Bogazici University

Introduction to Object Oriented Programming

CMPE160

Assignment 3

Author:
Ibrahim Berkay Ceylan
Student ID: 2023400327

May 10, 2024

Contents

1 Introduction

In this assignment, we tackle a hypothetical scenario for a delivery system for Migros using a Java program that addresses the Traveling Salesman Problem. This classical problem involves finding the shortest possible route that a delivery vehicle must take to visit each designated location exactly once and return to the starting point, which in our case is a Migros store. The assignment assesses two computational approaches to solve this problem, providing insights into algorithmic strategies and their application in theoretical contexts.

The first method we implement is the Brute-Force approach. This technique exhaustively searches all permutations of routes to ensure the discovery of the optimal path. Although computationally demanding and not feasible for large datasets, this method serves as a crucial benchmark for evaluating the efficiency of more sophisticated algorithms.

The second method we examine is the Ant Colony Optimization method. Inspired by the natural behavior of ants in finding paths to food sources, this method simulates ants moving between nodes, leaving pheromone trails and the evaporation of pheromones. These trails help other ants find short and effective routes, making it a suitable algorithm for finding near-optimal solutions more efficiently than the brute-force approach.

2 Methodology

This section details the methods implemented to tackle the Traveling Salesman Problem in our Java program. The challenge was approached using two distinct strategies: the Brute-Force method and the Ant Colony Optimization method. Both methods were developed and tested within controlled datasets and conditions to evaluate their performance in identifying the shortest route for a delivery scenario involving multiple stops. This comparison helps in understanding the trade-offs between computational complexity and the quality of solutions, essential for algorithm selection in real-world applications.

2.1 Brute-Force Approach

The Brute-Force method is a straightforward, yet computationally expensive approach that involves calculating the distance for every possible permutation of the route order to find the shortest possible path that visits each location once and

returns to the starting point. This method, while guaranteeing finding the optimal solution, scales factorially with the number of stops, making it impractical for scenarios with a large number of destinations. Despite its simplicity, it provides a critical benchmark for evaluating the efficiency of more advanced algorithms. In our implementation, the program generates all permutations of the destinations and calculates the total distance for comparison with results from the Ant Colony Optimization method.

2.2 Ant Colony Optimization Method

The Ant Colony Optimization method is a copy of the behavior of ants searching for food, where they leave pheromone trails on their paths that guide other ants to efficient routes. In the context of the Travelling Salesman Problem, the Ant Colony Optimization algorithm simulates a group of ants that move randomly between cities, favoring routes with higher pheromone concentrations which correlate with shorter paths and distance between cities. Pheromones are updated based on the distance of the route traveled by each ant, allowing the algorithm to improve its route choices iteratively. This method's performance hinges on several parameters: the number of ants (M), the number of iterations (N), the pheromone evaporation coefficient, and the influence levels of pheromone trails and heuristic information (α and β).

3 Results and Analysis

This section presents the results gotten from the implementation of the Brute-Force and Ant Colony Optimization methods, applied to solve the Traveling Salesman Problem in our Migros delivery scenario. The results are evaluated based on the route efficiency (shortest path) and computational performance (execution time). Each method's findings are visualized through graphical outputs and summarized in comparative tables to provide a clear understanding of their respective efficiencies.

3.1 Console Outputs and Graphical Representations

This subsection presents the console outputs and the graphical representations generated by both the Brute-Force and Ant Colony Optimization methods. The data include route details such as the shortest distance achieved and the computational time required for each method. The graphical outputs visually depict the routes, visualizing the practical results of each algorithm.

3.1.1 Brute-Force Method Outputs

The Brute-Force method produced the following 4 console outputs for the inputs respectively, detailing the shortest route and the time taken to compute it. Input 5 is not computationally feasible using this method.

