Project 3

Cap 4630

Dr. Marques

Andrew Donate

Traveling Salesman Problem: Baseline Solution using Ant Colony Optimization

Prologue

The following document describes of Project 3 for the Intro to Artificial Intelligence course (CAP 4630). This project was done entirely by Andrew Donate as the “architect”, “developer”, and “reporter”. This project does take inspiration from another project but is sourced down below in the references section.

Introduction

In this Python program, the objective was to develop an AI solution using ACO (Ant Colony Optimization) to efficiently solve the TSP (Traveling Salesman Problem). By simulating the behavior of ants and the pheromone trails they create, the program aims to iteratively construct and refine tours until an optimal or near-optimal route is obtained. The ants select nodes based on certain criteria and deposit pheromone on the edges, which influences the decision-making process of subsequent ants. Through the process of pheromone updates, the ACO algorithm explores the solution space and converges towards an optimal solution to the TSP.

Functions

1. __init__ (Edge Class): represents a connection between two cities in the TSP. It stores the indices of the cities, the weight (distance) of the edge, and the amount of pheromone on the edge.
2. __init__(Ant Class): represents an ant agent in the ACO algorithm. It stores the parameters alpha and beta, which control the importance of pheromone and heuristic information in the ant's decision-making process. It also keeps track of the number of nodes, the edges connecting them, the ant's tour, and the distance traveled.
3. _select_node: used by an "ant" to choose the next node to visit in the TSP. It calculates a roulette wheel selection based on the pheromone levels and heuristic information of the unvisited nodes. The alpha and beta parameters influence the probabilities of node selection, favoring higher pheromone levels and lower distances.
4. find_tour: used by an "ant" to construct a tour by iteratively selecting the next node based on the _select_node function until all nodes are visited. It returns the completed tour.
5. get_distance: calculates the total distance traveled by an ant in its tour by summing the weights (distances) of the edges between consecutive nodes. It returns the calculated distance.
6. __init__(SolveTSPUsingACO Class): sets up various parameters and data structures, including the generation_distances list to store the best distance at each generation, the colony size, the scaling factor for pheromone, the evaporation rate, the weight of pheromone deposit, the number of steps (generations), the number of nodes, the nodes' coordinates, and the labels for the nodes. It also creates the edges between nodes, initializes the ants with the specified parameters, and sets initial values for the global best tour, global best distance, and initial tour.
7. _add_pheromone: "deposits" pheromones on the edges of the tour based on the distance traveled by the "ant". It calculates the amount of pheromone to add and updates the pheromone levels accordingly.

8. _max_min: implements the Max-Min Ant System (MMAS) strategy for ACO. It performs iterations for the specified number of steps and updates the pheromone trails accordingly. It tracks the best tour found in each iteration and updates the global best tour and distance if necessary. The function also maintains a record of the best distance at each generation and plots the best tour, the generational change in distance, and the initial path.
9. plot: generates a visual representation of the ACO algorithm's results. It creates a figure with three subplots, each displaying different information such as the best tour found, generational change in distance, and the initial path. The function utilizes Matplotlib to plot the tour, distances, and paths with customizable line widths, point radii, and annotation sizes.
10. userSettings: prompts the user to enter parameters for the ACO algorithm, such as the number of cities, population size, and number of generations. It then creates an instance of the SolveTSPUsingACO class with the specified parameters, runs the ACO algorithm using the _max_min method, and plots the results using the plot method. It also includes input validation functions to ensure the user provides valid numerical inputs.
11. checkUserInputInt: takes input from user and ensures the value is an integer, if not returns false.
12. checkUserInputFloat: takes input from user and ensures the value is a float, if not returns false.
13. __name__ (Main): serves as the entry point of the program.

Initial/Revised Implementation Issues

Originally, I was trying to only convert the Genetic Algorithm functions in my previous project thinking that all I needed was to change what algorithm to use and I would have no further issues. Halfway through development I reached a roadblock as most of the code added onto project two because more cluttered and unreadable. Due to this issue, I had decided to start from scratch and only imported parts of my previous project code when needed instead of directly modifying a copy of the program code. Other issues the began to arise were my implementation of a graphing system, my previous code began to show more of its flaws as I began testing and the amount of processing time became atrocious. So I decided to reuse smaller chunks of the previous plotting function and rewrote the function containing only a handful of lines from it.

