SUTD SHARP Term 3 Honours sessions Metaheuristic Optimization Homework

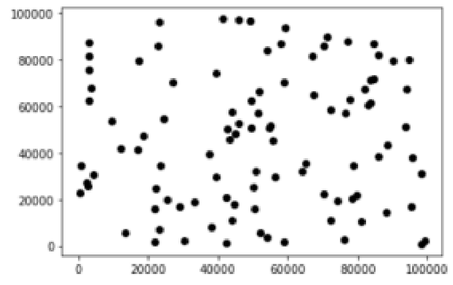
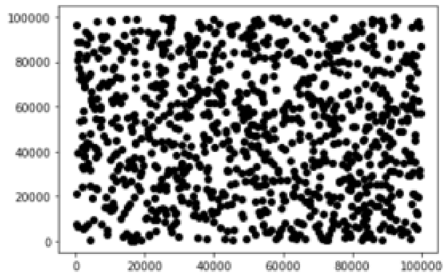
Goh Jet Wei 1007655

Consider the p-median location problem: Given a set of n locations representing demand points and a set of m potential facility locations, the p-median problem aims to determine the optimal selection of p facilities from the set of potential locations. The objective is to minimize the total distance associated with serving all demand points while ensuring that each demand point is assigned to its nearest facility among the selected p.

Consider the two random instances to be generated in python using random.seed(1).
The python file can be found in edimension under the project folder - **Instances.ipynb**
Note that other software can be used to code the algorithms – yet the instances must be generated in python to make sure that the instances run are the same to everyone.

Instance 1 considers 100 locations, with a requirement for exactly 15 facilities to be open.
Instance 2 considers 1000 locations, with a requirement for exactly 30 facilities to be open.
Please verify whether the generated instances match those provided below.

Note that in this exercise the set of demand locations is the same as the set of facility locations

Instance 1 (n=m=100, p=15)	**Instance 2 (n=m=1000, p=30)**
	
Exactly 15 facilities to be open	Exactly 30 facilities to be open

Exercise 1: Propose and implement a random sampling algorithm to solve the problem;

(2.1) solution encoding representation;

Instance 1 (n=m=100, p=15)	Instance 2 (n=m=1000, p=30)
[58, 72, 34, 4, 28, 10, 52, 38, 91, 26, 60, 87, 53, 90, 49]	[142, 893, 259, 514, 988, 452, 455, 347, 746, 900, 737, 149, 692, 654, 218, 67, 372, 250, 953, 795, 888, 45, 165, 328, 494, 966, 235, 648, 244, 230]

(2.2) explanation of the search procedure (no screenshots of the code);

The Random Sampling Optimisation algorithm works by first taking the objective value of the initial objective value from the initial solution. It then repeatedly generating random sets of p facilities and evaluating their performance in the set of demand points by calculating the total distance, etc. the objective value.

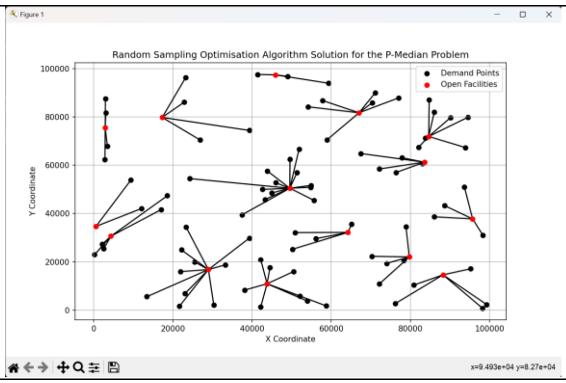
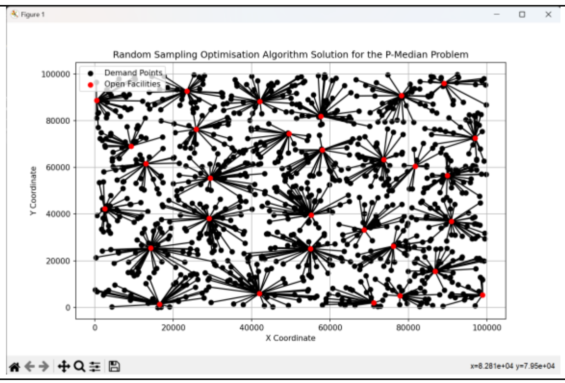
If the randomly generated set of p facilities obtained an objective value that is smaller than the initial objective value from the initial solution, this set of p facilities is now declared as the current best set of p facilities. Else, if the randomly generated set of p facilities obtained an objective value that is larger than the initial objective value from the initial solution, than this set of p facilities will be ignored and will not be updated as the current best set of p facilities.

