A Project report

on

“Facility Location Problem-Warehouse Location”

Submitted By,

Dasari Rishikesh	19BCS035
Shinde Rohit Sagar	19BCS099
Suryawanshi Randheer Ajit	19BCS104
Mane Param Dattatraya	19BCS121

Under the guidance of

Dr. B. Jayalakshmi

Asst. Professor

Computer Science & Engineering

Indian Institute of Information Technology, Dharwad

Certificate

This is to certify that the work contained in the project report titled "**Facility Location Problem-Warehouse Location"** by **Dasari Rishikesh, Shinde Rohit Sagar, Suryawanshi Randheer Ajit, and Mane Param Dattatraya** was completed during the VI semester - III Year as a Minor Project under the guidance of **Dr. B. Jayalakshmi**, Asst. Professor, CSE dept.

Signature of the Supervisor

Dr. B. Jayalakshmi

Asst. Professor

Computer Science & Engineering

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will cause disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Rishikesh	19bcs035
Rohit	19bcs099
Randheer	19bcs104
Param	19bcs121

Approval Sheet

This project report entitled **'Facility Location Problem-Warehouse Location'** by **Rishikesh (19bcs035), Rohit (19bcs099), Randheer (19bcs104), Param (19bcs121)** of Indian Institute of Information Technology, Dharwad is approved for the degree of Bachelor of Technology in Computer Science and Engineering.

Supervisor & Examiner

Dr. B. Jayalakshmi

Asst. Professor

Computer Science & Engineering

Abstract

This project elaborates on the “Facility Location Problem-Warehouse Location” from the perspective of three different algorithms such as greedy, local search, and tabu meta-heuristic algorithms. These methods/algorithms present solutions that are optimal in the perspective of cost efficiency and many other real-time constraints. We came up with a comparison of outputs of the above-mentioned algorithms with respect to the problem of facility location of warehouses providing different services in the prescribed area that is in reach of the considered facilities. In this facility location problem, we have considered an area with a different number of facility centers (warehouses) and the different number of customers as their in OR instances spread wide over the area. This solution to the Facility Location Problem can be implemented in many real-life applications such as emergency services, drone ports, delivery systems, etc.

Keywords*: facility location, Greedy algorithm, local search algorithm, tabu search metaheuristic algorithm.*

Table of Contents

Layout of the Project Report **08**

Chapter 1: Introduction **09-10**

1.1 Capacitated facility location problem 09-10

Chapter 2: Review of Literature **11**

Chapter 3: Problem Statement and Model Formulation **12-14**

3.1 Problem Statement 12

3.2 Model Formulation Hypothesis 12

3.3 Objective Function 12

3.4 Parameters for Model 13

3.5 Three models- SSCFLP, CFLP, UFLP 13

3.6 Dataset 14

Chapter 4: Algorithms **15-21**

4.1 Greedy Approach 15-16

4.1.1 Introduction to Greedy Algorithm 15

4.1.2 Greedy Algorithm Steps 15-16

4.1.3 Stopping Criteria of Algorithm 16

4.1.4 Advantages of Greedy Approach 16

4.1.5 Disadvantages of Greedy Approach 16

4.2 Local Search 17-18

4.2.1 Introduction to Local Search Algorithm 17

4.2.2 Local Search Algorithm Steps 17

4.2.3 Stopping Criteria of Algorithm 17

4.2.4 Advantages of Local Search 17-18

4.2.5 Disadvantages of Local Search 18

4.3 Tabu Search 19-21

4.3.1 Introduction to Tabu Search Algorithm 19

4.3.2 Tabu Search Algorithm Steps 20

4.3.3 Stopping Criteria of Algorithm 20

4.3.4 Flow Chart 21

4.3.5 Advantages of Tabu Search 21
4.3.6 Disadvantages of Tabu Search 21

Chapter 5: Execution of Algorithms and Results 22-31
5.1 Execution Greedy approach 22-24
5.1.1 How does the algorithm work in our model ? 22
5.1.2 Stopping Criteria 23-24
5.2 Execution for Local Search 24-26
5.2.1 How does the algorithm work in our model ? 24-25
5.2.2 Stopping Criteria 25-26
5.3 Execution for Tabu Search 26-27
5.3.1 How does the algorithm work in our model ? 26
5.3.2 Stopping Criteria 27
5.4 Results 28-31
5.4.1 Final Mean Result Table 28-29
5.4.2 Calculations 30-31

Summary 32

Conclusion 33

References 34

Acknowledgement 35

Layout of the Project Report

Chapter 1: Introduction

Facility location Problem is analyzing the locations for optimal placement of facilities reducing the costs and satisfying the customer demands. The Facility location is a vast area with numerous applications in the public and private sectors. The facility location problem has a wide range of applications in the industrial sector like warehouses. It includes Capacitated Facility Location problem and Uncapacitated Facility Location Problem.

Chapter 2: Literature Review

The Literature Review includes a few previous studies on the facility location problem. These papers included the heuristic techniques for the different problems. By understanding the research papers, the project is implemented and executed.

Chapter 3: Problem Statement and Model Formulation

The Single Source Capacitated Facility Location Problem is built for facilitating the warehouses in a location. It will be built, ensuring that minimizing the total cost, and reaching customer demands. OR-Library is used for source datasets.

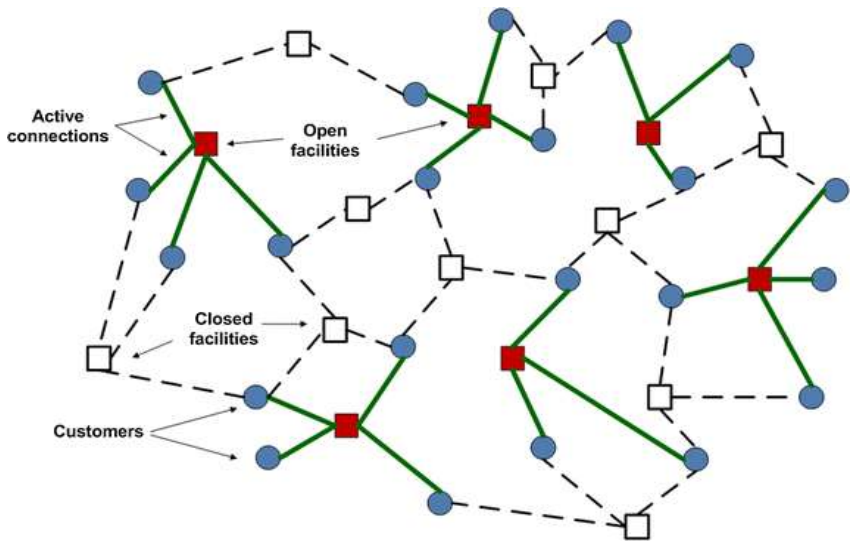
Chapter 4: Algorithms

For solving this facility location problem the heuristic and metaheuristic techniques were used. They include the Greedy Approach, Local Search, and Tabu Search. The Greedy approach takes the current best option available. Local Search uses the swapping procedure over the facilities. Tabu Search is a guided local search algorithm that explores the search space beyond the local solution.

Chapter 5: Execution of Algorithms

Steps of executing all the three algorithms in the project are mentioned. The algorithms that intake the dataset inputs from OR-Library, executes the respective results of each. The results were giving accuracy over the algorithms.

Chapter 1: Introduction



The Fermat-Weber problem, formulated in the 17th century, was one of the first devised plant position problems. The Fermat-Weber problem can be described as Find the fourth point using the three points on the plane and the sum of the distances to the given three points is the minimum. This problem can be thought of as a variant of the object location problem, which assumes that the cost of transport per distance is the same for all destinations.

Object placement study, also known as positional analysis, is a branch of computational research and computational geometry that relates to optimal placement of objects to minimize resources. In most cases, the resource minimized is total cost. There are different areas where facility location problems could be considered. For this project, a warehouse as a facility. Facility location is a classic optimization problem that determines the simplest location for a plant or warehouse to be supported geographical demands, facility setup prices, and transportation distances. The two types of location problem here are, capacitated and uncapacitated facility location.

