SHARP
Metaheuristic Optimisation - Project Activity 1

Ong Jung Yi
ID: 1006655

October 2023

1 Introduction

Location planning is essential for a variety of sectors to ensure the strategic positioning of facilities such that they most efficiently serve operational demands. One popular approach is the p-median location problem. This model tries to find the best locations for facilities by minimizing the total distance or cost needed to serve all customers. Our report will explore the p-median problem in detail and discuss methods based on random sampling and local search to find solutions.

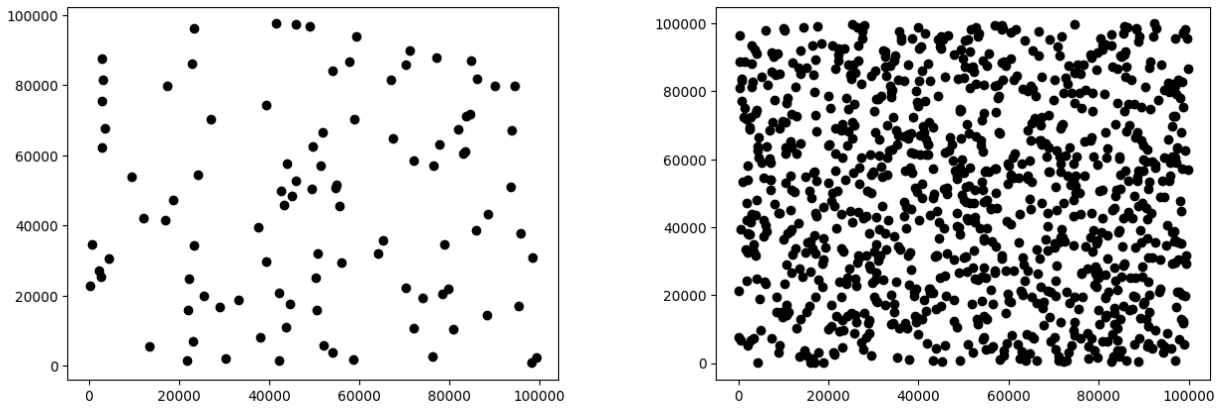
2 Problem Statement

Given a set of n locations representing demand points and a set of m potential facility locations, the challenge is to determine the optimal selection of p facilities from the potential set. This project aims to provide a computational approach to solve the problem using a random sampling algorithm and a local search algorithm.

For this project 2 instances will be investigated for both the Random Sampling Algorithm and the Local Search algorithm.

The first instance will consist of 100 city locations, 15 of which will be designated as open facilities.

The second instance will consist of 1000 city location, 30 of which will be designated as open facilities.



(a) Instance 1

(b) Instance 2

Figure 1: Problem representation of instance 1 and 2

3 Exercise 1: Random Sampling Algorithm

The Random Sampling Algorithm, as the name suggests, employs randomness to explore a solution space. By making randomized decisions at multiple steps, it provides a way to avoid being trapped in local optima and offers a broad exploration of potential solutions. In the context of the p-median problem, it involves randomly selecting facilities to be opened and then determining which cities they should serve.

3.1 Binary Solution Encoding Representation

Binary encoding solution representation uses a sequence of '0's and '1's to depict potential solutions to a problem, with each binary digit indicating the presence or absence of a specific feature or decision.

3.1.1 Facility Solution Encoding

Each potential open facility location is represented in binary format. Open facility location is marked with a *"1"* while City (closed facilities) locations is marked with a *"0"*.

For instance, with five potential facilities, if the second and fourth are chosen to be opened, the encoding will be 01010.

3.2 City-to-Facility Solution Encoding

For each opened facility (represented by a 1 in the facility solution encoding), another binary encoding determines which cities it serves. Cities being served by the facility are marked with 1, while those not being served are marked with 0.

If an open facility serves the first and third out of five cities, its city-serving encoding would be 00[10100]00.

