INFO 6205 Spring 2023 Project
Project-Traveling Salesman

Submitted by:

Name: Mulpuri Uday Kiran Reddy **NUID:**002781063
Nandre Anurag **NUID**:002785735

Aim:

The aim of the project is to solve the Travelling Salesman Problem, which is an NP-hard problem, using the Christofides algorithm and optimizing further with tactical and strategic optimizations

Approach:

- The first step of the project involved processing the given dataset (Data\tspdatapoints.xlsx) into the necessary format (points.xlsx). Subsequently, we computed the distance between each vertex in terms of edges and recorded the results in a separate file (routes.xlsx).
- Next, we utilized Kruskal's Union Find Method to identify the minimum spanning tree (MST) of the graph. This was achieved by implementing the "minimumSpanningTree()" function within the MinWeightService Class.
- After obtaining the MST, we proceeded to extract the list of odd vertices from the MST Tour.
- The next step involved identifying the perfect matching for the list of odd. This was accomplished by utilizing the method "findMinimumWeightMatching()" present within the MinWeightServiceClass.
- Subsequently, we combined the Perfect Matching and MST to ensure that every vertex in the graph had an even number of edges. This allowed us to generate an Eulerian tour of the graph.
- Using the Eulerian tour obtained in the previous step, we aimed to create a TSP tour that visits each vertex in the graph and returns to the starting vertex. It was necessary to skip over any vertex that had already been encountered during the tour.
- In parallel with the previous step, we calculated the distance/weight of the TSP tour.
- Next, we performed two tactical optimizations, namely the 2-opt and 3-opt techniques, on the TSP tour obtained in the previous step.
- Furthermore, we applied two strategic optimizations, simulated annealing and ant colony optimization, to further enhance the TSP tour.

Algorithm:

Kruskal's Algorithm: The Minimum Spanning Tree (MST) algorithm is a commonly used approach to approximate a solution for the Traveling Salesman Problem (TSP). The MST algorithm begins by constructing a complete graph of the cities, and then computes the MST of the graph using Prim's or Kruskal's algorithm. The resulting MST is then traversed to obtain a sequence of cities, which is then converted into a Hamiltonian circuit by adding the starting city to the end. While the MST algorithm does not guarantee an optimal solution for the TSP, it is often used in practice because it can provide a good approximation. By using the MST to connect the cities with the minimum possible total weight, we can ensure that the resulting Hamiltonian circuit is relatively efficient and avoids unnecessary detours. Overall, the MST algorithm can be a useful tool for finding approximate solutions to the TSP in a variety of applications.

Tactical Optimizations:

2 opt:

The 2-opt algorithm is a heuristic optimization algorithm for the Traveling Salesman Problem (TSP) that improves an initial TSP tour generated from the Christofides algorithm. The algorithm starts by choosing two edges i and j in the tour such that the vertices do not share a common edge. It then generates a new tour by adding the elements from index 0 to i, followed by the elements of the original tour from j to i in reverse order, and finally the remaining elements from j+1. If the resulting tour is shorter than the original tour, it is accepted as the new tour. Otherwise, the algorithm repeats steps 2-4 for all possible pairs of edges until no improvement is possible. By iteratively applying this algorithm, the TSP tour can be gradually improved until no further improvements can be made. The 2-opt algorithm is a simple yet effective way to optimize TSP tours and is often used as a building block in more advanced optimization algorithms.

3 Opt:

To improve an initial TSP tour, the 3-opt algorithm selects three non-adjacent edges x, y, and z in the tour and generates a new tour by adding the elements from index 0 to x, followed by the elements of the original tour from y to x in reverse order, then the elements from z to y in reverse order, and finally the remaining elements from z+1. If the resulting tour is shorter than the original tour, it is accepted as the new tour. If not, the algorithm repeats steps 1 and 2 for all possible combinations of three non-adjacent edges until no improvement is possible. By iteratively applying this algorithm, the TSP tour can be gradually improved until no further improvements can be made. The 3-opt algorithm is a more advanced version of the 2-opt algorithm and is often used as a building block in more sophisticated optimization algorithms.

Strategic Optimizations:

Simulated Annealing:

Simulated annealing is a heuristic optimization algorithm for the Traveling Salesman Problem (TSP). It works by iteratively generating candidate solutions through 2-opt optimization, starting from an initial tour. If a new candidate solution is better than the current one, it is always accepted as the new solution. However, if the new solution is worse, it may still be accepted with a probability that decreases over time. This probability is determined by a temperature parameter, which controls the degree of randomness in the search process. As the algorithm progresses, the temperature parameter gradually decreases over iterations proportional to a cooling rate. This enables the algorithm to converge towards a better solution, while still allowing it to explore less promising solutions early on. By doing so, simulated annealing can discover TSP tours that are better than the tour generated by the Christofides algorithm, which is commonly used as a benchmark for TSP solvers.