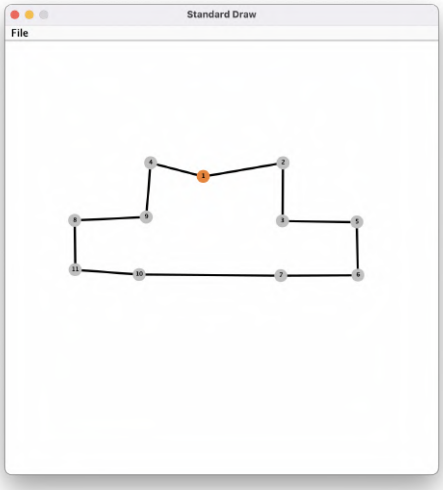
```
Method: Brute-Force Method
Shortest distance: 1.7952913856772432
Shortest Path: [1, 4, 9, 8, 11, 10, 7, 6, 5, 3, 2, 1]
Time it takes to find the shortest path: 0.266 seconds
```

```
Method: Brute-Force Method
Shortest distance: 2.935877143237598
Shortest Path: [1, 8, 7, 11, 10, 9, 6, 5, 4, 12, 3, 2, 1]
Time it takes to find the shortest path: 3.428 seconds
```

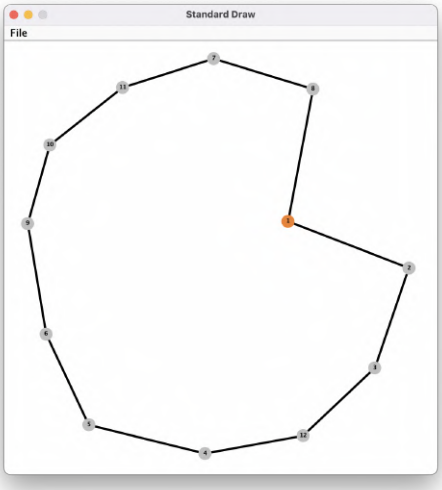
```
Method: Brute-Force Method
Shortest distance: 3.802919361826042
Shortest Path: [1, 2, 4, 3, 5, 9, 8, 13, 12, 11, 10, 7, 6, 1]
Time it takes to find the shortest path: 34.444 seconds
```

```
Method: Brute-Force Method
Shortest distance: 3.710908906673479
Shortest Path: [1, 2, 3, 13, 10, 14, 11, 9, 5, 6, 4, 8, 7, 12, 1]
Time it takes to find the shortest path: 454.219 seconds
```

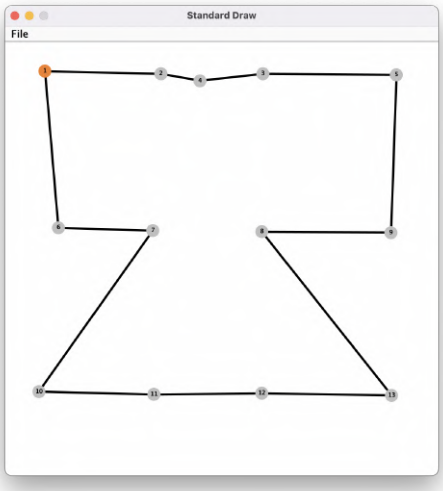
Figures below show the optimal route found by the Brute-Force method. Routes are illustrated on a map generated using the StdDraw library.



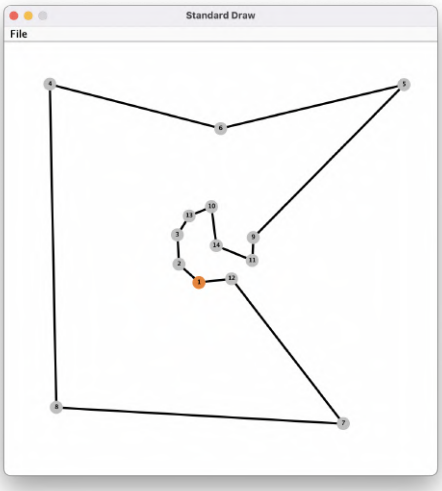
(a) Output of Input1



(b) Output of Input2



(c) Output of Input3



(d) Output of Input4

Figure 1: Brute-force graphical outputs for inputs.