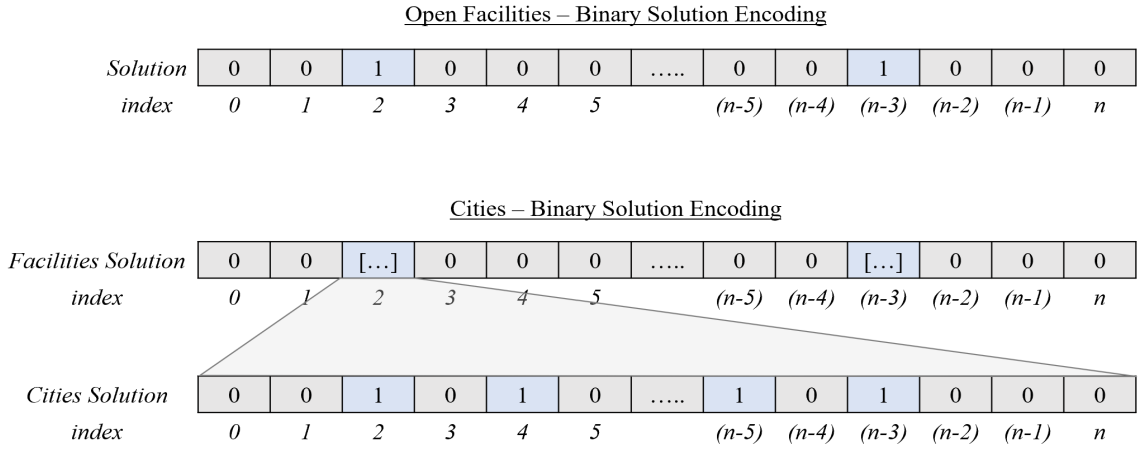


Figure 2: Instance 2

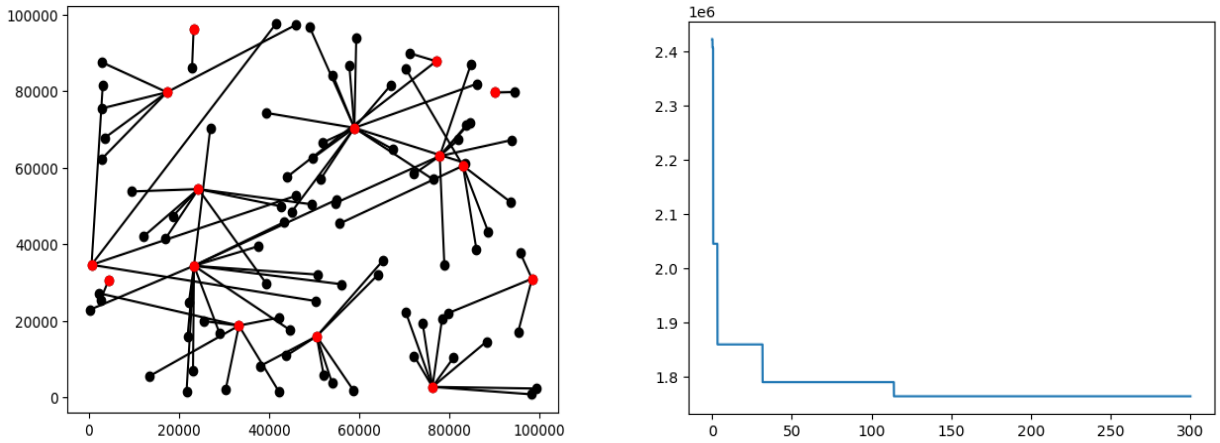
3.3 Search Procedure

1. n data points (representing cities) are randomly generated on a map. Out of these, p facilities are chosen to be opened.
2. The chosen facility points are removed from the list of remaining cities.
3. The remaining list is then sorted based on the straight-line distance from the center, prioritizing facilities closer to the center.
4. Iterating through each of the facilities:
   (a) A random number determines how many cities the opened facility will serve.
   (b) Based on this number, a random sample of cities is generated to be mapped to the facility. A normalized weighted selection ensures cities closer to the facility have a higher likelihood of being chosen.
   (c) These chosen cities are then removed from the list of cities left to be served.
5. The total distance, representing the sum of distances from facilities to their respective served cities, is computed.
6. If this calculated total distance is less than the previously recorded best solution, the best solution is updated.
7. The algorithm continues to run until a specified time limit is reached and displays the best solution

