Facility Location Problem

A B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Pranav Vinchurkar and Tattukolla Lokesh
(190101064 and 190101094)

under the guidance of

Pinaki Mitra

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Facility Location Problem**" is a bonafide work of **Pranav Vinchurkar and Tattukolla Lokesh** (**Roll No. 190101064 and 190101094**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Pinaki Mitra**

Associate Professor,

April, 2023

Department of Computer Science and Engineering,

Guwahati.

Indian Institute of Technology Guwahati, Assam.

Acknowledgements

We wish to express our sincere gratitude to our guide Professor Pinaki Mitra for his constant support and guidance (also resources) throughout the semesters. We are also grateful for the resources like Library, research papers, journal access and MATLAB computing resources, and support provided by IIT Guwahati which helped us in going forward and exploring the algorithms and simulating them.

Abstract

In this paper, the general ideas surrounding the facility location problem are discussed, which is a classic optimization problem that has received attention for decades due to its practical application background, primarily in both computer science and operations research. Here facilities (such as servers, warehouses, and factories) are to be constructed/placed such that the cost of conveying between the facilities and demand places (customers, towns) is minimized. Various models are formulated for this problem in this; we will deal with the uncapacitated version of the problem. Firstly we will go through the simplest formulation with only one facility, which is generally known as the Weber problem, and then further the discrete facility location problems in which the possible locations for facilities are finite, the most basic one, the Uncapacitated Facility Location Problem on which we reviewed many algorithms from greedy to linear programming. Later, we analyzed the single facility location problem that tries to identify the center for the facility by creating demand regions, where each demand region represents close towns where the conveying costs can be ignored. We assume that all the regions are disks of some arbitrary radius r and use the rectilinear L_1 norm as the distance metric. Then we extended this algorithm to multi-facility by dividing the set of towns into multiple clusters. Later, we explored a few meta-heuristic techniques: Teaching Learning Based Optimization (TLBO), Particle Swarm Optimization (PSO), Genetic Algorithm (GA), etc. We used these meta-heuristic techniques as the solution tool by making some variations to these algorithms for solving the problem by defining the objective function and using the penalty approach for the con-

straints on MATLAB. Although computationally costly, a Mixed Integer Linear Programming (MILP) approach was also used to obtain optimal solutions by simulating on Python's pulp library (CBC MILP Solver) so as to compare results with the other techniques.

Contents

List of Figures **vii**

List of Tables **viii**

1 Introduction **1**

1.1 Problem Formulation and Notation 1

1.2 The Fermat-Weber Problem 3

1.3 Relation to other NP-Hard Problems 3

1.3.1 k-Medians Problem 3

1.3.2 Set Covering Problem 4

2 Review of Prior Works **6**

2.1 Forward Greedy Algorithm 6

2.2 Reverse Greedy Algorithm 6

2.3 Linear Programming 7

2.3.1 Rounding Algorithm 7

2.3.2 Analysis of Rounding Algorithm 8

2.4 Local Search 9

2.5 Polyhedral, Block Norms and Disk-Shaped Demand Regions 10

2.5.1 Introduction 10

2.5.2 Problem Formulation 11

2.5.3 Algorithm and Observations 12
2.6 K-means algorithm by Lloyd 13
2.6.1 Pseudo Code 14
2.7 Metaheuristic Techniques 14
2.7.1 Particle Swarm Optimization (PSO) 15
2.7.2 Teaching Learning based Optimization (TLBO) 16
2.7.3 Genetic Algorithms (GA) 17

3 Proposed Approaches **18**
3.1 Extending A. Berger et al Algorithm to Multi-Facility 18
3.1.1 Introduction 18
3.1.2 Notation and Pseudo Code 18
3.2 Using Metaheuristic Techniques 19
3.2.1 Introduction 19
3.2.2 Decision Variables 20
3.2.3 Objective Function 21
3.2.4 Penalty 21
3.2.5 Binary Integer Constraint 22

4 Experimental Results and Discussion **24**
4.1 Sample Datasets 24
4.2 Mixed-Integer Linear Programming Simulation and Results 26
4.3 TLBO/PSO/GA Results for Sample Datasets 27
4.4 Further Discussion 29
4.4.1 Fitness Value vs Iteration 29
4.4.2 Comparison Metric 30
4.4.3 Summary 30

5 Conclusion **32**

References **33**

List of Figures

1.1 Example of FLP 2

1.2 k-Medians Problem 4

1.3 Set Covering Problem 5

2.1 Relation between αL_j and L_j 8

2.2 Regions for 4 points in 2 Dimension) 11

2.3 Basic Drawing for single point x 12

2.4 Grid 12

2.5 Cell 13

2.6 Block of Cells 13

3.1 Objective Function Code for TLBO 23

4.1 Defining Variables, Objective Function and Constraints in the Python Simulation 26

4.2 Observed Results for 3 sample datasets 28

4.3 Fitness Function Value vs Iterations for D1 29

4.4 Fitness Function Value vs Iterations for D2 29

4.5 Fitness Function Value vs Iterations for D3 29

List of Tables

4.1 Dataset-1 (D1) 25

4.2 Dataset-2 (D2) 25

4.3 Dataset-3 (D3) 25

4.4 MILP Results for Dataset-1 27

4.5 MILP Results for Dataset-2 27

4.6 MILP Results for Dataset-3 27

4.7 Summary Table 30

Chapter 1

Introduction

In this chapter, first, we will model/formulate our problem and give some notations which we will use thereafter. Then instead of going directly to our actual problem, we will see the historical background and a much simpler version of the Facility Location Problem (from here on FLP), the Fermat-Weber Problem.

1.1 Problem Formulation and Notation

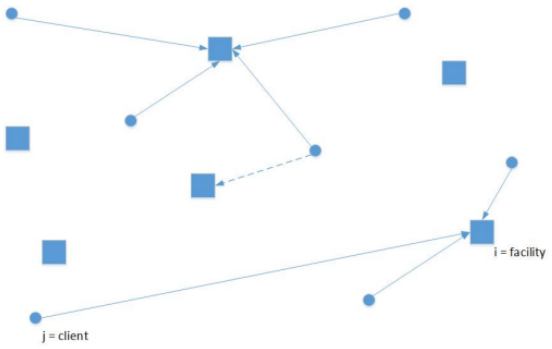
As stated earlier, we are going to deal with possible locations of facilities as finite, where we are provided a metric space with towns (customers) T, facilities F. Now we need to consider some say m facility locations of the potential places from F and need to assign them to towns such that each town is assigned to one facility. We can categorize the FLP into 2 variants, uncapacitated facility location problem and capacitated facility location problem.