This process repeats for a fixed number of time/iterations, where random sets of p facilities will keep being generated where randomly generated set of p facilities with an objective value that is smaller than the initial objective value from the initial solution will be updated as the current best set of p facilities. And the final current best set of p facilities will be returned as the best set of p facilities found by the Random Sampling Optimisation algorithm, along with its objective value.

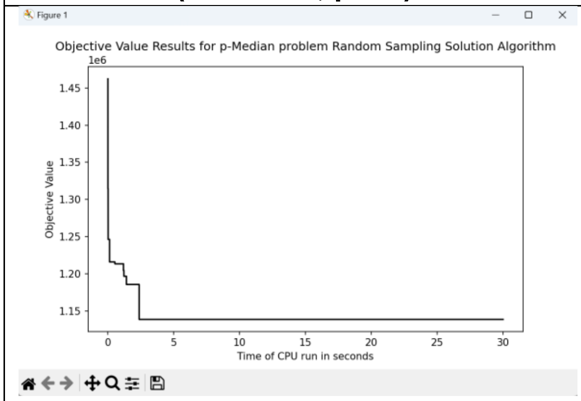
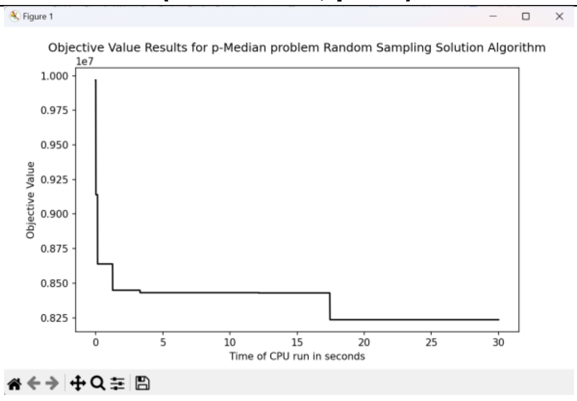
(2.3) results obtained for each exercise. The results should include:

(2.3.1) a plot displaying all the selected facility locations, along with connecting links to each of these Metaheuristic Optimization Page 2 of 2 facilities

Instance 1 (n=m=100, p=15)	Instance 2 (n=m=1000, p=30)

(2.3.2) a plot showcasing the Total Distance (Y-Axis) vs Computing Time (X-Axis)

Instance 1 (n=m=100, p=15)	**Instance 2 (n=m=1000, p=30)**
 Smallest Objective Value found: 1138426.5411935954	 Smallest Objective Value found: 8236285.194069174

Exercise 2: Propose and implement a local search algorithm to solve the problem;

(2.1) solution encoding representation;

Instance 1 (n=m=100, p=15)	Instance 2 (n=m=1000, p=30)
[60, 52, 86, 4, 26, 28, 73, 72, 77, 34, 66, 67, 53, 92, 41]	[261, 507, 667, 886, 981, 429, 588, 57, 571, 633, 434, 250, 939, 259, 859, 823, 542, 922, 191, 479, 636, 497, 499, 969, 696, 169, 552, 219, 876, 332]

(2.2) explanation of the search procedure (no screenshots of the code);

The Hill Climbing Local Search Optimisation algorithm works by first taking the objective value of the initial objective value from the initial solution. It then iteratively searches for better neighbouring set of p facilities (set of p facilities that is like the current set of p facilities, but slightly modified) using a search operator (can be either a swap random, insert random, or 3-opt search operator) and evaluating their performance in the set of demand points by calculating the total distance, etc. the objective value.