Limitations and Future Improvements

Like the previous project, due to time constraints I was unable to optimize the program to the fullest, there are still some slow downs which could have been addressed and potentially some bugs I was not able to find. In the future once the class ends and I have more time to work on personal projects, I would like to redo the projects I have taken in this class and to improve upon them.

Questions

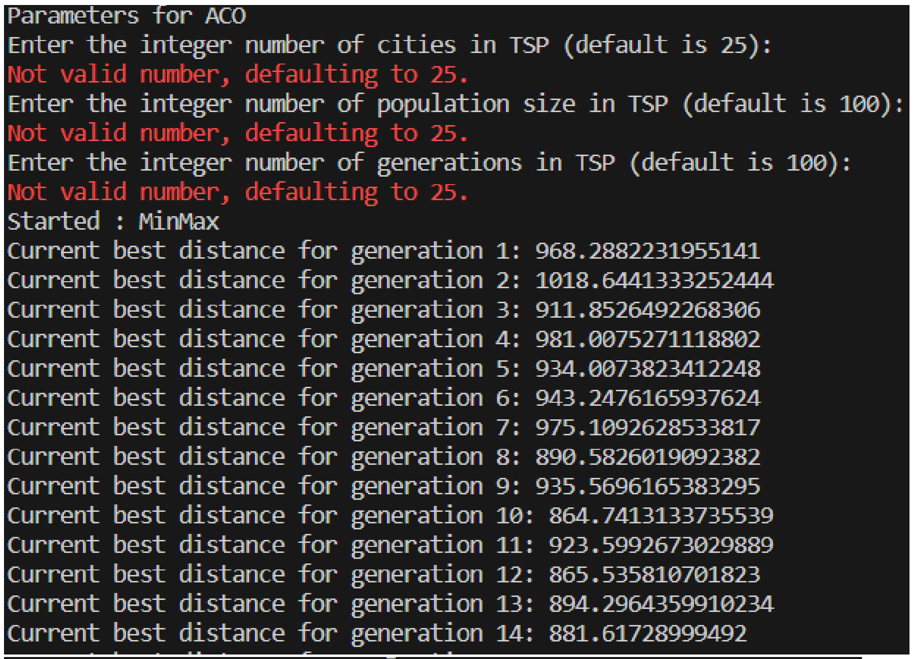
1. How were the cities and distances represented (as a data structure)?
    a. The cities and distances are represented using a class called "edges." The "Edge" class represents a connection between two cities. It contains attributes such as the city indices (a and b), the distance between them, and the amount of pheromone on that edge.
2. How did you encode the solution space?
    a. The solution space is encoded using a list of ants. Each “ant” represents a potential solution to the traveling salesman problem (TSP). The ants construct tours by selecting nodes based on certain criteria, and the quality of the solution is evaluated based on the distance traveled.
3. How did you handle the creation of the initial ant population?
    a. The creation of the initial ant population is handled in the constructor of the SolveTSPUsingACO class. The specified number of ants is created, each with its own alpha and beta parameters.
4. How did you handle the updating of the pheromone trails?
    a. The updating of the pheromone trails is done in the "_add_pheromone" method. After each iteration, the pheromone on the edges traversed by the best ant tour

and the global best tour is increased by a certain amount. The pheromone on other edges is reduced by a factor (rho).

