The goal of this project is to create 2 algorithms to find a solution to the Travelling salesman problem (TSP) by getting the shortest path to travel to all campuses.

The two algorithms being used are:

1. Iterated Local Search (ILS)
2. Simulated Annealing (SA)

The project is being developed in Java.

Iterated Local Search (ILS)

The Iterated Local Search algorithm is a metaheuristic that iteratively improves a candidate solution by applying a local search algorithm and perturbing the solution. The algorithm is as follows:

1. Generate an initial solution.
2. Apply a local search algorithm to the solution.
3. Perturb the solution (in this case by swapping 2 random campuses)
4. Apply the local search algorithm to the perturbed solution .
5. If the new solution is better than the current solution, replace the current solution with the new solution.
6. Repeat steps 3-5 until a stopping criterion is met or a maximum number of iterations is reached.

Note: in this case I have only perturbed the solution by swapping 2 random campuses as swapping 2 adjacent campuses and then swapping 2 random

campuses may result in swapping the 2 adjacent campuses back to their original position, this would make no progress and may cause the algorithm to terminate early due to the stopping criteria.

Simulated Annealing (SA)

The Simulated Annealing algorithm is a probabilistic metaheuristic that is inspired by the annealing process in metallurgy.

This algorithm Is meant to find the best path to take however in the interest of not getting stuck in a local minima when a good solution is found, there is a probability that the algorithm could accept a worse solution in 1 iteration to take a different path to see if an even better solution can be found.

The algorithm is as follows:

1. Generate an initial solution.

2. Set the initial temperature and cooling rate.
3. Repeat until the stopping criterion is met: (In this case when the temp value gets lower than the temperature is too low to accept any new solution)
    a. Generate 3 new solutions by swapping random campuses.
    b. Get the best route from the 3 new solutions.
    c. Calculate the cost difference between the new solution and the current solution.
    d. If the new solution is better than the current solution, replace the current solution with the new solution.
    e. If the new solution is worse than the current solution, accept the new solution with a probability based on the cost difference and the current temperature.
    f. Update the temperature.

Results Table

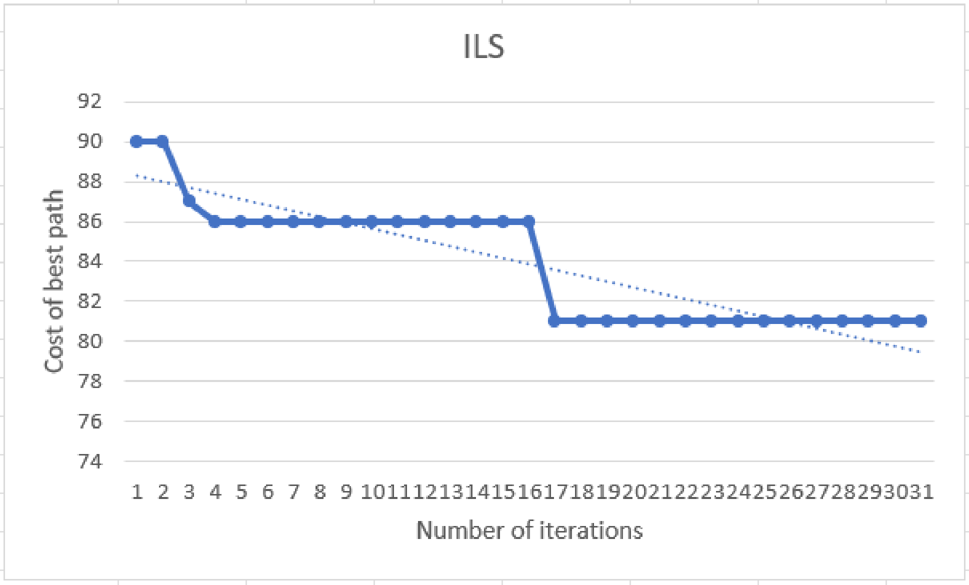
Problem Set	**ILS**	**SA**
Best Solution(route)	Hatfield -> Groenkloof -> Hillcrest -> Prinsof -> Mamelodi -> Hatfield	Hatfield -> Hillcrest -> Groenkloof -> Prinsof -> Mamelodi -> Hatfield
Objective Function Val	81	81
Runtime	31	45
Av Obj Func	85	82

For runtime I opted to use iterations rather than time as actual runtime can be inconsistent but iterations are a fixed solution.

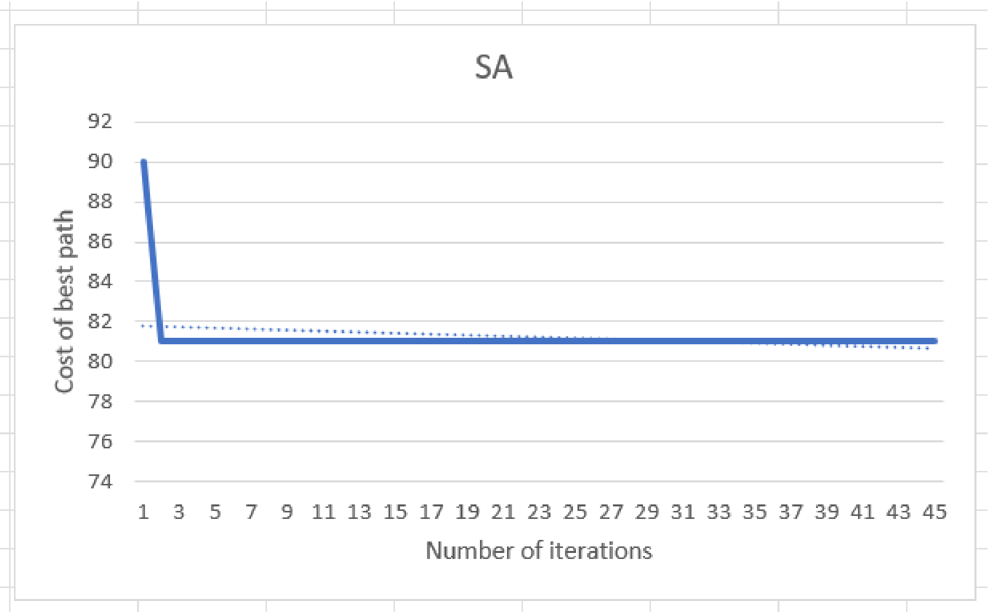
Av Obj Func I did the average of all the runs distances

Graphs

ILS:



SA:



Comparison

For this specific example SA managed to get to the shortest path a lot faster than ILS, it had a lower average objective distance but using my tests it did have a longer runtime than ILS.

However, when changing the starting values SA did not always manage to get to the shortest route even after 10000 iterations.

Taking this into consideration as well as the trendline, ILS should perform better with a larger data set.