In the uncapacitated version, we can assign any number of towns to the facility (no capacity restriction on the facility), whereas in the capacitated version, we have a cap to the number of towns that can be assigned to a particular facility, and we can't assign more than that limit. We will focus on the uncapacitated FLP from here on. Given metric space X:

- Possible facilities location finite set: F: (1, 2...m-1, m)

- The set of n finite towns: T: (1,2...n-1, n)
- Conveying (travel) cost for town j using facility i: d_{ij}
- Condition variable x_{ij}: 1 if the town i is assigned to facility j; otherwise 0
- Facilities to be opened: k
- Condition variable y_j: 0 if the facility j of F is not opened; otherwise 1
- A fixed facility cost f_i for opening them (we are not considering this, assuming it to be zero)

Fig. 1.1 Example of FLP



And we are trying to figure out the following:

- k facilities (say set S) of the set F of m facilities
- allocating one facility to each of the towns, an assignment $\sigma : T \to S$
- while our objective is the above assignment, but also to do it in an optimal way using the following objective function (minimizing the conveying cost, which is measured as the distance between the facility and town)

$\sum_{j\epsilon T, i\epsilon F} d_{ij} x_{ij} + \sum_{i\epsilon F} f_i y_i$

1.2 The Fermat-Weber Problem

This problem looks very simple but there are various possibilities and traps that it has been working since the 17th century and continue to produce enormous literature. Various names for this have been used due to various contexts: the Weber problem, the generalized Fermat problem, the Steiner problem, the Fermat-Toricelli problem, the Steiner-Weber problem, the mini-sum problem, the spatial median problem, and so on. Coming to the definition of the problem, we need to find the "minisum" point, which minimizes the sum of weighted Euclidean distances from itself to some fixed (many cases taken as 3) n points. The problem can be formulated as below: Find x and y to minimize the W(x, y) where W is defined as below:

$W(x, y) = \sum_{i=1}^{n} w_i d_i(x, y)$, where d_i(x, y) is the distance (euclidean) between (x, y) and one of the fixed point (a_i, b_i) $\sqrt{(x - a_i)^2 + (y - b_i)^2}$

This can also be seen as the extension of the simple 1-D median of n-weighted values to two dimensions, also known as the spatial-median finding problem.

In this case, we have taken distance as euclidean distances. Instead, we can use the rectilinear or Manhattan distances (more often used in grid cases) where $d_i(x, y) : |x - a_i| + |y - b_j|$. Here basically, the facility is set to 1, and our problem becomes the multi-facility version of the Fermat-Weber problem. There have been extensive records on solving the Fermat-Weber problem from the Torricelli point formulation, Simpson Lines, using the Dual approach, the Weiszfield algorithm, and many more.

1.3 Relation to other NP-Hard Problems

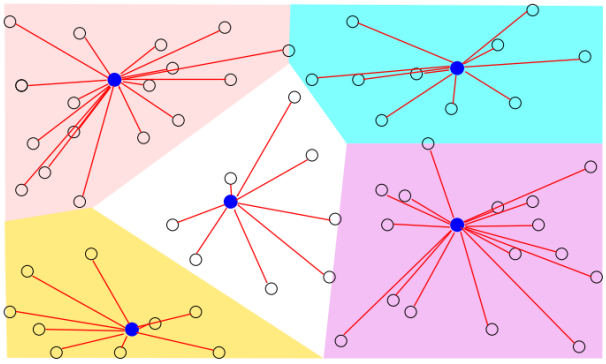
1.3.1 k-Medians Problem

The facility location problem has many similarities to the k-median problem. The k-median problem can be defined as: given a set of points, we try to create k disjoint clusters, which involve finding k cluster centers for which the summation of the distances from all

points to the cluster's center is minimized. We can use different means to infer the "center." If the metric used is L2 Norm, it's the mean (k-means), while if we use L1 Norm, it's the median (k-median). Given a metric space X,

- The input provided: Finite set S $\subset$ X; integer k
- Output: T $\subset$ S with the cardinality of T = k
- Minimizing the cost(T) = $\sum_{x \epsilon S} \rho(x, T)$

Fig. 1.2 k-Medians Problem



In terms of facilities and customers, it can be defined as follows: a set of customers C ϵ X, and a set of facilities F ϵ X such that X = F U C = n, open k facilities in F to minimize the summation of distances from every customer to its nearest facility which is open. Here the opening costs of the facility are not considered, and we are pre-determined to open k facilities, so there is an upper bound to the number of facilities. Kariv and Hakim reduced the k-medians problem to the dominating set problem and, there, proved that determining k-medians in a given graph is NP-Hard.

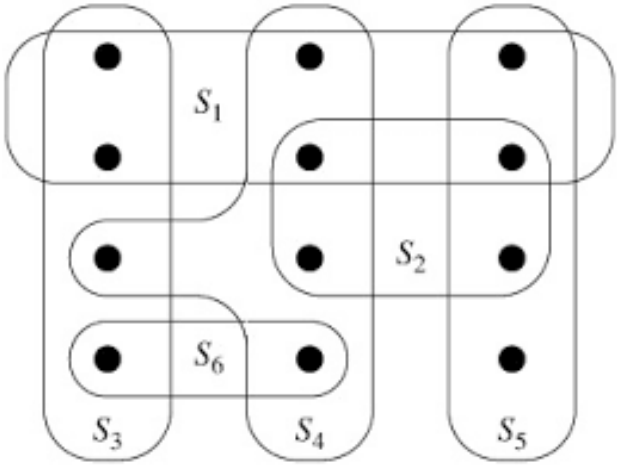
1.3.2 Set Covering Problem

Let's see the relation of FLP to the well-known Set Covering Problem. It's formulated as we are given two sets, a set U (Universe) and a set S, which is the set of subsets of U. Each of the subsets in S has a weight "c" (or cost) and taking the union of all the subsets

in S gives the set U. Now, the problem is finding the optimal number of subsets such that their union is the set U with the objective of minimizing the total cost of included subsets.

We can see that Set Covering Problem is a special case of Uncapacitated Facility Location Problem: Consider set U as the customer set (T:= U), set S as the facilities set (F:= S), opening cost of the facility f_s = c(s), s ϵ S, and consider the conveying cost d_sj as 0, for j ϵ s ϵ S and ∞ for j ϵ U-s. Also, conversely, we can say given an instance of FLP as a special case of the Set Covering Problem. The set cover is well known to be hard. Considering the

Fig. 1.3 Set Covering Problem



universe set U, it has been proven that there doesn't exist a polynomial time algorithm for any constant $\varrho > 0$ such that unless NP = P, whose complexity is at most $\varrho \ln|U|$ times the optimal solution. This hardness can be said even to the FLP due to the above relation with the set cover problem.

https://www.overleaf.com/project/64426a2a538e532186a9f73a

Chapter 2

Review of Prior Works

2.1 Forward Greedy Algorithm

In this approach, at a step t, consider a set S_t of facilities. Make $S_{t+1} = S_t U f$, for a particular facility $f \notin S$, so the cost of S_{t+1} is minimized. Repeat till $|S|$ = k.

It has been established that the approximation ratio for this algorithm is at least n.

Algorithm 1

$S \leftarrow \{\}$
while $|S| < k$ **do**
    $S \leftarrow$ S U $\{argmin_i(cost(SU\{i\}))\}$;
end while
Return S

2.2 Reverse Greedy Algorithm

Consider F as the set of all possible locations for facilities. In thus approach, consider every possible location having a facility initially. At every step, a facility is removed from a location, such that the effect on cost is the least. This is repeated till there are k facilities left.