3.1.2 Ant Colony Optimization Method Outputs

The Ant Colony Optimization method's console outputs include the shortest route found, the distance of this route, and the computational time with parameters Number of Iterations (N) = 400, Number of Ants (M) = 75, Degradation Factor = 0.9, $\alpha = 0.8$, $\beta = 1.5$, Initial Pheromone Intensity = 0.1, $Q = 0.0001$ are shown below, lines with the highest thickness indicate the optimal route found by the method while the less thick lines indicate pheromones present on those routes during the iteration of the best route.

```
Method: Ant Colony Optimization Method
Shortest distance: 1.795291385677243
Shortest Path: [1, 4, 9, 8, 11, 10, 7, 6, 5, 3, 2, 1]
Time it takes to find the shortest path: 0.423 seconds
```

```
Method: Ant Colony Optimization Method
Shortest distance: 2.9358771432375974
Shortest Path: [1, 2, 3, 12, 4, 5, 6, 9, 10, 11, 7, 8, 1]
Time it takes to find the shortest path: 0.499 seconds
```

```
Method: Ant Colony Optimization Method
Shortest distance: 3.802919361826042
Shortest Path: [1, 6, 7, 10, 11, 12, 13, 8, 9, 5, 3, 4, 2, 1]
Time it takes to find the shortest path: 0.581 seconds
```

```
Method: Ant Colony Optimization Method
Shortest distance: 3.710908906673479
Shortest Path: [1, 2, 3, 13, 10, 14, 11, 9, 5, 6, 4, 8, 7, 12, 1]
Time it takes to find the shortest path: 0.628 seconds
```

```
Method: Ant Colony Optimization Method
Shortest distance: 4.8393098410382915
Shortest Path: [1, 27, 17, 9, 10, 11, 6, 7, 28, 29, 3, 2, 4, 30, 5,
13, 8, 12, 14, 19, 20, 18, 21, 24, 26, 25, 15, 16, 23, 22, 1]
Time it takes to find the shortest path: 2.762 seconds
```

Shortest path for the Ant Colony Optimization is the same with the Brute-Force method for inputs 1 through 4. An example of a route for input 5 is shown below.

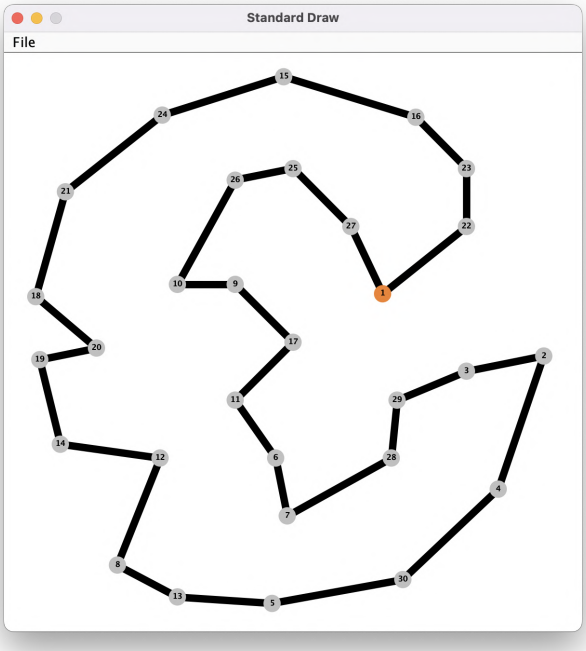
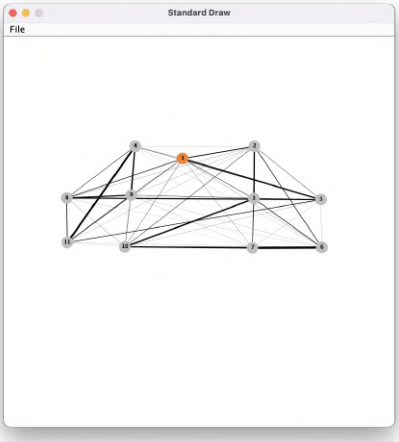
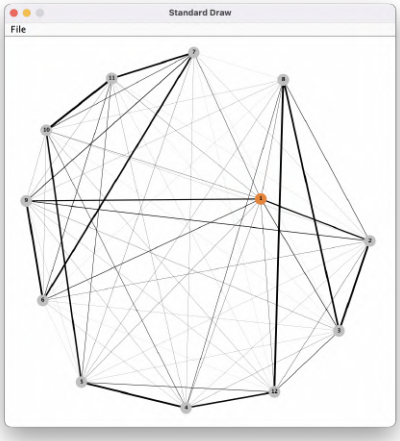


Figure 2: Example of a Ant Colony Optimization shortest path result for input5.

