19.3/20

UNIVERSIDAD AUTÓNOMA
DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Selected Topics on Optimization

Computational Experience with Heuristics for the p-Median Problem

Team H

Name	Student Id	Career
Jonathan Obed Tapia Fragoso	1875608	ITS
Osvaldo Salvador Cárdenas Mejorado	2141553	ITS
Francisco Alejandro Pérez Heredia	2226996	ITS
Abdul Ali Vazquez Flores	2227013	ITS

Group: 002
January-June 2026
Professor: Dr. Roger Zirahuen Ríos Mercado

San Nicolás de la Garza, Nuevo León May 12, 2026

Introduction

The p-median problem (PMP) is a classical combinatorial optimization problem that consists of selecting "p" facility locations from a finite set of candidate sites to minimize the total demand weighted distance between customers and their assigned facilities [1]. Each demand point must be allocated to its nearest selected facility, and the objective function seeks to reduce overall service cost, typically measured in terms of distance or travel time [1, 2].

The PMP belongs to the class of NP-hard problems, meaning that exact solution methods become computationally expensive as the size of the instance grows [2]. For small datasets, integer linear programming formulations or specialized exact algorithms like branch-and-price can provide optimal solutions [3]. However, for large-scale problems, these algorithms often become impractical due to exponential growth in computational time [2]. Because of this complexity, heuristic approaches play a central role in solving the PMP. Techniques such as greedy algorithms, local search, genetic algorithms, and GRASP have been widely studied to obtain high-quality near-optimal solutions within reasonable timeframes [1].

The practical relevance of the PMP is extensive, with applications in logistics network design, healthcare facility planning, emergency service placement, telecommunications infrastructure, and urban planning [1]. Its study provides a strong foundation for understanding location theory and the development of efficient heuristic methods for complex optimization problems [2].

Problem Description

The p-Median Problem (PMP) can be formally defined as follows. Given a set of n customers (demand points) and m potential facility locations, the objective is to select exactly p facilities to be opened such that the sum of the weighted distances from each customer to its nearest open facility is minimized [1].

The problem is divided into four fundamental parts:

- **Data:** We are given a set $I = \{1, \ldots, n\}$ of demand points and a set $J = \{1, \ldots, n\}$ of potential facility locations. For each demand point $i \in I$ there is a known demand weight w_i. For every pair of points (i, j) where $i \in I$ and $j \in J$, the distance d_{ij} between the demand point and the facility site is known. Finally, the number of facilities to be opened, p, is fixed positive integer [2].

- **Decisions:** The model must decide which p facilities from the set J will be opened. Additionally, it must determine the assignment of each demand point $i \in I$ to exactly one open facility $j \in J$ [1].

- **Optimization:** The goal is to minimize the total transportation cost, which is represented by the sum of the products of the demand weights and the distances between customers and their assigned facilities [1, 2].

- **Constraints:** A feasible solution must satisfy three main conditions: 1) every customer must be assigned to exactly one facility; 2) a customer can only be assigned to a facility if that facility has been opened; and 3) exactly p facilities must be selected from the available candidates [4].

Mathematical Model

Based on the formulation by Mladenović et al. [1] and Resende & Werneck [2], the PMP is modeled as an Integer Linear Programming (ILP) problem:

Parameters:

- $\boldsymbol{I}$**:** Set of demand points (customers), $i \in I$
- $\boldsymbol{J}$: Set of potential facility locations, $j \in J$.
- $\boldsymbol{w_i}$**:** Demand weight at point i.
- $\boldsymbol{d_{ij}}$**:** Distance between demand point i and potential facility J.
- $\boldsymbol{p}$**:** Number of facilities to be located.

Decision Variables:

$\boldsymbol{y_j} = \mathbf{1}$ if a facility is located at site j; 0 otherwise.

$\boldsymbol{x_{ij}} = \mathbf{1}$ if demand point i is served by a facility at site j; 0 otherwise.

