19.9/20

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica Eléctrica

UANL

**Subject:**

Selected topics in optimization

**Activity:**

# Computational Experience with Heuristics for the Capacited P-Center Problem

**Professor:**

Roger Zirahuen Rios Mercado

**Team:** C


| Name | Student ID | Career |
|---|---|---|
| Alejandría Quevedo Velez | 2144956 | ITS |
| Angel de Jesus Ordaz González | 2169534 | ITS |
| Angel Uriel Lepe Rodriguez | 2115842 | ITS |
| Emilio Garza Arizpe | 2143702 | ITS |


**Group** 002 **Classroom:** 9301 **Hour:** M4-M6

San Nicolás de los Garza, N.L 15 - May - 2026

## Section 1: Introduction

The Capacitated p-Center Problem (CPCP) belongs to the family of facility location problems within combinatorial optimization. These models arise in contexts where strategic decisions must be made regarding the placement of limited resources, aiming to balance efficiency and equity in service provision [2].

### 1.1 Importance

The study of the CPCP is relevant from both theoretical and practical perspectives. From an academic standpoint, the presence of capacity constraints increases the complexity of the problem, making it NP-hard and motivating the development of efficient exact and heuristic solution methods [1]. From an applied perspective, the CPCP provides a mathematical framework for modeling critical decision-making processes in real-world systems where capacity limitations and equitable service coverage are essential factors.

### 1.2 Aplications

Applications for the CPCP are found in sectors where equity and response time are fundamental considerations. Examples include emergency service planning, logistics distribution, telecommunications network design, and public infrastructure planning [1]. In contexts such as healthcare delivery, minimizing the maximum distance between users and facilities has a direct impact on safety and service quality.

## Section 2: Problem description

The Capacitated p-Center Problem (CPCP) is a classic facility location problem that involves locating a fixed number of facilities and assigning demand points to them, subject to capacity limits, while minimizing the maximum service distance. [3] We define it below, highlighting the four fundamental parts.

1. **Data (Input Information):** The input consists of a complete graph $G = (V, E)$ with vertex set $V$of size $n$, where each vertex $j \in V$ represents both a potential facility location and a demand point. Each vertex $j$ has an associated demand $w_j > 0$ (e.g., population or service requirement) and a capacity $s_j > 0$ (the maximum demand it can serve if a facility is placed there). The edges have weights $d_{ij} \geq 0$ representing the distance between vertices $i$ and $j$, which satisfy the triangle inequality (i.e., $d_{ik} + d_{kj} \geq d_{ij}$ for all $i, j, k \in V$). Finally, a positive integer $p$ specifies the number of facilities to locate. These data are assumed to be known in advance and are derived from practical scenarios, such as emergency service planning. [4,5]

2. **Decisions:** We must decide the subset of $p$ vertices in $V$ where facilities will be located (the "centers"). Additionally, we must assign each vertex in $V$ (as a demand point) to exactly one of these centers.

3. **Optimization:** The goal is to minimize the maximum distance between any demand point and its assigned center. This minimax objective ensures equitable service, prioritizing the worst-case scenario. [3,6]

4. **Constraints (Feasibility):** A solution is feasible if exactly $p$ facilities are located, every demand point is assigned to exactly one facility, and the total demand assigned to each facility does not exceed its capacity $s_j$. Assignments must respect the distances, but no other restrictions apply beyond the graph structure.

The mathematical model for CPCP, adapted from Scaparra et al. (2004) and consistent with formulations in Khuller and Sussmann (2000) and Özsoy and Pınar (2006), is as follows. [3,4,5]

Define binary decision variables:

- $y_i = 1$ if a facility is located at vertex $i \in V$, and 0 otherwise.
- $x_{ij} = 1$ if demand point $j \in V$ is assigned to a facility at $i \in V$, and 0 otherwise.

