

Genetic Algorithms

The Traveling Salesman Problem

Guidoux Vincent & Guillaume Hochet | HEIG-VD, Machine Learning | June 11, 2019

# Introduction

The travelling salesman problem (TSP) asks the following question: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city? It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

In Machine Learning course, we were asked to resolve a TSP problem given fourteen latitudes and longitudes located in Burma, after some training on simple genetic algorithms problems. Those simple problems leaded us to imagine, create and test a solution for this problem

Our solution tends to test a lot of possibilities, we tested mutation, crossover, population size, number of generations.

# C:\Users\Vincent Guidoux\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\EF00C314.tmpOur shortest path

0: (16.47, 94.44)

1: (20.09, 94.55)

2: (20.09, 92.54)

3: (22.39, 93.37)

4: (25.23, 97.24)

5: (22.0, 96.05)

6: (21.52, 95.59)

7: (20.47, 97.02)

8: (19.41, 97.13)

9: (17.2, 96.29)

10: (16.53, 97.38)

11: (16.3, 97.38)

1 : Solution of the shortest tour

12: (14.05, 98.12)

13: (16.47, 96.1)

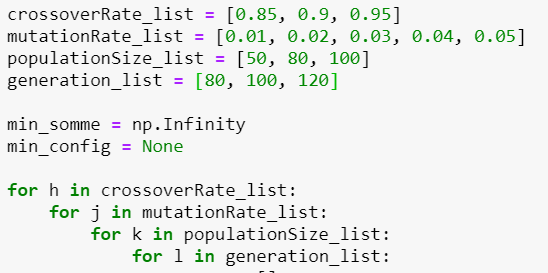
This is the best solution we found, with the vincenty formula for the distance, it’s 3346,762 Km long. Our fitness solution was to sum the distance between the cities, in the order of the tour. We can’t verify if it’s the shortest path, we could have run a script that test all the possible paths, but it would take us 4 days, this is due to the fact that the TSP is a NP-complete problem. We managed to ask our collaborators and it’s the shortest path found by people.

# Our solution

Our chromosome would look like a list of number between 0 and 13 which one represent the 14 cities. So, it would represent in which order the tour is to make.



2: example of one of our chromosome



We made 5-nested loops to test 4 parameters and run the experience a multiple time to compute the mean of an experience. We change the parameters during the practical work, we searched a way not to have always a best solution, but the quickest way to have a good solution, while decreasing the population size and the number of generations

One of the best results was with these parameters :

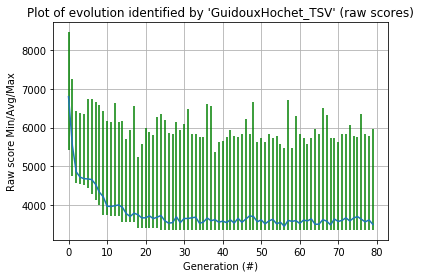
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Crossover rate** | **Mutation rate** | **Population size** | **Generation nb** | **Score (mean)** |
| 0.9 | 0.05 | 80 | 100 | 3403.011365 |
| 0.95 | 0.03 | 100 | 120 | 3405.802383 |
| 0.9 | 0.05 | 100 | 80 | 3408.707522 |
| 0.85 | 0.05 | 100 | 80 | 3409.995064 |
| 0.9 | 0.03 | 50 | 120 | 3410.844585 |

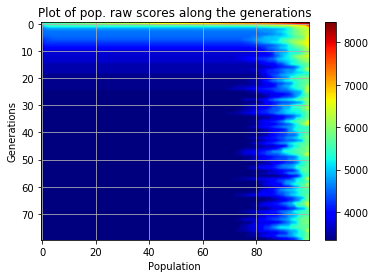
Ce n’est pas déterministe, alors parfois ça nous donne la meilleure solution, quelquefois pas. C’est pour ça que nous nous permettons de mettre plusieurs solutions

# Results

These plots show when we test with those parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| **Crossover rate** | **Mutation rate** | **Population size** | **Generation nb** |
| 0.95 | 0.02 | 100 | 80 |

The blue line seen in the plot above represents the shortest path in km for each generation. In our fitness function, there is no penalty, like a really bad score if path cross each over. Therefore the range varies a lot.

This heatmap represents the plot of population scores by the number of generations. There’s a lot of variations, but we see that the more there’s generation

# Conclusion

It would have taken us 4 days to compute all the 6227020800 tour possibilities, with with solutions, it took us seconds, it’s really powerful