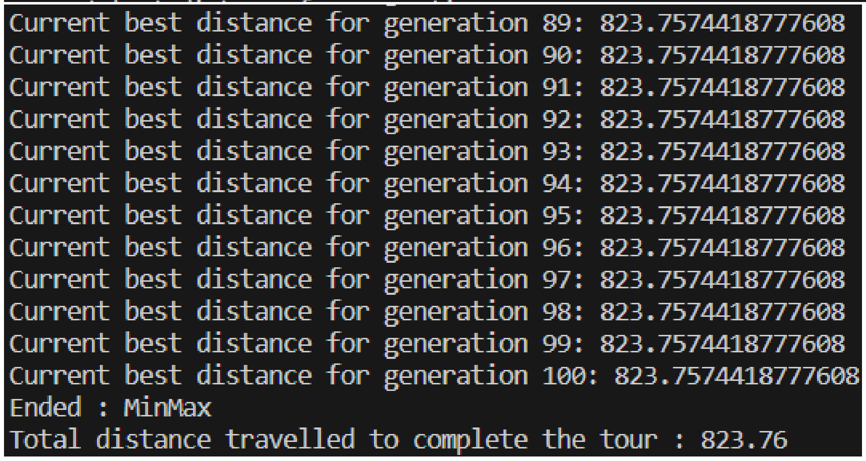
5. Which strategy(ies) did you use to compute the best solution?
    a. The Max-Min Ant System (MMAS) was used to compute the best solution. During the first approximate 75% of the iterations, the pheromone is updated based on the best tour found so far. After that, the pheromone is updated based on the global best tour.
6. Which stopping condition did you use? Why?
    a. The stopping condition is based on the number of iterations (steps) specified by the user input. The algorithm iterates for the given number of steps, updating the pheromone trails and evaluating the best tour at each step. It was chosen as to mimic the previous project, giving the user more input on how to control the outcome.
7. What other parameters, design choices, initialization and configuration steps are relevant to your design and implementation?
    a. Other relevant parameters, design choices, initialization, and configuration steps in the implementation include:
    - Alpha and beta parameters
    - Rho
    - Pheromone deposit weight
    - Initial pheromone
    - Min scaling factor
    - Plotting of initial, generational, and final outcomes.
8. Which (simple) experiments have you run to observe the impact of different design decisions and parameter values? Post their results and your comments.
    a. The experiments done were based on some initial inputs from project 2, and others were just the default inputs multiplied by an arbitrary amount.

Demos

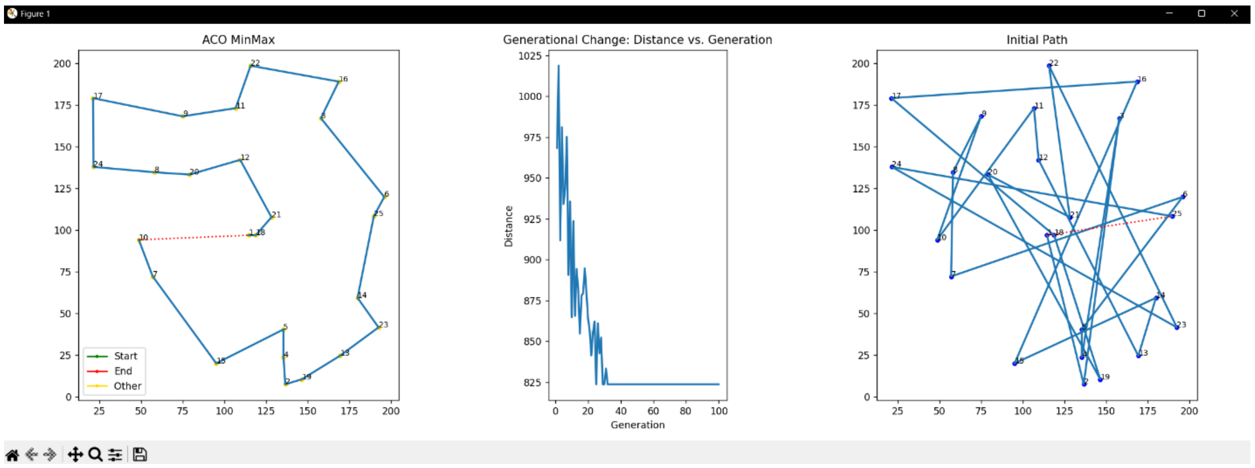
Demo1:



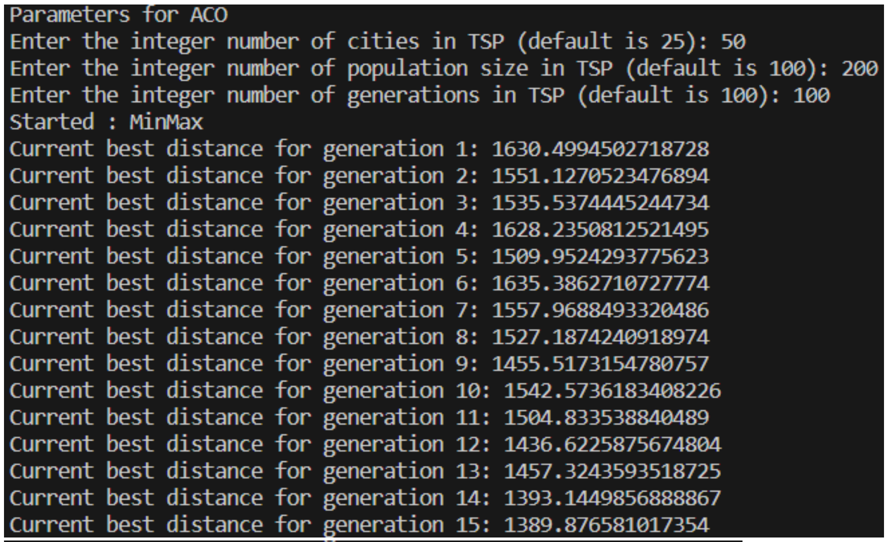
```
Parameters for ACO
Enter the integer number of cities in TSP (default is 25):
Not valid number, defaulting to 25.
Enter the integer number of population size in TSP (default is 100):
Not valid number, defaulting to 25.
Enter the integer number of generations in TSP (default is 100):
Not valid number, defaulting to 25.
Started : MinMax
Current best distance for generation 1: 968.2882231955141
Current best distance for generation 2: 1018.6441333252444
Current best distance for generation 3: 911.8526492268306
Current best distance for generation 4: 981.0075271118802
Current best distance for generation 5: 934.0073823412248
Current best distance for generation 6: 943.2476165937624
Current best distance for generation 7: 975.1092628533817
Current best distance for generation 8: 890.5826019092382
Current best distance for generation 9: 935.5696165383295
Current best distance for generation 10: 864.7413133735539
Current best distance for generation 11: 923.5992673029889
Current best distance for generation 12: 865.535810701823
Current best distance for generation 13: 894.2964359910234
Current best distance for generation 14: 881.61728999492
```



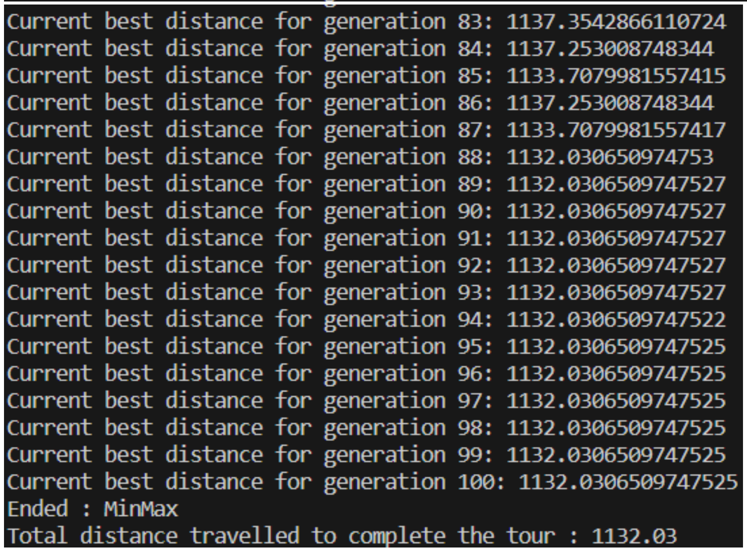
```
Current best distance for generation 89: 823.7574418777608
Current best distance for generation 90: 823.7574418777608
Current best distance for generation 91: 823.7574418777608
Current best distance for generation 92: 823.7574418777608
Current best distance for generation 93: 823.7574418777608
Current best distance for generation 94: 823.7574418777608
Current best distance for generation 95: 823.7574418777608
Current best distance for generation 96: 823.7574418777608
Current best distance for generation 97: 823.7574418777608
Current best distance for generation 98: 823.7574418777608
Current best distance for generation 99: 823.7574418777608
Current best distance for generation 100: 823.7574418777608
Ended : MinMax
Total distance travelled to complete the tour : 823.76
```



