]
      #print(Ruta(route5))


routeRANDOM=[random.randint(0,9),random.randint(0,9),random.randint(0,9),random
.randint(0,9),random.randint(0,9),random.randint(0,9),random.randint(0,9),rando
m.randint(0,9),random.randint(0,9),random.randint(0,9)]
      np.unique(routeRANDOM)
      routeRANDOMAUX=routeRANDOM + route1
      indexes = np.unique(routeRANDOMAUX, return_index=True)[1]
      route6=[routeRANDOMAUX[index] for index in sorted(indexes)]
      #print(Ruta(route6))


routeRANDOM=[random.randint(0,9),random.randint(0,9),random.randint(0,9),random
.randint(0,9),random.randint(0,9),random.randint(0,9),random.randint(0,9),rando
m.randint(0,9),random.randint(0,9),random.randint(0,9)]
      np.unique(routeRANDOM)
      routeRANDOMAUX=routeRANDOM + route1
      indexes = np.unique(routeRANDOMAUX, return_index=True)[1]
      route7=[routeRANDOMAUX[index] for index in sorted(indexes)]
      #print(Ruta(route7))


      #Crossover

      routeSon=route4[1:6] + route5[3:8]
      np.unique(routeSon)
      routeSon=routeSon + route1
      indexes = np.unique(routeSon, return_index=True)[1]
      routeSon=[routeSon[index] for index in sorted(indexes)]
      #print(Ruta(routeSon))

      routeSon2=route6[0:3] + route7[4:9]
      np.unique(routeSon2)
      routeSon2=routeSon2 + route1
      indexes = np.unique(routeSon2, return_index=True)[1]
```

```
routeSon2=[routeSon2[index] for index in sorted(indexes)]
#print(Ruta(routeSon2))

routeSon3=route7[1:6] + route5[3:8]
np.unique(routeSon3)
routeSon3=routeSon3 + route1
indexes = np.unique(routeSon3, return_index=True)[1]
routeSon3=[routeSon3[index] for index in sorted(indexes)]
#print(Ruta(routeSon3))

routeSon4=route4[0:3] + route7[4:9]
np.unique(routeSon4)
routeSon4=routeSon4 + route1
indexes = np.unique(routeSon4, return_index=True)[1]
routeSon4=[routeSon4[index] for index in sorted(indexes)]
#print(Ruta(routeSon4))



#Mutation & calculate the cost of the route muted

rutaMUTADA=funcion_mutar(routeSon)
custoTotalRuta=funcion_calcular_custo(rutaMUTADA)
if custoTotalRuta<custoMIN:
      custoMIN=custoTotalRuta
      rutaBOA=rutaMUTADA


rutaMUTADA=funcion_mutar(routeSon2)
custoTotalRuta=funcion_calcular_custo(rutaMUTADA)
if custoTotalRuta<custoMIN:
      custoMIN=custoTotalRuta
      rutaBOA=rutaMUTADA

rutaMUTADA=funcion_mutar(routeSon3)
custoTotalRuta=funcion_calcular_custo(rutaMUTADA)
if custoTotalRuta<custoMIN:
      custoMIN=custoTotalRuta
      rutaBOA=rutaMUTADA


rutaMUTADA=funcion_mutar(routeSon4)
custoTotalRuta=funcion_calcular_custo(rutaMUTADA)
if custoTotalRuta<custoMIN:
      custoMIN=custoTotalRuta
      rutaBOA=rutaMUTADA

#time.sleep(1)

print "Iteration", x+1, "COST MIN",custoMIN, "FOR ROUTE", Ruta(rutaBOA)


#We "kill" fathers and now the sons will be the fathers
route4=routeSon
route5=routeSon2
route6=routeSon3
route7=rutaBOA  #Here we save the best route ever
                #to create a father which will be the best one
```

3 - Notes

- This code is only valid for this exercise, is not for general purpose.
- We suppose that the salesman ends his route at the last city and he doesn't finish the route at the first city.
- Always the best son will be father. In other words, we save the best route and use it in the next iteration to create a son.
- The population is always constant.
- The probability of mutation is 15%