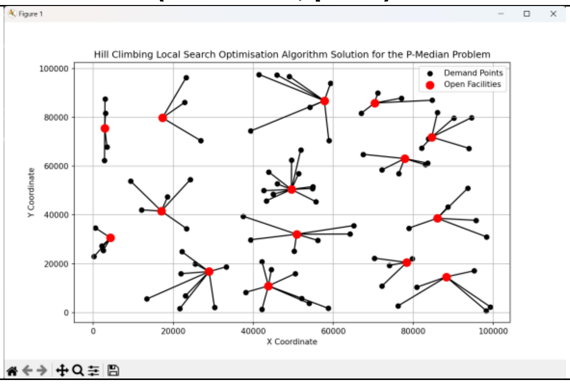
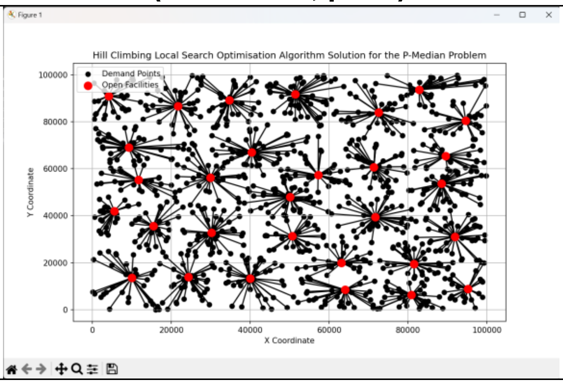
If the neighbouring set of p facilities has a smaller objective value than the initial objective value from the initial solution, this set of p facilities is now declared as the current best set of p facilities. Else, if the neighbouring set of p facilities obtained an objective value that is larger than the initial objective value from the initial solution, than this set of p facilities will be ignored and will not be updated as the current best set of p facilities.

This process repeats for a fixed number of time/iterations, where neighbouring set of p facilities will keep being searched for where neighbouring set of p facilities with an objective value that is smaller than the initial objective value from the initial solution will be updated as the current best set of p facilities. And the final current best set of p facilities will be returned as the best set of p facilities found by the Hill Climbing Local Search Optimisation algorithm, along with its objective value.

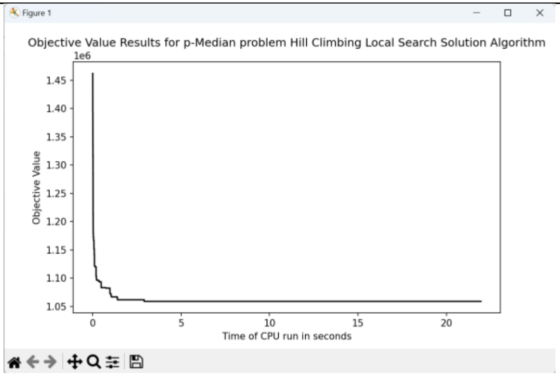
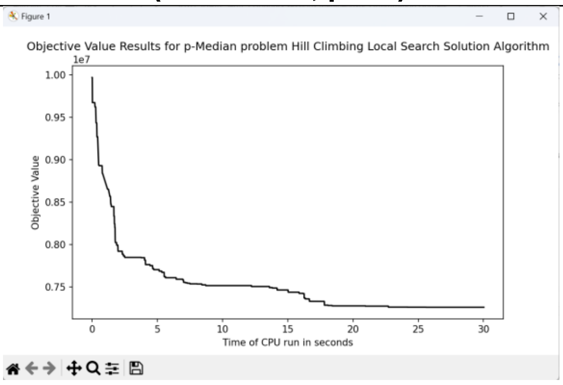
My code uses the swap random search operator with the Hill Climbing Local Search Optimisation algorithm. It works by randomly swapping one facility from the currently set of open p facilities with one from the set of closed facilities (i.e., facilities that are not currently open), creating a new neighbouring set of p facilities.

(2.3) results obtained for each exercise. The results should include:

(2.3.1) a plot displaying all the selected facility locations, along with connecting links to each of these Metaheuristic Optimization Page 2 of 2 facilities

Instance 1 (n=m=100, p=15)	**Instance 2 (n=m=1000, p=30)**
	

(2.3.2) a plot showcasing the Total Distance (Y-Axis) vs Computing Time (X-Axis)

Instance 1 (n=m=100, p=15)	**Instance 2 (n=m=1000, p=30)**
	
Smallest Objective Value found: 1059094.5132397362	Smallest Objective Value found: 7246276.764371948

Exercise 3: Propose and implement a simulated annealing algorithm to solve the problem;

(2.1) solution encoding representation;