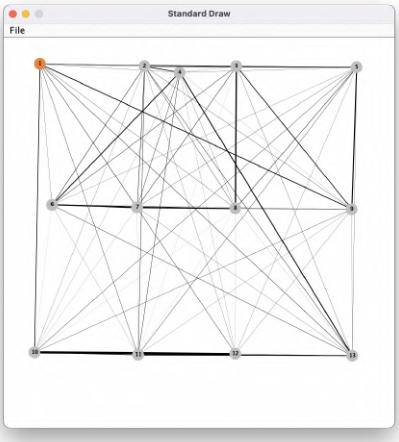
Figures in the next page illustrate the pheromone intensity on each path segment, reflecting the frequency and efficiency of each route used by the ants.



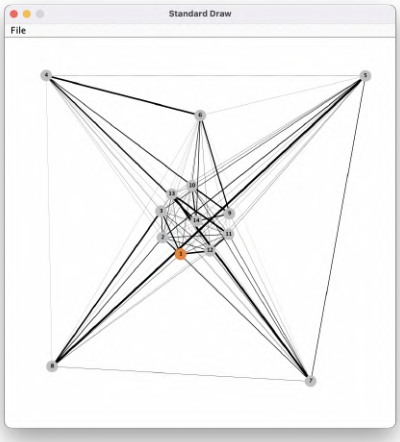
(a) Output of Input1



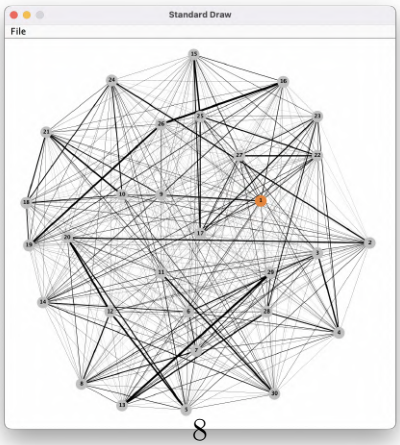
(b) Output of Input2



(c) Output of Input3



(d) Output of Input4



(e) Output of Input4

Figure 3: Pheromone intensities after 40 iterations with 50 ants for all inputs. Thicker lines indicate higher pheromone levels and intensity levels are min-max normalized.

3.2 Performance Comparison

This subsection provides a comparison of the Brute-Force and Ant Colony Optimization methods based on their performance. The key indicators compared are the computational time and the shortest path distances found by each method. The speed-up factor, which shows the relative performance of the Ant Colony Optimization method over the Brute-Force method, is also shown. This comparison helps in understanding the efficiency gains achieved through the heuristic approach of the Ant Colony Optimization method against the exhaustive search strategy of the Brute-Force method.

Table 1: Comparison of methods. Values are arbitrary and may not be correct.

Input File	Number of Houses + Migros	Brute-Force Time (seconds) (Distance: km)	Ant Colony Time (seconds) (Distance: km)	Speed Up Factor
Input1	11	0.266 (1.7952)	0.423 (1.7952)	0.628 times faster
Input2	12	3.428 (2.9358)	0.499 (2.9358)	6.870 times faster
Input3	13	34.444 (3.8029)	0.581 (3.8029)	59.283 times faster
Input4	14	454.219 (3.7109)	0.628 (3.7109)	723.279 times faster
Input5	30	-infeasible- (?)	2.762 (4.839)	∞ times faster

3.3 Ant Colony Optimization Performance Over Iterations

The performance of the Ant Colony Optimization method was further analyzed by observing how the shortest path distance varied across iterations. This analysis was visualized through a graph plotting the shortest distance against the iteration number, showing a clear trend of distance reduction as the number of iterations increased.