The approximation ratio is between Ω (log n / log log n) and O(log n)

Algorithm 2

$S \leftarrow \{\text{F}\}$
while $|S| > k$ **do**
  $S \leftarrow S - \{argmin_i cost((S - \{i\}))\}$;
end while
Return S

2.3 Linear Programming

Let's try formulating a linear program this time. As per the notation in the introduction, there are two decision variables used:

$x_{ij} = 1$, if town j is allocated to facility i, otherwise 0

$y_i = 1$ if the facility i is open else, 0

Clearly, x_{ij} and y_i are integers (binary) that can take either 0 or 1, so we get an Integer Linear Program (ILP); the ILP formed:

min $\sum_{j \epsilon T, i \epsilon F} d_{ij} x_{ij} + \sum_{i \epsilon F} f_i y_i$

s.t.

$\sum_{i \epsilon F} x_{ij} = 1 \; \forall \; j \; \epsilon \; T$

$x_{ij} \leq y_i \; \forall \; j \; \epsilon \; T, \; i \; \epsilon \; F$

$x_{ij}, \; y_i \; \epsilon \; \{0, 1\} \; \forall \; j \; \epsilon \; T, \; i \; \epsilon \; F$

We can relax the integrality constraints to convert this into a linear programming problem making the complexity to polynomial: $0 \leq x_{ij} \leq 1$, $0 \leq y_i \leq 1$. Now the value we get from this LP is, at most, the optimal value of the original ILP. Let (x, y) be the optimal solution we get from the LP; we then round this to integers (x^*, y^*) which will be our solution to the FLP. We need to bound the value of the cost of (x^*, y^*) as compared to the (x, y).

2.3.1 Rounding Algorithm

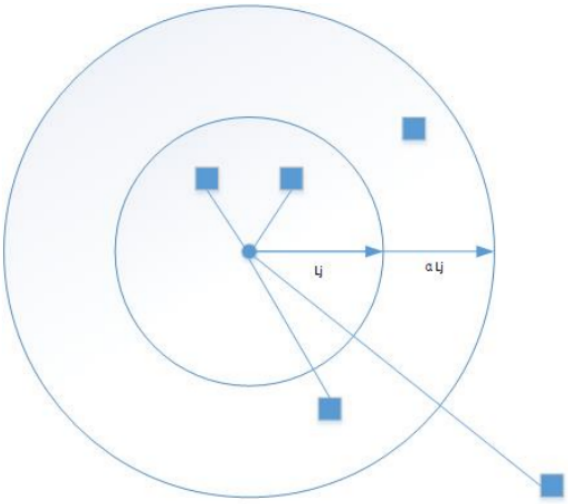
The following is a rounding algorithm which is at most 4 times optimal.

Consider a set of locations, V = F U T, a union of the sets of facilities and towns.

Lets define $B(V, r) = \{u \in V : d(v, u) \leq r\}$, a ball with vertex v and radius r.

Let $L_j = \sum_{i \epsilon F} \mathrm{d_{ij}x_{ij}}$, be defined as the LP connection cost for a town j. Consider a ball

Fig. 2.1 Relation between αL_j and L_j



with vertex at town j and radius αL_j, for a particular α as shown in Fig 2.1

Now, consider the following steps for opening of facilities:

1. Sort towns based on values of L_j, in ascending order.
2. A maximal set of disjoint balls is picked greedily, in the order based on sorted values of L_j.
3. Consider the set of chosen balls as $\{B_j, j \in I\}$. Open the cheapest facility $\pi(j)$ in each $\{B_j, j \in I\}$. Take S as the set of facilities opened.

2.3.2 Analysis of Rounding Algorithm

Consider the following in the solution:

Facility Cost, $\mathrm{F^* = \sum_{i \epsilon F} f_i y_i}$

Connection Cost, $\mathrm{D^* = \sum_{i \epsilon F, j \epsilon T} d_{ij} x_{ij}}$

Bounding the Facility Cost

Consider the claim $\sum_{k \epsilon B_j} x_{jk} \geq 1 - \frac{1}{\alpha}$, for all $j \in T$.

If it is not true, $\sum_{k \notin B_j} x_{jk} > \frac{1}{\alpha}$ and thus $L_j = \sum_{i \in F} d_{ji} x_{ij} \geq \sum_{k \notin B_j} x_{kj} \alpha L_j > \alpha \frac{L_j}{\alpha} = L_j$ which is a contradiction.

Since, $\{B_j, j \in I\}$ is disjoint, we can sum for $j \in I$

This leads to the result of the Facility Cost being bounded by $\frac{1}{1 - \frac{1}{\alpha}}$ F*.

Bounding the Connection Cost

For each town j, distance d(j, S) $\leq 3\alpha L_j$. Adding for all j $\in$ T, Connection Cost $\leq 3\alpha L_j$.

Proving that it is an at most 4-approximation algorithm

We know, Facility Cost $\leq \frac{1}{1 - \frac{1}{\alpha}}$ F* and Connection Cost $\leq 3\alpha L_j$

So ALG $\leq \frac{1}{1 - \frac{1}{\alpha}}$ F* + $3\alpha L_j \leq \max(\frac{\alpha}{\alpha - 1}, 3\alpha)(F* + D*)$

Considering α as $\frac{4}{3}$, ALG $\leq$ 4(F* + D*), hence it is an at most 4-approximation algorithm.

2.4 Local Search

Consider S as the set of Consider an assignment of facilities after applying an arbitrarily feasible solution like the Forward Greedy Algorithm. In a step of the algorithm, a median from S is swapped with an element from F - S, where the cost is minimized. The algorithm can be generalized with multi-swaps are considered, in which up to p facilities can be swapped simultaneously. It has been established that Local Search with p swaps is an 3 + 2/p approximation algorithm.

Consider $B(S)$ as the sets that can be obtained from S after p swaps. The Local Search algorithm is as follows.

Algorithm 3 Local Search with p swaps

$S \leftarrow$ Arbitrary feasible solution
while $\exists T \in B(S)$ such that cost(T) < cost(S) **do**
    $S \leftarrow$ T;
end while
Return S

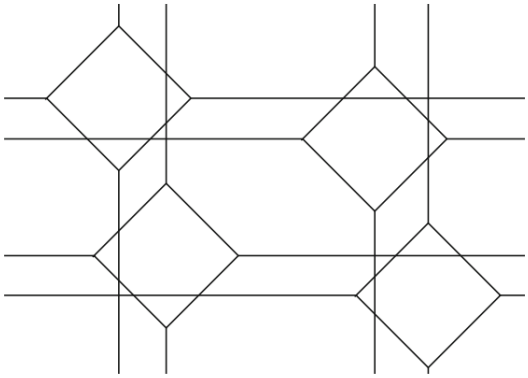
2.5 Polyhedral, Block Norms and Disk-Shaped Demand Regions

2.5.1 Introduction

Here we will deviate from the regular way of looking at the problem. Firstly, this algorithm is designed for a single facility (k is 1). In the convention FLP each town is represented by a single point, but in most applications, these points are too many to be considered as a single region(Dinler). Instead, these towns, which have some similarities, can be combined into one "demand region."

