SELECTED TOPICS ON OPTIMIZATION

P-CENTER PROBLEM

TEAM D

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 TO THE PROBLEM

The p-center model is a minimax strategy that minimizes the maximum distance to any client. Instead of averages, it focuses on the worst-case scenario to ensure service equity, making it essential for emergency infrastructure like fire stations or ambulances.

IN THIS PRESENTATION, WE WILL:

1. **Analyze the formal mathematical structure and constraints of the problem.**

2. **Explore its variations, such as the difference between vertex-center and absolute p-problems.**

3. **Investigate its computational behavior, looking at how to implement the model and evaluate the quality of solutions in real-world contexts.**

MATHEMATICAL MODEL

First, we define how the "decisions" are represented in the system:

w_j **(Facility Location):** A binary variable that takes the value **1** if a facility is opened at candidate site j, and **0** otherwise.

Y_{ij} **(Customer Assignment):** A binary variable that takes the value **1** if demand node (customer) i is assigned to an open facility j, and **0** otherwise.

D **(Maximum Distance):** A continuous variable representing the distance to the customer farthest from the service.

Instead of looking at averages, we focus on the worst-case scenario:

Minimize D

This objective ensures that we minimize the maximum distance between any demand node and the facility serving them.

These rules ensure the solution is practically feasible:

Each customer must be assigned to exactly one facility:

$$\sum_{j \in J} Y_{ij} = 1, \quad \forall i \in I$$

A customer can only be assigned to a facility that is actually open:

$$Y_{ij} \leq w_j, \quad \forall i \in I, j \in J$$

The total number of open facilities must be exactly p:

$$\sum_{j \in J} w_j = p$$

Distance definition (The "Minimax" Link):

$$D \geq \sum_{j \in J} d_{ij} Y_{ij}, \quad \forall i \in I$$

Restrictions

$$w_j, Y_{ij} \in \{0, 1\} \quad \forall i \in I, j \in J$$

In optimization modeling, this indicates that these are binary decision variables, typically used to represent "yes/no" or "on/off" logic.

CONSTRUCTIVE HEURISTIC

FARTHEST-FIRST STRATEGY

Core Concept

- An iterative Greedy approach to minimize the maximum distance (D).
- Focuses on "worst-served" nodes to ensure maximum spatial dispersion.

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 U {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
```

Input:

- Demand nodes
- Distance matrix
- Number of facilities: p

Initialization:

- Pick an initial node as the first facility.

Distance Computing:

- Calculate the minimum distance from every demand node to the current set of facilities C.

Selection:

- Identify node i∗ with the maximum of these distances.

Expansion:

- Add $i*$ to the facility set.

Termination:

- Repeat until p facilities are selected.

Output:

- Final set of facilities C.
- Maximum service distance D (Objective Value).

EXAMPLE

FARTHEST-FIRST STRATEGY

Demand nodes:

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

$d(i,j)$	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

Number of facilities:

- p=2
- Now, let's pick c1=1
- So, C={1}

Compute dmin(i) for each node:

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

The farthest node is the Node 5, so our i*=5 ⟶ We update C={1,5}

Now | C | = p, so, we evaluate the solution. For each node, compute distance to nearest facility:

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

Maximum of these = 5

D=5

Final Result

Facilities chosen: C={1,5}

Objective value: D=5

LOCAL SEARCH

BEST IMPROVEMENT FOUND

Core Concept

- Starts from an initial feasible solution.
- Explores neighboring solutions using 1-swap moves.
- Evaluates all neighbors and selects the best one.
- Repeats until no better solution is found.

Pseudocode

