19,9/25 +4

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Computational Experience with Heuristics for the P - Center Problem

---

REPORT

COURSE : *Selected Topics on Optimization*

PROFESSOR: *Roger Zirahuen Ríos Mercado*

SEMESTER: *January – June 2026*

DUE DATE: *15/05/2026*

TEAM D		
STUDENT NAME:	ID STUDENT	DEGREE
Jesús Eduardo Moreno Osoria	*2136063*	*ITS*
Jesús Emmanuel Camacho Moreno	*2153842*	*ITS*
Oscar Felipe Renaut Vega	*2226683*	*ITS*
Lesly Adamaris Cotlame Nieto	*2226958*	*ITS*

INTRODUCTION

The p-Center Problem, also known as the minimax location problem, is one of the best-known NP-hard discrete location problems. Its purpose is to determine the optimal location of exactly p facilities, minimizing the maximum distance between any demand node and the nearest facility. Simply put, this problem seeks to answer the following question: if we have to open exactly p facilities (for example, fire stations or ambulances), can we ensure that the client furthest away receives the best possible service under the given conditions? Unlike other models that attempt to minimize average cost or average distance, the p-Center Problem focuses on the worst-case scenario. That is, it doesn't focus so much on what happens "on average," but rather on ensuring that no one is too far from the service. This is why it is known as a minimax model: it minimizes the maximum distance between any demand node and its nearest facility. [1] [3]

The importance of this topic lies primarily in its focus on equity. In many real-world situations, especially in emergency services, it is unacceptable to neglect any one area simply because the overall average is fine. For example, when locating fire stations, ambulance bases, or disaster relief centers, the most important thing is that even the client living farthest away can receive help within a reasonable timeframe. In these cases, reducing the most critical response time can literally save lives. [2]

Furthermore, the problem presents different variations that model distinct real-world contexts. For example, the vertex-center p-problem assumes that facilities can only be located at specific nodes in a network, while the absolute p-problem allows them to be placed at any point along the edges (Kariv and Hakimi, 1979). Weighted versions also exist where nodes have varying importance or associated population, which is useful in urban and regional applications. Regarding its mathematical modeling, several formulations have been proposed to represent the problem efficiently. Elloumi, Labbé, and Pochet (2004) developed a new linear integer formulation that strengthens the classic model and improves the lower bounds, allowing for better computational performance in certain cases. These contributions demonstrate that, although the problem's structure is conceptually simple, its mathematical treatment requires careful analysis. [2]

In this proposal, we will investigate the behavior of the p-Center Problem from a computational perspective, analyzing its mathematical formulation, its fundamental constraints, and how it can be implemented for specific instances. We will also study how to evaluate the quality of a solution and how to build a model that adequately represents real-world service location situations. Our aim is to understand not only the formal structure of the problem but also its practical relevance and its impact on applications where ensuring equitable coverage is a priority.

SECTION 2: PROBLEM DESCRIPTION

In this section, we present the formal description of the p-Center problem. While the introduction explained a general, intuitive idea, it is now necessary to define it with greater mathematical precision. In optimization, a problem must clearly specify four fundamental elements: the input data (what information is known), the decisions (what we must choose), the objective function (what we want to optimize), and the constraints (what conditions must be met for a solution to be feasible). We will now explain each of these components for the p-Center problem.

1.- DATA INPUT: First, the input data consists of a set of demand nodes, representing the customers or users requiring service, and a set of candidate locations where the facilities could be installed. In many cases, both sets coincide and correspond to the nodes of a network. Additionally, a distance matrix d_{ij} is available, indicating the distance between each customer i and each possible location j. Finally, the value of p is known; this value is fixed and represents the exact number of facilities that must be located. [1]

I = set of demand nodes, $I = \{1, \dots, N\}$,
J = set of candidate facility sites, $J = \{1, \dots, M\}$,
d_{ij} = distance between demand node $i \in I$ and candidate site $j \in J$,
P = number of facilities to be located,

2.- DECISIONS: Secondly, the model's decisions involve determining which locations to select for opening facilities and how each customer will be assigned to an open facility. Binary variables are used to represent this mathematically. The variable w_j takes the value 1 if a facility is located at candidate site j, and 0 otherwise. Similarly, the variable Y_{ij} takes the value 1 if demand node (customer) i is assigned to an open facility j, and 0 if they are not. These variables allow the real-world decisions to be translated into a mathematical language that can be processed by an algorithm. [1]

$$w_j = \begin{cases} 1 & \text{if a facility is located at candidate site } j \in J, \\ 0 & \text{otherwise,} \end{cases}$$

$$Y_{ij} = \begin{cases} 1 & \text{if demand node } i \in I \text{ is assigned to an open facility at candidate site } j \in J, \\ 0 & \text{otherwise,} \end{cases}$$

3.- OPTIMIZATION: Third, the objective function must reflect the main purpose of the problem: to minimize the maximum distance between any demand node and the facility serving them. It is important to understand that the goal here is not to minimize the sum of distances or the average, but the maximum value. For this reason, the model introduces an additional variable, D, which represents this maximum distance. The objective is to minimize D. Conceptually, this means that we are trying to minimize the worst-case scenario (i.e., the customer farthest from the service).

4.- CONSTRAINS: Regarding the constraints, these ensure that the solution is practically feasible. First, each customer must be assigned to exactly one facility. Second, a customer can only be assigned to a facility that is actually open. Third, the total number of open facilities must be exactly p. Finally,

the variable D must be greater than or equal to the distance between each customer and their assigned facility, ensuring that D accurately represents the maximum distance in the system. With all this, the mathematical model should be as follows:

(1) Minimise D
subject to:

(2) $\sum_{j \in J} Y_{ij} = 1 \qquad \forall i \in I,$

(3) $Y_{ij} \leq w_j \qquad \forall i \in I,\ j \in J,$

(4) $\sum_{j \in J} w_j = P,$

(5) $D \geq \sum_{j \in J} d_{ij} Y_{ij} \qquad \forall i \in I,$

(6) $w_j,\ Y_{ij} \in \{0, 1\} \qquad \forall i \in I,\ j \in J,$

The objective function (1) minimises the maximum distance between each demand node and its closest open facility. Constraint (2) ensures that each demand node is assigned to exactly one facility, while constraints (3) restrict demand nodes to be assigned to open facilities. Constraint (4) stipulates that P facilities are to be located. Constraints (5) define the maximum distance between any demand node i and the nearest facility at node j . Finally, constraints (6) refer to integrality constraints. [1]

SECTION 3: PROBLEM EXAMPLE

In this section, we present the formal description of the p-Center problem. While the introduction explained a general, intuitive idea, it is now necessary to define it with greater mathematical precision. In optimization, a problem must clearly specify four fundamental elements: the input data (what information is known), the decisions (what we must choose), the objective function (what we want to optimize), and the constraints (what conditions must be met for a solution to be feasible). We will now explain each of these components for the p-Center problem.

Now we will apply the algorithm to the next problem example:

1.- INPUT

A weighted network of n demand nodes and edges with non-negative travel distances, plus an integer p (number of facilities).

Given:

- $N = \{P, Q, R, S, T, U, V, W\}$
- $n = 8$
- $p = 1$ (parts a and b)
- $p = 2$ (parts c)

2.- DECISIONS

The location of p facilities on the network at any node or interior point of an edge (absolute), or at nodes only (vertex).

- Where to place P* (parts a, b) or P1 and P2 (part c) on the tree.

3.- OPTIMIZATION

Minimize the coverage radius $D = MAX\, d_{ij}$ over all demand nodes i: the worst-case distance from any node to its nearest facility.

- $D = 23$ (part a) $D = 26$ (part b) $D = 18.5$ (part c)

4.- CONSTRAINTS

Exactly p facilities must be placed. Each node is assigned to its nearest facility. Locations must lie on the network.

- p = 1 or p = 2 facilities on the 8-node tree.

To illustrate the p-Center Problem, we construct a small custom instance: a tree of 8 nodes (labeled P through W) connected by 7 weighted edges, where each weight represents the travel distance between two adjacent nodes. Because the network is a tree connected graph with no cycles there is exactly one path between any pair of nodes, so the distance between them is simply the sum of the edge weights along that path.