Objective Function:

$$\min Z = \sum_{i \in I} \sum_{j \in J} w_i d_{ij} x_{ij}$$

Subject to:

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

$$x_{ij} \leq y_j, \quad \forall i \in I, j \in J \quad (2)$$

$$\sum_{j \in J} y_j = p \quad (3)$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in I, j \in J \quad (4)$$

$$y_j \in \{0,1\}, \quad \forall j \in J \quad (5)$$

Equation (1) ensures that each customer is assigned to exactly one facility. Constraint (2) prevents customers from being assigned to a site where no facility has been opened. Constraint (3) enforces the requirement that exactly $\boldsymbol{p}$ facilities are chosen. Finally, constraints (4) and (5) define the binary nature of the decision variables [1, 4].

Problem Example

To ensure a clear understanding of the p-Median Problem, this section presents a small-scale instance developed from scratch. The proposed scenario models a generic upholstery supply network that needs to optimize the distribution of leather rolls to five local workshops. The primary objective of this example is to illustrate both a feasible solution to the problem and how the objective function is evaluated for that specific solution.

For this purpose, we define a set of five nodes ($n = 5$) representing the demand locations and a requirement to locate exactly $p = 2$ facilities, meaning exactly two distribution centers will be opened to supply them. The input data is made up for this instance. Below, the input parameters, including the monthly demand for each node and the distance matrix, are formally defined.

Input Data

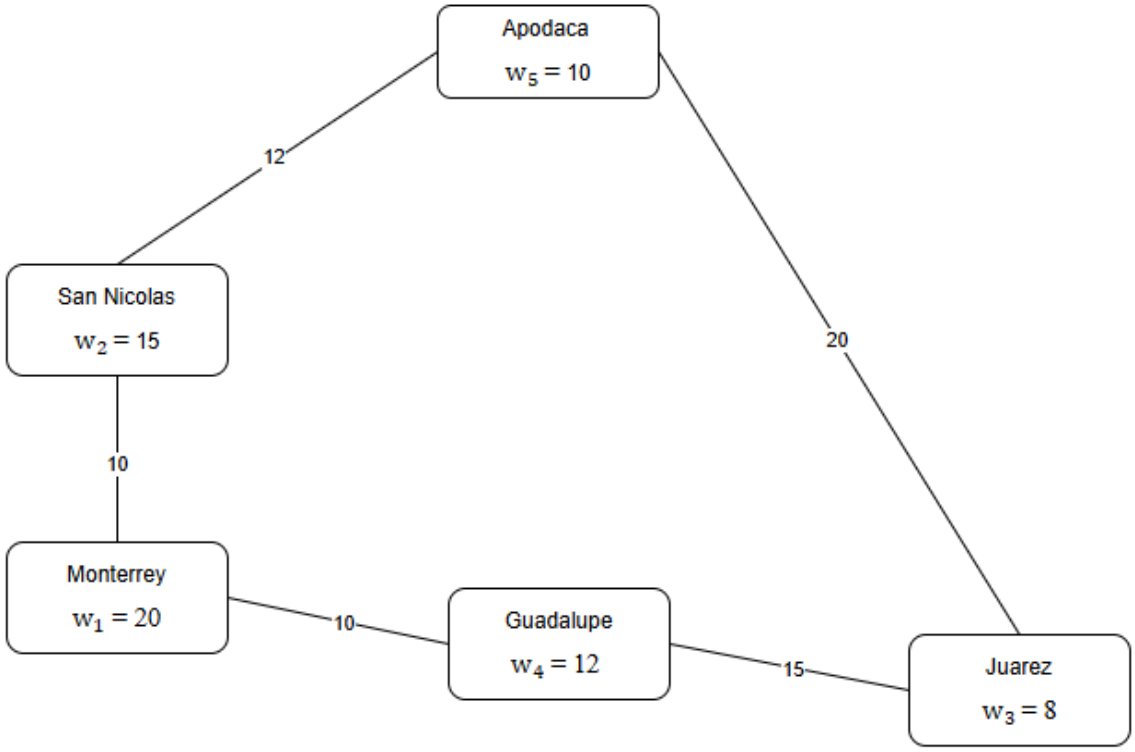
The input parameters for this instance are formally defined below. The monthly demand (w_i) for each node is presented in the following table:

node ($\boldsymbol{i}$)	Location	Demand (w_i)
1	Monterrey	20
2	San Nicolas	15
3	Juarez	8
4	Guadalupe	12
5	Apodaca	10