In this section, the conveying cost to a particular demand region is basically the cost from the nearest point in the selected demand region to the facility (that point is the center) the costs for conveyance within the demand region can be ignored. Here we define the demand regions as balls or disks of some radius r (≥ 0), and the metric for measuring the distance is the rectilinear norm like the L_1 norm(Manhattan distance). For l_1 norm, the disk shape comes out to be diamond. If we consider r as 0, it's basically the k-median problem where k is 1. Here A. Berger et al. developed an algorithm for the single facility location problem from here on SFLP, with disk-shaped demand regions in d dimensions using the rectilinear distance metric, solving for arbitrary disk radius r (≥ 0).

Fig. 2.2 Regions for 4 points in 2 Dimension)



2.5.2 Problem Formulation

A. Berger et al. solved for generalized polyhedral norms, but here we will limit ourselves to L_1 norm, and in this, the disks take the shape of diamonds. Let's consider the dimension d to be 2. Now given an input set of towns = x_1, x_2, ..., x_n and r (non-negative), the Single FLP with diamond regions is to find a center c, that minimises the following:

f(c) = $\sum_{i=1}^{n} \max(0, (\rho(x_i - c) - r))$, where $\rho(x) = \|x\|$ (the L_1 norm)

The above equation can be re-formulated as LP with generalization up to d dimensions. Consider j ϵ 1,2,3,..., d which tells us the jth coordinate of the point:

min $\sum_{i=1}^{n} z_i$

s.t.

$y_{i,j} \geq x_{i,j} - c_j$; $y_{i,j} \geq -x_{i,j} + c_j \; \forall \; i \in \{1,2,..., n\}, j \in \{1,2,..., d\}$

$z_i \geq \sum_{i=1}^{n} y_{i,j} - r, \; \forall \; i \in \{1,2,..., n\}$

$z_i \geq 0, \; \forall \; i \in \{1,2,..., n\}$

where $y_{i,j}$ is the distance between x_i projected to its jth coordinate and c, and z_i is the distance between x_i - r and c if its non-negative, else 0.

Fig. 2.3 Basic Drawing for single point x

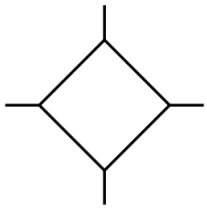
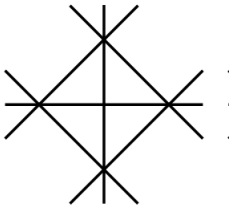


Fig. 2.4 Grid



2.5.3 Algorithm and Observations

As mentioned earlier, the demand regions here are diamonds, for a single point it looks like Fig 2.3 a basic drawing, when there are many of them as like in Fig 2.2, we embed many drawings of single points in the plane centering at respective points. This union of drawings of each point create embedding (also the line segments can be extended to lines), this gives us grid (as in Fig 2.4), which is formed of lines that are horizontal and vertical; and their 45 degree rotations, Now we define a cell to be the maximal polyhedron which is not crossed by any line as in Fig 2.5, We can see in the Fig 2.6, the union of such cells between any 2 consecutive parallel lines of the grid as block (of cells). Also we can observe that f(c) is convex and linear in pieces (piecewise linear).

In Location Theory, in location problems with polyhedral gauges we can formulate linear programs (Ward, Wendell) which can be easily solved. As defined earlier if we draw the fundamental directions starting at each point (town/demand point) we get multiple polyhedral cells and the objective function (f) is linear in each cell, these regions known as linearity regions or simple cells. So, as the function f is linear within cell, solving this takes linear O(n) time only.

We can see there are 4 types of lines as per Fig 2.4 (at least in 2D case), so each cell

Fig. 2.5 Cell

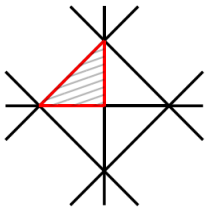
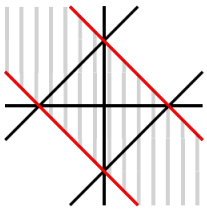


Fig. 2.6 Block of Cells



has at most 8 faces and 8 vertices, so the evaluation of f(c) at all corner points of a cell takes linear time and in all of them finding the minimum one will also take linear time. In general, there are $O(n^2)$ cells, so instead of looping over these quadratic no. of cells we can use binary search over the 4 directions (in the case of 2D) corresponding to 4 block types. As f(c) is convex, so, the binary search returns the cell correctly which minimizes the function.

It takes at most O(log n) to perform binary search and we need to do in 4 directions making it $O(\log^4 n)$ in 2D case, and further at each step we the objective function is evaluated making the time complexity $O(n\log^4 n)$.

2.6 K-means algorithm by Lloyd

Lloyd's algorithm partitions n points into k clusters, such that each point is part of the cluster with the nearest mean or center.

Initially, we are given a non-optimal arbitrary clustering. In a step of the algorithm, each point is reassigned to a cluster based on the nearest center to the point. For each cluster, the center is then updated to the mean of the points of the cluster. This is repeated

until a convergence criterion is met.

We added this algorithm in this section as we will be using this in the subsequent section. Next, we will see the pseudo-code of the algorithm.

2.6.1 Pseudo Code

Algorithm 4 k-means algorithm by Lloyd

D = $\{t_1, t_2 ... t_n\}$ //set of elements
$K \leftarrow$ Arbitrary positions of k centers
while Convergence Criteria is not met **do**
    Assign each element t_i to the cluster with the closest mean
    Update centers of each cluster;
end while
Return K

2.7 Metaheuristic Techniques

Metaheuristics Techniques are a family of optimization algorithms that iteratively explore the solution space to find near-optimal solutions. They are used to solve optimization problems where exact solutions are difficult to obtain. These algorithms are inspired by phenomena occurring in nature such as swarm intelligence, evolution, and physical phenomena. These techniques are flexible and can handle a wide range of optimization problems, and can often find good solutions even when the problem is not well-defined or difficult to evaluate.

These techniques do not require detailed knowledge of the problem structure. They are thus useful for complex problems where the objective function is difficult to specify or where the search space is very large. They are also computationally efficient.

These techniques usually first generate randomly a single solution or a set of solutions. A set of rules are used to update the candidate solutions based on their fitness value. Fitness functions measure how well solutions meet the objective function. In each iteration,

new candidate solutions are generated by the application of intelligent operators. These operators enable the algorithm to explore a large portion of the solution space.