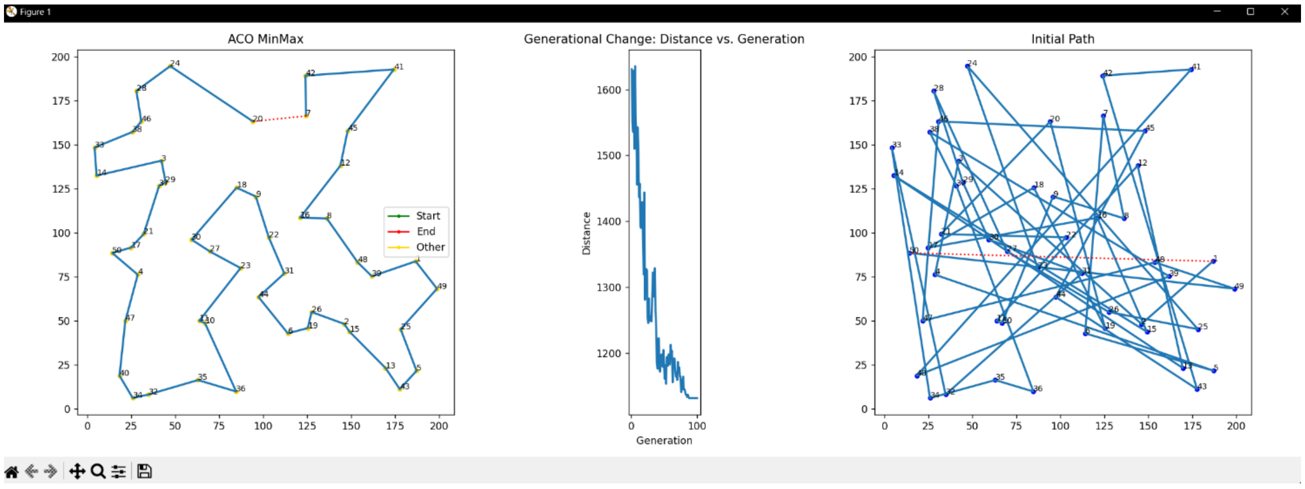
Demo2:



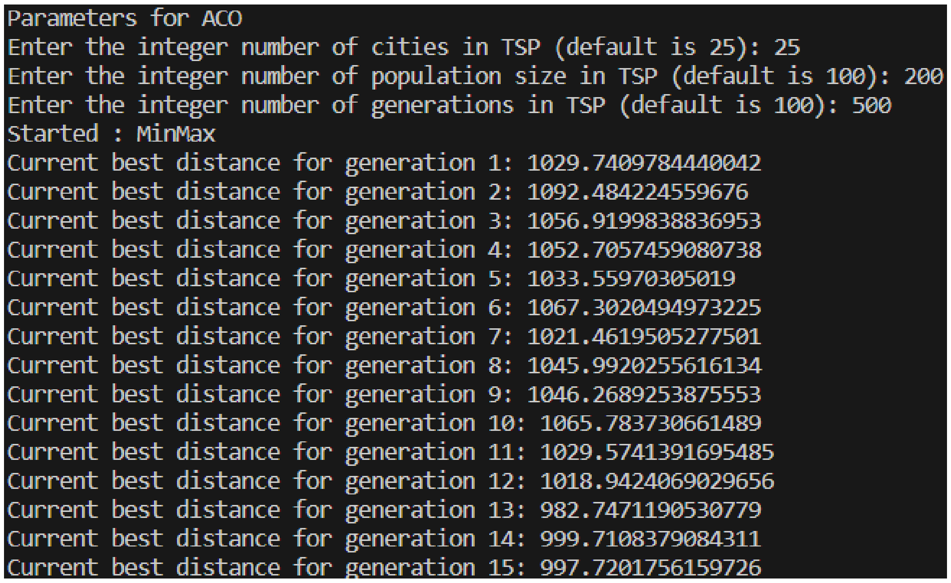
```
Parameters for ACO
Enter the integer number of cities in TSP (default is 25): 50
Enter the integer number of population size in TSP (default is 100): 200
Enter the integer number of generations in TSP (default is 100): 100
Started : MinMax
Current best distance for generation 1: 1630.4994502718728
Current best distance for generation 2: 1551.1270523476894
Current best distance for generation 3: 1535.5374445244734
Current best distance for generation 4: 1628.2350812521495
Current best distance for generation 5: 1509.9524293775623
Current best distance for generation 6: 1635.3862710727774
Current best distance for generation 7: 1557.9688493320486
Current best distance for generation 8: 1527.1874240918974
Current best distance for generation 9: 1455.5173154780757
Current best distance for generation 10: 1542.5736183408226
Current best distance for generation 11: 1504.833538840489
Current best distance for generation 12: 1436.6225875674804
Current best distance for generation 13: 1457.3243593518725
Current best distance for generation 14: 1393.1449856888867
Current best distance for generation 15: 1389.876581017354
```