3.4 Instance 1 Solution: n=100, p=15

Using the Random Sampling Algorithm the optimised total distance was evaluated to be:

$$1764116.2357082907 \approx 1764116$$



(a) Visual representation of solution

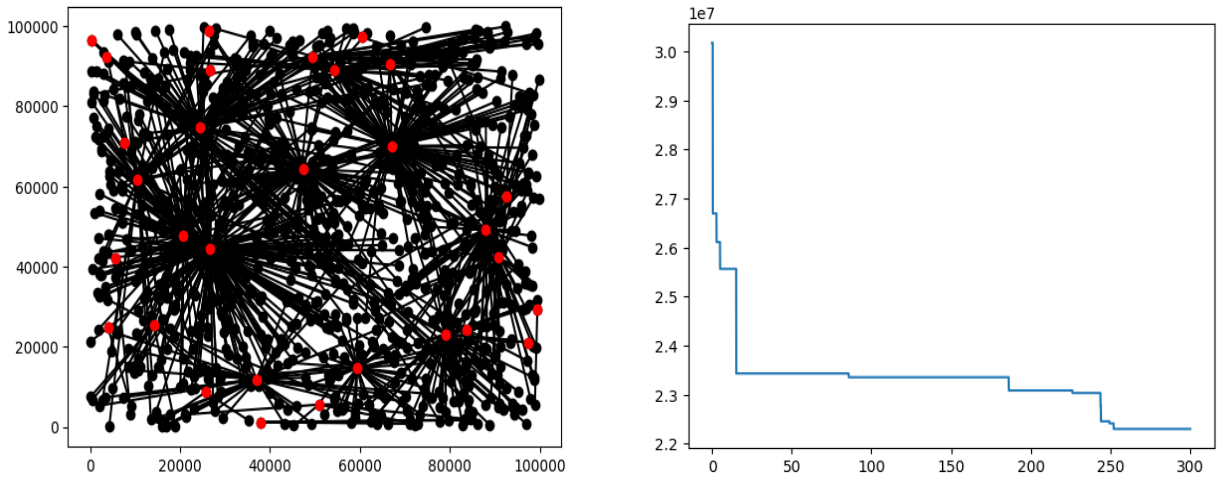
(b) CPU performance graph

Figure 3: Random Sampling Algorithm Solution optimised for Instance 1

3.5 Instance 2 Solution: n=1000, p=30

Using the Random Sampling Algorithm the optimised total distance was evaluated to be:

$$22298061.97677116 \approx 22298061$$



(a) Visual representation of solution

(b) CPU performance graph

Figure 4: Random Sampling Algorithm Solution optimised for Instance 2

4 Exercise 2: Local Search Algorithm

The Local Search Algorithm offers a systematic approach to refining solutions by exploring their immediate neighborhood. By making small, incremental changes, the algorithm searches for a better solution close to the current one. This iterative improvement strategy is particularly effective in increasing the performance of a solution.

4.1 Binary Solution Encoding

The encoding for the Local Search Algorithm bears resemblance to that of the Random Sampling Algorithm.

4.1.1 Facility Solution Encoding

Each potential open facility location is represented in binary format. Open facility location is marked with a *"1"* while City (closed facilities) locations is marked with a *"0"*.

For instance, with five potential facilities, if the second and fourth are chosen to be opened, the encoding will be 01010.

4.2 City-to-Facility Solution Encoding

For each opened facility (represented by a 1 in the facility solution encoding), another binary encoding determines which cities it serves. Cities being served by the facility are marked with 1, while those not being served are marked with 0.

If an open facility serves the first and third out of five cities, its city-serving encoding would be 00[10100]00.