2.7.1 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO), (Kennedy and Eberhart, 1995) is a metaheuristic technique inspired by Swarm Intelligence, or the collective behavior of animal societies or insect colonies . Swarm Intelligent Behaviour involves self-organization and division of labout. In PSO, a population of particles is initialized randomly in the solution space, with each particle representing a potential solution. Each particle has an associated position and velocity. Particles modify their position by adjusting their velocity. Each particle memorizes the best location identified by it and particles communicate the information among themselves. The particles move through the solution space iteratively, guided by the best solution found so far by the particle itself (personal best) and the best solution found by the entire swarm (global best). The movement of each particle is governed by its velocity, which is updated at each iteration based on its own experience and the experience of the swarm.

Algorithm 5 Pseudocode: PSO

$x \leftarrow$ Arbitrary initial positions of particles
$v \leftarrow$ Arbitrary initial velocities of particles
$p_{best,i} \leftarrow x_i$ //Personal best of ith particle
$g_{best} \leftarrow x$ //Global best
$r_1, r_2 \leftarrow$ Random numbers $\epsilon[0, 1]$
while Stopping Criteria is not met **do**
    $v_i = v_i + r_1(p_{best,i} - x_i) + r_2(g_{best} - x_i)$ //particle velocity
    $x_i = x_i + v_i$ //Update particle position
    Evaluate objective function f_i and update the population.
    Update p_{best} and g_{best}
end while
Return g_{best}

2.7.2 Teaching Learning based Optimization (TLBO)

Teaching-Learning-Based Optimization (TLBO), proposed by Rao et. al. in 2011 is a stochastic population based metaheuristic optimization technique inspired by the teaching and learning process that takes place in classrooms.

The problem is viewed as a classroom where the teacher (the best solution) guides the students (candidate solutions) to improve their performance. The students learn from the teacher and from each other by sharing knowledge and experience. At each iteration, a best solution is identified, and the teacher guides the students to move towards the best solution by adjusting their positions.

The algorithm has two phases, the Teacher Phase and the Learner Phase. Each solution goes through the Teacher Phase followed by the Learner Phase. In the Teacher Phase, a new solution is generated using the mean of the population and the best solution. After that, the current solution is greedily replaced with the new solution, if the new solution is better. In the Learner Phase, a new solution is generated with the help of a partner solution which is randomly selected from the population. After that, again, there is greedy replacement if applicable.

Algorithm 6 Pseudocode: TLBO

```
X ← Population of arbitrary initial solutions
F ← Fitness value of solutions
while Stopping Criteria is not met do
    Find the best and worst solution in the population
    Calculate the teaching vector as the mean of all solutions except the worst one
    For each solution in the population:
      //r is a random number between -1 and 1.
      Generate a new solution by adding r multiplied by the teaching vector
      If the new solution is better than the current solution, replace it
    Update the objective function values for all solutions
end while
Return Best Solution found
```

2.7.3 Genetic Algorithms (GA)

Genetic Algorithm (GA) is a metaheuristic technique inspired by Darwin's principle of natural evolution which involves natural selection and genetic evolution. The algorithm works by iteratively generating and evaluating a population of candidate solutions. Solution vectors or strings of parameters are termed as chromosomes. A Parent Solution is a solution from which new solutions are generated, and offspring are newly generated solution. In each iteration, the algorithm applies genetic operators, such as mutation, crossover, and selection, to generate a new population of candidate solutions or offspring, based on a fitness function. In a environment of limited resources, better solutions have a greater chance of survival. The better solutions are selected for the following iterations and the process continues until a stopping criterion is met.

Algorithm 7 Pseudocode: GA

$X \leftarrow$ Population of arbitrary initial solutions
$F \leftarrow$ Fitness value of solutions
while Stopping Criteria is not met **do**
    Select parents for mating using a selection method based on fitness
    Create offspring by applying genetic operators like mutation/crossover to the parents
    Evaluate the fitness of the offspring
    Select the fittest individuals for the next generation
    Preserve the best solutions from the previous generation by applying elitism
    Update the fitness values for solutions in the new population
end while
Return Best Solution found

Chapter 3

Proposed Approaches

3.1 Extending A. Berger et al Algorithm to Multi-Facility

3.1.1 Introduction

Our proposal is to extend the algorithm by A. Berger et al. on SFLP to mutli-facility. This we can be achieved by initially partitioning the towns into the k clusters, here k is the number of facilities we want setup.

Initially we make k clusters of towns, using the Lloyd's algorithm and then for each cluster we create demand regions and use the SFLP algorithm of A. Berger et al. to find location of the facility of that particular cluster. In the same way find for each of the cluster. After this iterate over each town to see if any of the unallocated k-1 facilties are closer than the alloted facility, if yes the change allocation to the closest one.

3.1.2 Notation and Pseudo Code

Let T be the given set of towns, k be the no. of facilities to be constructed and denote them with F (set). BergerSFLP to denote the algorithm introduced in last section, which takes some n towns and returns the position for the single facility to be setup for those towns. $d_{i,\,j}$ denotes the distance between the points i, j.

Algorithm 8 Proposed Algorithm

```
T = {t_1, t_2...t_n} //set of towns
S ← LloydAlgo(T, k) // applying lloyds algorithm to cluster
S = {s_1, s_2...s_k} //set of k-clusters of towns
C = {c_1, c_2...c_k} //initialise k centers for each cluster
i = 0
while i < k do
    c_i ← BergerSFLP(s_i)
    i++
end while
for i ← 1 to k do
    m ← size(s_i)
    for  j ← 1 to m do
        l ← min(d_{s_ij c_1}, ..., d_{s_ij c_k})
        If (i == l)
        continue
        Else
        s_i.remove(s_ij); s_l.insert(s_ij)
    end for
end for
Return C
```

3.2 Using Metaheuristic Techniques

3.2.1 Introduction

In this section, we proposed how the FLP can be solved using the various metaheuristic techniques discussed. Here, the problem statement is tweaked a little, so I will briefly define the problem here: In our problem, each town should be serviced by a facility center. It is possible that a facility gets no town allocations, in that case, it won't be opened.
Multiple criteria (factors), such as facility opening costs and travel costs (distance), to towns, are considered in deciding the opening of facilities and town allocations. For the following algorithms, we did not put a cap on how many facilities to be opened as we have decided it for the algorithm to decide based on fixed and variable costs. Of course, we can force it to restrict to, say, k facilities to be opened.

The generic framework of metaheuristic algorithms is as follows:

- Generate a single solution or a set of solutions randomly
- Based on the fitness of the current solutions suggest other solutions with the help of "intelligent" operators
- Various parameters like population size and maximum iterations need to be initialized depending on the situation.

In general, the metaheuristic algorithm works like a black box where we give some objective function and random solutions and it gives out better solutions based gradually over the iterations. I have used MATLAB to code the following algorithms to solve the FLP. The notations used (unless mentioned otherwise) are:

- n = number of potential facility locations
- m = number of demand points (towns)
- c_{ij} = cost of serving demand point j from facility i
- x_{ij} = binary decision variable indicating whether demand point j is served by facility i ($x_{ij} = 1$ if demand point j is served by facility i, and 0 otherwise)
- f_i = fixed cost of establishing facility i
- y_i = binary decision variable indicating whether facility i is established ($y_i = 1$ if facility i is established, and 0 otherwise)

I have explained this section extensively using the TLBO algorithm, although I implemented other techniques like GA, PSO which were already mentioned in other papers. (not TLBO)

3.2.2 Decision Variables

- First, we need to define our decision variables (the representation format). I have considered all the variables binary, where the first n values denote whether the potential facility is established (1 if yes, 0 otherwise).