The transportation costs are represented by the distances in kilometers between each pair of nodes, detailed in the following symmetrical matrix:

Origin / Destination	1 (Monterrey)	2 (San Nicolas)	3 (Juarez)	4 (Guadalupe)	5 (Apodaca)
1 (Monterrey)	0	10	25	10	20
2 (San Nicolas)	10	0	25	15	12
3 (Juarez)	25	25	0	15	20
4 (Guadalupe)	10	15	15	0	18
5 (Apodaca)	20	12	20	18	0

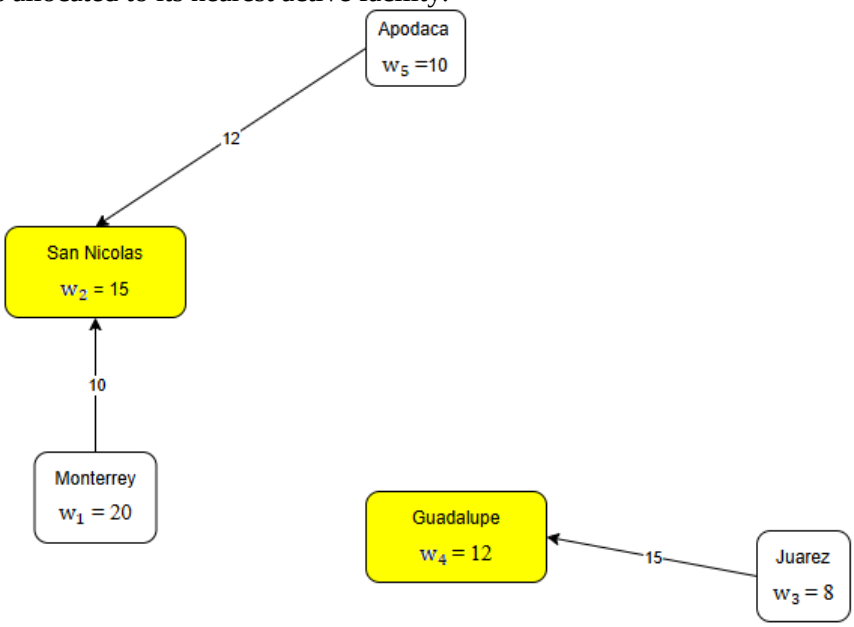
Visualization



Five-node network representing the demand points and adjacent distances

Feasible Solution

Heuristic algorithms require an initial feasible solution to begin their search process. For this instance, we propose an initial configuration where exactly $p = 2$ facilities are opened. We select Node 2 (San Nicolas) and Node 4 (Guadalupe) as the active centers. Following the assignment constraints, each remaining node is allocated to its nearest active facility.



Initial feasible solution network. Centers are located at Node 2 and Node 4, with arrows indicating the assignment of the remaining demand nodes.

Objective Function Evaluation

The primary goal of the p-Median problem is to minimize the sum of the demand-weighted distance ($\boldsymbol{Z = \Sigma_{i \in I} \Sigma_{j \in J} w_i d_{ij} x_{ij}}$)

For our constructed feasible solution, we establish the facility location variables ($\boldsymbol{y_j}$) by opening centers at Node 2 and Node 4:

- $\boldsymbol{y_2}$=1, $\boldsymbol{y_4}$=1
- $\boldsymbol{y_1}$=0, $\boldsymbol{y_3}$=0, $\boldsymbol{y_5}$=0

