Technical Specification

Note both algorithms were given the same initial population

Iterated Local Search (ILS) Algorithm

Introduction:

Iterated Local Search (ILS) is a metaheuristic optimization algorithm used to find near-optimal solutions for combinatorial optimization problems. It iteratively explores the solution space by refining the current solution through local search and perturbation.

Experimental Setup:

Hardware: AMD Athlon Gold 3150U CPU, 8GB RAM.

Software: Java.

Environment: Window 11.

Algorithm Configuration:

Initial Solution Generation: Randomly place a campus at a random index using Primary Hash. If there is a clash, use primary and secondary hash to find new index to place campus. If there is a clash again, start the process again and use primary key. Repeat process until all campus are placed into an array.

Local Search: Randomly generate a index, swap the elements of the index generated and the forward neighbor. If the generated index does not have a forward neighbor, then swap with back neighbor.

Perturbation: Apply Shaking operator. Randomly generate two different indexes. Swap the elements at the two indexes.

Acceptance Criterion: Accepts new solutions based on improvements (e.g., if the new solution is cost less than the best solution, accept it unconditionally as the new best solution.

Stopping Criterion: Maximum number of iterations which was set 20.

Experimental Procedure:

1)Initialization: Generate an initial solution using the random construction heuristic.

Iterations:

2)Apply the local search operator to the best solution.

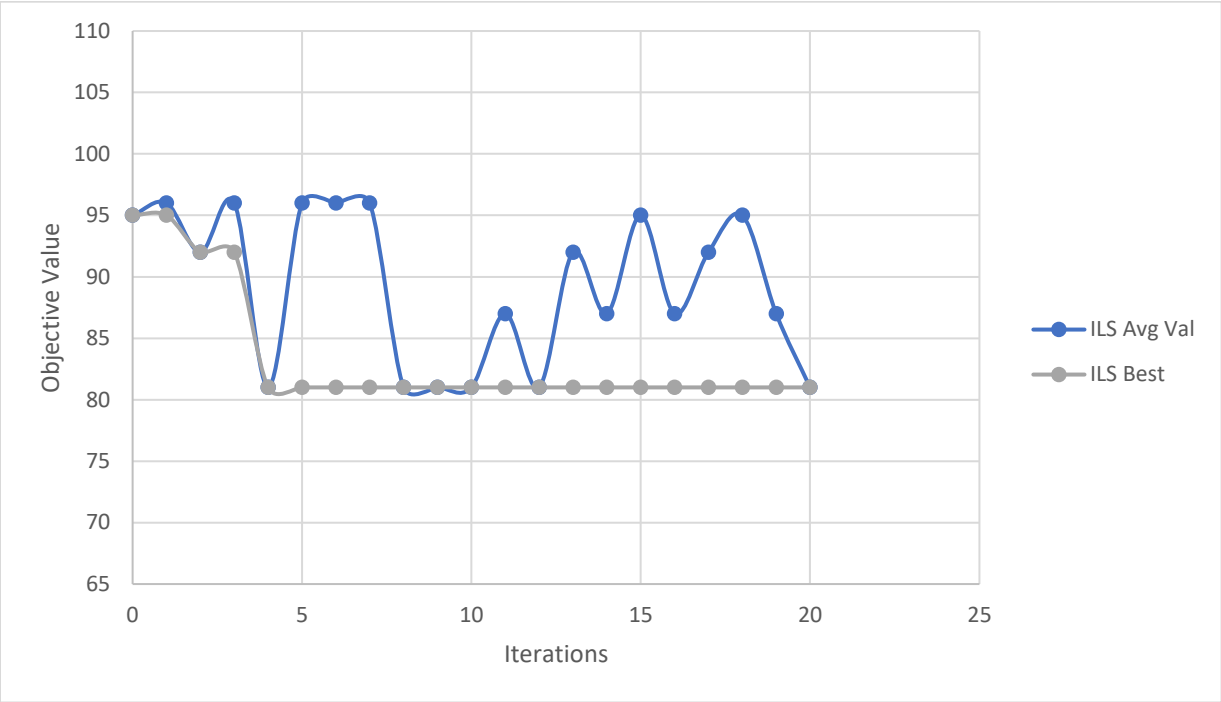
3)Perturb the solution using the shaking operator.

4)Update the Best solution based on the acceptance criterion.

Repeat steps 2-3 until the stopping criterion is met.