- The next n*m values basically is the allocation matrix, each value denoting which facility is serving the town (basically x_{ij} values when stretched).
- We dissect these variables and convert them to our convenient forms. (see Fig 3.1 Lines 5 and 7)

This is an important step because we will be defining our objective function using these decision variables, and we will be required to know the lower and upper bounds of these (have some idea) in a few algorithms like TLBO.

3.2.3 Objective Function

- Now, we define the cost function incorporating both fixed and variable cost terms using the decision variables.
- The fixed cost is added if the facility is established; see line 13 in Fig. 3.1 (second term on RHS).
- The variable cost is basically the distance cost between the allocated town and facility see line 13 in Fig. 3.1 (first term on RHS).

3.2.4 Penalty

- We also have constraints like a facility can be allocated only when it is established, and each town should be allocated only one town. To incorporate this, we use the penalty approach in our objective function.
- We basically add penalty terms (as in Fig 3.1, lines 16 and 21) if the above constraints are violated and penalize those solutions by adding the penalty with the cost determined in the fitness function value (Fig 3.1, line 33).
- Sometimes, the penalty approach might not work (as happened in our case initially). On looking for it, there might be many reasons like the Penalty factor is too high

(the algorithm may be biased towards infeasible solutions), an Insufficient number of iterations, poor initialization of the population, or can be wrong implementation itself. Our case was due to poor initialization.

- Combining this penalty and objective cost function gives us our fitness function (Fig. 3.1 line 33) (in some places, fitness and objective terms are used interchangeably).

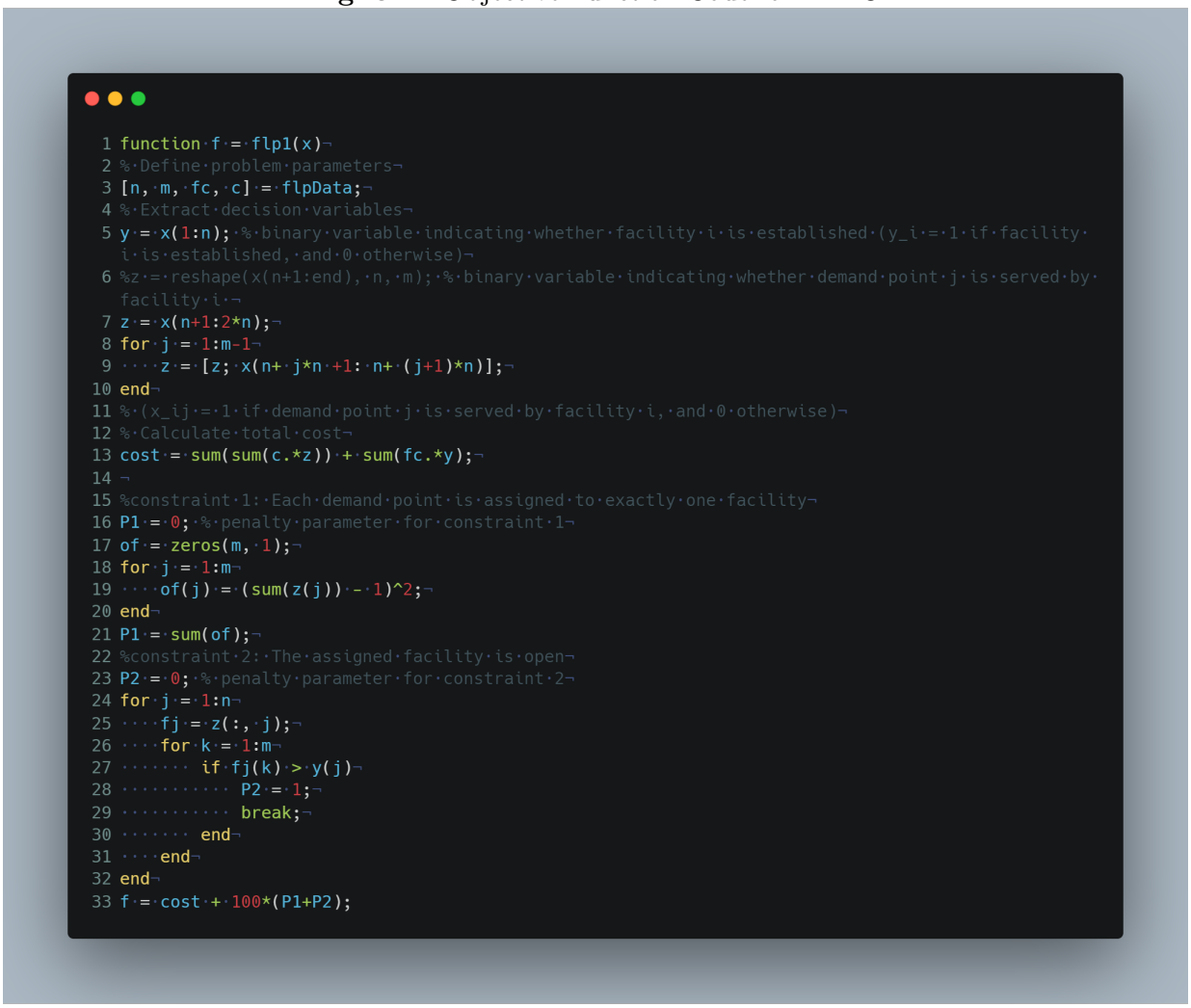
3.2.5 Binary Integer Constraint

We need to make changes in the algorithm (in general, it's written for continuous variables) to encode the binary variables and make sure the population is made in that way which will also influence the results. In the TLBO case, specifically on MATLAB, we use the "rand" function in general, but here we need to use "randi" and restrict the numbers to [0 1] while generating the population and during bounding conditions.

In this way, by enforcing the integer constraints in the algorithm and penalizing the objective function when constraints aren't met, we developed a solution to FLP using other metaheuristics as well. The implementation code of the discussed algorithm and others can be found in the following link: GitHubRepo

Also, for a better understanding of how things are happening, we memorized the best fitness function value in each iteration and then plotted them against the iteration number, which can be seen next chapter.