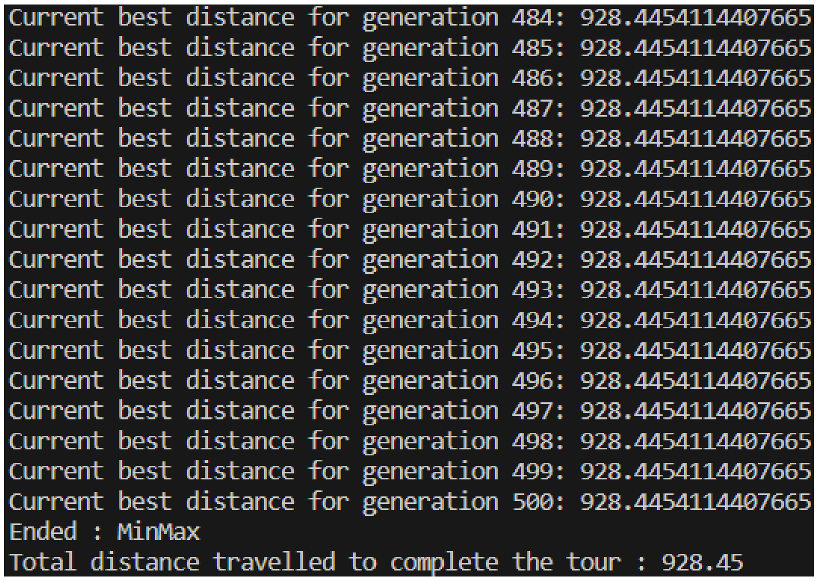
```
Current best distance for generation 83: 1137.3542866110724
Current best distance for generation 84: 1137.253008748344
Current best distance for generation 85: 1133.7079981557415
Current best distance for generation 86: 1137.253008748344
Current best distance for generation 87: 1133.7079981557417
Current best distance for generation 88: 1132.030650974753
Current best distance for generation 89: 1132.0306509747527
Current best distance for generation 90: 1132.0306509747527
Current best distance for generation 91: 1132.0306509747527
Current best distance for generation 92: 1132.0306509747527
Current best distance for generation 93: 1132.0306509747527
Current best distance for generation 94: 1132.0306509747522
Current best distance for generation 95: 1132.0306509747525
Current best distance for generation 96: 1132.0306509747525
Current best distance for generation 97: 1132.0306509747525
Current best distance for generation 98: 1132.0306509747525
Current best distance for generation 99: 1132.0306509747525
Current best distance for generation 100: 1132.0306509747525
Ended : MinMax
Total distance travelled to complete the tour : 1132.03
```