Instance 1 (n=m=100, p=15)	Instance 2 (n=m=1000, p=30)
[4, 72, 26, 73, 34, 60, 28, 52, 41, 66, 86, 67, 92, 77, 53]	[460, 499, 456, 67, 620, 565, 318, 177, 783, 46, 946, 809, 870, 66, 181, 292, 985, 922, 796, 427, 838, 55, 310, 130, 233, 212, 519, 841, 402, 850]

(2.2) explanation of the search procedure (no screenshots of the code);

Similar to the Hill Climbing Local Search Optimisation algorithm, the Simulated Annealing Optimisation algorithm works by first taking the objective value of the initial objective value from the initial solution. It then iteratively searches for better neighbouring set of p facilities (set of p facilities that is like the current set of p facilities, but slightly modified) using a search operator (can be either a swap random, insert random, or 3-opt search operator) and evaluating their performance in the set of demand points by calculating the total distance, etc. the objective value.

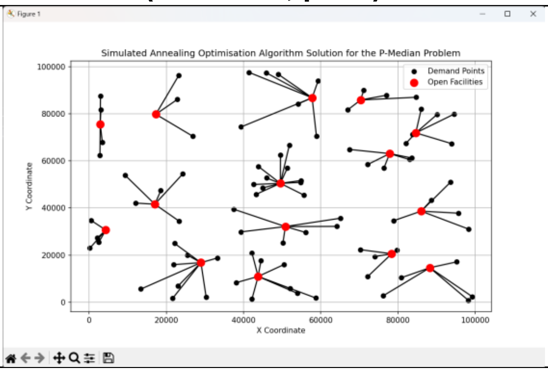
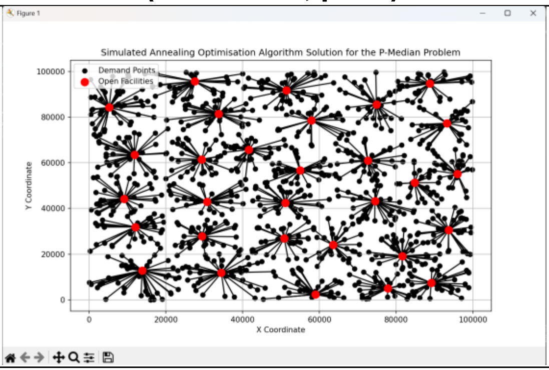
However, it differs from the Hill Climbing Local Search Optimisation algorithm as it has a probability of accepting a worse neighbouring set of p facilities, which is controlled by the temperature and the Metropolis Acceptance Criterion (which causes the increase in objective value as time passes occasionally). This probability/temperature starts out high in accepting worse neighbouring set of p facilities than the initial solution, in order to encourage exploration to avoid being trapped in a local minima. It then decreases/'cools' with time, making it less likely to accept worse neighbouring set of p facilities than the initial solution, to encourage exploitation of the current best set of p facilities to find even better set of p facilities. If the neighbouring set of p facilities has a smaller objective value than the initial objective value from the current best set of p facilities, this set of p facilities will definitely be selected with a probability of 1.

This process repeats for a fixed number of time/iterations, where neighbouring set of p facilities will keep being searched for where there will be a probability/temperature (which decreases with time) of a worse neighbouring set of p facilities than the current best set of p facilities/neighbouring set of p facilities with a smaller objective value than the initial objective value from the current best set of p facilities, being updated as the current best set of p facilities. And the final current best set of p facilities will be returned as the best set of p facilities found by the Simulated Annealing Optimisation algorithm, along with its objective value.