1.1. Capacitated facility location problem

Capacitated Facility Location Problem (CFLP) is a variant of the FLP, which includes capacities for the facilities unlike UCFLP, which have unlimited capacity. First, we need to fixed the number of potential sites where one can open the facility and number of demand points i.e., customers which vary according to the instances in OR-library, here no need to stick with the same numbers as there is a chance of biases towards a particular instance and respective algorithm. Secondly the fixed cost associated with each facility here warehouse to open and respective

capacities of the warehouses that are also different in different instances because of the same reason. Furthermore in this project demand points are directly customers which have their own demand i.e. their requirements which should be fulfilled by one assigned facility, and a customer is not assigning to multiple facilities at the same time i.e., model here built up is based on a single-source capacitated location problem where each customer served from a single facility. At last the comparison between 3 different outputs through 3 algorithms with predefined optimum value in OR-library is performed.

In this project, the main focus is on finding the best optimal algorithm in the capacitated facility. For that three algorithms are used namely greedy, local search and tabu-meta heuristic algorithm. Project is not focusing on any study on the uncapacitated facility location problem as there are not any constraints on the capacity of warehouses and also not any ideal warehouse exists in reality which has unlimited capacity.

Facility location issues apply to a variety of industries. For supply chain management and logistics, you can use this issue to find the best location for stores, factories, warehouses, and so on. Other applications range from public order and morals (e.g. placing police officers in cities) to telecommunications (e.g. cell towers in the network). And particle physics (e.g. separation distance between repulsive charges). Another use of the plant location problem is to locate natural gas transport equipment. Finally, the facility location issue can also be applied to cluster analysis.

Chapter 2: Literature Review

Referred to a few literature that was published before on this topic. Those papers implemented the various heuristic and meta heuristic algorithms and compared them to get the optimal algorithm for the facility location problem with the parameter of total cost as the central idea. The literatures are:

1. Facility location: Greedy and Local search algorithms by Anupam (CMU, USA).

2. A tabu search heuristic procedure for CFLP by Minghe Sun (University of Texas).

3. Metaheuristics for solving facility location optimization problems by ASTES journal.

4. Efficient solution of large scale, single-source, capacitated plant location problem by the journal of OR society.

Chapter 3: Problem Statement and Model Formulation

3.1. Problem Statement

Say a large supermarket chain needs to build warehouses but the locations of the warehouses have yet to be determined. Also opening many warehouses would be advantageous as this would reduce the average distance a vehicle has to drive from the warehouse to the customer, and hence reduce the delivery cost. However, opening a warehouse includes a fixed cost related to it. By deciding which facility/warehouse to open and, for each customer, who serves it, determine the minimum cost solution that satisfies all customers' demands while respecting the capacities of the facilities in CFL, in the hypothesis of each customer being served by a single facility.

3.2. Model Formulation Hypothesis

1. While doing model formulation we took this into consideration like, there are fixed costs for opening the plants and transportation costs to supply the commodity or the standard product mix from potential location sites to clients.
2. The CFLP considers a situation within which the plants have a capacity expressed in units of demand and additionally assumes that every shopper will be served from completely different open plants and purchasers even have their own demand.
3. If each customer served from a single facility then its single source capacitated facility location (ssc flp) problem & assumption is, transportation costs from facilities to markets are linear according to the quantity transported.
4. Used Java Development Kit (JDK) as a development environment.

3.3. Objective Function

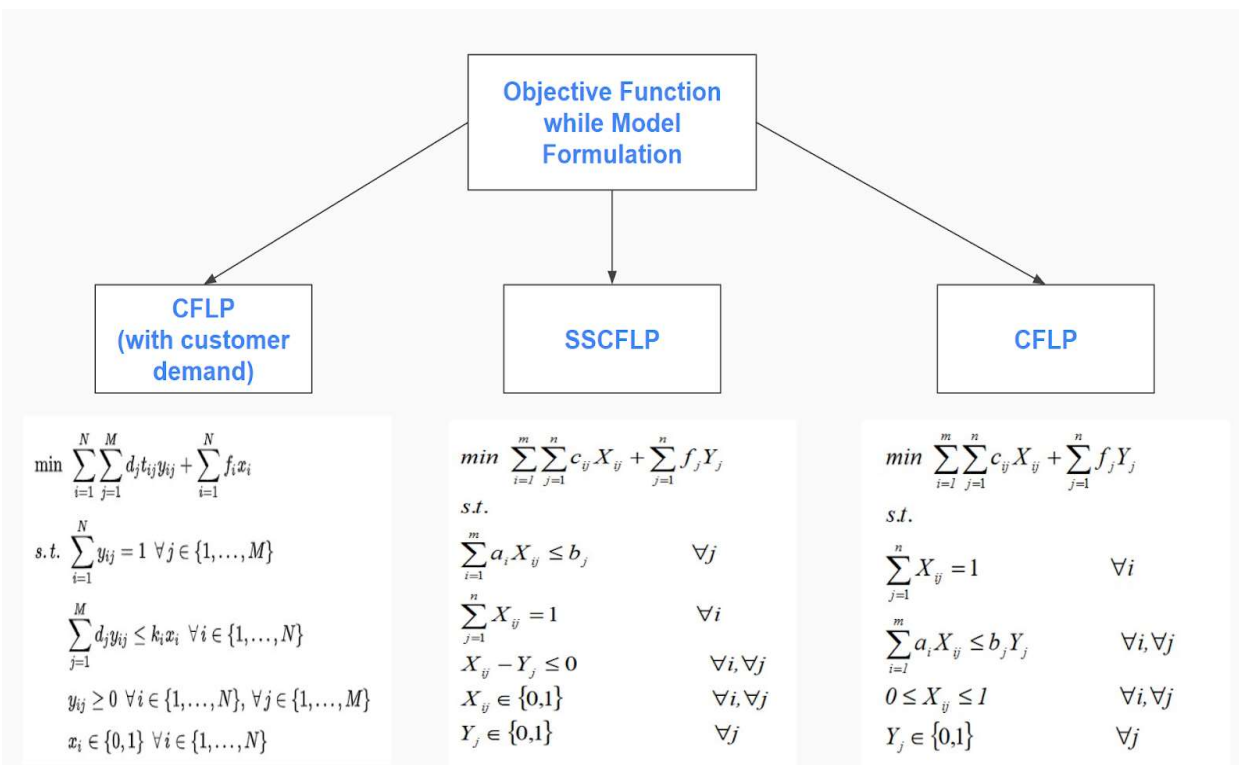
1. Minimizes the cost of assigning customers to open facilities and the cost of establishing such facilities.
2. Capacity constraints in CFLP (or the facility constraint), ensures that the customer demand served by a certain facility does not exceed its capacity.
3. Demand constraints (or the customer constraints), ensures that each customer is assigned to exactly one facility and demand should be fulfilled.
4. Assignments are made only to open facilities.
5. All decision variables are binary.

3.4. Parameters for Model

1. n potential sites where a facility that provide a service could be installed: J = {1, .., n}
2. The fixed cost of opening each facility j: F = {f1, .., fn}
3. The capacity of each facility j (i.e. the maximum amount of service that can be delivered by the facility): S = {s1, .., sn}
4. m customers: I = {1, .., m}
5. The service demand of each customer i: D = {d1, .., dm}
6. The cost of allocating all the demand of customer i to facility j: c_ji for each (j=1,...,|J|), (i=1,...,|I|)

3.5. Three Models

Used Single Source Capacitated Facility Location Approach (SSCFL)