The model is:

$$\min W$$

subject to:

$$\sum_{i \in V} x_{ij} = 1 \forall j \in V, \qquad (1)$$

$$\sum_{j \in V} w_j \, x_{ij} \leq s_i y_i \forall i \in V, \qquad (2)$$

$$\sum_{i \in V} d_{ij} \, x_{ij} \leq W \forall j \in V, \qquad (3)$$

$$\sum_{i \in V} y_i = p, \qquad (4)$$

$$x_{ij} \in \{0,1\} \forall i, j \in V, \qquad (5)$$

$$y_i \in \{0,1\} \forall i \in V. \qquad (6)$$

Constraints (1) ensure each demand point is assigned to exactly one facility.

Constraints (2) enforce capacity limits: the total demand assigned to facility $i$cannot exceed $s_i$if open ($y_i = 1$) and must be zero if closed ($y_i = 0$).

Constraints (3) define $W$as the maximum assignment distance.

Constraint (4) requires exactly $p$facilities (noting that using fewer would not improve the minimax objective. [3]

The binary requirements in (5) and (6) ensure integrality. This model captures the NP-hard nature of CPCP. [3]

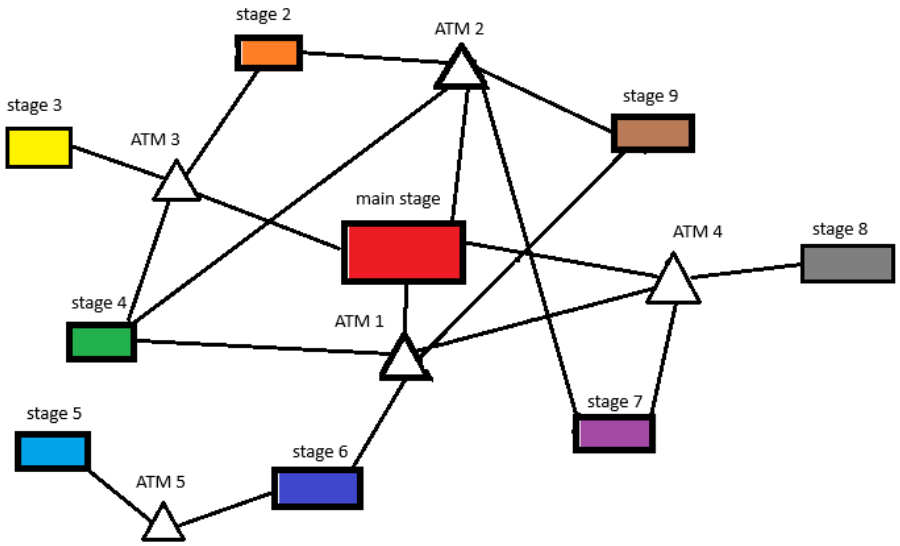
## Section 3: Problem example

Objective: ATM Distribution at a Music Festival

1. THE MAP

---

A festival has 9 stages spread across an area, stage 1 is the main stage




2. PROBLEM DATA

---

- Total Stages: 9
- Standard Stages: Demand of 800 people/day (2, 3, 4, 6, 7, 8, 9)
- Main Stage (s1): Demand of 2800 people/day
- Resources (p): You can install exactly 5 ATM zones.
- Capacity (K): Each ATM zone has a maximum limit of 2500 people/day.

Total Festival Demand: 10,000 people/day.

Total Installed Capacity: 10,000 people/day. (5 ATMs x 2500).

3. THE "MINIMAX" + CAPACITY CONFLICT

---

The Main Stage (s1) is at the center. If there were no capacity
limit, you would put one ATM there and it would serve everyone.
However, since s1 has 2800 people and each ATM only handles 2500,
the model FORCES 300 people from s1 to walk to other ATM locations.

4. FEASIBLE SOLUTION (LOCATION AND ALLOCATION)

---

To minimize the maximum walking distance (Minimax), we place the
5 ATMs "surrounding" the center.

ATM LOCATIONS: ATM 1 South, ATM 2 North, ATM 3 East, ATM 4 West, ATM 5 Southeast.