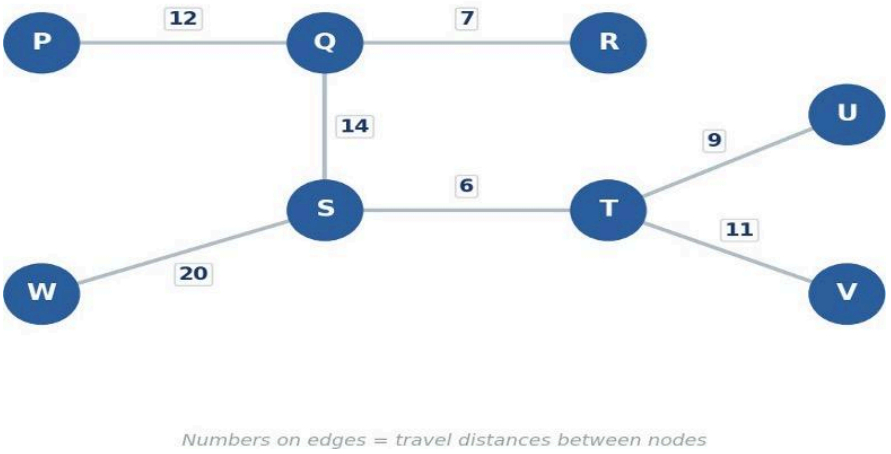

FIGURE 1. NETWORK INSTANCE.

d_{ij}	P	Q	R	S	T	U	V	W
P	0	12	19	26	32	41	43	46
Q	12	0	7	14	20	29	31	34
R	19	7	0	21	27	36	38	41
S	26	14	21	0	6	15	17	20
T	32	20	27	6	0	9	11	26
U	41	29	36	15	9	0	20	35
V	43	31	38	17	11	20	0	37
W	46	34	41	20	26	35	37	0

TABLE 1. MATRIX DISTANCE.

PART (A).

The Absolute 1-Center asks: where on the tree (at any node or any interior point of an edge) should a single facility be placed to minimize the worst-case distance to any node? Formally, if x is any point on the tree, we want to minimize $D(x) = max d_{xj}$

Theorem (Daskin, 1995): On a weighted tree, the absolute 1-center is the midpoint of the diameter — the longest shortest-path between any two nodes. This follows because D(x) is convex along any path in the tree, so its minimum is at the midpoint of the diameter [4].

From Table 1. , the diameter is d(P, W) = 12+14+20 = 46 units, achieved along path P-Q-S-W. The optimal facility location P* sits at the midpoint of this path, 46/2 = 23 units from each endpoint. Tracing from P: after 12 units we reach Q, after 26 units we reach S — so the midpoint at 23 units falls 11 units past Q on edge Q–S, leaving 3 units to S.

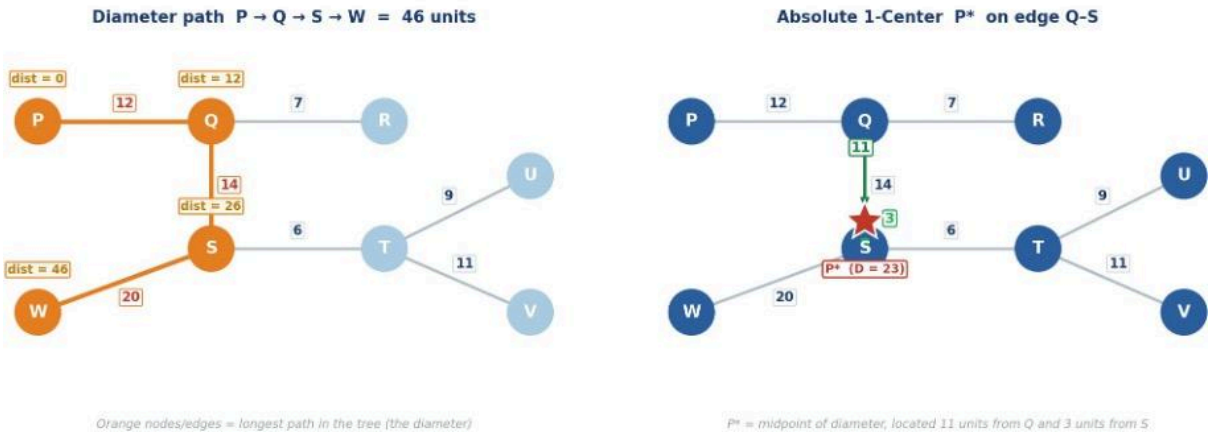

FIGURE 2. FINDING THE DIAMETER AND PLACING P*.

To confirm the radius, we check the distance from P* to every node by routing through the nearest endpoint of its edge (Q at distance 11, or S at distance 3). Nodes P and W are each exactly 23 units away, confirming the radius. All other nodes are closer: Q (11), R (18), S (3), T (9), U (18), V (20).

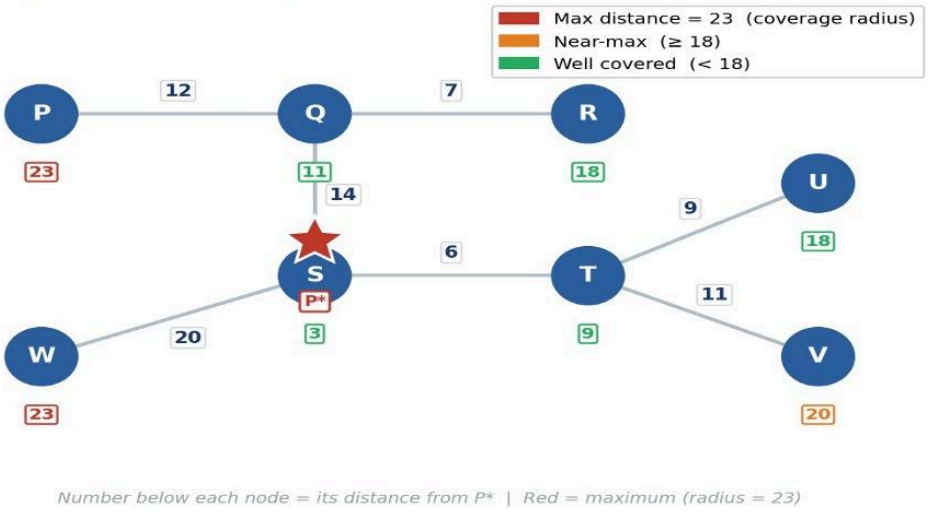

FIGURE 3. COVERAGE RADIUS OF P*.

Result (a): The absolute 1-center is P* on edge Q–S, 11 units from Q. Optimal radius = D = 23 units.

PART (B).

In the Vertex 1-Center variant, the facility must be placed at one of the eight nodes. For each candidate node v, its eccentricity e(v) $=max_{i \epsilon N}$d(v,i) measures the worst-case coverage if v is chosen as the facility. The vertex 1-center is the node with the smallest eccentricity.

Scanning the rows of Table 3.1, node S has eccentricity 26 (its farthest node is P or W at distance 26), which is the minimum across all nodes. Every other node would leave at least one demand point farther than 26 units away. Compared to the absolute 1-center (D=23), the vertex solution is 3 units worse because the optimal location P* lies strictly inside edge Q–S, unavailable to the vertex formulation.

PART (C).

With two facilities, each node is served by its nearest facility. The key result for trees is that the optimal solution always partitions the tree into two connected subtrees by removing exactly one edge.

Theorem (Daskin, 1995): On a weighted tree, the optimal 2-center solution splits the tree into two connected subtrees by removing one edge. It therefore suffices to test all 7 edges as candidate cuts and pick the one that minimizes max(R1, R2), where Ri is the 1-center radius of subtree i [4].

CUT	T1 (NODES)	R1	T2 (NODES)	R2	MAX(R1,R2)
P-Q	{P}	0	{Q,R,S,T,U,V,W}	20.5	20.5
Q-R	{R}	0	{P,Q,S,T,U,V,W}	23	23
Q-S	**{P,Q,R}**	**9.5**	**{S,T,U,V,W}**	**18.5**	**18.5**
S-T	{P,Q,R,S,W}	23	{T,U,V}	10	23
T-U	{U}	0	{P,Q,R,S,T,V,W}	23	23
T-V	{V}	0	{P,Q,R,S,T,U,W}	23	23
S-W	{W}	0	{P,Q,R,S,T,U,V}	21.5	21.5

TABLE 2. ALL 7 EDGE CUTS.

The optimal cut is edge Q–S, splitting the tree into T1 = {P, Q, R} (diameter 19, center P1 on edge P–Q at 9.5 fromP) and T2 = {S, T, U, V, W} (diameter 37 along path V-T-S-W, center P2 on edge S–W at 1.5 from S). Both radii equal 18.5.

