Workshop #2: Swarm Intelligence and Sinergy:
Ant Colony for the Traveling Salesman Problem

Juan Diego Contreras Yepes 20211020069

System Analisis

Ing. Carlos Andrés Sierra Virgüez

Systems engineer

Universidad Distrital Francisco Jose De Caldas

2024-1

Bogota D.C

Workshop Definition:

In this second workshop, the idea is you solve the Traveling Salesman Problem (TSP) using Ant Colony Optimization (ACO) algorithm. It is time to apply a very popular swarm intelligence algorithm to a problem in the real world.

Imagine you have been hired as an artificial intelligence engineer in an important logistic company. Your boss, the Chief Technology Officer, wants to get the shortest path to deliver some products. You analyze the problem, and figure out it is like TSP, and you remember you know some swarm intelligent algorithms useful to solve this problem, in this case, ACO.

Here you will have some tasks to complete this workshop:

- Create some diagrams and explanations to represent/understand the problem following a system thinking approach.
- Generate several random 3D space points to define cities to be visited by the salesman.
- With random points, generate a list to define the requirements for the route.
- Implement ACO to solve TSP problem. Test different parameters combination.
- Draw a 3D plot to see the output of the algorithm.
- Think of some conclusions based on output analysis.

Write any technical concern/decision/difficulty you think is relevant regarding your work. You must deliver a full report detailing each one of the previous steps. I strongly recommend you use a Jupyter Notebook or a COLAB to write/execute your code. Also, your work must include the report, the code, and a README.md file in repo of the course, everything in a folder called workshop_2.

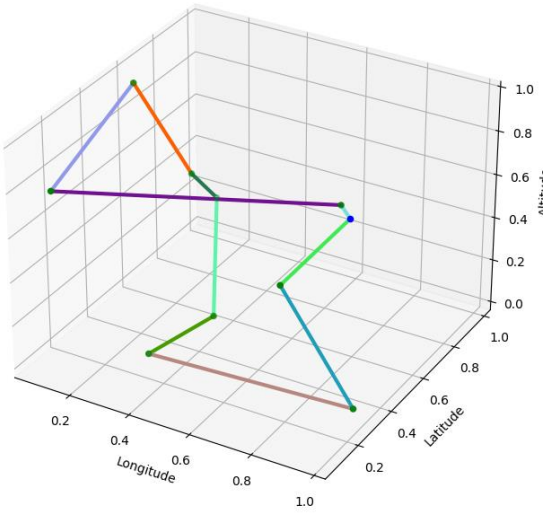
Steps made to complete the workshop

The steps made to complete the TCO with the use of an ACO were:

1. At first, we need to define the TCO, to be more exactly on this step we define the functions to create the list of cities that we're going to use on each try of the code that we made, this will return a list of coordinates (X,Y,Z), and the amount of the list will be define on the parameters of the program, for this function we have use a NumPy option that is np.random.rand(n), this option will select a random number between [0,1] and each of the random numbers are assigned to the coordinates X,Y,Z
2. After that we created the function to define the Euclidean distance between 2 arrays of coordinates, this making use of the normal formula to find the distance and returning a float that will be the answer of the formula
3. The next step that we started on the exercise was the process of make the ACO or Ant colony optimization, for this we started reviewing the amount of cities that we will be recurring and also we started a matrix for the pheromones that will start in 1 with the dimension of the cities that we're recurring, this was made making use of the np.ones option, that will receive the parameters of the dimensions on this case the number of cities and then will create a matrix filled all with ones
4. We will start also programming the simulation on how ants explore cities to find an optimal path, we take the parameters that will be defined for the program (number of ants, iterations, etc.) and create empty lists in order to save each path that we will be finding with each ant that we use on each interaction, for each ant, we start the path with a random select that will choose a starting city, it will check the way in where it can be visited, after that the ant will mark it as visited, and initialize a path. Added to this part of the code, we also solve the randomly option to select the next city that an ant will be visiting, this while there are unvisited cities, each ant needs to calculate probabilities for moving to each unvisited city based on pheromone levels and distances.
5. To this part we also need to add a normalization of the probabilities that we have find to visit the different options for next cities, the normalization is a process in where all the probabilities addition will need to be 1, after checking on different sites, I found that the best option to make this will be to divide each probability on the addition for all the probabilities find on the process and this will be saved as the probabilities variable.
6. As we already notice the process that we were making were for each ant, this process is also repeated for the number of interactions that were selected on the parameters of the program, and on each