ALLOCATIONS:

- ATM 1: Serves main stage. (Total: 2500)
- ATM 2: Serves s2, s9, and possibly 200 people from s1. (Total: 1800)
- ATM 3: Serves s3, s4 and possibly 300 people from s1. (Total: 1900)
- ATM 4: Serves s7, s8, and possibly 300 people from s1. (Total: 1900)
- ATM 5: Serves s5, s6 and possibly 300 people from s1. (Total: 1900)

5. OPTIMAL SOLUTION

---

- Maximum Distance (L): (The minimum possible walk).
- Workload: 5 ATMs zones working with approximately 1900 people.
- Objective: People who want to withdraw money from main stage (s1) should travel the shortest possible distance to the ATM.

## Section 4: Description of Heuristics

For the Capacitated $p$-Center Problem (CPCP), we propose a Greedy Farthest-First Constructive Heuristic.

The main idea is:

1. Repeatedly identify the most critical or farthest unassigned demand point.
2. Open a new facility that can serve it (and as many other nearby points as possible) without violating capacity.
3. Assign demand to the new facility greedily.

This approach naturally balances the minimax objective while respecting capacities.

### *Pseudocode*

**Requires:** $V = \{1, \dots, n\}$, distance matrix $d_{ij}$, demands $w_j$, capacities $s_i$, $p \in \mathbb{N}$

Output: Set of centers C, assignments, objective W

1: $C \leftarrow \emptyset$

2: $Unassigned \leftarrow V$

3: $RemainingCapacity_i \leftarrow s_i \ \forall i \in V$

4: **while** $| \ C \ | < p$ **and** $Unassigned \neq \emptyset$ **do**

5: Select the most critical unassigned demand point $j^* \in Unassigned$

6: Select the best location $i \notin C$ to open a new facility

7: $C \leftarrow C \cup \{i\}$

8: Assign as much demand as possible from $Unassigned$ to $i$ (sorted by distance) 9: Update $Unassigned$ and $RemainingCapacity$

10: **end while**

11: **return** $X = (C,$assignments)