Fig. 3.1 Objective Function Code for TLBO



```
function f = flp1(x)
%·Define·problem·parameters
[n, m, fc, c] = flpData;
%·Extract·decision·variables
y = x(1:n);·%·binary·variable·indicating·whether·facility·i·is·established·(y_i·=·1·if·facility·i·is·established,·and·0·otherwise)
%z·=·reshape(x(n+1:end),·n,·m);·%·binary·variable·indicating·whether·demand·point·j·is·served·by·facility·i·
z = x(n+1:2*n);
for j = 1:m-1
····z = [z; x(n+ j*n +1: n+ (j+1)*n)];
end
%·(x_ij·=·1·if·demand·point·j·is·served·by·facility·i,·and·0·otherwise)
%·Calculate·total·cost
cost = sum(sum(c.*z)) + sum(fc.*y);

%constraint·1:·Each·demand·point·is·assigned·to·exactly·one·facility
P1 = 0;·%·penalty·parameter·for·constraint·1
of = zeros(m, 1);
for j = 1:m
····of(j) = (sum(z(j)) - 1)^2;
end
P1 = sum(of);
%constraint·2:·The·assigned·facility·is·open
P2 = 0;·%·penalty·parameter·for·constraint·2
for j = 1:n
····fj = z(:, j);
····for k = 1:m
·····  if fj(k) > y(j)
·····  P2 = 1;
·····  break;
·····  end
····end
end
f = cost + 100*(P1+P2);
```

Chapter 4

Experimental Results and Discussion

In this chapter, we will discuss the output generated while running the various algorithms discussed earlier and see how the compare with the solutions obtained by simulating the MILP. We ran the algorithms on 100 data sets/test cases and compared the results Out of those 100 test cases, first we look at 3 sample datasets, before moving on to comparison with all 100 test cases.

4.1 Sample Datasets

The following are 3 random sample datasets out of all the datasets used while executing the algorithms discussed earlier. These are just 3 samples to get an understanding of how the results look like for a few cases of the various algorithms. Later, when we compare the performances of the algorithms, we consider all 100 test cases, and not just these 3.

Facility locations	1	2	3	4	5
Fixed cost	12	5	3	7	9
Town1	2	3	6	7	1
Town2	0	5	8	4	12
Town3	11	6	14	5	8
Town4	19	18	21	16	13
Town5	3	9	8	7	10
Town6	4	7	9	6	0

Table 4.1 Dataset-1 (D1)

Facility locations	1	2	3	4
Fixed cost	6	8	10	3
Town1	1	3	5	6
Town2	7	2	8	4
Town3	2	1	3	9
Town4	4	2	5	3
Town5	3	7	1	6
Town6	5	4	8	9
Town7	9	7	1	2

Table 4.2 Dataset-2 (D2)

Facility locations	1	2	3	4
Fixed cost	9	6	2	10
Town1	2	5	8	6
Town2	0	3	7	5
Town3	10	4	12	3
Town4	18	15	20	14
Town5	1	8	7	6

Table 4.3 Dataset-3 (D3)

4.2 Mixed-Integer Linear Programming Simulation and Results

Fig. 4.1 Defining Variables, Objective Function and Constraints in the Python Simulation

```
m = 5 #number of potential facility locations
n = 6 #number of demand points

#Costs of establishing facilities (In potential locations)
f = [12, 5, 3, 7, 9]

#Costs of serving demand points.
c = [[2, 3, 6, 7, 1],
[0, 5, 8, 4, 12],
[11, 6, 14, 5, 8],
[19, 18, 21, 16, 13],
[3, 9, 8, 7, 10],
[4, 7, 9, 6, 0]]

#Defining decision variables
y = LpVariable.dicts("y", Facility, 0, 1, LpBinary) #0 or 1, binary variable.
x = LpVariable.dicts("x", [(i,j) for i in Town for j in Facility], 0, 1)

flp = LpProblem("LP_Problem", LpMinimize)

flp += lpSum(f[j]*y[j] for j in Facility)+lpSum(x[(i, j)]*c[i][j] for i in Town for j in
Facility)

#Setting up Objective Function
for i in Town:
    flp += lpSum(x[(i, j)] for j in Facility) == 1

#Setting up Constraints
for i in Town:
    for j in Facility:
        flp += x[(i,j)] <= y[j]
```

As we have seen in Section 2.3, the problem can be solved using Linear Programming. We have simulated it in Python using the CBC MILP Solver of Pulp Library.

There are n towns and m potential facility locations. Array f in the code snippet of the simulation below represents the costs of opening the facilities and array c represents the costs of serving demand points.

The decision variables used are $x_{ij} = 1$, if town i is allocated to facility j, otherwise 0 and $y_j = 1$ if the facility j is open else, 0.

We get an Integer Linear Program (ILP):

min $\sum_{i \epsilon T, j \epsilon F}$ $c_{ij} x_{ij}$ + $\sum_{j \epsilon F} f_j y_j$

s.t.

$\sum_{j \epsilon F} x_{ij} = 1 \; \forall$ i ϵ T

$x_{ij} \leq y_j \; \forall$ i ϵ T, j ϵ F

x_{ij}, $y_i \; \epsilon \; \{0, 1\} \; \forall$ j ϵ T, i ϵ F

The integrality constraints are relaxed to convert it into a linear programming problem, which can be solved in polynomial time with solutions with $0 \leq x_{ij} \leq 1$, $0 \leq y_i \leq 1$. These solutions are then rounded by the solver using algorithms, such as in Section 2.3.1, to give the solution to our FLP.

The observed results for the sample datasets used are as follows:

Objective Function Value	Facility locations	1	2	3	4	5
46	Towns Allocated				2,3,5	1,4,6

Table 4.4 MILP Results for Dataset-1

Objective Function Value	Facility locations	1	2	3	4
29	Towns Allocated	1,3,6			2,4,5,7

Table 4.5 MILP Results for Dataset-2

Objective Function Value	Facility locations	1	2	3	4
37	Towns Allocated	1,2,5	3,4		

Table 4.6 MILP Results for Dataset-3

4.3 TLBO/PSO/GA Results for Sample Datasets

The observed results for the 3 sample datasets using TLBO/PSO/GA are as follows:

Fig. 4.2 Observed Results for 3 sample datasets

TLBO	Objective Function Value	Facility Locations	1	2	3	4	5
Dataset 1	51	Towns Allocated	1,2,3,4,5,6				
Dataset 2	37	Towns Allocated	1,2,3,4,5,6,7				
Dataset 3	40	Towns Allocated	1,2,3,4,5,6				

PSO	Objective Function Value	Facility Locations	1	2	3	4	5
Dataset 1	49	Towns Allocated	2,5			3	1,4,6
Dataset 2	61	Towns Allocated	2,5	1,4	6	7,3	
Dataset 3	60	Towns Allocated	4,5	2	1	3	

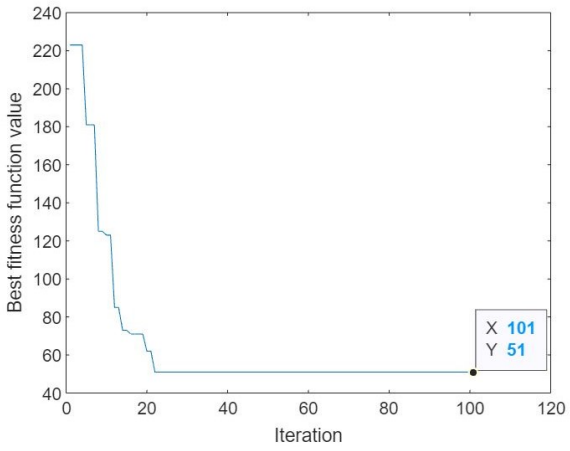
GA	Objective Function Value	Facility Locations	1	2	3	4	5
Dataset 1	53	Towns Allocated					1,2,3,4,5,6
Dataset 2	37	Towns Allocated	1,2,3,4,5,6,7				
Dataset 3	40	Towns Allocated	1,2,3,4,5,6				

4.4 Further Discussion

4.4.1 Fitness Value vs Iteration

We also observed how the fitness function (best) value changes throughout the iterations for a better understanding of the methodology. The following three plots (Fig. 4.3, 4.4, 4.5) have been generated for the three datasets mentioned earlier using the TLBO Algorithm. We can see with iterations increasing the fitness value decreasing and it varies less as it progresses.