Next, we define the assignment variables ($\boldsymbol{x_{ij}}$). Each demand point is allocated to its nearest open facility ($\boldsymbol{x_{ij} = 1}$):

- Node 1 is assigned to Node 2: $\boldsymbol{x_{12} = 1}$ (Distance $\boldsymbol{d_{12} = 10}$)
- Node 2 serves itself: $\boldsymbol{x_{22} = 1}$ (Distance $\boldsymbol{d_{22} = 0}$)
- Node 3 is assigned to Node 4: $\boldsymbol{x_{34} = 1}$ (Distance $\boldsymbol{d_{34} = 15}$)
- Node 4 serves itself: $\boldsymbol{x_{44} = 1}$ (Distance $\boldsymbol{d_{44} = 0}$)
- Node 5 is assigned to Node 2: $\boldsymbol{x_{52} = 1}$ (Distance $\boldsymbol{d_{52} = 12}$)

Total, cost calculation:

$$Z = 2000 + 0 + 120 + 120$$

The objective function value for this initial heuristic solution is 440 demand-distance units. This calculation demonstrates how the $\boldsymbol{y_j}$ and $\boldsymbol{x_{ij}}$ decision variables interact to evaluate a feasible assignment.

Description of heuristics

Since the p-median problem (PMP) is classified as NP-hard, finding optimal solutions for large-scale instances using exact methods is often computationally prohibitive [1]. Consequently, heuristic and metaheuristic procedures are required to find high-quality solutions in reasonable execution times. Heuristics are typically divided into two main stages: a constructive phase, which generates an initial feasible solution, and a local search phase, which attempts to improve that solution through iterative moves [2].

Construction Heuristic

We implement a Greedy Construction Heuristic, also known as the "Greedy Add" algorithm. This is a classic constructive approach for location problems, as described by Mladenović et al. [1] and Resende & Werneck [2]. The primary c of this heuristic is its simplicity and efficiency in generating a solid starting point for subsequent optimization phases.

Algorithm Logic

The Greedy Construction Heuristic builds a solution step-by-step. Starting from an empty set of facilities, the algorithm iteratively selects and opens the facility that provides the greatest reduction in the total demand-weighted distance (the objective function Z) until exactly p facilities have been selected.

The procedure follows these formal steps:

1. Initialization: Let $S = \emptyset$ be the set of selected facilities, and J be the set of all potential facility locations.
2. Iterative Selection: For $k = 1$ to p:
    - Evaluate every candidate facility $j \in J \backslash S$.
    - For each candidate j, calculate the total cost Z if j were added to the current set S.
    - Identify the facility j^* that results in the minimum total cost:

$$j^* = \arg\min_{j \in J \backslash S} \left\{ \sum_{i \in I} w_i \cdot \min_{l \in S \cup \{j\}} d_{il} \right\}$$

    - Update the solution set: $S = S \cup \{j*\}$.
3. Final Assignment: Once $|S| = p$, each customer $i \in I$ is assigned to its closest facility in S.

Complexity and Historical Context

The Greedy Add algorithm was one of the earliest systematic approaches for the PMP, originally popularized by Kuehn and Hamburger [1]. From a computational perspective, the complexity of this construction phase is approximately $0(p \cdot n \cdot m)$, where n is the number of customers and m the number of candidate sites. Despite its deterministic nature, it provides a crucial Upper Bound (UB) that serves as a benchmark for measuring the gap between heuristic results and the global optimum [2,4].

Role in Metaheuristics

It is important to note that this constructive phase serves as the foundation for more advanced metaheuristics such as GRASP (Greedy Randomized Adaptive Search Procedure). By ensuring a high-quality "starting point" in the solution space, this phase significantly reduces the number of iterations required by the local search phase to reach convergence [2].