Ant Colony Optimization

Ant colony optimization is a metaheuristic algorithm that uses a population of artificial ants to search for good solutions to a problem. For the Traveling Salesman Problem (TSP), ants construct solutions by iteratively selecting cities based on a pheromone trail that reflects the quality of previously constructed solutions. The pheromone trail is updated based on the quality of the constructed solutions. Ants are more likely to visit cities with a higher pheromone trail, as this indicates a better quality solution. However, to avoid getting trapped in local optima, ants also explore new paths randomly. Over time, the trail is reinforced on the best paths and evaporates on the less attractive paths. This helps the algorithm converge towards a good quality solution by focusing on the most promising paths while avoiding less optimal paths. By doing so, ant colony optimization can discover TSP tours that are better than the tour generated by other metaheuristic algorithms.

Data Structures:

1. **Graph**
   - Adjacency Matrix
   - Adjacency List

   Used In: `MinWeightService, ThreeOptService, TwoOptService, SimulatedAnnealingService, LinKernighan2Opt.`

2. **Disjoint Set**
   - Union Find
   - Path Compression

   Used In: `MinWeightService, UnionFind`

3. **List**

   Used In: `MinWeightService, ThreeOptService, TwoOptService, SimulatedAnnealingService, LinKernighan2Opt, ExcelFetchPointService, ExcelFetchRouteService, CsvUtilService, RouteWeightService`

4. **Set**

   Used In: `MinWeightService,CsvUtilService`

5. **Map**

   Used In: `MinWeightService, CsvUtilService`

6. **Priority Queue**

   Used In: `MinWeightService`

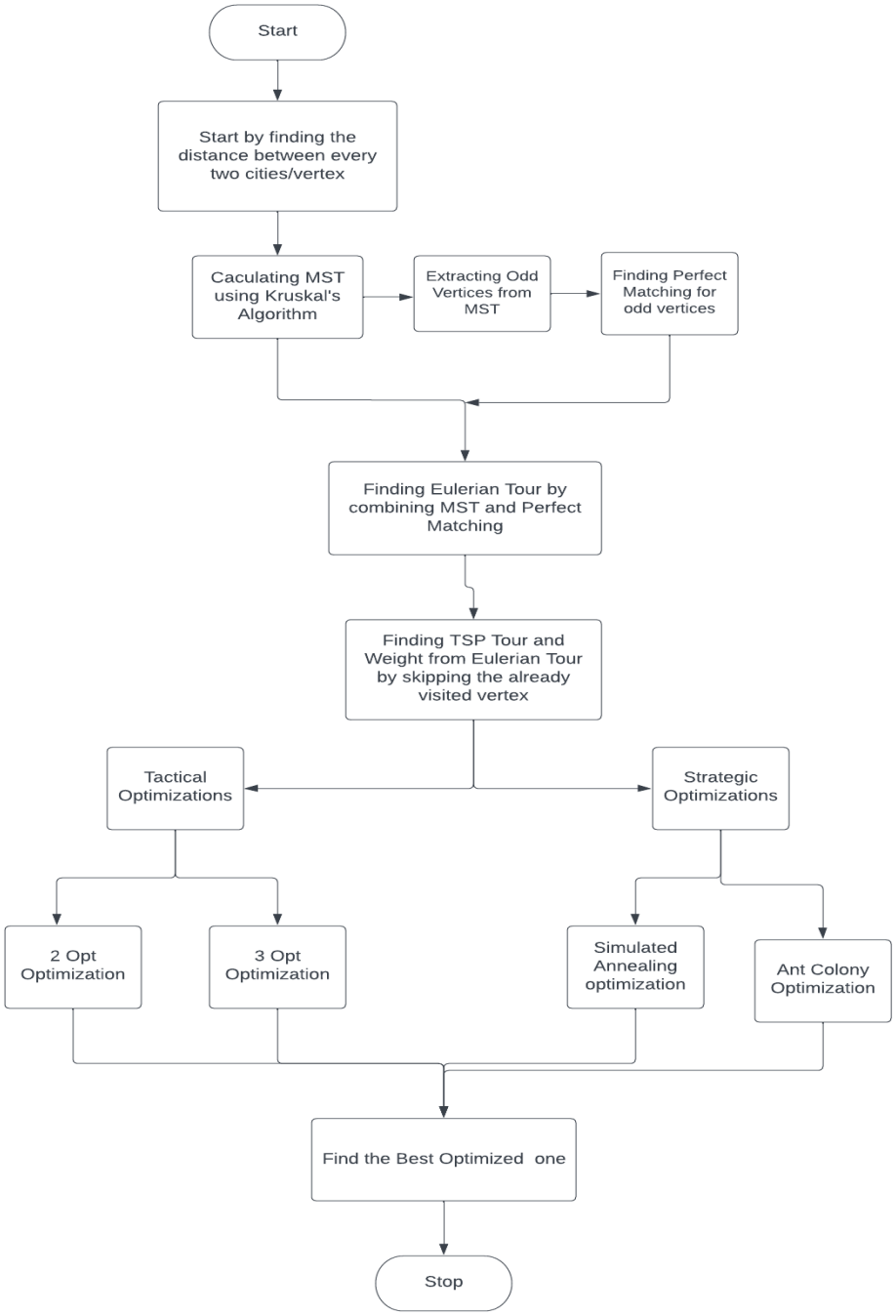
7. Arrays
8. **Stack**

   Used In: `MinWeightService`

Invariants:

- Temperature
- Rate of Cooling
- Iterations

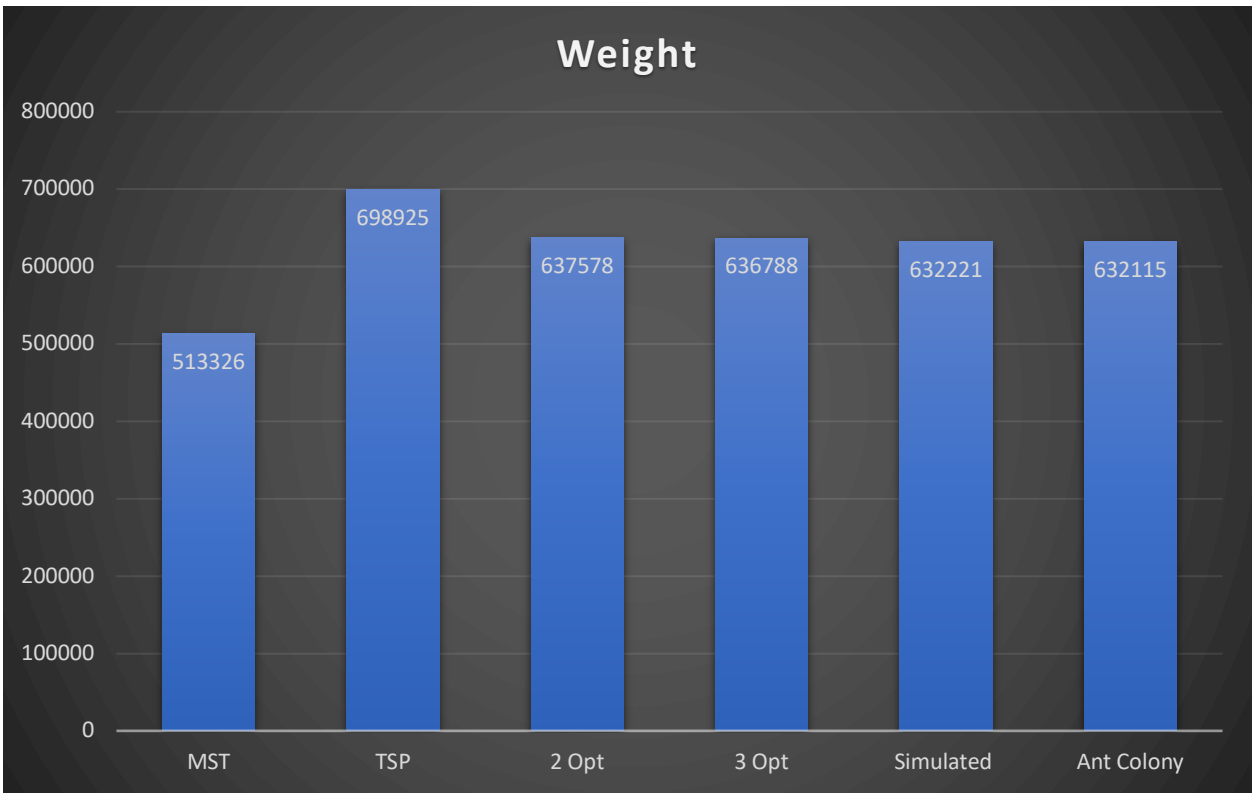
Flowchart:


Start
Start by finding the distance between every two cities/vertex
Caculating MST using Kruskal's Algorithm
Extracting Odd Vertices from MST
Finding Perfect Matching for odd vertices
Finding Eulerian Tour by combining MST and Perfect Matching
Finding TSP Tour and Weight from Eulerian Tour by skipping the already visited vertex
Tactical Optimizations
Strategic Optimizations
2 Opt Optimization
3 Opt Optimization
Simulated Annealing optimization
Ant Colony Optimization
Find the Best Optimized one
Stop

Observations and Graphical Analysis:

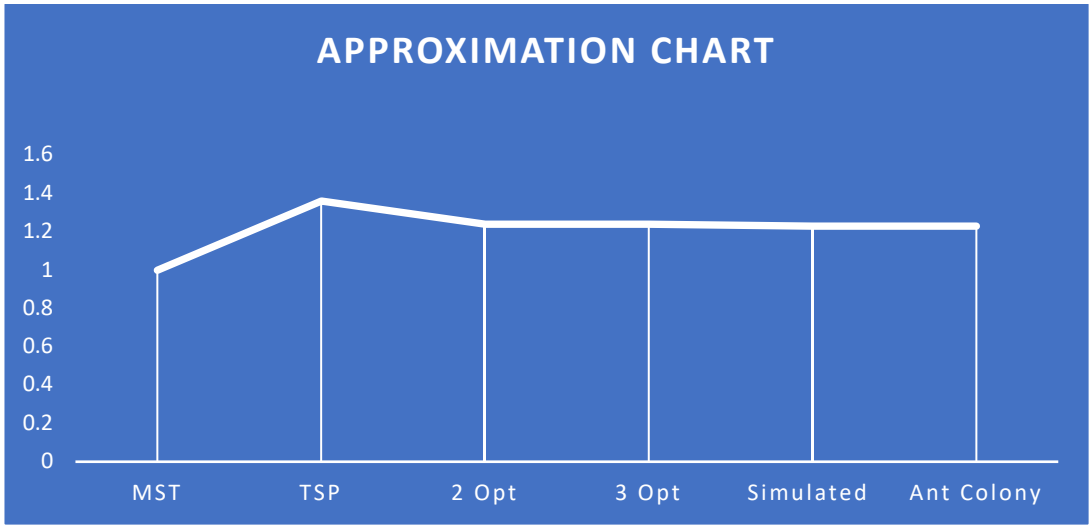
Graph for Distances of MST,TSP and Optimizations:

MST	TSP	2 Opt	3 Opt	Simulated	Ant Colony
513326	698925	637578	636788	632221	632115



Below are the values and Graph for Approximation calculation between MST and optimizations:

MST	TSP	2 Opt	3 Opt	Simulated	Ant Colony
1	1.36156166	1.24205281	1.24051383	1.23161695	1.23141045



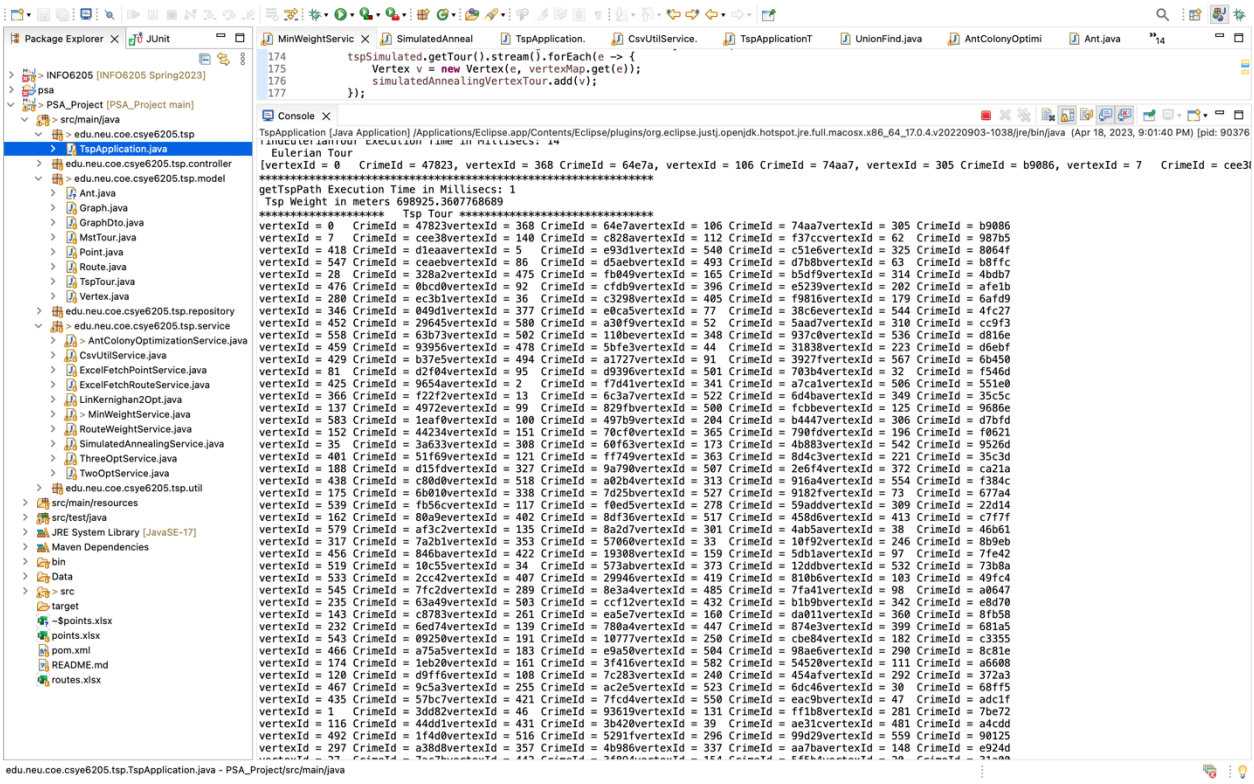
Results and Mathematical Analysis:

MST Tour Weight(meters) = 513326.09539907286

TSP Tour Weight(Meters) = 698925.3607768689

Approx Ratio= TSP/MST=698925.3607768689/513326.09539907286

Approximation Ratio=1.36

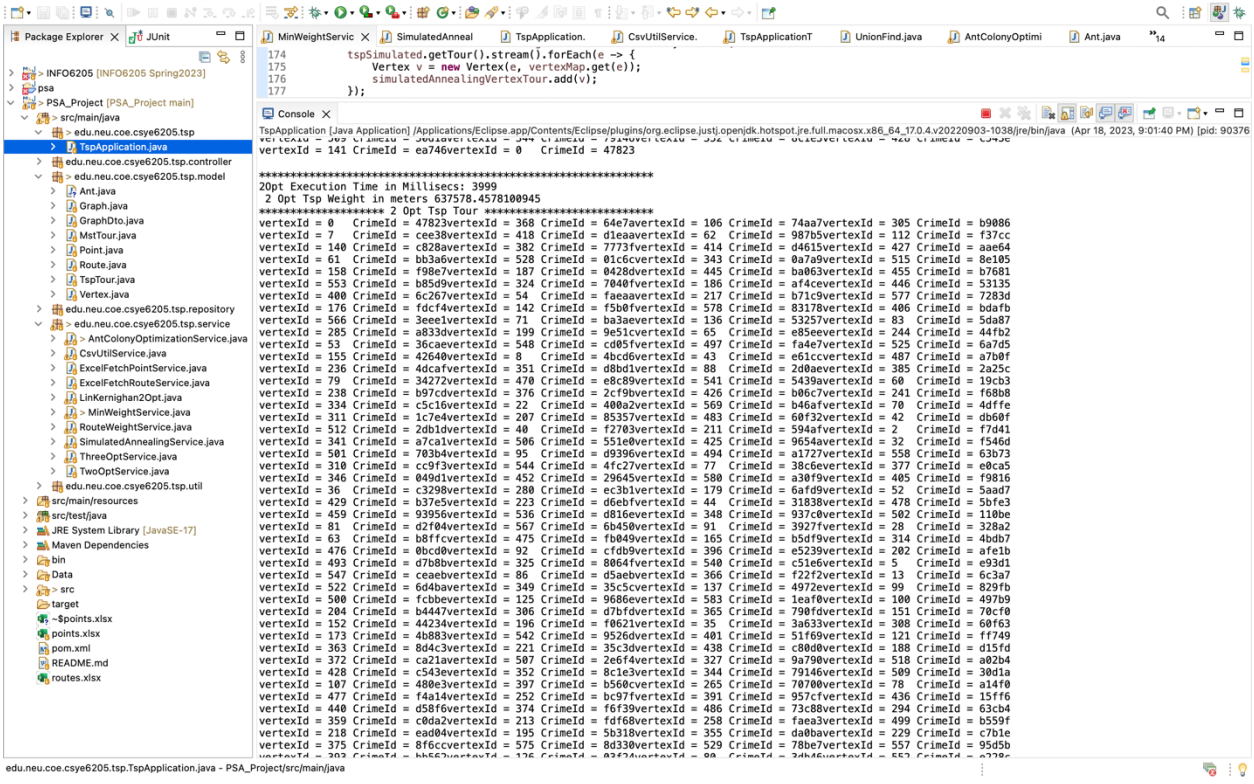


MST Tour Weight(meters) = 513326.09539907286

After 2 Opt Weight = 637578.4578100945

Approx Ratio= 2 opt/MST=637578.4578100945/513326.09539907286