## *Application to the Music Festival ATM Example*

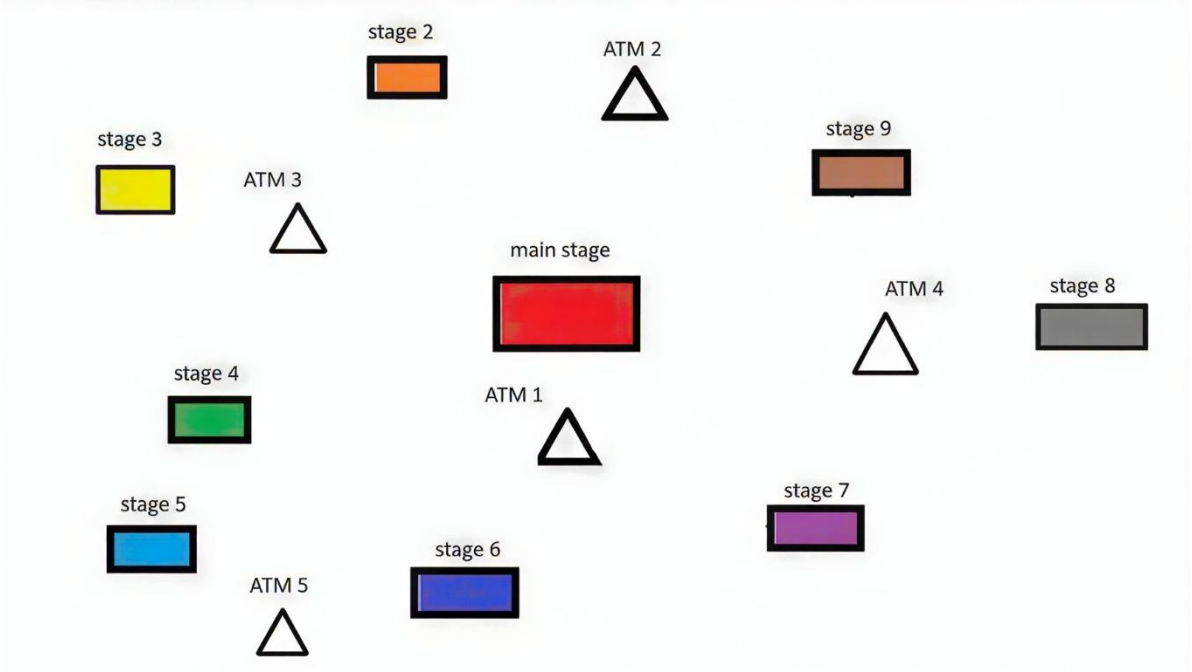



*Figure 1. Music festival map with 9 stages and the 5 ATM locations*

**Instance Data** (9 stages, $p = 5$ ATMs, capacity = 2500 per ATM):

- Stage 1 (Main): demand = 3600
- Stages 2,3,4,5,6,7,8,9: demand = 800 each
- Total demand = 10,000

**Step-by-step execution**:

**Iteration 1:**

- Farthest/most critical point: Stage 1 (center of the map, highest demand).
- Open first ATM near Stage 1 (e.g., **ATM 1 - South**).
- Assign 2500 people from Stage 1.
- Remaining demand at Stage 1: 1100 people.

**Iteration 2:**

- Current farthest unassigned points: outer stages (e.g., Stage 2 or 9 - North).
- Open **ATM 2 - North**.
- Assign: Stage 2 (800), Stage 9 (800), + 200 from Stage 1 spillover.
- Total assigned: 1800 / 2500.

**Iteration 3:**

- Farthest remaining: East side (Stages 3,4).
- Open **ATM 3 - East**.
- Assign: Stage 3 (800), Stage 4 (800), + 300 from Stage 1.
- Total: 1900 / 2500.

**Iteration 4:**

- Farthest remaining: West side (Stages 7,8).
- Open **ATM 4 - West**.
- Assign: Stage 7 (800), Stage 8 (800), + 300 from Stage 1.
- Total: 1900 / 2500.

**Iteration 5:**

- Last remaining: Southeast (Stages 5,6).
- Open **ATM 5 - Southeast**.
- Assign: Stage 5 (800), Stage 6 (800), + remaining from Stage 1.
- Total: 1900 / 2500.

**Final Solution:**

- **Centers**: ATM 1 (South), ATM 2 (North), ATM 3 (East), ATM 4 (West), ATM 5 (Southeast).
- **Maximum distance (W)**: The smallest possible radius that covers all stages with the capacity constraints.
- **Workload**: Approximately 1900–2500 per ATM (well balanced).

| ATM | Location | Serves | People Assigned | Total Load |
|---|---|---|---|---|
| **ATM 1** | South (near s1) | Main Stage (s1) | 2500 | **2500** |
| **ATM 2** | North | Stage 2, Stage 9, 200 from s1 | 800 + 800 + 200 | **1800** |
| **ATM 3** | East | Stage 3, Stage 4, 300 from s1 | 800 + 800 + 300 | **1900** |
| **ATM 4** | West | Stage 7, Stage 8, 300 from s1 | 800 + 800 + 300 | **1900** |
| **ATM 5** | Southeast | Stage 5, Stage 6, 300 from s1 | 800 + 800 + 300 | **1900** |