Evaluation: Measure the quality of the final solution using the cost function to calculate the total cost of the path

Current and Best Graphical Plot



Simulated Annealing Algorithm

Introduction:

Simulated Annealing (SA) is a probabilistic optimization algorithm inspired by the annealing process in metallurgy. It is used to find near-optimal solutions for combinatorial optimization problems by iteratively exploring the solution space while gradually decreasing the search intensity.

Experimental Setup:

Hardware: AMD Athlon Gold 3150U CPU, 8GB RAM.

Software: Java.

Environment: Window 11.

Algorithm Configuration:

Initial Solution Generation: Randomly place a campus at a random index using Primary Hash. If there is a clash, use primary and secondary hash to find new index to place campus. IF there is a clash again, start the process again and use primary key. Repeat process until all campus are placed into an array.

Temperature Schedule: Cooling schedule defining how the temperature decreases over iterations. In this case temperature decreases in exponential way, where temperature decrease by 15% at every iteration.

Neighborhood Exploration: Randomly generate an index, swap the elements of the index generated and the forward neighbor. If the generated index does not have a forward neighbor, then swap with back neighbor.

Acceptance Criterion: Accepts new solutions based on improvement criteria (e.g., if the new solution cost less the known the best solution, accept it unconditionally; otherwise, accept it with a probability determined by a temperature parameter. Where a random value (0-100) is generated if it less than the temperature accepts the solution as current solution).

Stopping Criterion: Maximum number of iterations which was set 20.

Experimental Procedure:

Initialization: Generate an initial solution using a random construction heuristic or a predefined starting solution.

Initialize the temperature to 100.

Repeat the following until the stopping criterion is met:

Neighborhood search the current solution to obtain 3 neighboring solutions.

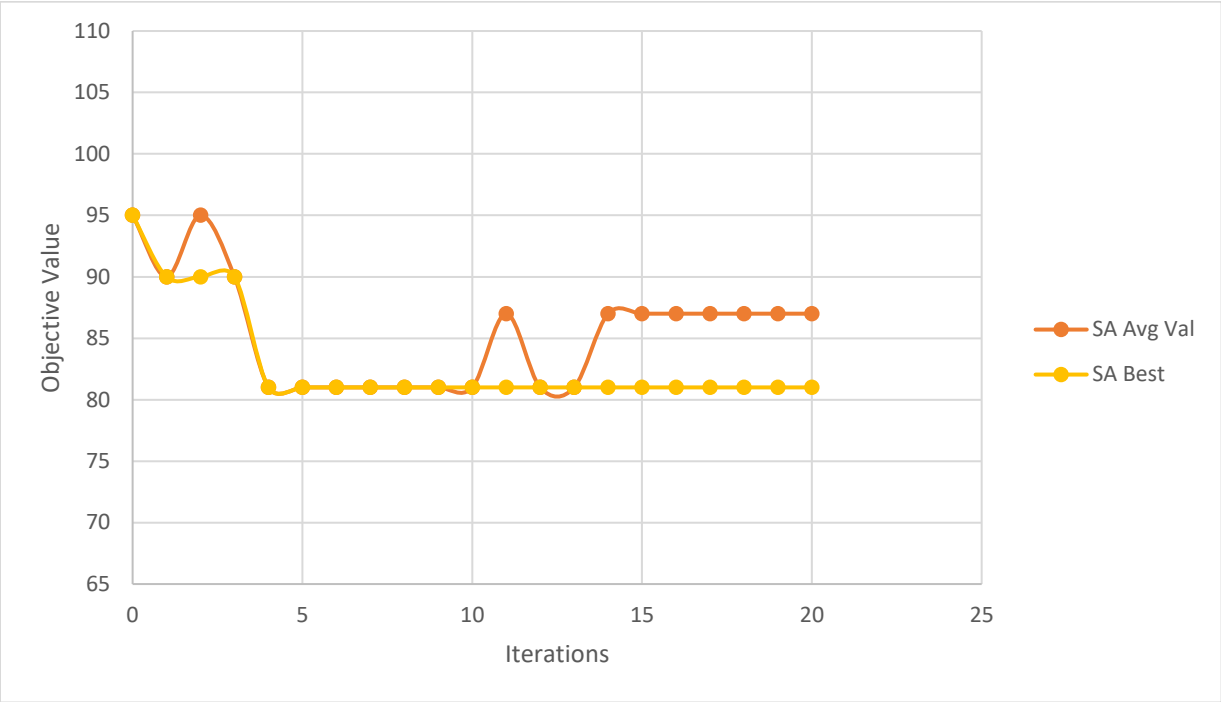
Choose the best out of the 3 to be next.