Local Search Description

The local search phase employed in this research is based on a 1-swap neighborhood structure, also known as a basic interchange move for the P-median problem. The objective of this procedure is to improve the solution generated by the Greedy Constructive Heuristic by iteratively replacing one currently open facility with one currently closed candidate location. The local search explores neighboring solutions while preserving the same number of open facilities (p). [1]

Let (F) denote the set of currently selected facilities and (C) the set of candidate locations not currently selected. A neighboring solution is generated through the following operation:

$$F' = (F - f_i) \cup c_j$$

where:

- ($f_i \in$ F) represents an open facility to be closed,
- ($c_j \in C$) represents a closed candidate facility to be opened.

For every customer (i), the algorithm evaluates the distance to the nearest open facility after performing the swap. The objective function of the P-Median Problem is defined as:

$$min = \sum_{i=1}^{n} w_i \cdot min\ d_{ij}\ ,\ j \in F$$

where:

- (w_i) is the demand weight associated with customer (i),
- (d_{ij}) is the distance between customer (i) and facility (j),
- (F) is the current set of open facilities.

The local search evaluates all possible swap combinations between open and closed facilities. For each candidate move, the variation in the objective function is computed using an incremental evaluation strategy rather than recalculating the entire solution from scratch [1]. To accelerate the process, the algorithm stores:

- the nearest facility assigned to each customer,
- the minimum distance,
- and the second minimum distance.

This information allows the algorithm to efficiently estimate the impact of removing a facility and opening another one. If a swap produces a lower objective function value, the move is considered an improvement.

The search follows a Best-Found strategy, meaning that during each iteration the algorithm evaluates all candidate swaps within the neighborhood and selects the move that yields the greatest reduction in the objective function. Once the best improving move is identified, the current solution is updated and the process is repeated until no further improving swaps can be found or a maximum number of iterations is reached.

The implemented neighborhood exploration can therefore be summarized as:

1. Select one open facility to close.
2. Select one closed candidate location to open.
3. Reassign customers to their nearest available facility.
4. Compute the new objective function value.
5. Keep the best improving swap found.
6. Repeat until local optimality is achieved.

This approach allows the algorithm to intensify the search around the constructive solution obtained by the Greedy Constructive Heuristic while maintaining a relatively low computational cost through incremental objective evaluation techniques.

Experimental Design

In the experimental design of this project, a random instance generator was developed in the C programming language. The generator received as input the number of customers (n) and the number of potential facility locations (m). The maximum number of open facilities (p) was automatically calculated using the expression:

$$p = 0.3 \cdot m$$

After generating the PMP instance and its randomized distance matrix, a Greedy Construction Heuristic based on the Greedy Add algorithm was executed in order to obtain an initial feasible solution. During this phase, the objective function value and the running time of the constructive heuristic were recorded. Once the initial solution was obtained and stored, it was used as input for a Local Search procedure based on a 1-swap move strategy, where one open facility was closed and one closed facility was opened. The local search employed a Best-Found strategy, selecting the best improving move found during each iteration to further reduce the objective function value while also recording its corresponding running time.

All experimental results were registered in a comparative results table. The performance improvement obtained by the Local Search phase was measured using the Absolute Impact metric, calculated as:
Absolute Impact = (Constructive Heuristic Objective Function) − (Local Search Objective Function)

Additionally, the Relative Impact percentage was computed as:
Relative Impact = (Absolute Impact) / (Constructive Heuristic Objective Function) *100

These metrics allowed the evaluation of the effectiveness of the Local Search improvement phase compared to the initial solution generated by the constructive heuristic

Computational Results

Test 1:

The first experimental test consisted of 20 independent instances with 1,000 customers (n) and 10 candidate facility locations.