interaction we will be finding a path, we use minimization to review if the path that we have find is shorter than the actual best path, if it's the variable best path will be changed for the actual path, each interaction we also remove a bit of pheromone to all the path, this just to verify that the ants that we're using will no continue using the same paths otherwise them will be finding new paths that can be slower than the actuals that have been already find

7. To finish with this, we define the parameters that we will be using, this parameters will be the number_cities that will define how much cities the ants will be traveling, the number_ants that will define the amount of ants that will be checking the paths available, the number_iterations that will define how much try's we will be making in order to find the best option to travel between the cities, the alfa an beta, that will affect the probability of decision for the ants between selecting a path with a shorter distance or a path with a greater amount of pheromone, the evaporation_rate = 0.5 that will define how much amount of the pheromones will be removed for the map on each interaction, and the Q that is a variable that will define how much amount of pheromone an ant will be leaving on each travel for a path; and also with this parameters that we have decided we will be calling the function of create the cities and also making the ACO on the list of cities created, leaving us the best path available and also the length of this path
8. To finish with the exercise and also to provide all the correct information regarding to the best path that we find we made the process with Matplotlib that is a library for python, that will allow us to create and generate an image regarding to the path that we already find, into a 3D space, showing us the principal city that we've used and also the path needed in order to complete all the cities on the list (example on the following image)



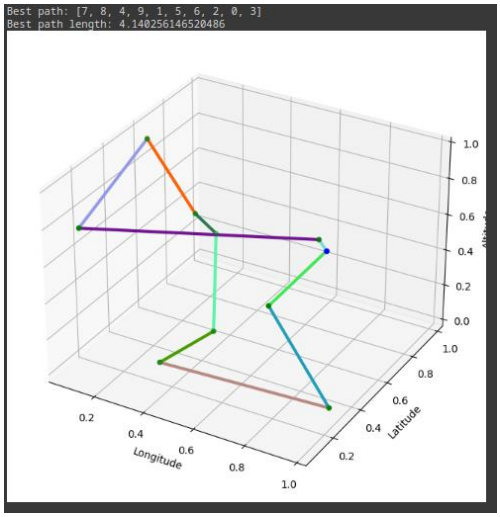
Example of the graphic created with Matplotlib

Parameters Combinations

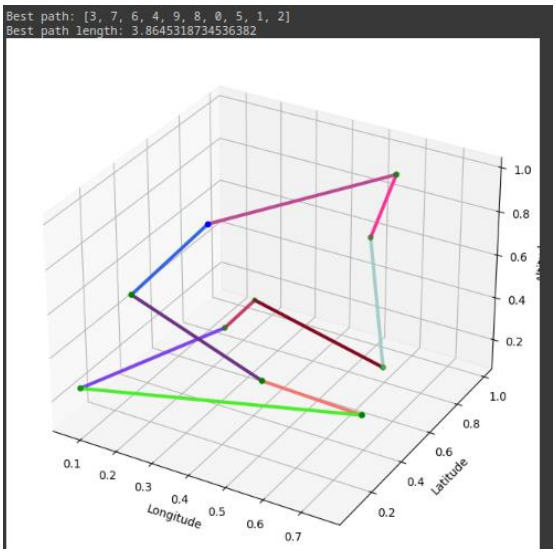
As we already said on the explanation of the steps, we define the different parameters available into the program, and we can find different answers and different conclusions base on which parameter we change, the following examples are different use of parameters for different number of cities.