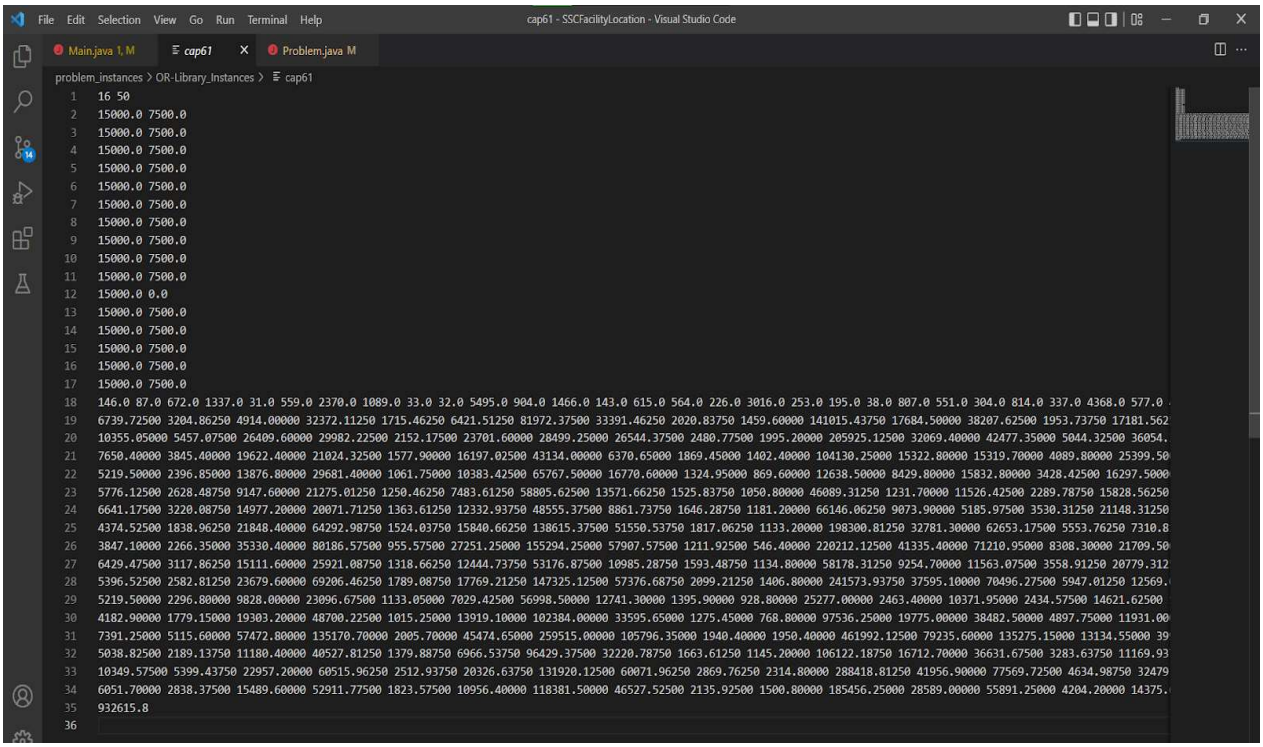


3.6. Dataset

OR-library 24 instances.

These data files are set OR-Library in Guastaroba G. and Speranza M.G. A Heuristic for BILP Problems: The Single Source Capacitated Facility Location Problem, currently under review in the European Journal of Operational Research.

E.g. cap61 instance is :




```
16 50
15000.0 7500.0
15000.0 7500.0
15000.0 7500.0
15000.0 7500.0
15000.0 7500.0
15000.0 7500.0
15000.0 7500.0
15000.0 7500.0
15000.0 7500.0
15000.0 7500.0
15000.0 0.0
15000.0 7500.0
15000.0 7500.0
15000.0 7500.0
15000.0 7500.0
15000.0 7500.0
146.0 87.0 672.0 1337.0 31.0 559.0 2370.0 1089.0 33.0 32.0 5495.0 904.0 1466.0 143.0 615.0 564.0 226.0 3016.0 253.0 195.0 38.0 807.0 551.0 304.0 814.0 337.0 4368.0 577.0
6739.72500 3204.86250 4914.00000 32372.11250 1715.46250 6421.51250 81972.37500 33391.46250 2020.83750 1459.60000 141015.43750 17684.50000 38207.62500 1953.73750 17181.562
10355.05000 5457.07500 26409.60000 29982.22500 2152.17500 23701.60000 28499.25000 26544.37500 2480.77500 1995.20000 205925.12500 32069.40000 42477.35000 5044.32500 36054.
7650.40000 3845.40000 19622.40000 21024.32500 1577.90000 16197.02500 43134.00000 6370.65000 1869.45000 1402.40000 104130.25000 15322.80000 15319.70000 4089.80000 25399.50
5219.50000 2396.85000 13876.80000 29681.40000 1061.75000 10383.42500 65767.50000 16770.60000 1324.95000 869.60000 12638.50000 8429.80000 15832.80000 3428.42500 16297.5000
5776.12500 2628.48750 9147.60000 21275.01250 1250.46250 7483.61250 58805.62500 13571.66250 1525.83750 1050.80000 46089.31250 1231.70000 11526.42500 2289.78750 15828.56250
6641.17500 3220.08750 14977.20000 20071.71250 1363.61250 12332.93750 48555.37500 8861.73750 1646.28750 1181.20000 66146.06250 9073.90000 5185.97500 3530.31250 21148.31250
4374.52500 1838.96250 21848.40000 64292.98750 1524.03750 15840.66250 138615.37500 51550.53750 1817.06250 1133.20000 198300.81250 32781.30000 62653.17500 5553.76250 7310.8
3847.10000 2266.35000 35330.40000 80186.57500 955.57500 27251.25000 155294.25000 57907.57500 1211.92500 546.40000 220212.12500 41335.40000 71210.95000 8308.30000 21709.50
6429.47500 3117.86250 15111.60000 25921.08750 1318.66250 12444.73750 53176.87500 10985.28750 1593.48750 1134.80000 58178.31250 9254.70000 11563.07500 3558.91250 20779.312
5396.52500 2582.81250 23679.60000 69206.46250 1789.08750 17769.21250 147325.12500 57376.68750 2099.21250 1406.80000 241573.93750 37595.10000 70496.27500 5947.01250 12569.
5219.50000 2296.80000 9828.00000 23096.67500 1133.05000 7029.42500 56998.50000 12741.30000 1395.90000 928.80000 25277.00000 2463.40000 10371.95000 2434.57500 14621.62500
4182.90000 1779.15000 19303.20000 48700.22500 1015.25000 13919.10000 102384.00000 33595.65000 1275.45000 768.80000 97536.25000 19775.00000 38482.50000 4897.75000 11931.00
7391.25000 5115.60000 57472.80000 135170.70000 2005.70000 45474.65000 259515.00000 105796.35000 1940.40000 1950.40000 461992.12500 79235.60000 135275.15000 13134.55000 39
5038.82500 2189.13750 11180.40000 40527.81250 1379.88750 6966.53750 96429.37500 32220.78750 1663.61250 1145.20000 106122.18750 16712.70000 36631.67500 3283.63750 11169.93
10349.57500 5399.43750 22957.20000 60515.96250 2512.93750 20326.63750 131920.12500 60071.96250 2869.76250 2314.80000 288418.81250 41956.90000 77569.72500 4634.98750 32479
6051.70000 2838.37500 15489.60000 52911.77500 1823.57500 10956.40000 118381.50000 46527.52500 2135.92500 1500.80000 185456.25000 28589.00000 55891.25000 4204.20000 14375.
932615.8
```

Chapter 4: Algorithms

4.1 Greedy Algorithm

4.1.1. Introduction to Greedy Algorithm

A greedy algorithm is an approach to solving a problem by choosing the best option currently available. Don't care whether the current best results will lead to the overall optimal results. The algorithm never over-turns a previous decision, even if the choice is wrong. It works top-down. This algorithm may not give the best results for all operations. It's as a result of it continuously goes for the native best option to provide the global best result. However, we can determine if the algorithm is often used with any problem if the problem has the subsequent properties:

1. Greedy Choice Property:

If an optimal solution to the problem will be found by choosing the best choice at each step without reconsidering the previous steps once chosen, the problem is often solved employing a greedy approach. This property is termed greedy choice property.

2. Optimal substructure:

If the optimal general solution of the problem corresponds to the optimal solution of the subproblem, then a greedy approach can be used to solve the problem. This property is called the optimal substructure.

4.1.2. Greedy Algorithm Steps

let C = number of serving candidates in a facility

1. Initialize an empty solution set, Solution = {}

2. While (C is not empty):

```
 # step 1: select associate item from C

        current_choice = Select(C)

# step 2: if current_choice is feasible, add to Solution

   if(isFeasable(current_choice)):

       Solution.add(C)

   # step 3: remove the current_choice from C

   C.remove(current_choice)

3. Return Solution
```

4.1.3. Stopping Criteria of Algorithm

When C (list of a number of candidates) gets empty (Empty means each candidate gets served), the algorithm gets terminated.

4.1.4. Advantages of Greedy Approach

1. All greedy algorithms are very intuitive and maybe guessed simply.
2. Greedy algorithms are usually simple to grasp moreover as implement.
3. If correct, greedy algorithms realize the optimal solution a lot quicker moreover lot with efficiency than its alternatives, like dynamic programming.