```
Input:  X̄ ← initial solution (p centers)
Output: X*

X* ← X̄
Improve ← 1

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

Return X*
```

Input:

- The algorithm starts with an initial feasible solution X, representing an initial set of selected facilities.

Initialization:

- The current best solution X* is set equal to X, and the variable Improve is initialized to 1 to start the search process.

Neighborhood Generation:

- Neighbor solutions are created by swapping one selected facility with one non-selected node. This allows the algorithm to explore alternative solutions.

Selection:

- All neighboring solutions are evaluated, and the one with the lowest objective value is selected.

Improvement check :

- If the new solution is better than the current one:
- Update X*
- Set Improve = 1
- Otherwise, no update is made.

Stopping Condition:

- The algorithm continues while improvements are found.

Output:

- **The algorithm returns:**
- **The final set of facilities C**
- **The final solution $X*$**

EXAMPLE

BEST IMPROVEMENT FOUND

Problem Setup

Parameters

- p = 2
- I = {1, 2, 3, 4, 5} (demand nodes)
- Initial solution: C = {1}

Objective

Minimize the maximum distance
from any node to its nearest facility:

min D = max dmin(i)

Distance Matrix d(i, j)

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

Step 1 — Evaluate Initial Solution

Initial solution: $C = \{1\}$ → $|C| < p = 2$, so we must add one more facility.

Using the Farthest-First strategy, we compute dmin(i) for each node with $C = \{1\}$:

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

The farthest node is Node 5 (distance = 9)

→ We update $C = \{1, 5\}$ → Now $|C| = p = 2$ ✓

Now evaluate $C = \{1, 5\}$: for each node, compute dmin(i) = distance to nearest facility.

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

Step 1 — Initial Objective Value

For C = {1, 5}, compute dmin(i) = min distance from node i to any facility in C:

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

Maximum of dmin(i) = 5 → D = 5

Initial Solution (from Farthest-First)

C = {1, 5} Objective value: D = 5

This is our starting point for Best Improvement Local Search.

Step 2 — Generate Neighbors (1-swap moves)

Current solution: $C^* = \{1, 5\}$ $D^* = 5$ Possible 1-swap neighbors:

Swap out	Swap in	New C	D (max dmin)
1	2	{2, 5}	4
1	3	{3, 5}	5
1	4	{4, 5}	4
5	2	{1, 2}	6
5	3	{1, 3}	6
5	4	{1, 4}	7

Best Improvement: $\hat{X} = \{2, 5\}$ or $\{4, 5\}$ with $D = 4 < D^* = 5$

→ We accept {2, 5} (ties broken arbitrarily) and update $C^* = \{2, 5\}$, $D^* = 4$

Step 3 — Next Iteration $C^* = \{2, 5\}$

Current solution: $C^* = \{2, 5\}$ $D^* = 4$ Generate all 1-swap neighbors:

Swap out	Swap in	New C	D (max dmin)
2	1	{1, 5}	5
2	3	{3, 5}	5
2	4	{4, 5}	4
5	1	{2, 1}	6
5	3	{2, 3}	4
5	4	{2, 4}	4

No neighbor has $D < D^* = 4$ → Improve = 0 → Local optimum reached!

Algorithm terminates. Return $X^* = \{2, 5\}$ with $D^* = 4$

Final Result

Iteration	Solution C*	Objective D*	Status
0 (initial)	{1, 5}	5	Farthest-First
1	**{2, 5}**	**4**	**Improved ✓**
2	{2, 5}	4	No improvement — STOP

✦ Final Solution

Facilities chosen: $C^* = \{2, 5\}$

Objective value: $D^* = 4$

Farthest-First gave D = 5 → Best Improvement LS improved to D = 4

COMPUTATIONAL WORK

CONSTRUCTIVE HEURISTICS VS. LOCAL SEARCH (BEST IMPROVEMENT)

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%

p = 25

For small instances, the Best Improvement Local Search (LS) improved the Constructive Heuristic (CH) in all experiments. The average relative improvement was 11.80%, while execution times remained below one second. These results show that local search can significantly improve the initial solution with very low computational cost.

COMPUTATIONAL WORK

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	10.1284%
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	3.4321%
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	13.7034%
data5000_15	1132.5317	0.000000	1046.0043	3.802000	86.527400	7.6402%
data5000_16	1130.4588	0.014000	1114.4528	0.758000	16.006000	1.4159%
data5000_17	1182.8884	0.000000	987.9474	5.475000	194.941000	16.4801%
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	8.4362%
data5000_20	1143.1645	0.000000	1029.8762	2.158000	113.288300	9.9101%
number of winning times	0		20		Average REL IMP	9.3302%

p = 50

In medium-size instances, LS also outperformed CH in all cases, obtaining an average relative improvement of 9.33%. Although the computational times increased because of the larger neighborhood exploration, the algorithm still achieved good solution quality within reasonable execution times.

COMPUTATIONAL WORK

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	5.9155%
data10000_3	799.0150	0.000000	742.8923	24.502000	56.122700	7.0240%
data10000_4	811.3279	0.003000	730.6415	32.633000	80.686400	9.9450%
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	4.6485%
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	4.6948%
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	7.2470%
data10000_12	869.1657	0.016000	746.0965	38.651000	123.069200	14.1595%
data10000_13	791.6375	0.005000	747.2998	25.849000	44.337700	5.6008%
data10000_14	806.1675	0.004000	765.9772	13.417000	40.190300	4.9854%
data10000_15	853.6633	0.002000	769.3120	39.662000	84.351300	9.8811%
data10000_16	865.0017	0.001000	773.8863	14.426000	91.115400	10.5336%
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	6.4507%
data10000_19	806.8562	0.004000	726.0337	21.609000	80.822500	10.0170%
data10000_20	791.1618	0.004000	777.0927	2.408000	14.069100	1.7783%
number of winning times	0		20		Average REL IMP	7.5392%

p = 100

For large-scale instances, LS continued improving the constructive solutions in every experiment. The average relative improvement was 7.54%, showing that the constructive heuristic already provides better initial solutions for larger problems. However, local search still produced important improvements, although with higher computational times due to the increased complexity of the search space.

CONCLUSIONS

- This project allowed the team to analyze the p-Center Problem from both theoretical and computational perspectives.
- The results showed that heuristic methods are effective for solving medium and large instances with good solution quality and reasonable execution times.
- Constructive Heuristic 1 obtained better overall performance, while local search procedures further improved the initial solutions.
- Overall, the project highlighted the importance and practical value of heuristic optimization techniques in facility location problems.

THANK YOU