- **10 Cities:**

For the parameter of 10 cities, we've made the exercise of change the number of ants and the number of interactions that we will be using to find the best path, starting with 100 Ants and 100 interactions, and getting the following answer after 9 seconds of running the program



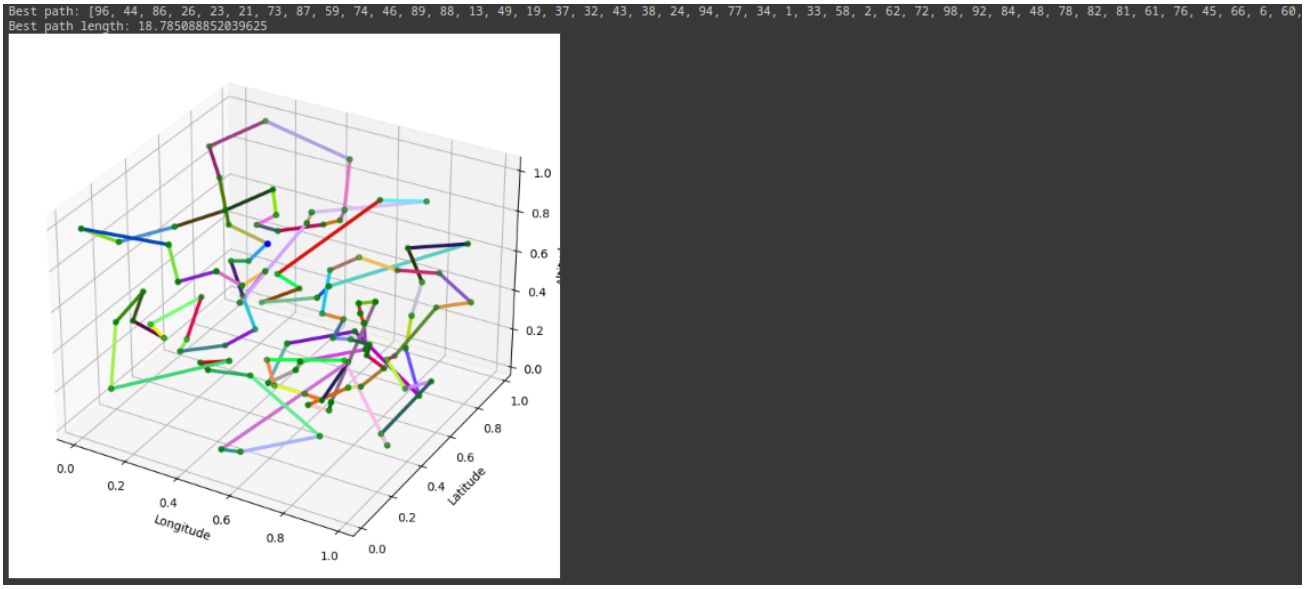
If we increase the number of Ants to 300 the time that the program will take to run will be increased to 16 seconds, but it's supposed that it will find the path into a more exactly way than with a shorter number of Ants, this also happen if we increase the number of interactions that we made, that if we increase it to up to 500 interactions will make that the program take 26 seconds to be completed, in this case and also with this try, we can check that the changes on the amount of ants and interactions will be proportional to the time that the program will take to run