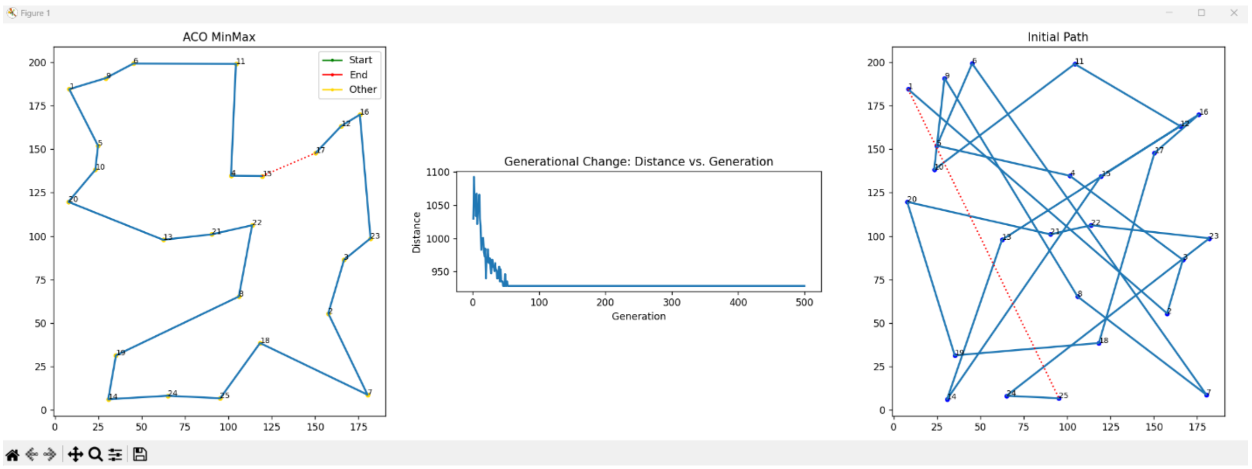
Demo3:



```
Parameters for ACO
Enter the integer number of cities in TSP (default is 25): 25
Enter the integer number of population size in TSP (default is 100): 200
Enter the integer number of generations in TSP (default is 100): 500
Started : MinMax
Current best distance for generation 1: 1029.7409784440042
Current best distance for generation 2: 1092.484224559676
Current best distance for generation 3: 1056.9199838836953
Current best distance for generation 4: 1052.7057459080738
Current best distance for generation 5: 1033.55970305019
Current best distance for generation 6: 1067.3020494973225
Current best distance for generation 7: 1021.4619505277501
Current best distance for generation 8: 1045.9920255616134
Current best distance for generation 9: 1046.2689253875553
Current best distance for generation 10: 1065.783730661489
Current best distance for generation 11: 1029.5741391695485
Current best distance for generation 12: 1018.9424069029656
Current best distance for generation 13: 982.7471190530779
Current best distance for generation 14: 999.7108379084311
Current best distance for generation 15: 997.7201756159726
```



```
Current best distance for generation 484: 928.4454114407665
Current best distance for generation 485: 928.4454114407665
Current best distance for generation 486: 928.4454114407665
Current best distance for generation 487: 928.4454114407665
Current best distance for generation 488: 928.4454114407665
Current best distance for generation 489: 928.4454114407665
Current best distance for generation 490: 928.4454114407665
Current best distance for generation 491: 928.4454114407665
Current best distance for generation 492: 928.4454114407665
Current best distance for generation 493: 928.4454114407665
Current best distance for generation 494: 928.4454114407665
Current best distance for generation 495: 928.4454114407665
Current best distance for generation 496: 928.4454114407665
Current best distance for generation 497: 928.4454114407665
Current best distance for generation 498: 928.4454114407665
Current best distance for generation 499: 928.4454114407665
Current best distance for generation 500: 928.4454114407665
Ended : MinMax
Total distance travelled to complete the tour : 928.45
```



References

T. Stutzle and H. Hoos, "MAX-MIN Ant System and local search for the traveling salesman problem," Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97), Indianapolis, IN, USA, 1997, pp. 309-314, doi: 10.1109/ICEC.1997.592327.

Rodrigues, N. (2019, April 4). *Traveling Salesman Problem using Ant Colony Optimization*. GitHub. https://github.com/nishnash54/TSP_ACO/blob/master/aco_tsp.py