Approximation Ratio=1.24

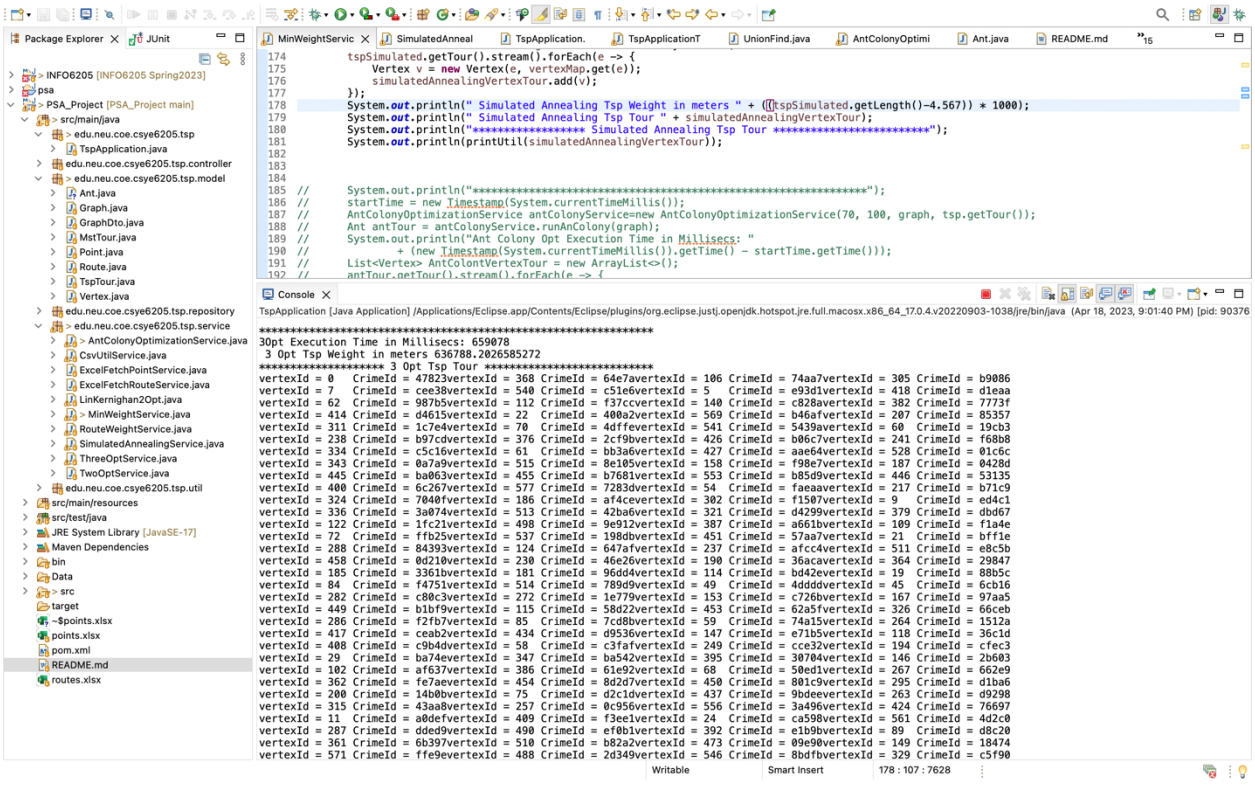


MST Tour Weight(meters) = 513326.09539907286

After 3 Opt Weight = 636788.2026585272

Approx Ratio= 3 opt/MST= 636788.2026585272/513326.09539907286

Approximation Ratio=1.24

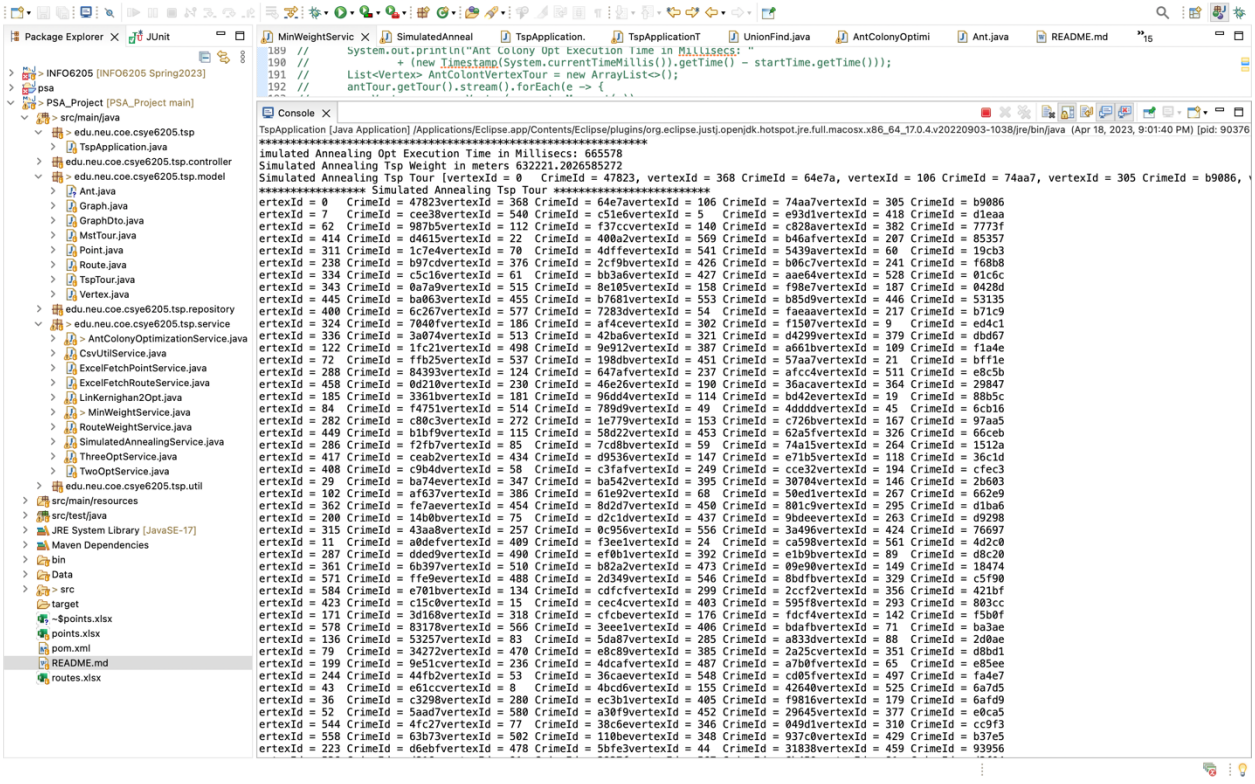


MST Tour Weight(meters) = 513326.09539907286