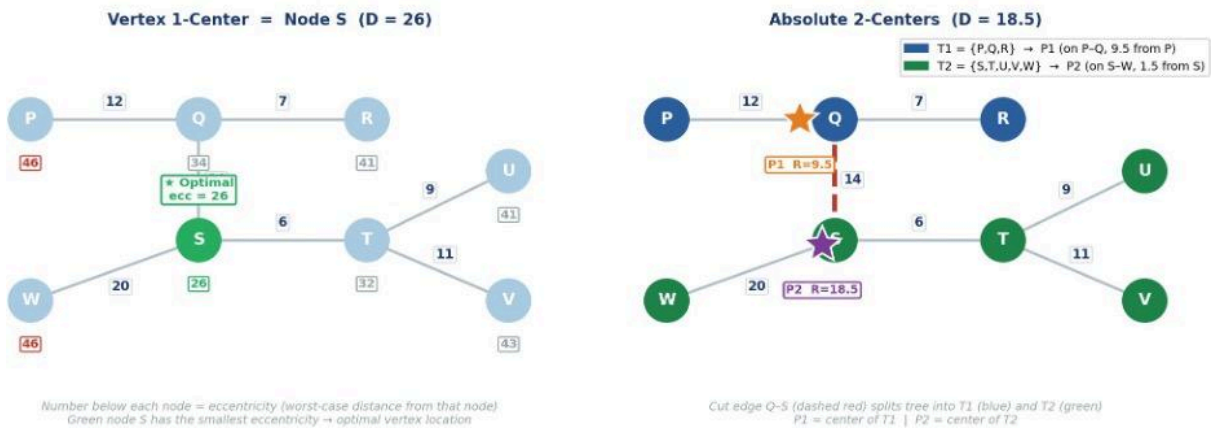

FIGURE 4. VERTEX 1-CENTER AND ABSOLUTE 2-CENTERS.

- **Result (b):** Vertex 1-center = Node S, D = 26 units.
- **Result (c):** P1 on edge P–Q (9.5 from P) and P2 on edge S–W (1.5 from S). Optimal radius = D = 18.5 units.

SECTION 4: DESCRIPTION OF HEURISTICS

In this section, we present a set of heuristic methods to address the p-Center Problem from a computational perspective. Given the combinatorial complexity of the problem, exact approaches may become impractical for medium and large instances, motivating the use of alternative solution strategies.

The proposed methods are organized into two categories: constructive heuristics and local search procedures. Constructive heuristics generate feasible solutions by progressively selecting facility locations based on a specific criterion. In contrast, local search methods operate over an existing solution and attempt to improve it by exploring neighboring configurations.

4.1 Constructive Heuristic 1

4.1.1 Pseudocode

```
Input:
    I = set of demand nodes
    d(i,j) = distance matrix
    p = number of facilities
Output:
    C = set of facilities
    D = maximum distance
1. Choose initial solution:
    Select c₁ ∈ I
    C ← {c₁}
2. While |C| < p do:
    For each i ∈ I:
        d_min(i) ← min { d(i,c) : c ∈ C }
    i* ← argmax { d_min(i) : i ∈ I }
    C ← C ∪ {i*}
End While
3. Compute objective value:
    For each i ∈ I:
        d_min(i) ← min { d(i,c) : c ∈ C }
     D ← max { d_min(i) : i ∈ I }
Return C, D
```

4.1.2 Explanation

This constructive heuristic describes a greedy constructive heuristic for the p-center problem based on a farthest-first strategy. The method begins by selecting an initial node c1 and setting it as the first facility in the solution.

At each iteration, the algorithm computes, for every demand node i, the minimum distance to the current set of facilities C. This value represents how well each node is served by the existing facilities. The node i∗ with the largest of these distances is then selected, as it corresponds to the worst-served node. This node is added to the set of facilities, and the process is repeated until p facilities have been selected. In this way, the algorithm progressively improves coverage by focusing on the nodes that are farthest from the current solution.

Finally, the solution is evaluated by computing the maximum distance between each demand node and its nearest facility, which defines the objective value D. The algorithm returns the selected facilities C and the corresponding maximum distance.

4.1.3 Illustrative Example

Demand nodes: $I = \{1,2,3,4,5\}$

Distance Matrix:

$d(i,j)$	1	2	3	4	5
1	0	2	6	7	9
2	2	0	4	6	8
3	6	4	0	3	5
4	7	6	3	0	4
5	9	8	5	4	0

TABLE 3. DISTANCE MATRIX FROM THE NODE.

Number of facilities: $p = 2$

Now, let's pick $c_1 = 1$

So, $C = \{1\}$

While: $|C| < p = 1 < 2$

We need one more facility

Compute $d_{min}(i)$ for each node:

- **Node 1:** $d_{min}(1) = d(1,1) = 0$

- **Node 2:** $d_{min}(2) = d(2,1) = 2$
- **Node 3:** $d_{min}(3) = d(3,1) = 6$
- **Node 4:** $d_{min}(4) = d(4,1) = 7$
- **Node 5:** $d_{min}(5) = d(5,1) = 9$

The farthest node is the Node 5, so our $i^* = 5$

We update $C = \{1, 5\}$
Now, we evaluate the solution. For each node, compute distance to nearest facility:

- **Node 1:** $min(0, 9) = 0$
- **Node 2:** $min(2, 8) = 2$
- **Node 3:** $min(6, 5) = 5$
- **Node 4:** $min(7, 4) = 4$
- **Node 5:** $min(9, 0) = 0$

Maximum of these = 5
$D = 5$

Final Result

- **Facilities chosen:** $C = \{1, 5\}$
- **Objective value:** $D = 5$

4.2 Constructive Heuristic 2

4.2.1 Pseudocode

```
Input:
    I = set of demand nodes
    d(i,j) = distance matrix
    p = number of facilities
Output:
    C = set of facilities
    D = maximum distance
1. Initialize:
    C ← ∅
2. While |C| < p do:
    For each i ∈ I not in C:
        Compute:
```

```
        D_i ← max { min d(j,c) , d(j,i) } for all j ∈ I, c ∈ C
    i* ← argmin { D_i : i ∈ I not in C }
    C ← C ∪ {i*}
End While
3. Compute objective value:
    For each i ∈ I:
        d_min(i) ← min { d(i,c) : c ∈ C }
    D ← max { d_min(i) : i ∈ I }
Return C, D
```

4.2.2 Explanation

This pseudocode describes a greedy constructive heuristic based on a minimax strategy for the p-center problem. The method starts with an empty set of facilities C and iteratively builds the solution until p facilities are selected.

At each iteration, every candidate node i not yet in C is evaluated by temporarily considering it as a new facility. For each candidate, the algorithm computes D_i, which represents the maximum distance between all demand nodes and their nearest facility when node i is added to the current set. This evaluation considers both the already selected facilities and the candidate node.

The node $i*$ that minimizes this maximum distance is selected and added to the set C. In this way, each step aims to reduce the worst-case distance, which is consistent with the objective of the p-center problem.

Once p facilities have been chosen, the final solution is evaluated by assigning each demand node to its closest facility and computing the maximum distance D. The algorithm returns the selected facilities C and the corresponding objective value D.

4.2.3 Illustrative Example

Demand nodes: $I = \{1,2,3,4\}$

Distance Matrix:

$d(i,j)$	1	2	3	4
1	0	4	6	8
2	4	0	5	7
3	6	5	0	3

4	8	7	3	0

TABLE 4. DISTANCE MATRIX FROM THE NODE.

Number of facilities: $p = 2$

Initialize: Start with an empty set: $C = \varnothing$

Iteration 1: Evaluate each node as the first facility and we will identify the worst distance and pick the minimum.

$d(i,j)$	**1**	**2**	**3**	**4**
1	0	4	6	**8**
2	4	0	5	**7**
3	**6**	5	0	3
4	**8**	7	3	0

TABLE 5. ITERATION 1.

The minimum worst distance is the node 3, so we choose the node 3 as our first facility

Update: $C = \{3\}$

Iteration 2: Now test adding another facility

Candidate $i = 1$

- **Node 1:** $min(6, 0) = 0$
- **Node 2:** $min(5, 4) = 4$
- **Node 3:** $min(0, 6) = 0$
- **Node 4:** $min(3, 8) = 3$

Candidate $i = 2$

- **Node 1:** $min(6, 4) = 4$
- **Node 2:** $min(5, 0) = 0$
- **Node 3:** $min(0, 5) = 0$
- **Node 4:** $min(3, 7) = 3$

Candidate $i = 4$

- **Node 1:** $min(6, 8) = 6$
- **Node 2:** $min(5, 7) = 5$
- **Node 3:** $min(0, 3) = 0$

- **Node 4:** $min(3, 0) = 0$

$D_1 = 4,\ D_2 = 4,\ D_4 = 6$

So, our i^* could be the node 1 or 2
We take $C = \{1, 3\}$

Now, we evaluate the solution. For each node, compute distance to nearest facility:

- **Node 1:** $min(0, 6) = 0$
- **Node 2:** $min(4, 5) = 4$
- **Node 3:** $min(6, 0) = 0$
- **Node 4:** $min(8, 3) = 3$

Maximum of these = 4
$D = 4$

Final Result

- **Facilities chosen:** $C = \{1, 3\}$
- **Objective value:** $D = 4$

4.3 Local Search Heuristic 1 – Best Improvement

4.3.1 Pseudocode

```
Input:
      X ← initial solution obtained from Constructive Heuristic 1 (p centers)
Output:
      X_best

X_best ← X
Improve ← 1
While (Improve = 1) do:
      Improve ← 0
      Generate N(X_best) by swapping one facility i ∈ X_best with one node j
∉ X_best
      X* ← argmin { f(x') : x' ∈ N(X_best) }
      If f(X*) < f(X_best) then:
      X_best ← X*
      Improve ← 1
      End If
End While
```

```
Return X_best
```

4.3.2 Explanation

This pseudocode describes a Best Improvement local search strategy applied over the initial solution produced by Constructive Heuristic 1. The method begins with the set of p facilities returned by the farthest-first constructive procedure and designates it as the current best solution X_best.