4.1.5. Disadvantages of Greedy Approach

1. Though very intuitive, it's terribly laborious to prove mathematically the correctness of a greedy algorithm to unravel the given problem at hand.
2. A greedy approach to unravel optimization problems might not continuously be correct, could even cause the worst possible solution.

4.2. Local Search Algorithm

4.2.1. Introduction to Local Search Algorithm

Local search is a heuristic method used for solving problems that are computationally hard to solve. Local search may be used on problems which will be formulated as finding a solution maximizing a criterion among a variety of candidate solutions. This algorithm moves from solution to solution in the space of candidate solutions by applying local changes, until a solution deemed optimal is found or a time-bound is elapsed.

4.2.2. Local search Algorithm Steps

1. S ← an arbitrary feasible solution in S.
2. While ∃ S ∈ B(S) such that cost(S)
3. return s

Procedure Partition in algorithm**;**

i = 0 while ∃ a bad facility in S do
1.i = i + 1 {iteration i}
2. Ai ← {b} where b ∈ S is any bad facility
3. Bi ← capture(Ai)
4. while |Ai| = |Bi|
do 4.1. Ai ← Ai ∪ {g} where g ∈ S \ Ai is any good facility
4.2. Bi ← capture(Ai)
5. S ← S \ Ai O ← O \ Bi Ar ← S Br ← O

4.2.3. Stopping Criteria of Algorithm

If a profitable exchange is found then the algorithm makes the candidates present within the local space swaps, opening new facilities if necessary, producing a replacement better solution. Otherwise, it terminates.

4.2.4. Advantages of Local Search

1. Problems with memory constraints,

    a. Local search algorithms keep just one current state in memory (or a set range of states, if using algorithms like local beam search and genetic algorithm)
    b. However, path-constructing algorithms typically take exponential order of memory for gaint search trees

2. problems with changes in state space, for instance, local search,
    a. Local search refers to searching dynamically. In alternative words, the agent is looking in associate in nursing evolving setting.
    b. If, for instance, A* is adopted, by the time it gives out a solution for a particular state, the environment has already changed to a state where the solution can not apply.
    c. However, local search algorithms can unceasingly modify its solution in keeping with the ever-changing setting as they run.
    d. One other advantage of local search is that once a search is complete, the algorithm will still be invoked at a later time once the setting has modified thus on optimize the previous solution within the new setting. Apparently, this can be unattainable for path-constructing algorithms. + Problems which we only care about results instead of the process.

4.2.5. Disadvantages of Local Search Algorithm

1. Some neighborhoods will become very gaint (time overwhelming to look at all the neighbors)
2. The search stops once no improvement may be found: local optimum
3. Restarting the search may facilitate, however, is commonly not very effective in itself – want for a technique!

4.3. Tabu Search Algorithm

4.3.1. Introduction to Tabu Search Algorithm

Tabu Search is a meta-heuristic technique that guides a local heuristic search procedure and it explores the solution space beyond the local optimum by using a tabu list.

4.3.1.1. What is the tabu list and its role in the algorithm

The tabu list is a set of solutions that are not to be used anymore. In a sense, they are forbidden. It uses the flexible memory structure in conjunction with strategic restrictions and aspirations levels. The kind of problem it solves is of combinatorial nature obviously with finite solution space. Its neighborhood search method. It means once we have a point it searches around its neighbor and there is a flexible memory structure.

First of all, it is a primary way to exploit memory in the tabu search by classifying a subset of the moves in a neighborhood as forbidden or tabu. The neighborhood is constructed to identify adjustment solutions that can be reached from the current solution. The classification is based on the search history and the frequency that certain moves or solutions components are called attributes that have participated in generating solutions. Tabu list records prohibited moves which are referred to as tabu moves. The aspiration criteria provide an exception to tabu restrictions when a tabu move has a sufficiently attractive evaluation where it would result in a solution better than any visited so far then its tabu classification is overridden.

4.3.1.2. Strategies in tabu search based on the tabu list

1. Forbidden strategy
2. Freeing strategy
3. Short-term strategy

Forbidden Strategy: It controls what enters the tabu list. This is what goes into the tabu list.

Freeing Strategy: It controls what exits from the tabu list. Under which conditions do we get take out from the tabu list.

Short-term Strategy: It manages the interplay between the forbidding strategy and the freeing strategy to select trial solutions.

So now we have a question: what goes into the tabu list, and what comes out of the tabu list?

Let us suppose have an **Example** which we will give an overview of tabu list,

We have a problem i.e, the problem of traveling salesmen where a person goes from one city to all other cities and comes back to the same city. Suppose there is a 5 city problem. The person has to go to all 5 cities. Assume that,

2	1	5	4	3

is the current solution. Here we are minimizing the distance and current distance D =50. Assume I interchange 1and 4 and the solution obtained is

2	4	5	1	3

is a solution which is a neighboring solution. Let's say the improvement we got in the distance is 55. By interchanging 1 and 4 we got something better if we again interchange then it will become worse. So then we put a restriction on the Tabu list for a certain period. So, for that amount of time, the Tabu list won't switch places. But at later stages, it might so happen that we have another current solution which maybe

3	4	2	1	5

better than the previous one.
Sometimes times with these interchanges may actually get even better solutions.

4.3.2. Tabu Search Algorithm Steps

Step-1: Choose an initial solution i in S. Set i* = i and k=0.

Step-2: Set k=k+1 and generate a subset V* of solution in N(i,k) such that either one of the Tabu conditions is violated or a minimum one in all the aspiration conditions holds.

Step-3: Choose a best j in V* and set i=j.

Step-4: If f(i) <f(i*) then set i*=i.

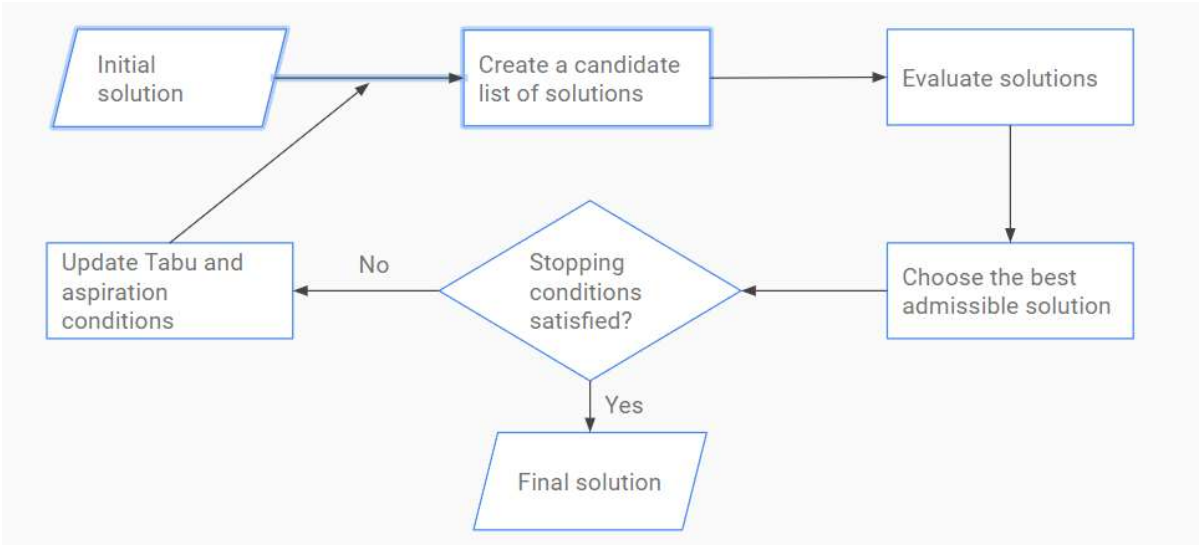
Step-5: Update Tabu and aspiration conditions.

Step-6: If a stopping condition is met then stop. Else go to Step 2.

4.3.3. Stopping Criteria of Algorithm

1. N(i, K+1) = 0. (no possible solution within the neighborhood of solution i).
2. K is larger than the most variety of iterations allowed.
3. The variety of iterations since the last improvement of i* is larger than a specified number.
4. There is proof that an optimum solution has been obtained.