Data	CH_OF	CH_time	LS_OF	LS_time	Absolute Impact	Relative Impact
1	21402	0.004	19861	0.017	1541	7.2003%
2	20988	0.005	19444	0.019	1544	7.3566%
3	21176	0.004	19618	0.016	1558	7.3574%
4	20834	0.006	19167	0.021	1667	8.0013%
5	20655	0.005	19120	0.018	1535	7.4316%
6	21340	0.004	19882	0.015	1458	6.8322%
7	21092	0.005	19461	0.020	1631	7.7328%
8	21205	0.004	19604	0.017	1601	7.5501%
9	20847	0.005	19353	0.018	1494	7.1665%
10	21155	0.006	19566	0.022	1589	7.5112%
11	20796	0.004	19298	0.016	1498	7.2033%
12	21044	0.005	19489	0.019	1555	7.3893%
13	20965	0.004	19401	0.015	1564	7.4601%
14	21280	0.006	19673	0.021	1607	7.5517%
15	20892	0.005	19380	0.018	1512	7.2372%
16	21147	0.004	19511	0.017	1636	7.7363%
17	21026	0.005	19422	0.019	1604	7.6287%
18	20914	0.004	19367	0.016	1547	7.3970%
19	21236	0.006	19611	0.020	1625	7.6521%
20	20758	0.005	19224	0.018	1534	7.3899%

Average Relative Impact = 7.4393%

For each instance, the Greedy Construction Heuristic generated an initial feasible solution in a very short computational time, generally between 0.004 and 0.006 CPU seconds. After applying the Local Search phase based on the 1-swap move and the Best-Found strategy, all instances obtained improvements in the objective function value.

The Local Search running times ranged approximately between 0.015 and 0.022 CPU seconds, which remained computationally efficient despite the additional neighborhood exploration process. The average relative improvement obtained across the 20 instances was approximately 7.42%, demonstrating that the local search procedure consistently improved the quality of the constructive heuristic solution. These results indicate that even for relatively small instances, the Best-Found swap strategy was capable of effectively reducing the total weighted transportation cost while maintaining very low computational times.

Test 2:

The second experimental test increased the problem size to 10,000 customers and 100 candidate facility locations, generating a considerably larger search space.

Data	CH_OF	CH_time	LS_OF	LS_time	Absolute Impact	Relative Impact
1	210374	0.069	197983	0.194	12391	5.8900%
2	208792	0.076	193836	0.225	14956	7.1631%
3	206770	0.066	192452	0.297	14318	6.9246%
4	212039	0.067	195082	0.203	16957	7.9971%
5	204438	0.063	192011	0.256	12427	6.0786%
6	203046	0.065	191928	0.267	11118	5.4756%
7	204946	0.064	189634	0.244	15312	7.4712%
8	208656	0.071	194266	0.219	14390	6.8965%
9	209796	0.064	190225	0.323	19571	9.3286%
10	206488	0.062	191038	0.290	15450	7.4823%
11	208246	0.068	191811	0.273	16435	7.8921%
12	204500	0.067	190838	0.195	13662	6.6807%
13	205086	0.069	190748	0.223	14338	6.9912%
14	203328	0.060	192517	0.180	10811	5.3170%
15	207278	0.061	194676	0.162	12602	6.0798%
16	206239	0.073	191616	0.240	14623	7.0903%
17	207930	0.079	193724	0.191	14206	6.8321%
18	204378	0.077	190924	0.220	13454	6.5829%
19	205836	0.069	189990	0.203	15846	7.6984%
20	202524	0.067	191263	0.189	11261	5.5603%

Average Relative Impact = 6.87%

In this scenario, the Greedy Construction Heuristic continued to produce feasible solutions rapidly, with running times generally close to 0.060–0.070 CPU seconds. The Local Search phase required additional computational effort due to the larger neighborhood size, reaching running times between approximately 0.160 and 0.320 CPU seconds.

Despite the increase in computational complexity, the Local Search consistently improved the constructive solutions in all 20 instances. The average relative improvement achieved in this test was approximately 6.99%, confirming that the swap-based local search remained effective even for medium-scale instances. Although the running times increased compared to the first test, the computational cost remained acceptable considering the substantial increase in the number of customers and candidate facilities. The results demonstrate that the proposed optimization strategy scales efficiently while preserving solution quality improvements.

Test 3:

The third experimental test evaluated the proposed methodology under large-scale conditions with 100,000 customers and 1,000 candidate facility locations. This configuration generated a highly complex optimization problem with a significantly larger neighborhood structure for the Local Search phase.