At each iteration, the algorithm generates the complete neighborhood N(X_best) by applying a 1-swap move: one facility currently in the solution is replaced by a node that does not belong to it. Every neighboring solution in N(X_best) is evaluated using the objective function f, which computes the maximum distance between any demand node and its nearest facility.

The algorithm selects the best neighbor X*, defined as the solution that minimizes f over all candidates in the neighborhood. If this neighbor strictly improves upon X_best, it replaces it and the search continues. Otherwise, no improving neighbor exists and the algorithm terminates, returning the locally optimal solution found.

4.3.3 Illustrative Example

Initial Data:

Consider the set of demand nodes:

$$I=\{1,2,3,4\},P=2$$

The initial solution is generated using the constructive heuristic.

Distance Matrix:

$d(i,j)$	1	2	3	4
1	0	4	6	8
2	4	0	5	7
3	6	5	0	3
4	8	7	3	0

TABLE 6. DISTANCE MATRIX FROM NODE.

Initial solution X_best (from farthest-first heuristic)

The constructive heuristic returns the initial solution:

$$X\_best = \{1,3\}$$

This means that facilities are located at nodes 1 and 3. To evaluate the solution, the nearest facility for each node is identified.

Node	Nearest Facility	Distance
1	Facility 1	0
2	Facility 1	4
3	Facility 3	0
4	Facility 3	3

TABLE 7. ITERATION 1.

The objective function is calculated as:

$$f(X\_best)=\max(di)=4$$

Therefore, the current best solution is:

$$X\_best=\{1,3\}\ ,\ f(X\_best)=4$$

Neighborhood Generation Using 1-Swap Moves

A 1-swap move removes one facility from the current solution and replaces it with a node that is not currently selected.

Since:

$$X\_best=\{1,3\}$$

and the non-selected nodes are:

$$\{2,4\}$$

a total of 4 neighboring solutions are evaluated.

Neighbor 1: Replace 1 with 2

$$X'=\{2,3\}$$

Node Assignment

Node	Nearest Facility	Distance
1	Facility 2	4
2	Facility 2	0
3	Facility 3	0
4	Facility 3	3

TABLE 8. NODE ASSIGNMENT.

$$f(X')=4$$

Neighbor 2: Replace 1 with 4

$$X'=\{3,4\}$$

Node Assignment

Node	Nearest Facility	Distance
1	Facility 3	6
2	Facility 3	5
3	Facility 3	0
4	Facility 4	0

TABLE 9. NODE ASSIGNMENT ITERATION 2.

$$f(X')=6$$

Neighbor 3: Replace 3 with 2

$$X'=\{1,2\}$$

Node Assignment

Node	Nearest Facility	Distance

1	Facility 1	0
2	Facility 2	0
3	Facility 2	5
4	Facility 2	7

TABLE 10. ITERATION 3 NODE ASSIGNMENT.

$$f(X')=7$$

Neighbor 4: Replace 3 with 4

$$X'=\{1,4\}$$

Node Assignment

Node	Nearest Facility	Distance
1	Facility 1	0
2	Facility 1	4
3	Facility 4	3
4	Facility 4	0

TABLE 11. ITERATION 4 NODE ASSIGNMENT.

$$f(X')=4$$

Neighbor Comparison

All neighboring solutions are compared using the objective function value.

Neighbor	Solution	f(X′)
Neighbor 1	{2,3}	4
Neighbor 2	{3,4}	6
Neighbor 3	{1,2}	7

Neighbor 4	{1,4}	4

TABLE 12. NEIGHBOR COMPARISON.

The smallest objective value obtained is:

$$f(X')=4$$

However, this value does not improve the current solution because:

$$f(X')=f(X\_best)$$

Final Result

Since no neighboring solution improves the current objective value, the algorithm stops.

Therefore, the local optimum obtained is:

$$X\_best=\{1,3\}$$

with:

$$f(X\_best)=4$$

This solution is considered the final locally optimal solution for the example.

4.4 Local Search Heuristic 1 – First Improvement

4.4.1 Pseudocode

```
Input:
      X ← initial solution obtained from Constructive Heuristic 1 (p centers)
Output:
      X_best

X_best ← X
Improve ← 1
While (Improve = 1) do:
      Improve ← 0
      Generate N(X_best) by swapping one facility i ∈ X_best with one node j
∉ X_best
      For each x' ∈ N(X_best) do:
      If f(x') < f(X_best) then:
            X_best ← x'
```

```
            Improve ← 1
            Break
        End If
        End For
End While
Return X_best
```

4.4.2 Explanation

This pseudocode describes a First Improvement local search strategy applied over the initial solution produced by Constructive Heuristic 1. The method begins with the set of p facilities returned by the farthest-first constructive procedure and designates it as the current best solution X_best.

At each iteration, the algorithm generates the neighborhood N(X_best) using a 1-swap move, where one facility in the current solution is exchanged with a node that is not part of it. The neighboring solutions are examined one by one in sequence. Instead of evaluating all neighbors before making a decision, the algorithm immediately accepts the first solution x' that improves the objective function f. Once such a neighbor is found, it replaces X_best, and the search resumes from this updated solution.

If no improving neighbor is encountered throughout the full neighborhood traversal, the algorithm terminates and returns the locally optimal solution X_best. Compared to the Best Improvement variant, this strategy requires less computational effort per iteration, since the neighborhood need not be exhaustively evaluated; however, the improvement accepted at each step may not be the largest possible one.

4.4.3 Illustrative Example

Initial Data:

Consider the set of demand nodes:

$$I=\{1,2,3,4\}\ ,\ P=2$$

The initial solution is generated using the constructive heuristic.

Distance Matrix:

$d(i,j)$	1	2	3	4
1	0	4	6	8
2	4	0	5	7
3	6	5	0	3
4	8	7	3	0

TABLE 13. DISTANCE MATRIX FROM NODE.

Initial Solution from Constructive Heuristic

The constructive heuristic generates the initial solution:

$$X\_best=\{1,3\}$$

This means that facilities are located at nodes 1 and 3.

Assignment of Nodes to Facilities

Node	Nearest Facility	Distance
1	Facility 1	0
2	Facility 1	4
3	Facility 3	0
4	Facility 3	3

TABLE 14. ITERATION 1 NODE ASSIGNMENT.

The objective function value is:

$$f(X\_best)=\max(di)=4$$

Therefore:

$$X\_best=\{1,3\}\text{ , }f(X\_best)=4$$

First Improvement Strategy

Unlike Best Improvement, which evaluates all neighbors and selects the best one, the First Improvement strategy evaluates neighbors sequentially and accepts the first solution that improves the objective value. The neighborhood is explored using 1-swap moves.

The evaluation order is:

$$(-1,+2),\ (-1,+4),\ (-3,+2),\ (-3,+4)$$

If a neighbor improves the current solution, the search stops immediately and the algorithm moves to the new solution.

Neighborhood Evaluation

Neighbor 1: Replace 1 with 2

$$X'=\{2,3\}$$

Node Assignment

Node	Nearest Facility	Distance
1	Facility 2	4
2	Facility 2	0
3	Facility 3	0
4	Facility 3	3

TABLE 15. ITERATION 2 NODE ASSIGNMENT.

$$f(X')=4$$

Since:

$$f(X')=f(X\_best)$$

there is no improvement, so the algorithm continues evaluating the next neighbor.

Neighbor 2: Replace 1 with 4

$$X'=\{3,4\}$$

Node Assignment

Node	Nearest Facility	Distance
1	Facility 3	6
2	Facility 3	5
3	Facility 3	0
4	Facility 4	0

TABLE 16. ITERATION 4 NODE ASSIGNMENT.

$$f(X')=6$$

Since:

$$f(X')>f(X\_best)$$

The solution is worse, so the search continues.