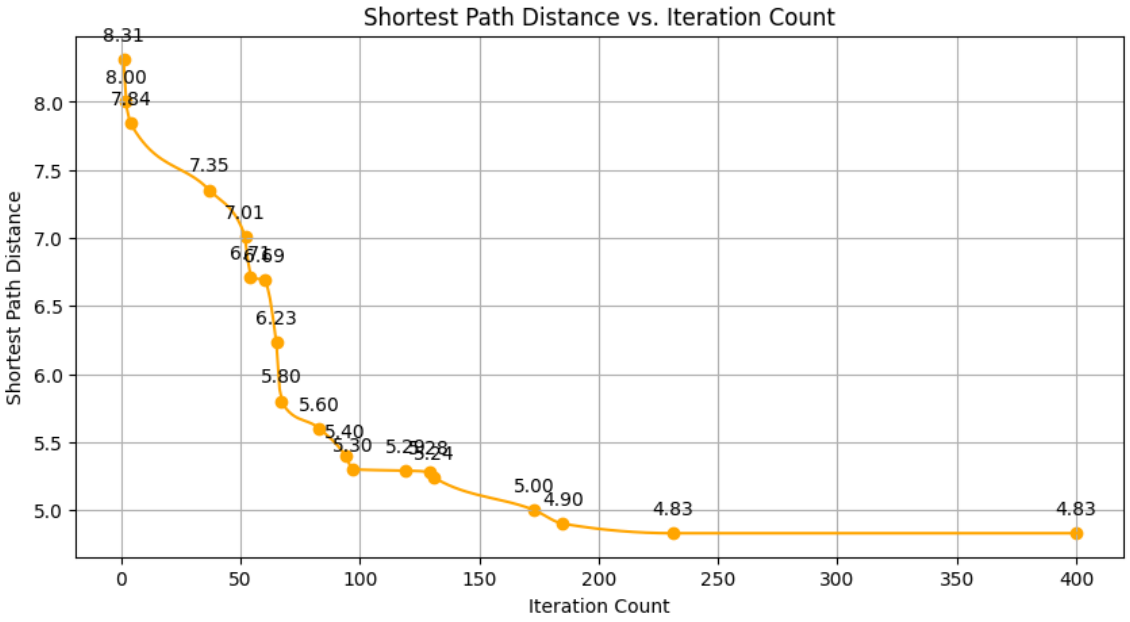


Figure 4: Shortest path distance found by the Ant Colony Optimization method over iterations.

4 Advantages and Disadvantages of Each Method

4.1 Brute-Force Method

Advantages:

- **Simplicity:** The Brute-Force method is straightforward to understand and implement. It does not require hard algorithms or understanding of advanced concepts.
- **Guaranteed Optimal Solution:** This method guarantees finding the shortest possible route, as it checks all possible permutations of routes.

Disadvantages:

- **Scalability:** The Brute-Force method is computationally expensive and scales terribly with the number of destinations. The time complexity grows factorially with the addition of more nodes, making it impractical for large datasets.
- **Resource Intensive:** Due to its exhaustive nature, the method requires significant computational resources, which is a major disadvantage in applications with limited processing power.

4.2 Ant Colony Optimization Method

Advantages:

- **Efficiency:** Ant Colony Optimization is more efficient than Brute-Force for larger problems. It significantly reduces computation time by using a probabilistic approach to find a near-optimal solution fast.

Disadvantages:

- **No Guarantee of Optimal Solution:** Unlike the Brute-Force method, Ant Colony Optimization method does not guarantee that the found solution is an optimal solution, but usually gets very close to the optimal solution, suitable for most practical purposes. The solution depends on the parameters set, and luck.
- **Complexity in Implementation:** Setting up an Ant Colony Optimization method to solve a problem requires a deeper understanding of the problem and the algorithm than the Brute-Force method, which increases the complexity of implementation, and development time.