Rethakgetse Manaka

U22491032

COS314 Assignment 1

Technical Specification

Program Description: Iterated Local Search

The first program implements an Iterated Local Search algorithm to solve the problem(which is based on the traveling salesman problem). The cost values are read in from a file which have the different distances between the different locations (campuses). The algorithm iteratively searches for the optimal route to visit all the campuses and then returns to the starting point, aiming to minimize the total distance traveled.

Experimental Setup:

- **Operating System:** Any system supporting Java (e.g., Windows, Linux, macOS)
- **Java Version:** Java 8 or later
- **IDE:** Any Java IDE or any IDE that supports Java programs (e.g., IntelliJ IDEA, Eclipse, Visual Studio Code)
- **External Libraries:** The standard Java libraries were used. ArrayList, Collections, List were the data structure libraries used to store the data for the algorithm. Random was used so that we can randomize initial solution generation. File and Scanner were used

for reading in the matrix of cost values from the file which contains the cost values of the different distances between the campuses.

- **Other requirements:** It should be noted that the storage of the data may be in a plethora of formats, It may be stored directly in a 2D array or can be read in from a text file.

Algorithm Configuration:

- **Input data:** A cost matrix representing the distances between different campuses.
- **Algorithm:** Iterative Local Search
  - **Initialization:** Random initialization of a solution.
  - **Iterations:** 1,10,100,1000
  - **Neighbourhood Search:** Local search using the hill climb local search algorithm to improve the current solution iteratively.
  - **Termination Criteria:** Maximum number of iterations reached.
- **Objective Function:** Total distance traveled in the route.
- **Output:** The best route found, total distance, and the average objective function value over a specified number of iterations.

Experiment Procedure:

- **Input data preparation:**
  a. Read the cost matrix from the “Cost_Matrix.txt” file.
  b. Initialize the campus objects to represent different locations

- **Algorithm Execution:**
  a. We generate an initial solution using a random selection as a solution.
  b. We initialize the best objective function value of the solution as the best total distance, there are chances this may change if better solutions are found.
  c. We apply the hill climbing algorithm as our local search to our solution in order to improve it.
  d. With the local search applied we obtain a local optimum.
  e. We perturb the local optimum by randomly swapping two campuses this allows us to get a new solution.
  f. If our new solution is better than the current solution we update the current solution.
  g. We repeat steps d,e,f until we have met our desired iterations

- **Result Analysis:**
  a. We output the best route that was able to found by the algorithm
  b. We output the calculated total distance traveled.
  c. We calculate and output the average objective function value.

d. We measure and output the runtime of the algorithm.

Evaluation Metrics:

- **Execution time**: Time taken to execute the algorithm
- **Solution Quality**: Total distance traveled by the best route found.
- **Convergence**: Average objective function value(Average total distance)

Additional Notes:

- Ensure that the file “Cost_Matrix.txt” exists and contains valid data
- Ensure to specify the size of the matrix in the program.

Program Description: Simulated Annealing

The second program implements a Simulated Annealing algorithm to solve the problem(which is based on the traveling salesman problem). The cost values are read in from a file which have the different distances between the different locations (campuses). The algorithm iteratively searches for the optimal route to visit all the campuses and then returns to the starting point, aiming to minimize the total distance traveled.

Experimental Setup:

- **Operating System:** Any system supporting Java (e.g., Windows, Linux, macOS)
- **Java Version:** Java 8 or later
- **IDE**: Any Java IDE or any IDE that supports Java programs (e.g., IntelliJ IDEA, Eclipse, Visual Studio Code)
- **External Libraries:** The standard Java libraries were used. Random was used so that we can generate . File and Scanner were used for reading in the matrix of cost values from the file which contains the cost values of the different distances between the campuses.
- **Other requirements:** It should be noted that the storage of the data may be in a plethora of formats, It may be stored directly in a 2D array or can be read in from a text file.

Algorithm Configuration:

- **Input data:** A cost matrix representing the distances between different campuses.
- **Algorithm:** Simulated Annealing
  - **Initialization:** Random initialization of a solution.
  - **Annealing Schedule:** We defined a temperature and a cooling rate which allowed us to adjust the number of iterations the algorithm went through.
  - **Iterations:** 1,10,100,1000

    - **Neighbourhood Search:** We used a perturbation mechanism to explore neighbouring solutions. Based on a random value and our acceptance probability this would allow us to randomly check other neighbouring solutions.
    - **Acceptance Criteria:** A probability criteria that allows us to randomly check neighbouring solutions and escape local optima.
    - **Termination Criteria:** The temperature at which the algorithm should stop iterating through solutions.
- **Objective Function:** Total distance traveled in the route.
- **Output:** The best route found, total distance, and the average objective function value over a specified number of iterations.

Experiment Procedure:

- **Input data preparation:**
    a. Read the cost matrix from the "Cost_Matrix.txt" file.

- **Algorithm Execution:**
    a. We generate an initial solution using a random selection as a solution.
    b. We initialize the best objective function value of the solution as the best total distance, there are chances this may change if better solutions are found. As well as we initialize the temperature and cooling rate.
    c. We run the core of the simulated annealing algorithm.
    d. We iteratively explore solutions by randomly swapping two campuses in the current solution. We may accept worse solutions which are determined by our acceptance probability, based on the current temperature.
    e. We repeat the algorithm until our temperature drops below our threshold. Whenever a solution which is better is encountered we use it as our best solution until we encounter a better solution.
- **Result Analysis:**
    a. We output the best route that was able to found by the algorithm
    b. We output the calculated total distance traveled.
    c. We calculate and output the average objective function value.
    d. We measure and output the runtime of the algorithm.