Neighbor 3: Replace 3 with 2

$$X'=\{1,2\}$$

Node Assignment

Node	Nearest Facility	Distance
1	Facility 1	0
2	Facility 2	0
3	Facility 2	5
4	Facility 2	7

TABLE 17. ITERATION 5 NODE ASSIGNMENT.

$$f(X')=7$$

This solution is also worse than the current solution, so the search continues.

Neighbor 4: Replace 3 with 4

$$X'=\{1,4\}$$

Node Assignment

Node	Nearest Facility	Distance
1	Facility 1	0
2	Facility 1	4
3	Facility 4	3
4	Facility 4	0

TABLE 18. ITERATION 6 NODE ASSIGNMENT.

$$f(X')=4$$

Again:

$$f(X')=f(X\_best)$$

Therefore, no improvement is found.

Evaluation Summary

Neighbor	Solution	f(X′)	Result
Neighbor 1	{2,3}	4	No improvement
Neighbor 2	{3,4}	6	Worse solution
Neighbor 3	{1,2}	7	Worse solution
Neighbor 4	{1,4}	4	No improvement

TABLE 19. SUMMARY OF RESULTS.

No neighboring solution improves the current objective value.

Final Result

Since no improvement is found during the neighborhood exploration, the algorithm terminates.

The final local optimum is:

$$X\_best=\{1,3\}$$

with objective value:

$$f(X\_best)=4$$

This example shows that the First Improvement strategy reaches the same local optimum as Best Improvement, but it may require fewer evaluations when an improving neighbor is found early in the search process.

SECTION 5: COMPUTATIONAL WORK

In this section, the proposed heuristics will be implemented and tested on different instances of the p-Center Problem. The objective is to evaluate the computational performance of each method in terms of solution quality and execution time. The experiments will compare the constructive heuristics and local search procedures using distance matrices of different sizes, analyzing how effectively each approach minimizes the maximum service distance. The obtained results will allow us to identify the strengths and limitations of each heuristic strategy from a practical computational perspective.

5.1 EXPERIMENT 1: CH VS LS (Best improvement)

This experiment compares the Constructive Heuristic (CH) with the Best Improvement Local Search (LS). The objective is to evaluate how much the local search procedure can improve the initial solution generated by the constructive heuristic. The comparison considers both solution quality and computational time for different instance sizes. Since local search explores neighboring solutions iteratively, it is expected to obtain better objective values than the constructive approach, although at the expense of higher execution times.

5.1.1 Small Instances

SET SMALL: n = 500						
INSTANCE	CH VALUE	CH_TIME (CPU SEC)	LS VALUE	LS_TIME (CPU SEC)	ABSOLUTE IMPROVEMENT	RELATIVE IMPROVEMENT
data1000_1	1566.5867	0.000000	1403.1180	0.201000	163.4687	10.4347%
data1000_2	1651.5378	0.000000	1380.5868	0.797000	270.9510	16.4060%
data1000_3	1664.0192	0.000000	1435.6775	0.300000	228.3417	13.7223%
data1000_4	1660.2918	0.000000	1429.2634	0.307000	231.0284	13.9149%
data1000_5	1577.6061	0.000000	1507.0269	0.147000	70.5792	4.4738%
data1000_6	1538.5464	0.000000	1362.1307	0.101000	176.4157	11.4664%
data1000_7	1287.5655	0.000000	1194.0976	0.991000	93.4679	7.2593%
data1000_8	1853.2809	0.000000	1738.2649	0.113000	115.0160	6.2061%
data1000_9	1664.4428	0.000000	1649.8830	0.053000	14.5598	0.8748%
data1000_10	1048.5633	0.000000	923.3125	0.810000	125.2508	11.9450%

data1000_11	1533.8230	0.000000	1324.7339	0.440000	209.0891	13.6319%
data1000_12	1155.2255	0.000000	1097.1472	0.531000	58.0783	5.0274%
data1000_13	2007.5981	0.000000	1853.7057	0.079000	153.8924	7.6655%
data1000_14	2168.9444	0.000000	1996.3569	0.059000	172.5875	7.9572%
data1000_15	3321.7208	0.000000	2155.2578	0.132000	1166.4630	35.1162%
data1000_16	1889.6140	0.000000	1658.3202	0.222000	231.2938	12.2403%
data1000_17	1693.2484	0.000000	1532.7182	0.146000	160.5302	9.4806%
data1000_18	2408.1902	0.000000	1956.9704	0.143000	451.2198	18.7369%
data1000_19	2223.6205	0.000000	1772.9774	0.338000	450.6431	20.2662%
data1000_20	1639.3737	0.000000	1488.0457	0.196000	151.3280	9.2308%
number of winning times	0		20		Average REL IMP	11.8028%

TABLE 20. RESULTS OF EXPERIMENT 1 FOR SMALL INSTANCES.

This table presents the comparison between the Constructive Heuristic (CH) and the Best Improvement Local Search (LS) for small instances. The results show that LS improved the constructive solution in all 20 experiments, obtaining lower objective values in every case. This confirms that the local search procedure is capable of refining the initial solution generated by CH and reducing the maximum service distance of the p-Center Problem.

The average relative improvement was 11.8028%, which represents a considerable enhancement in solution quality. However, the improvement percentages vary significantly between instances. In some cases, the relative improvement is below 1%, indicating that the constructive heuristic already generated a solution close to a local optimum. On the other hand, some instances achieved improvements greater than 30%, meaning that the initial constructive solution still had substantial room for optimization.

Regarding computational performance, CH required practically no execution time because it directly constructs a feasible solution without performing iterative refinements. In contrast, LS required additional CPU time because it evaluates multiple neighboring solutions before selecting the best movement at each iteration. Even so, all execution times remained below one second, which indicates that the Best Improvement strategy is computationally efficient for small-scale problems. Overall, the results demonstrate that applying local search after the constructive phase considerably improves solution quality while maintaining acceptable computational times.

5.1.2 Medium Instances

SET MEDIUM: n = 1000						
INSTANCE	**CH VALUE**	**CH_TIME (CPU SEC)**	**LS VALUE**	**LS_TIME (CPU SEC)**	**ABSOLUTE IMPROVEMENT**	**RELATIVE IMPROVEMENT**
data5000_1	1108.8214	0.000000	1004.3112	3.365000	104.510200	9.4253%
data5000_2	1106.0280	0.000000	1086.0373	0.996000	19.990700	1.8074%
data5000_3	1107.2741	0.000000	995.1251	2.845000	112.149000	10.1283%
data5000_4	1150.6120	0.000000	1033.0252	3.232000	117.586800	10.2195%

data5000_5	1265.2371	0.000000	1088.0869	3.278000	177.150200	14.0013%
data5000_6	1118.1596	0.000000	1079.7838	1.712000	38.375800	3.4320%
data5000_7	1098.2213	0.000000	954.3762	1.180000	143.845100	13.0980%
data5000_8	1118.9553	0.000000	998.9795	1.711000	119.975800	10.7221%
data5000_9	1102.2944	0.000000	1003.6419	2.784000	98.652500	8.9497%
data5000_10	1210.2202	0.000000	1042.9046	2.559000	167.315600	13.8252%
data5000_11	1216.6133	0.000000	1141.1223	2.757000	75.491000	6.2050%
data5000_12	1294.7409	0.000000	1164.9893	2.186000	129.751600	10.0214%
data5000_13	1130.0040	0.000000	986.2682	2.781000	143.735800	12.7199%
data5000_14	1170.4636	0.000000	1010.0703	4.084000	160.393300	13.7033%
data5000_15	1132.5317	0.000000	1046.0043	3.802000	86.527400	7.6401%
data5000_16	1130.4588	0.014000	1114.4528	0.758000	16.006000	1.4158%
data5000_17	1182.8884	0.000000	987.9474	5.475000	194.941000	16.4800%
data5000_18	1078.6552	0.000000	1030.5115	0.964000	48.143700	4.4633%
data5000_19	1201.7729	0.000000	1100.3890	2.181000	101.383900	8.4361%
data5000_20	1143.1645	0.000000	1029.8762	2.158000	113.288300	9.9100%
number of winning times	0		20		Average REL IMP	9.3302%

TABLE 21. RESULTS OF EXPERIMENT 1 FOR MEDIUM INSTANCES.

For medium-size instances, the Best Improvement Local Search continues outperforming the Constructive Heuristic in all experiments. LS achieved 20 winning cases out of 20, confirming that the local search phase consistently improves the initial constructive solution regardless of the instance analyzed.