4.3.4. Flow Chart



4.3.5. Advantages of Tabu-Search

1. Accepts non-improving solutions so as to escape from a local optimum.
2. Can be applied to each separate and continuous solution spaces.
3. Can be used for complex problems on scheduling, quadratic assignment, and vehicle routing. It obtains competitive solutions vis-à-vis other similar approaches.

4.3.6. Disadvantages of Tabu-Search

1. Too many parameters to be determined.
2. Number of iterations could be very large.
3. Global optimum may not be found, depending on parameter settings.

Chapter 5: Execution Of Algorithms and Results

This chapter elaborates on the execution of three algorithms and their respective outputs. Greedy algorithm, local search algorithm and meta-heuristic algorithm Tabu are used to get optimal solutions for facility location problems.

Data Set that we Extracted from cap61 OR-library:

```
Facilities: 16 - Customers: 50
Facility Capacities: 15000.0 15000.0 15000.0 15000.0 15000.0 15000.0 15000.0 15000.0 15000.0 15000.0 15000.0 15000.0 15000.0 15000.0 15000.0 15000.0
Facility Fixed Costs: 7500.0 7500.0 7500.0 7500.0 7500.0 7500.0 7500.0 7500.0 7500.0 7500.0 7500.0 0.0 7500.0 7500.0 7500.0 7500.0 7500.0
Customer Demands: 146.0 87.0 672.0 1337.0 31.0 559.0 2370.0 1089.0 33.0 32.0 5495.0 904.0 1466.0 143.0 615.0 564.0 226.0 3016.0 253.0 195.0 38.0 807.0 551.0 304.0 81
4.0 337.0 4368.0 577.0 482.0 495.0 231.0 322.0 685.0 12912.0 325.0 366.0 3671.0 2213.0 705.0 328.0 1681.0 1117.0 275.0 500.0 2241.0 733.0 222.0 49.0 1464.0 222.0
```

5.1. Execution for Greedy Algorithm

The greedy algorithm at each step opens the facility that can serve all the not served customers left at the minimum cost (even considering the fixed cost of opening). After that, it iterates over all these not served customers sorted by increasing serving cost (relatively to the just-opened facility) and it assigns them to this facility, one by one only if the facility has the necessary capacity left. The algorithm terminates after all the customers are served. This algorithm gives us the first rough solution.

5.1.1. How does the algorithm work in our model?

Step1: Calculates the total service/transportation cost associated with each facility to all customers and choose that facility which has less total service cost, where Total cost = Transportation/Service cost associated with each customer w.r.t. Their assigned facility + Fixed cost of opening all chosen facilities.

Step2: Again it checks the lower service cost associated with the customer w.r.t. The chosen facility extracts the demand of that customer.

Step3: Now demand of that customer gets minused from the capacity of chosen facility (Dataset predefined the capacity of individual facility).

Step4: After exhaustion of capacity of chosen facility, next facility which has less total service cost gets selected and Step2,3 repeated. Gives first rough solution and in coming

algorithms we reduced this error.

5.1.2. Stopping Criteria

When each customer gets served from a single facility, our greedy algorithm gets terminated.

```
Press ENTER to apply the greedy algorithm...
Greedy solution:
Opening facility 10 with service cost for all customers: 1248142.9
Assigning customer 22 to facility 10 - capacity left: 14449.0
Assigning customer 20 to facility 10 - capacity left: 14411.0
Assigning customer 9 to facility 10 - capacity left: 14379.0
Assigning customer 4 to facility 10 - capacity left: 14348.0
Assigning customer 8 to facility 10 - capacity left: 14315.0
Assigning customer 27 to facility 10 - capacity left: 13738.0
Assigning customer 47 to facility 10 - capacity left: 13689.0
Assigning customer 1 to facility 10 - capacity left: 13602.0
Assigning customer 31 to facility 10 - capacity left: 13280.0
Assigning customer 13 to facility 10 - capacity left: 13137.0
Assigning customer 11 to facility 10 - capacity left: 12233.0
Assigning customer 16 to facility 10 - capacity left: 12007.0
Assigning customer 25 to facility 10 - capacity left: 11670.0
Assigning customer 49 to facility 10 - capacity left: 11448.0
Assigning customer 18 to facility 10 - capacity left: 11195.0
Assigning customer 30 to facility 10 - capacity left: 10964.0
Assigning customer 0 to facility 10 - capacity left: 10818.0
Assigning customer 39 to facility 10 - capacity left: 10490.0
Assigning customer 28 to facility 10 - capacity left: 10008.0
Assigning customer 35 to facility 10 - capacity left: 9642.0
Assigning customer 34 to facility 10 - capacity left: 9317.0
Assigning customer 5 to facility 10 - capacity left: 8758.0
Assigning customer 23 to facility 10 - capacity left: 8454.0
Assigning customer 19 to facility 10 - capacity left: 8259.0
Assigning customer 2 to facility 10 - capacity left: 7587.0
Assigning customer 40 to facility 10 - capacity left: 5906.0
Assigning customer 12 to facility 10 - capacity left: 4440.0
Assigning customer 24 to facility 10 - capacity left: 3626.0
Assigning customer 46 to facility 10 - capacity left: 3404.0
Assigning customer 42 to facility 10 - capacity left: 3129.0
Assigning customer 7 to facility 10 - capacity left: 2040.0
Assigning customer 43 to facility 10 - capacity left: 1540.0
Assigning customer 14 to facility 10 - capacity left: 925.0
Assigning customer 32 to facility 10 - capacity left: 240.0
No more customers assignable to facility 10
```

```
Opened Facilities:
3 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 36 (Demand: 3671.0 - Cost: 83331.7), 44 (Demand: 2241.0 - Cost: 131938.88), 10 (Demand: 5495.0 - Cost: 12638.5),
4 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 29 (Demand: 495.0 - Cost: 15598.6875), 41 (Demand: 1117.0 - Cost: 20427.137), 45 (Demand: 733.0 - Cost: 45381.863),
 21 (Demand: 807.0 - Cost: 23493.787), 37 (Demand: 2213.0 - Cost: 20497.912), 48 (Demand: 1464.0 - Cost: 21905.1), 3 (Demand: 1337.0 - Cost: 21275.012), 38 (Demand: 705.0 - Cost
: 44494.312), 17 (Demand: 3016.0 - Cost: 23562.5), 15 (Demand: 564.0 - Cost: 18421.65), 6 (Demand: 2370.0 - Cost: 58805.625),
5 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 33 (Demand: 12912.0 - Cost: 229995.0),
8 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 26 (Demand: 4368.0 - Cost: 289434.6),
10 (Capacity: 15000.0 - Fixed Cost: 0.0) serving customers: 12 (Demand: 1466.0 - Cost: 10371.95), 30 (Demand: 231.0 - Cost: 4440.975), 4 (Demand: 31.0 - Cost: 1133.05), 35 (Dema
nd: 366.0 - Cost: 5984.1), 32 (Demand: 685.0 - Cost: 14693.25), 40 (Demand: 1681.0 - Cost: 10086.0), 43 (Demand: 500.0 - Cost: 14550.0), 19 (Demand: 195.0 - Cost: 7312.5), 46 (D
emand: 222.0 - Cost: 11183.25), 1 (Demand: 87.0 - Cost: 2296.8), 18 (Demand: 253.0 - Cost: 4123.9), 23 (Demand: 304.0 - Cost: 7106.0), 7 (Demand: 1089.0 - Cost: 12741.3), 16 (De
mand: 226.0 - Cost: 3062.3), 0 (Demand: 146.0 - Cost: 5219.5), 28 (Demand: 482.0 - Cost: 5916.55), 27 (Demand: 577.0 - Cost: 1601.175), 5 (Demand: 559.0 - Cost: 7029.425), 2 (De
mand: 672.0 - Cost: 9828.0), 9 (Demand: 32.0 - Cost: 928.8), 13 (Demand: 143.0 - Cost: 2434.575), 39 (Demand: 328.0 - Cost: 5658.0), 42 (Demand: 275.0 - Cost: 11550.0), 25 (Dema
nd: 337.0 - Cost: 3268.9), 22 (Demand: 551.0 - Cost: 0.0), 24 (Demand: 814.0 - Cost: 10541.3), 31 (Demand: 322.0 - Cost: 2398.9), 14 (Demand: 615.0 - Cost: 14621.625), 49 (Deman
d: 222.0 - Cost: 4001.55), 11 (Demand: 904.0 - Cost: 2463.4), 20 (Demand: 38.0 - Cost: 780.9), 47 (Demand: 49.0 - Cost: 2174.375), 8 (Demand: 33.0 - Cost: 1395.9), 34 (Demand: 3
25.0 - Cost: 6353.75),

TOTAL COST = 1298453.8
OPTIMUM VALUE = 932615.8 - RELATIVE ERROR = 39.227077%
```

5.2. Execution for Local Search Algorithm

Starting with the solution found by the greedy algorithm, the local search builds an improvement graph and makes use of it to find single customer exchanges between facilities (opened and closed) which will lower the cost of the solution. If a profitable exchange is found then the algorithm makes the customers swaps, opening new facilities if necessary, producing a new better solution. Otherwise it terminates.