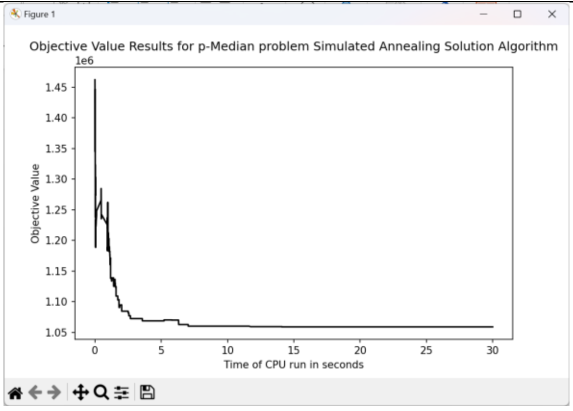
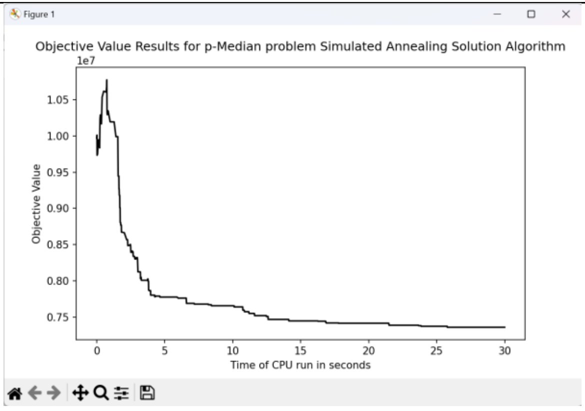
My code uses the swap random search operator with the Simulated Annealing Optimisation algorithm. It works by randomly swapping one facility from the currently set of open p facilities with one from the set of closed facilities (i.e., facilities that are not currently open), creating a new neighbouring set of p facilities.

(2.3) results obtained for each exercise. The results should include:

(2.3.1) a plot displaying all the selected facility locations, along with connecting links to each of these Metaheuristic Optimization Page 2 of 2 facilities

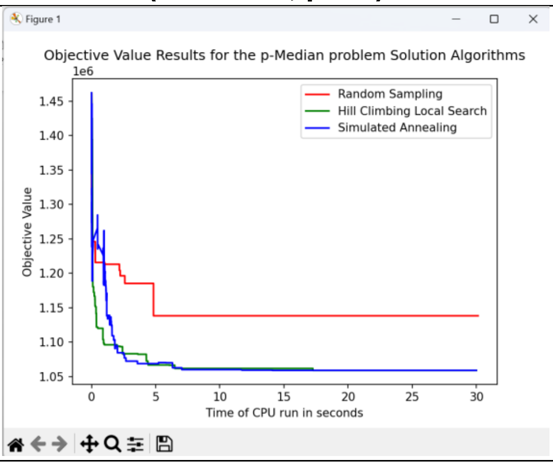
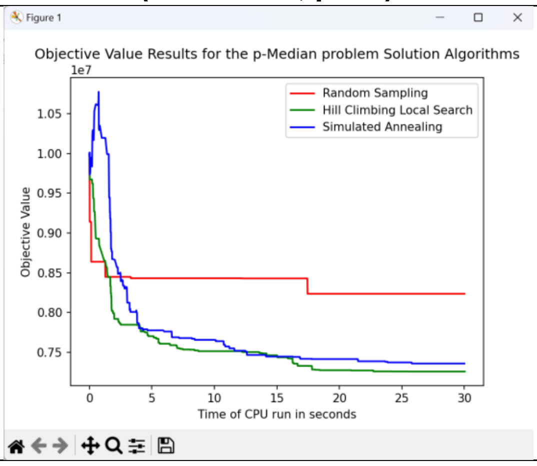
Instance 1 (n=m=100, p=15)	**Instance 2 (n=m=1000, p=30)**
	

(2.3.2) a plot showcasing the Total Distance (Y-Axis) vs Computing Time (X-Axis)

Instance 1 (n=m=100, p=15)	**Instance 2 (n=m=1000, p=30)**
	
Smallest Objective Value found: 1059094.5132397362	Smallest Objective Value found: 7326916.302433655

Extra Results:

Plotting a comparison of the changes in the Objective Value of the Optimisation Algorithms with respect to Computing Time

Instance 1 (n=m=100, p=15)	Instance 2 (n=m=1000, p=30)
 Figure 1 Objective Value Results for the p-Median problem Solution Algorithms 1e6 Random Sampling Hill Climbing Local Search Simulated Annealing Objective Value 1.45 1.40 1.35 1.30 1.25 1.20 1.15 1.10 1.05 0 5 10 15 20 25 30 Time of CPU run in seconds	 Figure 1 Objective Value Results for the p-Median problem Solution Algorithms 1e7 Random Sampling Hill Climbing Local Search Simulated Annealing Objective Value 1.05 1.00 0.95 0.90 0.85 0.80 0.75 0 5 10 15 20 25 30 Time of CPU run in seconds