The average relative improvement obtained was 9.3302%. Although this value is slightly lower than the one observed in small instances, it still represents an important reduction in the objective function. This suggests that the constructive heuristic generates relatively better initial solutions as the problem size increases, leaving less room for later optimization. Several instances still present improvements above 10%, which indicates that local search remains highly effective for refining facility locations and reducing the maximum distance between demand nodes and selected centers. In terms of computational time, the difference between CH and LS becomes more noticeable. Since medium-size instances contain larger neighborhoods, the local search algorithm must evaluate a significantly higher number of candidate solutions at every iteration. As a consequence, CPU times increase compared to small instances.

Nevertheless, the execution times are still reasonable considering the quality improvements obtained. Therefore, the Best Improvement strategy maintains a good balance between computational effort and solution quality for medium-size problems.

5.1.2 Large Instances

SET LARGE: n = 2000						
INSTANCE	**CH VALUE**	**CH_TIME (CPU SEC)**	**LS VALUE**	**LS_TIME (CPU SEC)**	**ABSOLUTE IMPROVEMENT**	**RELATIVE IMPROVEMENT**
data10000_1	792.0227	0.005000	739.7736	27.322000	52.249100	6.5969%
data10000_2	807.1443	0.005000	759.3978	20.594000	47.746500	5.9154%
data10000_3	799.0150	0.000000	742.8923	24.502000	56.122700	7.0239%
data10000_4	811.3279	0.003000	730.6415	32.633000	80.686400	9.9449%
data10000_5	1107.4787	0.001000	985.3979	29.336000	122.080800	11.0233%
data10000_6	780.2083	0.013000	739.8946	22.653000	40.313700	5.1670%
data10000_7	784.9395	0.008000	748.4517	29.058000	36.487800	4.6484%
data10000_8	795.3898	0.002000	728.9753	24.255000	66.414500	8.3499%
data10000_9	780.4672	0.010000	743.8259	19.883000	36.641300	4.6947%
data10000_10	775.5050	0.000000	731.9324	28.093000	43.572600	5.6186%
data10000_11	992.1275	0.002300	920.2282	12.232000	71.899300	7.2469%
data10000_12	869.1657	0.016000	746.0965	38.651000	123.069200	14.1594%
data10000_13	791.6375	0.005000	747.2998	25.849000	44.337700	5.6007%
data10000_14	806.1675	0.004000	765.9772	13.417000	40.190300	4.9853%
data10000_15	853.6633	0.002000	769.3120	39.662000	84.351300	9.8810%
data10000_16	865.0017	0.001000	773.8863	14.426000	91.115400	10.5335%
data10000_17	816.0815	0.005000	725.1103	37.226000	90.971200	11.1473%
data10000_18	815.8394	0.000000	763.2123	26.485000	52.627100	6.4506%
data10000_19	806.8562	0.004000	726.0337	21.609000	80.822500	10.0169%
data10000_20	791.1618	0.004000	777.0927	2.408000	14.069100	1.7782%
number of winning times	0		20		Average REL IMP	7.5392%

TABLE 22. RESULTS OF EXPERIMENT 1 FOR LARGE INSTANCES.

In large-scale instances, the Best Improvement Local Search again obtained better objective values than the Constructive Heuristic in all experiments. The algorithm achieved 20 winning cases out of 20, demonstrating that the local search procedure remains effective even when the problem size becomes significantly larger.

The average relative improvement was 7.5392%, which is lower than the percentages obtained for small and medium instances. This behavior suggests that the constructive heuristic becomes more competitive for larger problems, generating initial solutions that are already relatively close to good local optima.

Even though the average improvement decreases, several instances still show improvements above 10%. This indicates that local search continues identifying beneficial exchanges between facilities and demand nodes, allowing further reductions in the maximum service distance. The most significant change appears in computational time. Because the neighborhood size grows considerably for large

instances, LS requires much more CPU time to evaluate possible movements and identify the best improvement at every iteration. In some cases, execution times reach several tens of seconds.

Despite the increase in computational effort, the improvements in solution quality justify the additional runtime. Therefore, the local search strategy remains valuable for large-scale problems where obtaining better service distances is more important than minimizing execution time.

5.2 EXPERIMENT 2: CH1 VS CH2

This experiment compares two constructive heuristics, CH1 and CH2, for solving the p-Center Problem. The objective is to determine which construction strategy generates better initial solutions and how both methods behave as the problem size increases. Since constructive heuristics define the starting point for later optimization procedures, obtaining high-quality initial solutions is essential for improving the overall performance of the algorithms.

5.2.1 Small Instances

SET SMALL: n = 500						
INSTANCE	**CH1 VALUE**	**CH1_TIME (CPU SEC)**	**CH2 VALUE**	**CH2_TIME (CPU SEC)**	**ABSOLUTE IMPROVEMENT**	**RELATIVE IMPROVEMENT**
data1000_1	1843.2189	0.000000	1796.3719	0.027000	46.8470	2.5416%
data1000_2	1531.9011	0.000000	1869.478	0.032000	-337.5769	-22.0365%
data1000_3	1772.7315	0.000000	1867.5794	0.028000	-94.8479	-5.3504%
data1000_4	1674.6134	0.000000	1801.0044	0.021000	-126.3910	-7.5475%
data1000_5	1743.0677	0.000000	1958.0952	0.023000	-215.0275	-12.3362%
data1000_6	1854.9404	0.001000	1854.9501	0.047000	-0.0097	-0.0005%
data1000_7	1651.2847	0.000000	1865.5254	0.048000	-214.2407	-12.9742%
data1000_8	1662.8364	0.000000	1761.2405	0.040000	-98.4041	-5.9178%
data1000_9	1754.4928	0.000000	1828.4160	0.041000	-73.9232	-4.2134%
data1000_10	1573.4462	0.000000	1752.0925	0.034000	-178.6463	-11.3538%
data1000_11	1755.8189	0.000000	2010.2860	0.024000	-254.4671	-14.4928%
data1000_12	1538.2822	0.001000	1662.1655	0.037000	-123.8833	-8.0534%
data1000_13	1948.2141	0.000000	2091.6207	0.036000	-143.4066	-7.3609%
data1000_14	1658.2985	0.004000	1797.2868	0.031000	-138.9883	-8.3814%
data1000_15	1679.5824	0.000000	1729.9549	0.041000	-50.3725	-2.9991%
data1000_16	1600.3181	0.000000	1709.0655	0.037000	-108.7474	-6.7954%
data1000_17	1746.9116	0.000000	1865.0775	0.021000	-118.1659	-6.7643%
data1000_18	1464.1817	0.004000	1616.4684	0.040000	-152.2867	-10.4008%
data1000_19	1784.7983	0.000000	2021.1294	0.052000	-236.3311	-13.2413%
data1000_20	1955.3261	0.000000	2178.5738	0.022000	-223.2477	-11.4174%

number of winning times	19	1	Average REL IMP	-8.4548%

TABLE 23. RESULTS OF EXPERIMENT 2 FOR SMALL INSTANCES.

This table compares Constructive Heuristic 1 (CH1) and Constructive Heuristic 2 (CH2) for small instances. The results clearly show that CH1 produces better solutions in almost every experiment, obtaining 19 winning cases out of 20. The average relative improvement was -8.4548%, meaning that CH2 generally generated worse objective values than CH1. In several cases, the deterioration exceeded 10%, indicating that the second constructive strategy is less effective for selecting facility locations that minimize the maximum service distance.

One possible explanation is that CH1 follows a more effective selection mechanism during the construction phase, allowing it to generate more balanced center distributions. In contrast, CH2 appears to make less efficient choices during some iterations, which negatively impacts the final solution quality. Regarding execution time, both heuristics required very small CPU times due to the simplicity of constructive procedures. However, CH2 consistently required more time because it performs additional evaluations while selecting candidate facilities. Overall, the results indicate that CH1 is both faster and more effective for small instances.