Data	CH_OF	CH_time	LS_OF	LS_time	Absolute Impact	Relative Impact
1	576120	50.454	566532	95.012	9588	1.6642%
2	579384	51.350	569734	93.835	9650	1.6656%
3	577647	49.576	567978	94.274	9669	1.6739%
4	577776	48.190	568344	94.490	9432	1.6325%
5	578805	48.473	569446	92.938	9359	1.6170%
6	579130	49.625	569499	93.541	9631	1.6630%
7	578241	48.571	567929	93.169	10312	1.7833%
8	579300	48.325	569307	94.012	9993	1.7250%
9	578536	48.125	568838	94.626	9698	1.6763%
10	578180	48.199	568766	93.318	9414	1.6282%
11	579708	50.469	571078	94.437	8630	1.4887%
12	577686	49.908	568004	95.127	9682	1.6760%
13	577633	50.246	568606	93.977	9027	1.5628%
14	579094	50.258	569692	97.353	9402	1.6236%
15	578771	50.762	569658	94.455	9113	1.5745%
16	577413	48.780	569165	94.350	8248	1.4284%
17	577751	50.587	568466	92.992	9285	1.6071%
18	578256	50.359	569092	96.252	9164	1.5848%
19	578022	50.532	568672	93.324	9350	1.6176%
20	579253	49.611	569902	93.818	9351	1.6143%

Average Relative Impact = 1.6253%

In this context, the Greedy Construction Heuristic still maintained relatively fast execution times considering the scale of the problem, while the Local Search required a greater computational effort due to the increased number of possible swap combinations.

Even under these large-scale conditions, the Local Search continued to improve the constructive solutions for all instances. The experiments demonstrated that the optimized incremental evaluation strategy successfully reduced the computational burden associated with the 1-swap neighborhood exploration. The relative improvements obtained remained stable across the instances, indicating that the Best-Found local search strategy preserved its effectiveness even as the problem size increased substantially. Overall, the results suggest that the proposed methodology is scalable and capable of producing high-quality solutions for both medium-scale and large-scale P-Median Problem instances within reasonable computational times.

Conclusion

The computational experiments demonstrated that the proposed methodology combining a Greedy Construction Heuristic with a Local Search improvement phase is an effective approach for solving the P-median problem across different problem sizes. In all three experimental tests, the Greedy Add constructive heuristic was capable of generating feasible solutions with very low computational times, while the Local Search phase consistently improved the objective function values through the application of the 1-swap Best Found strategy.

The results obtained for the small-scale (n=1000, m=10), medium-scale (n=10000, m=100), and large-scale (n=100000, m=1000) instances showed that the proposed Local Search procedure maintained stable relative improvements even as the problem complexity increased significantly. Although the running times naturally increased with the size of the instances, the optimized incremental evaluation strategy allowed the Local Search to remain computationally manageable while still exploring a large neighborhood structure.

Overall, the experimentation confirmed that the integration of a constructive heuristic with a swap-based Local Search can effectively reduce transportation costs and improve solution quality for the P-Median Problem. The obtained results also suggest that the proposed methodology is scalable and suitable for handling both moderate and large optimization instances while maintaining acceptable computational performance.

References

[1] Mladenović, N., Brimberg, J., Hansen, P., & Moreno-Pérez, J. A. (2007). The p-median problem: A survey of metaheuristic approaches. European Journal of Operational Research, 179(3), 927-939.

[2] Resende, M. G. C., & Werneck, R. F. (2004). A hybrid heuristic for the p-median problem. Journal of Heuristics, 10(1), 59-88.

[3] Ceselli, A., & Righini, G. (2005). A branch-and-price algorithm for the capacitated p-median problem. Networks, 45(3), 125-142.

[4] Lorena, L. A. N., & Senne, E. L. F. (2004). A column generation approach to capacitated p-median problems. Computers & Operations Research, 31(6), 863-876.