Fig. 4.3 Fitness Function Value vs Iterations for D1



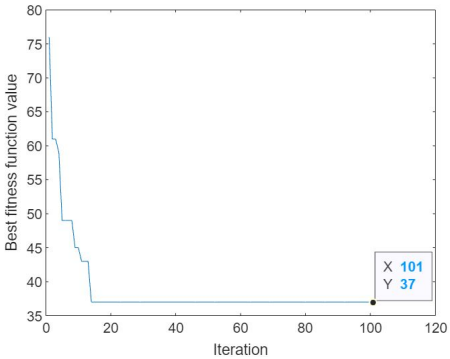


Fig. 4.4 Fitness Function Value vs Iterations for D2

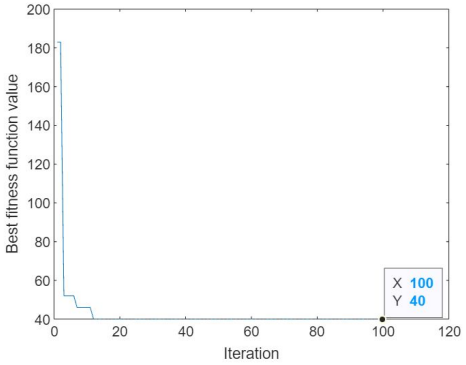


Fig. 4.5 Fitness Function Value vs Iterations for D3

4.4.2 Comparison Metric

For a particular test case, suppose after applying an algorithm, the value of the objective function obtained is X, and the optimal value is Y, with $Y < X$ (considering the minimizing case). We consider the comparison metric as Mean Absolute Error (pondered whether to use RMSE or MAE). Our metric for comparing the performance of different Metaheuristic techniques is the calculation of MAE for each technique over 100 cases. MAE $= \sum_{i=1}^{T} |Yi - Xi| /T$

4.4.3 Summary

We have summarized our inferences and results in the below comparison of techniques:

	TLBO	PSO	GA
Phases	Teacher, Learner	No Phases (Position, Velocity Update)	No phases (Mutation, Cross-over)
Parameters	Population Size, Termination Criteria	Population Size, Termination Criteria, w, c1, c2	Population size, termination criteria, pc , pm, and other parameters of variation operators
Generation of Solutions	Using other solutions, mean and best solution	Using velocity vector, pbest and gbest	Using other solutions
Best Solution	Part of population	Need not be part of population	Part of population
Selection	Greedy	Always accept new solution into the population	Survival of the fittest
no.of function evaluations	Np + 2NpT	Np + NpT	At max, Np + NpT
Obj Value for D1 (Optimal: 46)	51	49	53
Obj Value for D2 (Optimal: 29)	37	61	37
Obj Value for D3 (Optimal: 37)	40	60	40
MAE Value	10.33	15.5	9.83

Table 4.7 Summary Table

Although no approach gave the best results always, overall we can see GA performing slightly better in our testing conditions over the other two.

Chapter 5

Conclusion

We have seen various algorithms for FLP. In detail analysis of LP Rounding and the SFLP algorithm of André Berger et al. Then we extended the later algorithm to the multi-facility algorithm and added a greedy step at the end. Then we simulated the MILP for a few datasets to get the optimal results, which were later used to compare the results obtained using certain metrics. Finally, we summarized the various metaheuristic approach approaches used. Although it does not give exact optimal answers always, for large datasets, this metaheuristic approach will be of great use. In addition, to the best of our knowledge, this is the first application of this variant of TLBO is applied to the Uncapacitated FL problem. We tested for many cases and were concerned about normalizing and solving the data. Our computational experiments showed that no metaheuristic is easier and good under all circumstances; it depends upon the situation, like problem type, size, and level of detail involved. To extend and for further research, more rigorous model research and validation on benchmark problem instances and larger datasets can be done. Also, hybrid models can be proposed by combining various heuristics and exact techniques and verifying for better results.

References

[1] Zvi Drezner, Horst W.Hamacher. Facility Location Applications and Theory.

[2] Vijay V. Vazirani. Approximation Algorithms, Springer publication.

[3] Viswanath Nagarajan, University of Michigan. IOE 691: Approximation and Online Algorithms. Lecture Notes: Linear-Programming Methods.

[4] Jens Vygen, University of Bonn. Approximation Algorithms for Facility Location Problems (Lecture Notes).

[5] Christopher Whelan, Greg Harrell, Jin Wang. Understanding the K-Medians Problem.

[6] David Dohan, Stefani Karp, Brian Matejek (2015). K-median Algorithms: Theory in Practice.

[7] Vijay Arya, Naveen Garg, Rohit Khandeka. Local Search Heuristics for k-median and Facility Location Problems.

[8] An efficient algorithm for the single facility location problem with polyhedral norms and disk-shaped demand regions. André Berger1, Alexander Grigoriev1, Andrej Winokurow, Springer publication.

[9] Dinler, D., Tural, M.K., Iyigun (2015): Heuristics for a continuous multi-facility location problem with demand regions.

[10] Solving The FERMAT-WEBER Problem A Numerical and Geometric Approach Beat-Trachsler, MartinGuggisberg, Switzerland

[11] Computer Aided Applied Optimization, Prakash Kotecha, Debasis Maharana and Remya Kommadath, Department of Chemical Engineering, Indian Institute of Technology Guwahati

[12] Sumanta Basu, Megha Sharma (2013): Metaheuristic Applications on Discrete Facility Location Problems: A Survey

[13] Mohammed Al-Haidary et. al(2021): Metaheuristic Approaches to Facility Location, 4th ICSPIS

[14] Ali R. Guner and Mehmet Sevkli: A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem

[15] Teaching-learning-based optimization: An optimization method for continuous non-linear large scale problems, Information Sciences, Volume 183, Issue 1, 2012

[16] Genetic Algorithms in search, optimization and machine learning, Addison-Wesley publishing company, 1989

[17] Particle swarm optimization, Proceedings of ICNN'95 - International Conference on Neural Networks, Perth, WA, Australia, 4, 1942-1948, 1995

[18] University of Lagos. Metaheuristics for Solving Facility Location Optimization Problem in Lagos, Nigeria.