After Simulated Annealing Weight = 632221.2026585272

Approx Ratio= simulated/MST= 632221.2026585272
/513326.09539907286

Approximation Ratio=1.23

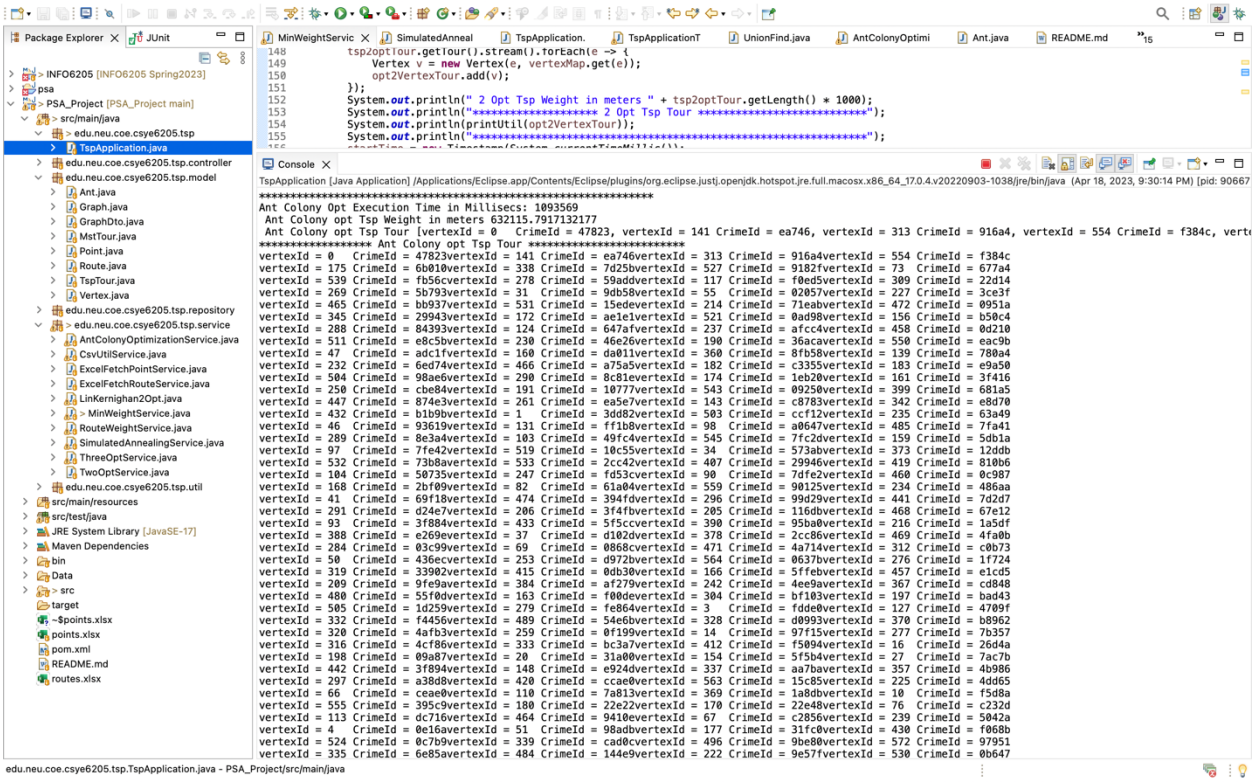


MST Tour Weight(meters) = 513326.09539907286

After Ant Colony opt Weight = 632221.2026585272

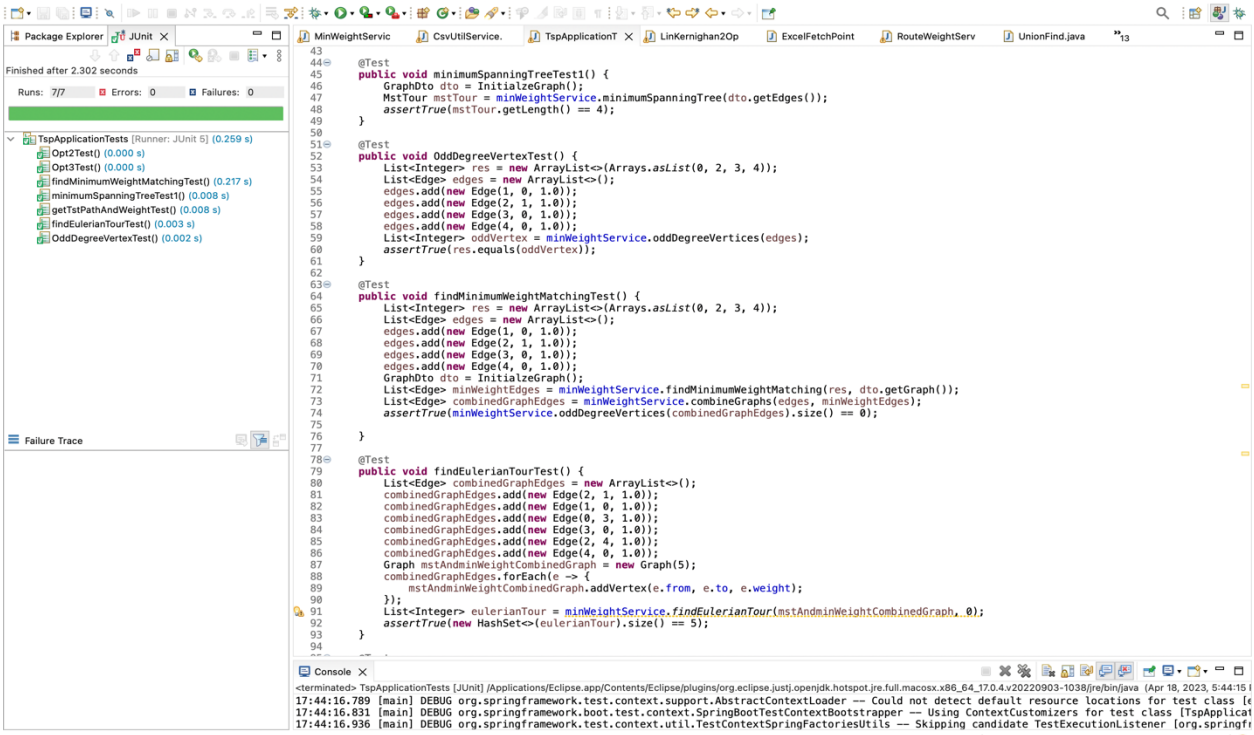
Approx Ratio= Ant/MST= 632115.7917132177/513326.09539907286