5.2.1. How does the algorithm work in our model ?

Step1: Selecting customers from opened facilities and swap to another facility whose capacity is available in such a way that after swap we get total service cost minimum than the candidate solution's total service cost.

New Facilities		1	2	3	4
Customer	Old Facilities	Saving cost after swap			
1	3	100	451	0	400
2	4	-20	12	124	0
3	2	74	0	713	-14
4	1	0	219	141	77
5	4	80	-111	398	0
6	2	-78	0	-47	200
7	3	12	69	0	222
8	1	0	27	341	47
9	1	0	741	10	104
10	4	870	-20	85	0

(Table for Cap61 OR_library DataSet)

In the above table we can see, customer 1 for the previous iteration was present in the 3rd facility but after running the next iteration it moves to the 1st facility with a saving cost of 100. That's how local search checks for each customer swap to another facility and chooses the best possible swap so that we get less total cost in the next iteration.

Step2: In iteration 2, iteration 1 is a candidate solution, repeat the step1 for the upcoming iteration till that iteration where no feasible improvement is possible i.e, the algorithm is not able to find any significant saving cost value where, saving cost=service cost before-service cost after.

5.2.2. Stopping Criteria

If the algorithm is not able to find a better solution than the previous iteration then it terminates.

First Iteration for Local Search:

```
Press ENTER to start the local search...                                                    c:\Users\RISHIKESH\Desktop\SSCFacilityLocat
ion\src\com\sscfacilitylocation\algorithms\localsearch\impro

Computing Local Search:

ITERATION 1

Best cycle found: [d12, 8, s] with saving of: 165383.39
        - Moving customers [26] from facility 8 to facility 12

Opened Facilities:
3 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 36 (Demand: 3671.0 - Cost: 83331.7), 44 (Demand: 2241.0 - Cost: 131938.88), 10 (Demand: 5495.0 - Cost: 12638.5),
4 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 29 (Demand: 495.0 - Cost: 15598.6875), 41 (Demand: 1117.0 - Cost: 20427.137), 45 (Demand: 733.0 - Cost: 45381.863),
 21 (Demand: 807.0 - Cost: 23493.787), 37 (Demand: 2213.0 - Cost: 20497.912), 48 (Demand: 1464.0 - Cost: 21905.1), 3 (Demand: 1337.0 - Cost: 21275.012), 38 (Demand: 705.0 - Cost
: 44494.312), 17 (Demand: 3016.0 - Cost: 23562.5), 15 (Demand: 564.0 - Cost: 18421.65), 6 (Demand: 2370.0 - Cost: 58805.625),
5 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 33 (Demand: 12912.0 - Cost: 229995.0),
10 (Capacity: 15000.0 - Fixed Cost: 0.0) serving customers: 12 (Demand: 1466.0 - Cost: 10371.95), 30 (Demand: 231.0 - Cost: 4440.975), 4 (Demand: 31.0 - Cost: 1133.05), 35 (Dema
nd: 366.0 - Cost: 5984.1), 32 (Demand: 685.0 - Cost: 14693.25), 40 (Demand: 1681.0 - Cost: 10086.0), 43 (Demand: 500.0 - Cost: 14550.0), 19 (Demand: 195.0 - Cost: 7312.5), 46 (D
emand: 222.0 - Cost: 11183.25), 1 (Demand: 87.0 - Cost: 2296.8), 18 (Demand: 253.0 - Cost: 4123.9), 23 (Demand: 304.0 - Cost: 7106.0), 7 (Demand: 1089.0 - Cost: 12741.3), 16 (De
mand: 226.0 - Cost: 3062.3), 0 (Demand: 146.0 - Cost: 5219.5), 28 (Demand: 482.0 - Cost: 5916.55), 27 (Demand: 577.0 - Cost: 1601.175), 5 (Demand: 559.0 - Cost: 7029.425), 2 (De
mand: 672.0 - Cost: 9828.0), 9 (Demand: 32.0 - Cost: 928.8), 13 (Demand: 143.0 - Cost: 2434.575), 39 (Demand: 328.0 - Cost: 5658.0), 42 (Demand: 275.0 - Cost: 11550.0), 25 (Dema
nd: 337.0 - Cost: 3268.9), 22 (Demand: 551.0 - Cost: 0.0), 24 (Demand: 814.0 - Cost: 10541.3), 31 (Demand: 322.0 - Cost: 2398.9), 14 (Demand: 615.0 - Cost: 14621.625), 49 (Deman
d: 222.0 - Cost: 4001.55), 11 (Demand: 904.0 - Cost: 2463.4), 20 (Demand: 38.0 - Cost: 780.9), 47 (Demand: 49.0 - Cost: 2174.375), 8 (Demand: 33.0 - Cost: 1395.9), 34 (Demand: 3
25.0 - Cost: 6353.75),
12 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 26 (Demand: 4368.0 - Cost: 124051.2),

TOTAL COST = 1133070.9
OPTIMUM VALUE = 932615.8 - RELATIVE ERROR = 21.493853%
```

Last iteration for Local Search:

```
ITERATION 17

No more improvement feasible.
Opened Facilities:
1 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 6 (Demand: 2370.0 - Cost: 28499.25),
2 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 33 (Demand: 12912.0 - Cost: 203364.0),
3 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 41 (Demand: 1117.0 - Cost: 13627.4), 10 (Demand: 5495.0 - Cost: 12638.5),
5 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 48 (Demand: 1464.0 - Cost: 20807.1), 37 (Demand: 2213.0 - Cost: 15076.0625), 36 (Demand: 3671.0 - Cost: 56946.387),
 3 (Demand: 1337.0 - Cost: 20071.713),
7 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 45 (Demand: 733.0 - Cost: 39050.574), 38 (Demand: 705.0 - Cost: 34351.125), 15 (Demand: 564.0 - Cost: 8135.7),
8 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 17 (Demand: 3016.0 - Cost: 11423.1),
9 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 21 (Demand: 807.0 - Cost: 11166.862),
10 (Capacity: 15000.0 - Fixed Cost: 0.0) serving customers: 12 (Demand: 1466.0 - Cost: 10371.95), 30 (Demand: 231.0 - Cost: 4440.975), 4 (Demand: 31.0 - Cost: 1133.05), 35 (Dema
nd: 366.0 - Cost: 5984.1), 32 (Demand: 685.0 - Cost: 14693.25), 40 (Demand: 1681.0 - Cost: 10086.0), 43 (Demand: 500.0 - Cost: 14550.0), 19 (Demand: 195.0 - Cost: 7312.5), 46 (D
emand: 222.0 - Cost: 11183.25), 1 (Demand: 87.0 - Cost: 2296.8), 18 (Demand: 253.0 - Cost: 4123.9), 23 (Demand: 304.0 - Cost: 7106.0), 7 (Demand: 1089.0 - Cost: 12741.3), 16 (De
mand: 226.0 - Cost: 3062.3), 0 (Demand: 146.0 - Cost: 5219.5), 28 (Demand: 482.0 - Cost: 5916.55), 27 (Demand: 577.0 - Cost: 1601.175), 5 (Demand: 559.0 - Cost: 7029.425), 2 (De
mand: 672.0 - Cost: 9828.0), 9 (Demand: 32.0 - Cost: 928.8), 13 (Demand: 143.0 - Cost: 2434.575), 39 (Demand: 328.0 - Cost: 5658.0), 42 (Demand: 275.0 - Cost: 11550.0), 25 (Dema
nd: 337.0 - Cost: 3268.9), 22 (Demand: 551.0 - Cost: 0.0), 24 (Demand: 814.0 - Cost: 10541.3), 31 (Demand: 322.0 - Cost: 2398.9), 14 (Demand: 615.0 - Cost: 14621.625), 49 (Deman
d: 222.0 - Cost: 4001.55), 11 (Demand: 904.0 - Cost: 2463.4), 20 (Demand: 38.0 - Cost: 780.9), 47 (Demand: 49.0 - Cost: 2174.375), 8 (Demand: 33.0 - Cost: 1395.9), 34 (Demand: 3
25.0 - Cost: 6353.75),
12 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 44 (Demand: 2241.0 - Cost: 89976.15), 26 (Demand: 4368.0 - Cost: 124051.2),
14 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 29 (Demand: 495.0 - Cost: 8049.9375),

TOTAL COST = 971987.0
OPTIMUM VALUE = 932615.8 - RELATIVE ERROR = 4.221587%
```