5 Search Procedure

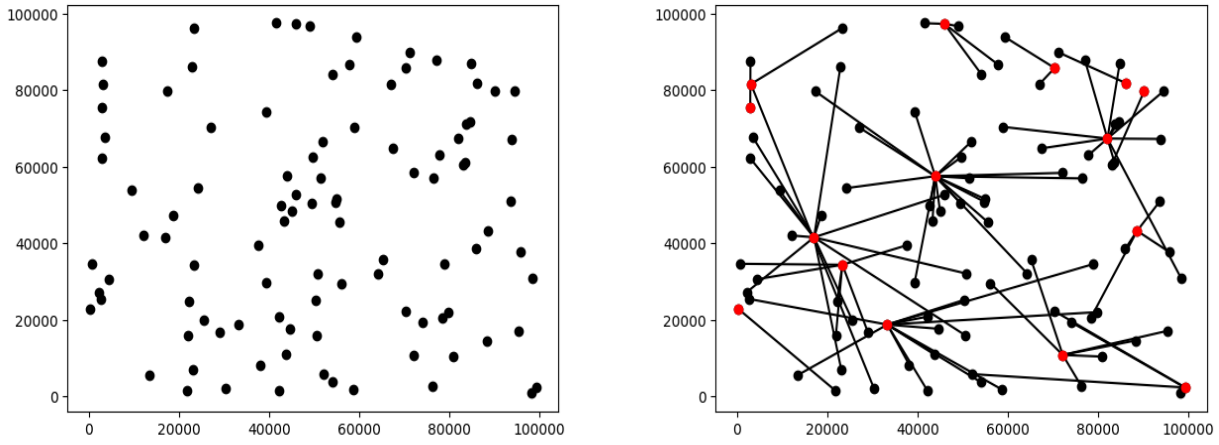
To begin the search procedure of the Local Search Algorithm, an initial solution is required. This could be obtained from a fixed method or a Random Sample Algorithm.

1. Start with an initial solution
2. Swap Function: Define a function to randomly swap the paths between two facility-to-city pairs. This aims to explore the neighborhood of the current solution for potential improvements.
3. Based on elapsed time, perform the following steps iteratively:
   (a) Based on elapsed time, perform the following steps iteratively:
   (b) Deep copy the initial solution to retain its original state.
   (c) Use the swap function to generate a new solution by swapping paths of two randomly chosen facility-to-city pairs.
   (d) Compute the total distance between each facility and its mapped cities in this new solution.
   (e) Compare the total distance of this new solution with that of the initial one.
   (f) If the new solution offers a lower total distance, replace the initial solution with this new one.
   (g) Repeat the process until the time limit is reached.

5.1 Instance 1 Solution: n=100, p=15

Using the Random Sampling Algorithm the program was able to find an initial solution of Total distance = 1715029.4537790015. By further implementing the Local Search Algorithm, the optimised total distance was evaluated to be:

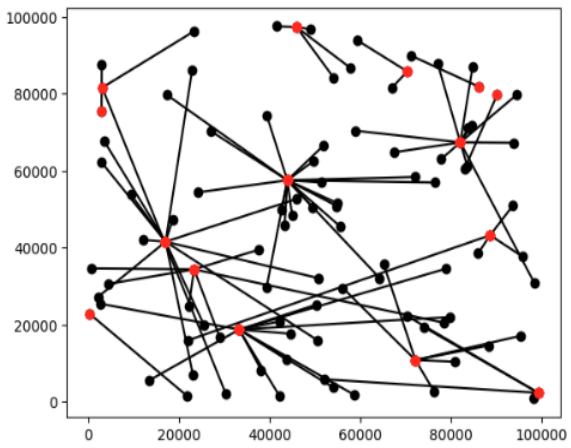
$$1598447.8997869375 \approx 1598447$$



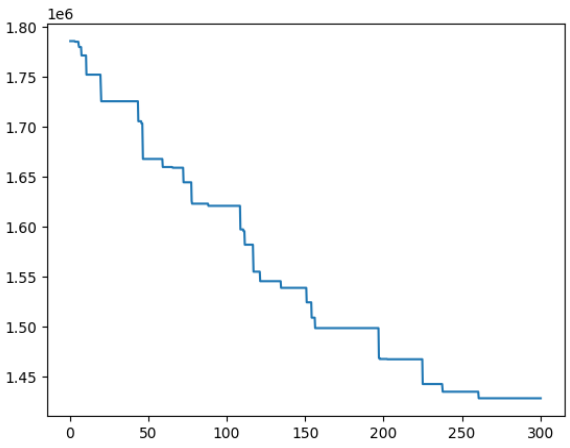
(a) Instance 1

(b) Initial Random Sampling Algorithm Solution

Figure 5: Part 1: Local Search Algorithm Solution optimised for Instance 1



(a) Visual representation of local search algorithm solution



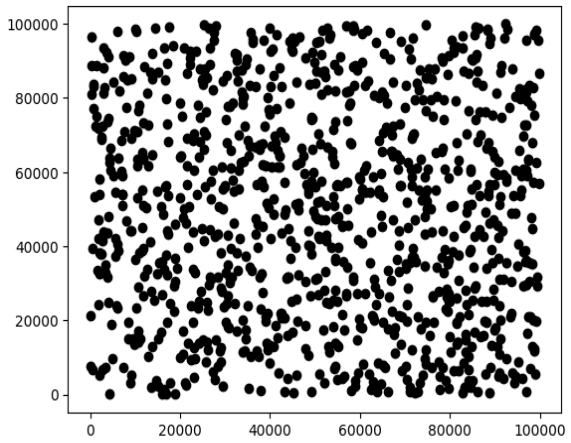
(b) CPU performance graph

Figure 6: Part 2: Local Search Algorithm Solution optimised for Instance 1

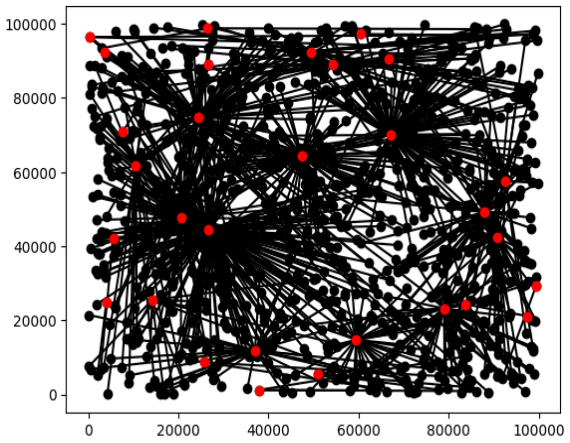
5.2 Instance 2 Solution: n=1000, p=30

Using the Random Sampling Algorithm the program was able to find an initial solution of Total distance = 22276292.04275504. By further implementing the Local Search Algorithm, the optimised total distance was evaluated to be:

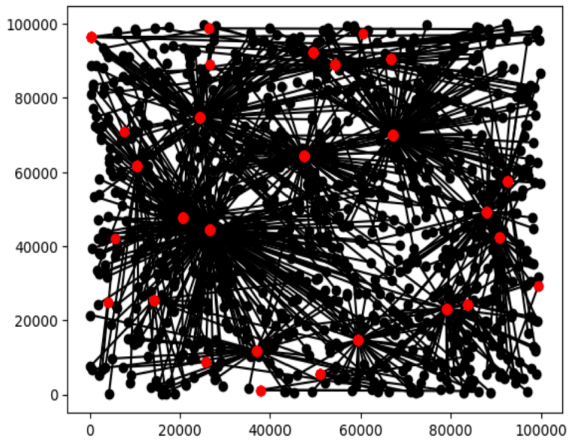
$$21994589.200536404 \approx 21994589$$



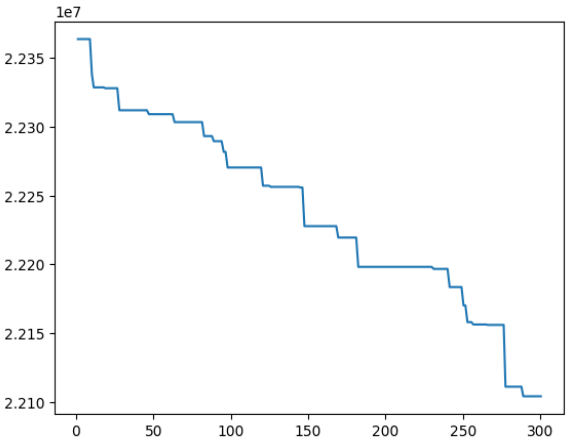
(a) Instance 2



(b) Initial Random Sampling Algorithm Solution



(c) Visual representation of local search algorithm solution



(d) CPU performance graph

Figure 7: Part 2: Local Search Algorithm Solution optimised for Instance 2