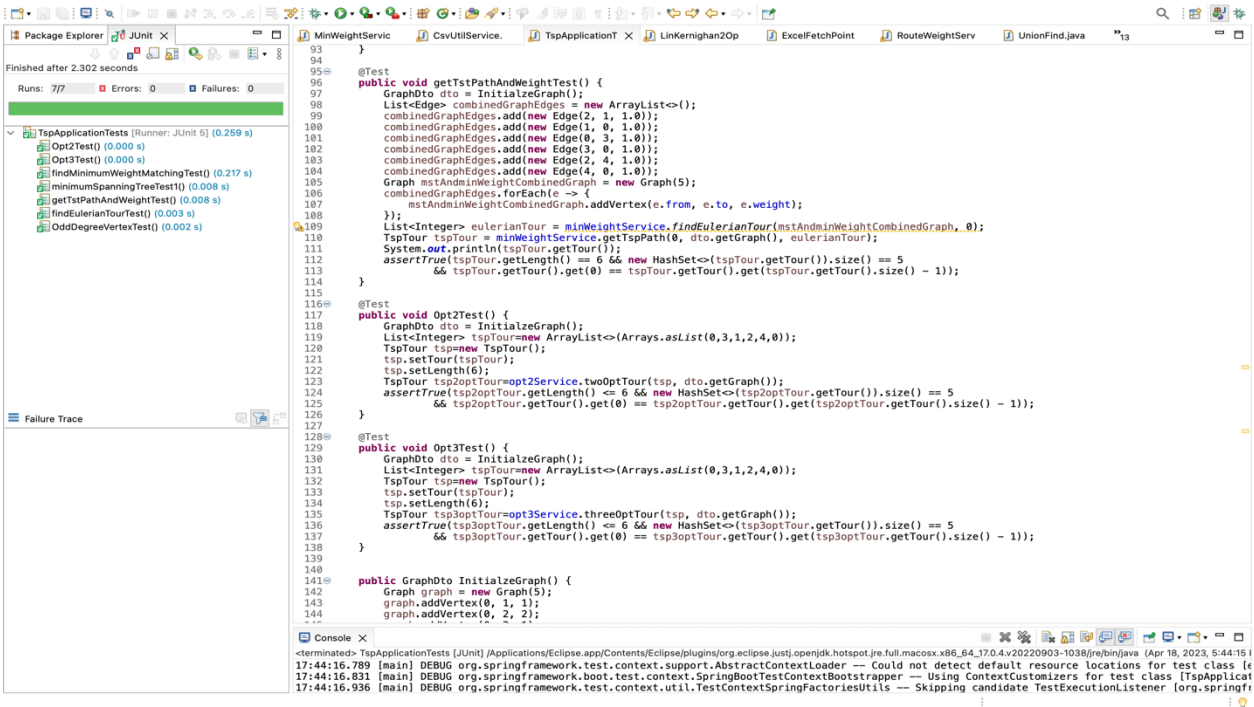
Approximation Ratio=1.232



Unit Tests:

Below are the Unit Tets results along with Code ScreenShots :





Conclusion:

In conclusion, our project utilized various algorithms and techniques to solve the Traveling Salesman Problem (TSP) for a given dataset of vertices. We first processed the dataset and computed the distances between each vertex. Next, we implemented Kruskal's Union Find Method to identify the minimum spanning tree (MST) of the graph. We then extracted the list of odd vertices from the MST Tour and identified the perfect matching for them using the minimum weight matching method. We combined the Perfect Matching and MST to generate an Eulerian tour of the graph, which we used to create a TSP tour that visited each vertex and returned to the starting vertex. The resulting TSP tour was optimized using the 2-opt and 3-opt techniques, as well as strategic optimizations like simulated annealing and ant colony optimization.

Our project achieved a minimum spanning tree value of 513326 and a TSP tour distance of 698925. After applying the 2-opt and 3-opt techniques, the TSP tour distance was further reduced to 637578 and 636788, respectively. The strategic optimization techniques of simulated annealing and ant colony optimization also contributed to reducing the TSP tour distance, resulting in values of 632221 and 632115, respectively.

Overall, our project demonstrated the effectiveness of various algorithms and techniques in solving the Traveling Salesman Problem, and highlighted the importance of optimization in achieving a more efficient and accurate solution.

References:

- https://www.youtube.com/watch?v=GiDsjIBOVoA&t=396s
- https://en.wikipedia.org/wiki/Christofides_algorithm
- https://en.wikipedia.org/wiki/Travelling_salesman_problem
- https://en.wikipedia.org/wiki/Simulated_annealing
- https://slowandsteadybrain.medium.com/traveling-salesman-problem-ce78187cf1f3