Compute the change in objective function value.

Accept or reject the next solution based on the acceptance criterion.

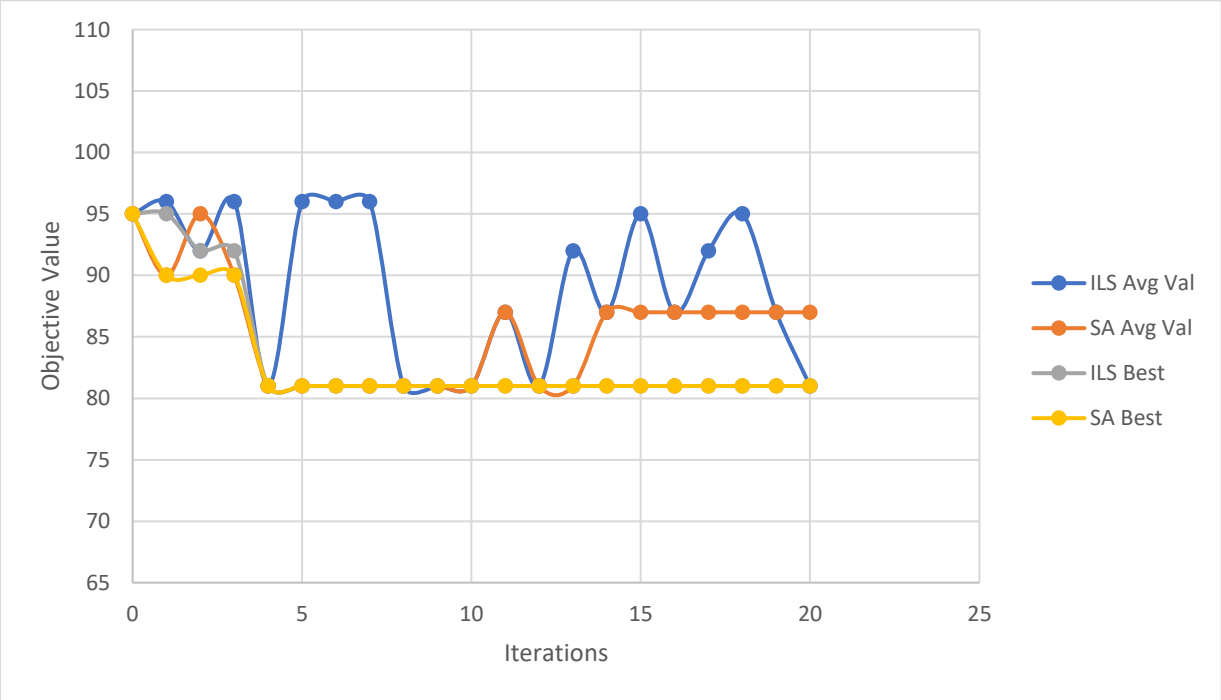
Update the temperature according to the cooling schedule.

Current and Best Graphical Plot



Results and Analysis

Problem Set	*Iterated Local Search*	*Simulated Annealing*
Best Solution	HATFIELD->MAMELODI->PRINSOF->GROENKLOOF->HILLCREST->	HATFIELD->HILLCREST->GROENKLOOF->PRINSOF->MAMELODI->
Objective Function Val	81	81
Runtime	201638966	191822412
Avg Obj	89	85



Discussion and Conclusion

We will compare the algorithms in the following aspects.

Minimum changes to Initial Candidate Solution

Initial Solution: HATFIELD -> GROENKLOOF -> HILLCREST -> MAMELODI -> PRINSOF

ILS Solution: HATFIELD->MAMELODI->PRINSOF->GROENKLOOF->HILLCREST

SA Solution: HATFIELD->HILLCREST->GROENKLOOF->PRINSOF->MAMELODI

Both algorithms have the same cost but have different solution. However SA solutions is closer to the initial solution compared to ILS.

Convergence Rate

According to the graphs, both algorithm have a very similar convergence rate.

Exploration (Escape local optima)

The Iterated local search explores a larger area of the search space compared to Simulated Annealing.

Exploitation

They both exploit the best-known solutions to find a better solution.

Sensitivity to Parameters

SA is more sensitive to parameters, especially the rate of change of the temperature and starting temperature. The Higher the iteration for both the algorithms the higher chances of finding a better solution.

The ITS outperforms the SA because it better equipped escaping local optima.