5.2.2 Medium Instances

SET MEDIUM: n = 1000						
INSTANCE	**CH1 VALUE**	**CH1_TIME (CPU SEC)**	**CH2 VALUE**	**CH2_TIME (CPU SEC)**	**ABSOLUTE IMPROVEMENT**	**RELATIVE IMPROVEMENT**
data5000_1	1139.7022	0.008000	1323.7099	0.171000	-184.007700	-16.1452%
data5000_2	1133.1390	0.005000	1355.0399	0.238000	-221.900900	-19.5828%
data5000_3	1061.5767	0.006000	1181.1808	0.239000	-119.604100	-11.2666%
data5000_4	1129.6004	0.000000	1334.7120	0.226000	-205.111600	-18.1578%
data5000_5	1120.3504	0.005000	1278.2226	0.263000	-157.872200	-14.0913%
data5000_6	1207.1541	0.005000	1315.5261	0.189000	-108.372000	-8.9774%
data5000_7	1147.9974	0.000000	1273.5042	0.187000	-125.506800	-10.9326%
data5000_8	1005.3721	0.010000	1194.9816	0.223000	-189.609500	-18.8596%
data5000_9	1040.4941	0.000000	1249.3558	0.244000	-208.861700	-20.0733%
data5000_10	1067.0033	0.005000	1252.4065	0.210000	-185.403200	-17.3760%
data5000_11	1129.7084	0.000000	1393.3564	0.222000	-263.648000	-23.3377%
data5000_12	1055.1986	0.004000	1250.2788	0.276000	-195.080200	-18.4875%
data5000_13	1090.4041	0.000000	1244.3653	0.278000	-153.961200	-14.1196%
data5000_14	1352.6955	0.004000	1553.3721	0.160000	-200.676600	-14.8353%
data5000_15	1237.1035	0.005000	1435.1226	0.190000	-198.019100	-16.0066%
data5000_16	1059.1959	0.008000	1293.4763	0.203000	-234.280400	-22.1187%
data5000_17	1102.7257	0.004000	1379.7753	0.223000	-277.049600	-25.1240%
data5000_18	1134.0216	0.004000	1413.3775	0.206000	-279.355900	-24.6340%
data5000_19	1118.8141	0.000100	1280.1973	0.192000	-161.383200	-14.4244%

data5000_20	1231.8385	0.003000	1340.7076	0.164000	-108.869100	-8.8379%
number of winning times	20		0		Average REL IMP	-16.8695%

TABLE 24. RESULTS OF EXPERIMENT 2 FOR MEDIUM INSTANCES.

For medium-size instances, the superiority of CH1 becomes even more evident. CH1 obtained better objective values in all 20 experiments, while CH2 failed to outperform CH1 in any case.

The average relative improvement reached -16.8695%, showing that the deterioration produced by CH2 becomes considerably larger as the problem size increases. In several instances, the difference exceeded 20%, which indicates that CH2 struggles to maintain solution quality for larger search spaces. These results suggest that the decision mechanism used by CH1 scales more effectively and is better suited for selecting facility locations in medium-size problems. Meanwhile, the choices made by CH2 tend to accumulate errors during construction, leading to poorer final solutions.

In terms of computational effort, CH2 also required noticeably larger execution times because of the greater number of candidate evaluations performed during the construction process. Consequently, CH1 not only provides better solutions but also achieves them with lower computational cost, making it the preferable constructive heuristic for medium-size instances.

5.2.3 Large Instances

SET LARGE: n = 2000						
INSTANCE	**CH1 VALUE**	**CH1_TIME (CPU SEC)**	**CH2 VALUE**	**CH2_TIME (CPU SEC)**	**ABSOLUTE IMPROVEMENT**	**RELATIVE IMPROVEMENT**
data10000_1	772.3341	0.024000	915.6184	1.266000	-143.284300	-18.5521%
data10000_2	802.7609	0.034000	906.2588	1.684000	-103.497900	-12.8927%
data10000_3	792.4557	0.037000	913.7494	1.488000	-121.293700	-15.3060%
data10000_4	836.9576	0.030000	912.5536	1.498000	-75.596000	-9.0322%
data10000_5	761.3212	0.042000	926.6677	1.466000	-165.346500	-21.7183%
data10000_6	820.1195	0.025000	939.5169	1.334000	-119.397400	-14.5585%
data10000_7	795.7701	0.033000	950.1521	1.483000	-154.382000	-19.4003%
data10000_8	851.9554	0.032000	979.6530	1.523000	-127.697600	-14.9887%
data10000_9	727.3575	0.051000	848.3684	1.865000	-121.010900	-16.6370%
data10000_10	931.2728	0.021000	1075.0377	1.195000	-143.764900	-15.4374%
data10000_11	805.4936	0.034000	979.2900	1.814000	-173.796400	-21.5763%
data10000_12	799.1808	0.038000	974.5112	1.517000	-175.330400	-21.9387%
data10000_13	806.7868	0.033000	911.8777	1.515000	-105.090900	-13.0258%
data10000_14	978.0644	0.014000	1219.7213	0.959000	-241.656900	-24.7076%
data10000_15	783.4858	0.034000	939.7133	1.593000	-156.227500	-19.9400%
data10000_16	1021.2159	0.015000	1303.3879	1.000000	-282.172000	-27.6309%
data10000_17	814.2954	0.046000	914.2237	1.489000	-99.928300	-12.2717%
data10000_18	845.1728	0.020000	1007.4483	1.362000	-162.275500	-19.2002%

data10000_19	846.0148	0.025000	981.7953	1.447000	-135.780500	-16.0494%
data10000_20	805.6035	0.035000	955.0916	1.538000	-149.488100	-18.5560%
number of winning times	20		0		Average REL IMP	-17.6710%

TABLE 25. RESULTS OF EXPERIMENT 2 FOR LARGE INSTANCES.

This table compares Constructive Heuristic 1 (CH1) and Constructive Heuristic 2 (CH2) for large-scale instances of the p-Center Problem. The results show that CH1 outperforms CH2 in all 20 experiments, obtaining better objective values in every tested instance. The average relative improvement was -17.6710%, which indicates that CH2 generally produces significantly worse solutions than CH1. In several cases, the deterioration exceeded 20%, suggesting that CH2 becomes less effective as the problem size increases. In contrast, CH1 maintains more stable and competitive solutions for large instances.

The computational times also reveal important differences. CH2 requires considerably more CPU time because it evaluates additional candidate solutions during the construction phase. As the instance size grows, these evaluations become more expensive computationally. Meanwhile, CH1 remains relatively fast while still obtaining better objective values. Overall, the results demonstrate that CH1 is both more efficient and more robust for large-scale problems.

5.3 EXPERIMENT 3: CH2 + LS1_BEST VS CH2 + LS2_FIRST

This experiment compares two local search strategies that start from the same initial solution generated by CH2. LS1 applies the Best Improvement strategy, while LS2 uses the First Improvement approach. The objective is to analyze the balance between solution quality and computational time. Since LS1 evaluates the complete neighborhood, it is expected to obtain slightly better solutions, whereas LS2 is expected to reduce execution times by accepting the first improving move found.

5.3.1 Small Instances

SET SMALL: n = 500						
INSTANCE	**LS1 VALUE**	**LS1_TIME (CPU SEC)**	**LS2 VALUE**	**LS2_TIME (CPU SEC)**	**ABSOLUTE IMPROVEMENT**	**RELATIVE IMPROVEMENT**
data1000_1	1376.6412	0.185000	1436.4585	0.059000	-59.8173	-4.3452%
data1000_2	1404.9434	0.151000	1404.7128	0.051000	0.2306	0.0164%
data1000_3	1396.3380	0.169000	1465.3873	0.090000	-69.0493	-4.9450%
data1000_4	1391.4485	0.155000	1391.4485	0.062000	0.0000	0.0000%
data1000_5	1444.1541	0.064000	1444.1541	0.035000	0.0000	0.0000%
data1000_6	1430.0013	0.323000	1381.0156	0.121000	48.9857	3.4256%
data1000_7	1423.4202	0.212000	1494.418	0.046000	-70.9978	-4.9878%
data1000_8	1427.5900	0.204000	1421.2703	0.068000	6.3197	0.4427%
data1000_9	1430.8099	0.068000	1430.8099	0.027000	0.0000	0.0000%
data1000_10	1379.7684	0.310000	1379.7684	0.102000	0.0000	0.0000%
data1000_11	1393.1260	0.285000	1374.1925	0.108000	18.9335	1.3591%
data1000_12	1468.7191	0.103000	1447.0079	0.027000	21.7112	1.4782%
data1000_13	1524.2998	0.096000	1524.2998	0.013000	0.0000	0.0000%
data1000_14	1448.1658	0.176000	1468.1281	0.113000	-19.9623	-1.3785%
data1000_15	1445.0447	0.140000	1445.0447	0.044000	0.0000	0.0000%

data1000_16	1428.6095	0.165000	1387.8588	0.041000	40.7507	2.8525%
data1000_17	1501.3127	0.161000	1446.5355	0.075000	54.7772	3.6486%
data1000_18	1445.9962	0.191000	1445.9962	0.032000	0.0000	0.0000%
data1000_19	1459.2340	0.172000	1414.3302	0.056000	44.9038	3.0772%
data1000_20	1453.9945	0.091000	1436.9513	0.040000	17.0432	1.1722%
number of winning times	4		9		Average REL IMP	0.0908%

TABLE 26. RESULTS OF EXPERIMENT 3 FOR SMALL INSTANCES.

This table compares the Best Improvement local search (LS1) with the First Improvement strategy (LS2) for small instances. The results show that both methods obtain very similar objective values, and several experiments even reach exactly the same final solution.