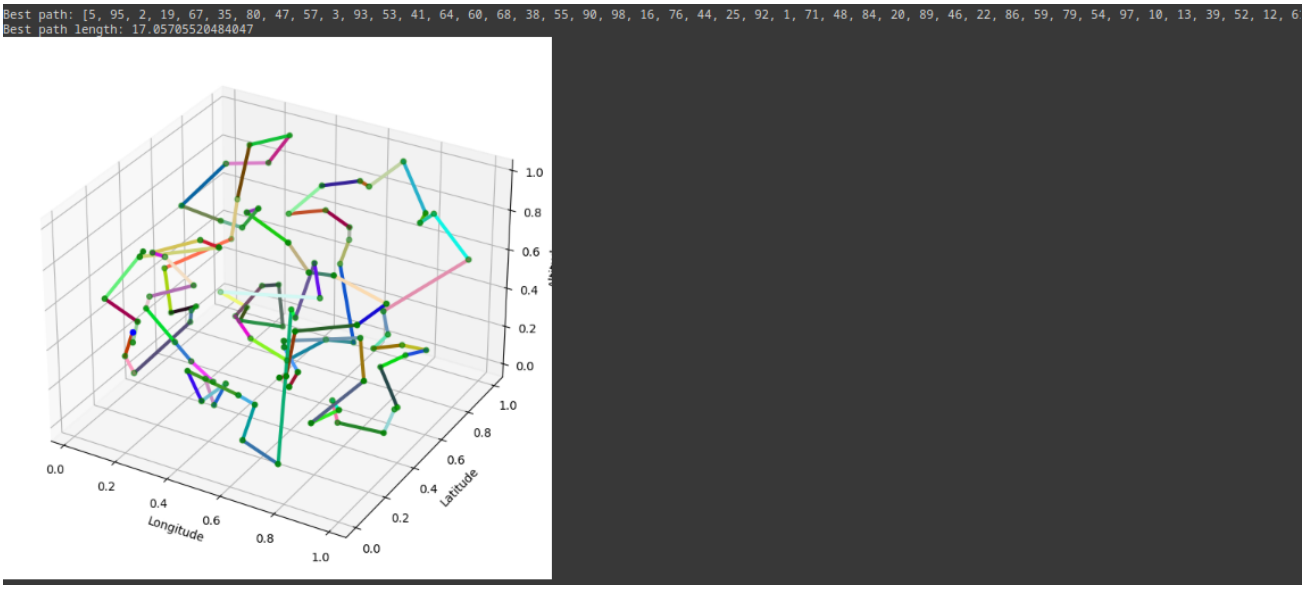
We need to clarify something while we're making this study, as we already know the algorithm balances exploration (following pheromone trails) and distance (choosing shorter paths). When we start making changes on the number of ants and iterations, we can control the trade-off between exploring diverse paths and converging toward optimal solutions, but these changes will make that the time and resourses that we use for the experiment will increase a lot

- **99 Cities:**

I also have made the try of check all the information but with a total amount of 99 cities, this to check the time that the program can take to make this and check the different paths that we can find with this number of cities, normally starting the exercise with the parameters of 100 Ants and 100 cities, and getting this results into a normal time of 9.25 minutes, a normal time in base of all the amount of paths that the ants find during all the travel, and also in base of the cities that we're traveling.



After this I wanted to review how much time it can take if we change the parameters of the number of ants and the iterations, I selected a number of 500 ants and the time that the program takes to run were exactly a total of 24minutes with 30 seconds, tiis compared to the time that i got when I've made the change of the iterations to 500 with 100 ants, this other experiment have taken a time of 38 minutes with 40 seconds to be completed, Also to be honest it caused the device where the test was being carried out to overheat a little and slow down as well.



Observations, difficulties and conclusions

- To start and to be sincerely at first, I was thinking that all this process will take a lot of hours to be fully completed and that it will be necessary a lot of knowledge regarding to manage the NumPy library on

the python language, but after making the process and get results of the code that I've created I can say that it’s easier as it appears with a naked eye

- Regarding to the difficulties that I got were the process of read some documentation about NumPy in order to find the best options that I could use to fill the blank spaces of code, as the creation of matrix started on 1 or also the process of find the Euclidean distance between the cities coordinates, for this I've used all the options that i got as checked on different webpages and forums to find the best code that I could implement on the program
- Another thing that were difficult were made the process of solve the different errors that can appear on the system with each change, for example I have used 3 different ways to find the Euclidean distance, and only the 3rd one worked fine, with the others I were just having issues on the program, other example were checking the difference between the tuple array that the function of generate_cities were returning and the NumPy arrays that can be used on the ACO, because I were getting an error that some operations cannot be made between tuple and tuple types
- Something that were interesting for me were trying to understand the code of Matplotlib in order to check how the graphics were made, this was the first contact that i have with this type of software and to be sincerely it interest me to continue making use of it and check which other things we can try to get with it
- To conclude regarding to all of this, all of this workshop shows me the different thigs that I can reach into this language of python and show me how does the ACO and the TCP worked