5.3. Execution for Tabu Search Algorithm

It works similar to local search, except that the recently swapped customers or involved facilities are not considered while building improvement graphs. In some cases, it accepts nonprofitable swaps to see if it can find better improvement in the second moment. In this way, we can escape the attraction basin of the local optimum found previously by the local search with the aspiration to find a better one.

5.3.1. How does the algorithm work in our model?

Step1: Choose a previous solution by local search as a candidate solution and create an empty tabu list.

Step2: Select customers from opened facilities and swap to another facility whose capacity is available (Swap may or may not always be profitable) and define aspirations.

Step3: Update the tabu list and the aspiration conditions for building the improvement graph.

Step4: Repeat the above steps until we meet our stopping condition, which is the number of iterations allowed in the model.

5.3.2. Stopping Criteria

When the number of iterations defined in the source code gets executed our tabu search algorithm gets terminated.

```
Press ENTER to start the tabu search...

Computing Tabu Search:

ITERATION 1
TabuList = []

Now exploring neighborhood of
Opened Facilities:
1 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 6 (Demand: 2370.0 - Cost: 28499.25),
2 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 33 (Demand: 12912.0 - Cost: 203364.0),
3 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 41 (Demand: 1117.0 - Cost: 13627.4), 10 (Demand: 5495.0 - Cost: 12638.5),
5 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 48 (Demand: 1464.0 - Cost: 20807.1), 37 (Demand: 2213.0 - Cost: 15076.0625), 36 (Demand: 3671.0 - Cost: 56946.387),
 3 (Demand: 1337.0 - Cost: 20071.713),
7 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 45 (Demand: 733.0 - Cost: 39050.574), 38 (Demand: 705.0 - Cost: 34351.125), 15 (Demand: 564.0 - Cost: 8135.7),
8 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 17 (Demand: 3016.0 - Cost: 11423.1),
9 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 21 (Demand: 807.0 - Cost: 11166.862),
10 (Capacity: 15000.0 - Fixed Cost: 0.0) serving customers: 12 (Demand: 1466.0 - Cost: 10371.95), 30 (Demand: 231.0 - Cost: 4440.975), 4 (Demand: 31.0 - Cost: 1133.05), 35 (Dema
nd: 366.0 - Cost: 5984.1), 32 (Demand: 685.0 - Cost: 14693.25), 40 (Demand: 1681.0 - Cost: 10086.0), 43 (Demand: 500.0 - Cost: 14550.0), 19 (Demand: 195.0 - Cost: 7312.5), 46 (D
emand: 222.0 - Cost: 11183.25), 1 (Demand: 87.0 - Cost: 2296.8), 18 (Demand: 253.0 - Cost: 4123.9), 23 (Demand: 304.0 - Cost: 7106.0), 7 (Demand: 1089.0 - Cost: 12741.3), 16 (De
mand: 226.0 - Cost: 3062.3), 0 (Demand: 146.0 - Cost: 5219.5), 28 (Demand: 482.0 - Cost: 5916.55), 27 (Demand: 577.0 - Cost: 1601.175), 5 (Demand: 559.0 - Cost: 7029.425), 2 (De
mand: 672.0 - Cost: 9828.0), 9 (Demand: 32.0 - Cost: 928.8), 13 (Demand: 143.0 - Cost: 2434.575), 39 (Demand: 328.0 - Cost: 5658.0), 42 (Demand: 275.0 - Cost: 11550.0), 25 (Dema
nd: 337.0 - Cost: 3268.9), 22 (Demand: 551.0 - Cost: 0.0), 24 (Demand: 814.0 - Cost: 10541.3), 31 (Demand: 322.0 - Cost: 2398.9), 14 (Demand: 615.0 - Cost: 14621.625), 49 (Deman
d: 222.0 - Cost: 4001.55), 11 (Demand: 904.0 - Cost: 2463.4), 20 (Demand: 38.0 - Cost: 780.9), 47 (Demand: 49.0 - Cost: 2174.375), 8 (Demand: 33.0 - Cost: 1395.9), 34 (Demand: 3
25.0 - Cost: 6353.75),
12 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 44 (Demand: 2241.0 - Cost: 89976.15), 26 (Demand: 4368.0 - Cost: 124051.2),
14 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 29 (Demand: 495.0 - Cost: 8049.9375),

TOTAL COST = 971987.0
OPTIMUM VALUE = 932615.8 - RELATIVE ERROR = 4.221587%

Best cycle found: [d15, 10, s] with saving of: -153.375
        - Moving customers [32] from facility 10 to facility 15
```

After running 100 iterations the best solution we got:

```
Best cycle found: [d10, 3, s] with saving of: 5715.4004

Stopping tabu search as it's not possible to update the best solution from 100 iterations.
Opened Facilities:
0 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 23 (Demand: 304.0 - Cost: 4883.0), 29 (Demand: 495.0 - Cost: 12480.1875), 2 (Demand: 672.0 - Cost: 4914.0), 5 (Dema
nd: 559.0 - Cost: 6421.5127),
1 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 6 (Demand: 2370.0 - Cost: 28499.25),
2 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 7 (Demand: 1089.0 - Cost: 6370.65), 33 (Demand: 12912.0 - Cost: 203364.0),
3 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 24 (Demand: 814.0 - Cost: 11050.05), 18 (Demand: 253.0 - Cost: 3542.0), 41 (Demand: 1117.0 - Cost: 13627.4), 28 (De
mand: 482.0 - Cost: 7458.95), 35 (Demand: 366.0 - Cost: 5307.0), 10 (Demand: 5495.0 - Cost: 12638.5), 34 (Demand: 325.0 - Cost: 7198.75), 49 (Demand: 222.0 - Cost: 4190.25),
5 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 12 (Demand: 1466.0 - Cost: 5185.975), 48 (Demand: 1464.0 - Cost: 20807.1), 37 (Demand: 2213.0 - Cost: 15076.0625),
36 (Demand: 3671.0 - Cost: 56946.387), 3 (Demand: 1337.0 - Cost: 20071.713), 39 (Demand: 328.0 - Cost: 5621.1),
7 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 0 (Demand: 146.0 - Cost: 3847.1), 45 (Demand: 733.0 - Cost: 39050.574), 38 (Demand: 705.0 - Cost: 34351.125), 46 (D
emand: 222.0 - Cost: 10134.3), 19 (Demand: 195.0 - Cost: 6371.625), 42 (Demand: 275.0 - Cost: 7885.625), 15 (Demand: 564.0 - Cost: 8135.7),
8 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 17 (Demand: 3016.0 - Cost: 11423.1),
9 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 21 (Demand: 807.0 - Cost: 11166.862), 14 (Demand: 615.0 - Cost: 12569.0625), 43 (Demand: 500.0 - Cost: 9543.75),
10 (Capacity: 15000.0 - Fixed Cost: 0.0) serving customers: 25 (Demand: 337.0 - Cost: 3268.9), 22 (Demand: 551.0 - Cost: 0.0), 31 (Demand: 322.0 - Cost: 2398.9), 4 (Demand: 31.0
 - Cost: 1133.05), 40 (Demand: 1681.0 - Cost: 10086.0), 1 (Demand: 87.0 - Cost: 2296.8), 16 (Demand: 226.0 - Cost: 3062.3), 11 (Demand: 904.0 - Cost: 2463.4), 20 (Demand: 38.0 -
 Cost: 780.9), 27 (Demand: 577.0 - Cost: 1601.175), 47 (Demand: 49.0 - Cost: 2174.375), 9 (Demand: 32.0 - Cost: 928.8), 13 (Demand: 143.0 - Cost: 2434.575), 8 (Demand: 33.0 - Co
st: 1395.9),
12 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 44 (Demand: 2241.0 - Cost: 89976.15), 26 (Demand: 4368.0 - Cost: 124051.2),
15 (Capacity: 15000.0 - Fixed Cost: 7500.0) serving customers: 30 (Demand: 231.0 - Cost: 3285.975), 32 (Demand: 685.0 - Cost: 7346.625),

TOTAL COST = 943817.6
OPTIMUM VALUE = 932615.8 - RELATIVE ERROR = 1.201117%
```

5.4. Results

There are **24 instances of dataset from OR-Library** and calculated mean relative error and mean total cost associated with each dataset instance such that there is no possibility of biased result.