LS1 achieved better solutions in 4 cases, while LS2 obtained better results in 9 instances. The remaining 7 experiments ended in ties, showing that both methods frequently converge to the same local optimum. In addition, the average relative improvement was only 0.0908%, confirming that the differences in solution quality between both methods are generally very small. The main difference appears in computational time. LS2 is consistently faster because it stops the search as soon as an improving move is found, while LS1 evaluates the entire neighborhood before selecting the best movement. Although LS1 performs a more exhaustive search, the additional computational effort does not always produce significantly better solutions. Therefore, LS2 provides a good balance between execution time and solution quality for small instances.

5.3.2 Medium Instances

SET MEDIUM: n = 1000						
INSTANCE	**LS1 VALUE**	**LS1_TIME (CPU SEC)**	**LS2 VALUE**	**LS2_TIME (CPU SEC)**	**ABSOLUTE IMPROVEMENT**	**RELATIVE IMPROVEMENT**
data5000_1	1035.5172	1.745000	1038.7849	0.398000	-3.267700	-0.3156%
data5000_2	1052.5446	2.070000	1038.1715	0.425000	14.373100	1.3656%
data5000_3	980.3510	2.617000	981.7418	0.745000	-1.390800	-0.1419%
data5000_4	995.3944	1.600000	995.3944	0.637000	0.000000	0.0000%
data5000_5	983.9720	2.204000	963.3883	0.618000	20.583700	2.0919%
data5000_6	1004.5720	2.046000	1004.5720	0.556000	0.000000	0.0000%
data5000_7	1041.2012	1.203000	1061.6897	0.250000	-20.488500	-1.9678%
data5000_8	1031.9424	2.010000	1031.9424	0.543000	0.000000	0.0000%
data5000_9	1014.3082	2.119000	990.8184	0.467000	23.489800	2.3158%
data5000_10	1002.4515	2.065000	1064.5078	0.558000	-62.056300	-6.1905%
data5000_11	1018.3934	2.210000	969.0103	0.664000	49.383100	4.8491%
data5000_12	1010.5865	4.175000	1030.8491	0.703000	-20.262600	-2.0050%
data5000_13	991.9299	3.172000	986.2297	0.950000	5.700200	0.5747%
data5000_14	1045.0192	2.101000	1015.3403	0.650000	29.678900	2.8400%
data5000_15	975.8099	2.243000	967.1882	0.733000	8.621700	0.8835%
data5000_16	988.2449	1.837000	1012.3364	0.400000	-24.091500	-2.4378%
data5000_17	954.4045	2.608000	954.4045	0.694000	0.000000	0.0000%
data5000_18	1060.9547	1.398000	1016.6459	0.414000	44.308800	4.1763%
data5000_19	989.8005	1.616000	995.6074	0.596000	-5.806900	-0.5867%

data5000_20	1001.0974	1.175000	994.2057	0.385000	6.891700	0.6884%
number of winning times	7		9		Average REL IMP	0.3070%

TABLE 27. RESULTS OF EXPERIMENT 3 FOR MEDIUM INSTANCES.

For medium-size instances, LS1 and LS2 continue producing highly similar objective values. Some experiments favor LS1, while others favor LS2, which explains why the relative improvement values contain both positive and negative percentages. LS1 obtained better solutions in 7 cases, while LS2 performed better in 9 instances. The remaining 4 experiments ended in ties, indicating that both strategies often converge to very similar local optima. Furthermore, the average relative improvement was only 0.3070%, showing that the differences in solution quality are generally very small.

LS1 evaluates all neighboring solutions at every iteration, which increases the computational effort. In contrast, LS2 accepts the first improving movement found, reducing the number of evaluations required during the search process. As a result, LS2 requires considerably less CPU time while still obtaining competitive solutions. Therefore, LS2 offers a better balance between computational efficiency and solution quality for medium-size instances.

5.3.3 Large Instances

SET LARGE: n = 2000						
INSTANCE	**LS1 VALUE**	**LS1_TIME (CPU SEC)**	**LS2 VALUE**	**LS2_TIME (CPU SEC)**	**ABSOLUTE IMPROVEMENT**	**RELATIVE IMPROVEMENT**
data10000_1	707.7803	39.431000	707.7803	9.780000	0.000000	0.0000%
data10000_2	733.8351	36.435000	730.8221	10.902000	3.013000	0.4106%
data10000_3	696.0417	39.778000	673.3417	10.131000	22.700000	3.2613%
data10000_4	690.5071	49.411000	729.9370	9.110000	-39.429900	-5.7103%
data10000_5	745.9149	29.272000	745.9149	6.216000	0.000000	0.0000%
data10000_6	707.3571	37.008000	713.6561	9.131000	-6.299000	-0.8905%
data10000_7	691.3559	38.527000	801.1548	2.267000	-109.798900	-15.8817%
data10000_8	736.7062	34.890000	691.4861	9.993000	45.220100	6.1381%
data10000_9	718.2485	31.832000	699.1452	9.779000	19.103300	2.6597%
data10000_10	712.0674	44.360000	702.4429	12.445000	9.624500	1.3516%
data10000_11	694.2751	48.222000	699.4662	7.227000	-5.191100	-0.7477%
data10000_12	767.8418	27.284000	742.9199	7.515000	24.921900	3.2457%
data10000_13	735.0170	38.159000	716.2472	7.904000	18.769800	2.5537%
data10000_14	761.6101	28.011000	761.8766	7.397000	-0.266500	-0.0350%
data10000_15	710.3478	39.730000	716.5927	9.505000	-6.244900	-0.8791%
data10000_16	696.3749	55.290000	704.8475	11.300000	-8.472600	-1.2167%
data10000_17	744.4421	28.170000	694.8792	10.274000	49.562900	6.6577%
data10000_18	730.2671	34.656000	739.1136	7.303000	-8.846500	-1.2114%
data10000_19	763.0865	46.705000	740.0169	11.649000	23.069600	3.0232%

data10000_20	726.8370	46.413000	753.4753	11.036000	-26.638300	-3.6650%
number of winning times	9		9		Average REL IMP	-0.0468%

TABLE 28. RESULTS OF EXPERIMENT 3 FOR LARGE INSTANCES.

In large-scale instances, the difference in computational time between LS1 and LS2 becomes much more significant. LS1 requires substantially higher CPU times because evaluating the complete neighborhood becomes increasingly expensive as the problem size grows.

In terms of solution quality, neither strategy clearly dominates the other. Both LS1 and LS2 obtained 9 winning cases, while the remaining 2 experiments ended in ties. Additionally, the average relative improvement was very close to zero (-0.0468%), indicating that both methods generally obtain very similar solutions. Although LS1 occasionally achieves slightly better objective values, the improvements are usually small compared to the additional computational time required. Meanwhile, LS2 greatly reduces execution time by accepting the first improving movement found during the search. Therefore, LS2 becomes a more practical alternative for large-scale problems where execution time is especially important

SECTION 6: CONCLUSIONS

Throughout this project, our team gained a deeper understanding of the p-Center Problem and the importance of optimization techniques in real-world decision-making. By studying both the mathematical model and the computational behavior of the heuristics, we learned how facility location problems aim to provide equitable service coverage by minimizing the maximum distance between demand nodes and facilities.

The computational experiments allowed us to compare different heuristic strategies and analyze the relationship between solution quality and execution time. The results showed that Constructive Heuristic 1 generally produced better initial solutions than Constructive Heuristic 2, while the local search procedures were able to further improve those solutions. In particular, the First Improvement strategy offered a good balance between computational efficiency and solution quality. We also observed how the behavior of the algorithms changed as the instance size increased, helping us better understand the strengths and limitations of each heuristic approach.

Another important aspect of this project was the opportunity to connect theoretical concepts with practical applications. Through the illustrative examples and computational analysis, we understood how optimization models can be used in situations such as emergency services, facility distribution, and coverage planning, where reducing the worst-case distance is essential. This helped us appreciate the practical value of operations research and heuristic methods beyond the classroom.

Overall, this project helped us strengthen our knowledge of heuristic optimization, algorithm analysis, and computational experimentation. Beyond the technical results, we also learned the importance of teamwork, critical thinking, and problem-solving when developing and evaluating optimization methods for complex real-world problems.

REFERENCES

[1] A. Al-Khedhairi y S. Salhi. Enhancements to Two Exact Algorithms for Solving the Vertex P-Center Problem. *Journal of Mathematical Modelling and Algorithms*, 4(2):129–147, 2005.

[2] Elloumi, S., Labbé, M., & Pochet, Y. A New Formulation and Resolution Method for the P-Center Problem. *INFORMS Journal on Computing*, 16(1):84-94, (2004).

[3] Chen, D., & Chen, R. New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. *Computers & Operations Research*, 36(5):1646-1655, 2009.

[4] Daskin, M.S. (1995). Network and Discrete Location: Models, Algorithms, and Applications. Wiley, New York.