Evaluation Metrics:

- **Execution time**: Time taken to execute the algorithm
- **Solution Quality**: Total distance traveled by the best route found.
- **Convergence**: Average objective function value(Average total distance)

Additional Notes:

- Ensure that the file "Cost_Matrix.txt" exists and contains valid data
- Ensure to specify the size of the matrix in the program.

Results Table

Problem Set	ILS (Iterated Local Search)	SA (Simulated Annealing)
Best Solution (Route)	2, 3, 4, 5, 1	5, 4, 3, 2, 1
Objective Function Value	81	81
Runtime (μs)	1078	393
Avg Objective Function Value	82.5	83

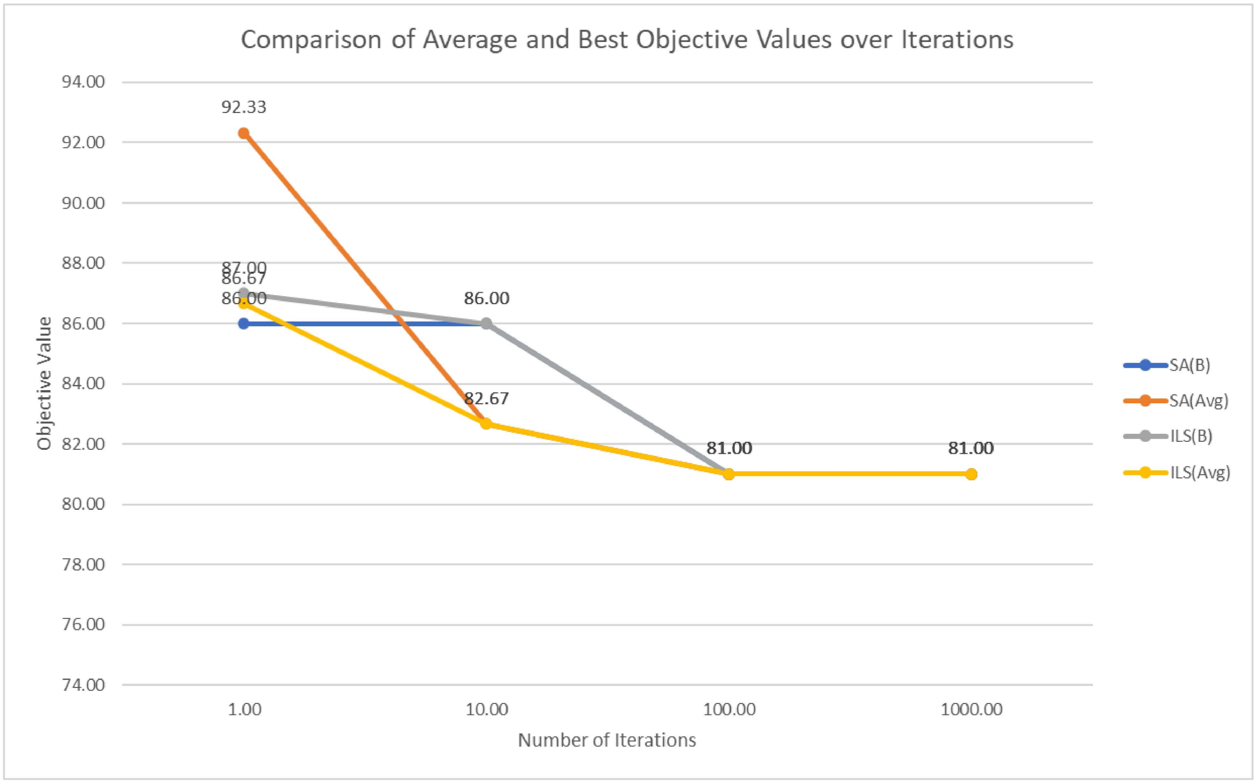
Number of Iterations: 10 per algorithm

Legend:
1. Hatfield Campus
2. Hillcrest Campus
3. Groenkloof
4. Prinshof
5. Mamelodi

Results

Data Collected across 4 different levels of iterations

Iterations	SA(B)	SA(Avg)	ILS(B)	ILS(Avg)
1.00	86.00	92.33	87.00	86.67
10.00	86.00	82.67	86.00	82.67
100.00	81.00	81.00	81.00	81.00
1000.00	81.00	81.00	81.00	81.00



Discussion & Conclusion

Based on our findings, it appears that the Iterated Local Search (ILS) algorithm performs better than the Simulated Annealing (SA) algorithm when it comes to solving the problem which is based on the Traveling Salesman Problem (TSP). Even though ILS takes more time to run, it consistently gives us better solutions. This suggests that ILS is better at exploring different options and avoiding getting stuck in less-than-ideal solutions.

On the other hand, Simulated Annealing doesn't seem to do as well in finding the best solutions every time. It tends to settle for solutions that aren't as good, as shown by its higher average objective function value. Also, when we look at the graph, we see that Simulated Annealing struggles to find the best solution when it doesn't have a lot of chances to try different options. While Simulated Annealing is faster than ILS, especially when we need to find the best solution quickly, it doesn't always give us the best results. So, even though ILS takes longer, it's more dependable in giving us high-quality solutions consistently.

In conclusion, based on our experiment, the Iterated Local Search algorithm seems to be the better choice for solving TSP problems because it consistently gives us better solutions over different situations. However, it's important to keep exploring and testing both algorithms to see how they perform in different situations and with different constraints.