Sl No.	Dataset Instance	Optimal Value	Greedy Algorithm		Local Search		Tabu search	
			Relative Error	Total Cost	Relative Error	Total Cost	Relative Error	Total Cost
1	Cap61	932615.8	39.2270%	1298453.8	4.2215%	971987.0	1.2011%	943817.6
2	Cap62	977799.4	34.8388%	1318453.8	6.4330%	1040701.7	2.71234%	1004320.7
3	Cap63	1014100.0	31.98439%	1338453.8	5.088428%	1065701.8	3.68422%	1051461.8
4	Cap64	1053197.0	29.93331%	1368453.8	5.748129%	1113736.1	1.81505%	1072313.1
5	Cap71	932615.7	33.83249%	1248142.9	7.795467%	1005317.4	3.56916%	965902.25
6	Cap72	977799.4	27.64815%	1248142.9	4.348342%	1020317.4	3.34453%	1010502.2
7	Cap73	1010641.0	23.50012%	1248142.9	2.158153%	1032452.2	2.15815%	1032452.2
8	Cap74	1034977.0	20.59619%	1248142.9	1.205358%	1047452.2	1.20535%	1047452.2
9	Cap91	796648.4	62.98956%	1298453.8	4.846068%	835254.5	2.80424%	818988.4
10	Cap92	858109.3	53.64636%	1318453.8	6.498502%	913873.56	3.25924%	886077.2

11	Cap93	900760.1	48.59158%	1338453.8	6.48589%	959182.44	3.73901%	934439.6
12	Cap94	950608.4	43.95557%	1368453.8	6.206328%	1009606.2	3.86149%	987316.06
13	Cap101	796648.4	56.67424%	1248142.9	4.846068%	835254.5	3.14957%	821739.4
14	Cap102	854704.2	46.03214%	1248142.9	4.284055%	891320.2	2.57602%	876721.56
15	Cap103	893782.1	39.64733%	1248142.9	3.521495%	925256.6	3.06098%	921140.7
16	Cap104	928941.8	34.36179%	1248142.9	2.853381%	955448.06	2.76735%	954648.94
17	Cap121	793439.6	63.84772%	1300032.8	3.317785%	819764.25	2.64798%	814449.75
18	Cap122	854900.5	54.40776%	1320032.8	8.187196%	924892.9	3.12369%	881605.0
19	Cap123	898266.1	49.17992%	1340032.8	8.564903%	975201.75	3.28513%	927775.4
20	Cap124	950608.4	44.12167%	1370032.8	8.680473%	1033125.7	3.32403%	982206.94
21	Cap131	793439.6	57.30785%	1248142.9	5.571885%	837215.2	3.00071%	817248.44
22	Cap132	851495.3	46.58247%	1248142.9	3.690258%	882917.7	2.29419%	871030.3
23	Cap133	893076.7	39.75763%	1248142.9	3.603281%	925256.75	3.14240%	921140.75
24	Cap134	928941.8	34.36179%	1248142.9	2.853381%	955448.06	2.76735%	954648.94

5.4.1. Final Mean Result Table

Approach	Mean Relative Error	Mean Total Cost
Greedy Algorithm	42.37%	1289811.5166
Local search	5.042%	953596.2916
Tabu search	2.8538%	937474.9589

Mean Optimum Value = 911588.2

(Optimum values pre-defined in dataset)

(The Cost is measured in terms of rupees)

5.4.2. Calculations

Greedy Algorithm-

Mean Relative Error = Total Error / 24

=1017.0246/24

=42.37%

Mean Total Cost = Total Cost Sum / 24

= 30955476.4 / 24

= 1289811.5166

Local Search Algorithm-

Mean Relative Error = Total Error / 24

=121.008/24

=5.042%

Mean Total Cost = Total Cost Sum / 24

= 22886311 / 24

= 953596.2916

Tabu Search Algorithm-

Mean Relative Error = Total Error / 24

=68.4906/24

=2.8538%

Mean Total Cost = Total Cost Sum / 24

= 22499399 / 24

= 937474.9589

Optimum Value-

Mean Optimum Value = Total Optimum / 24

= 21878116 / 24

= 911588.2

Summary

In this project, we tried to optimize the facility locations such that the total effective cost from the facility center i.e. warehouse should be minimum and the relative error should be least. We performed three heuristic algorithms such as greedy, local search and tabu meta-heuristic algorithm.

From the results derived in the above sections, we can say that a greedy algorithm, it Makes the best optimal choice at each small stage with the goal of leading to an optimal solution that has a reasonable accuracy as this algorithm performs only a single iteration.

In the local search algorithm, the algorithm starts from a candidate solution and then iteratively moves to a neighbor solution. This algorithm performs 17 iterations to reach the final optimal solution. This algorithm delivers better results than the greedy algorithm with less relative error.

In the Tabu meta-heuristic algorithm, it performs about 100 iterations to reach the final state. As this algorithm updates the maxima with the global maxima each time, we get a new optimal iteration until we get the final global maxima. In this project, we got that global maxima in around 100 iterations. This algorithm performs better than both greedy algorithms and local search algorithms and has least relative error.

Conclusion

After performing three algorithms on the capacitated facility location problem, we arrived at the conclusion that among three algorithms greedy, local search, tabu, the tabu algorithm i.e. meta-heuristic algorithm provides more optimal results than the other two algorithms. This performance is measured on the basis of parameters such as total cost and relative error. Using OR library we are provided with the dataset and costs required for each customer to get facility from specific facility centers. This concludes that the Tabu meta-heuristic algorithm is much superior and gives more optimum results in assigning the customers to respective facility centers or warehouses.

So, we can conclude that the **Facility Location Problem- Warehouse Location** can be solved optimally with help of tabu algorithm rather than other two algorithms such as greedy and local search.

References

[1] https://towardsdatascience.com/urban-drones-the-facility-location-problem-6137c59d7a7e

[2] https://kth.diva-portal.org/smash/get/diva2:1215912/FULLTEXT01.pdf

[3]https://www.researchgate.net/publication/222827602_Capacitated_facility_location_problem_with_general_setup_cost

[4] https://econ-papers.upf.edu/papers/642.pdf

[5] https://www.cs.cmu.edu/~anupamg/adv-approx/lecture6.pdf

[6] http://webia.lip6.fr/~codognet/PSAI/8-LS.pdf

[7]https://www.researchgate.net/publication/23547095_A_Tabu_Search_Heuristic_Procedure_for_the_Capacitated_Facility_Location_Problem

[8] https://paperzz.com/doc/7301198/local-search-heuristics-for-k-median-and-facility-locatio…

Acknowledgement

It gives us immense pleasure and satisfaction in presenting this mini project **'Facility Location Problem-Warehouse Location'**.

We would like to express our deep sense of gratitude towards **Dr.B.Jayalakshmi** for her continuous guidance and assistance.

We also wish to express our heartfelt gratitude to all those who have played a crucial role in the research for this project, without their active cooperation the preparation of this project could not have been completed within the specified time limit.

Dasari Rishikesh	19bcs035
Shinde Rohit Sagar	19bcs099
Suryawanshi Randheer Ajit	19bcs104
Mane Param Dattatraya	19bcs121