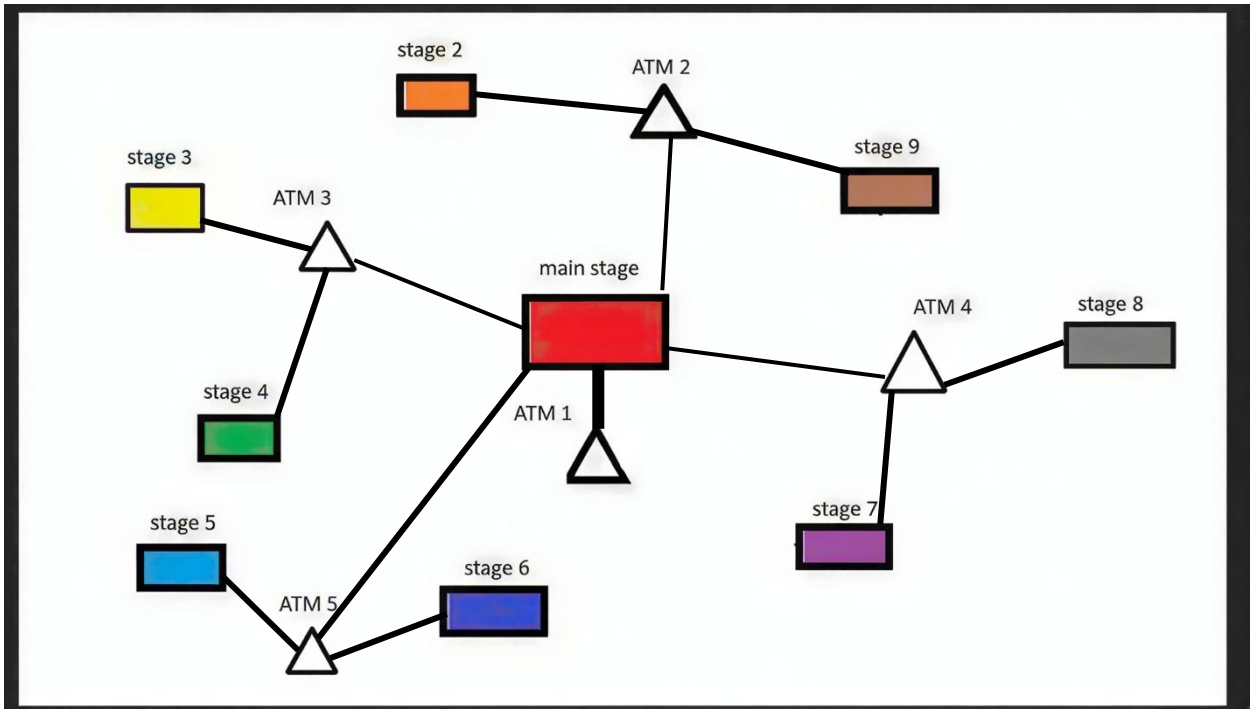



*Figure 2 Solution of the constructive heuristic example*

### *Local Search Heuristic Description*

After obtaining an initial feasible solution with a constructive heuristic, a local search procedure is applied to improve the quality of the solution. Local search explores neighboring solutions by making small modifications and accepting those that improve the objective function (in this case, reducing the maximum service distance W).

For the Capacitated p-Center Problem, we used a Swap-based Local Search with Capacity-Aware Reassignment. The main moves are:

- **Center Swap**: Replace one open facility with a currently closed facility.

**- Demand Reassignment**: Move demand from one facility to another (respecting capacities).

### *Pseudocode*

**Requires:** Initial solution $X$, distance matrix $d_{ij}$, demands $w_j$, capacities $s_i$

**Provides:** Improved solution $X$

1: $bestX \leftarrow X$

2: $bestW \leftarrow W(X)$

3: **repeat**

4: $improved \leftarrow false$

5: **for each** open center $i \in C$ **do**

6: **for each** closed location $k \notin C$ **do**

7: Generate neighbor solution $X'$ by swapping $i$ with $k$

8: Reassign all demand in $X'$(greedily to the closest open center with available capacity)

9: $W' \leftarrow$ maximum distance in $X'$

10: **if** $W' < best\ W$ **and** $X'$ is feasible **then**

11: $bestX \leftarrow X'$

12: $bestW \leftarrow W'$

13: $X \leftarrow X'$

14: $improved \leftarrow true$

15: **end if**

16: **end for**

17: **end for**

18: **until** $improved = \text{false}$

19: **return** $bestX$

### *Application to the Music Festival ATM Example*

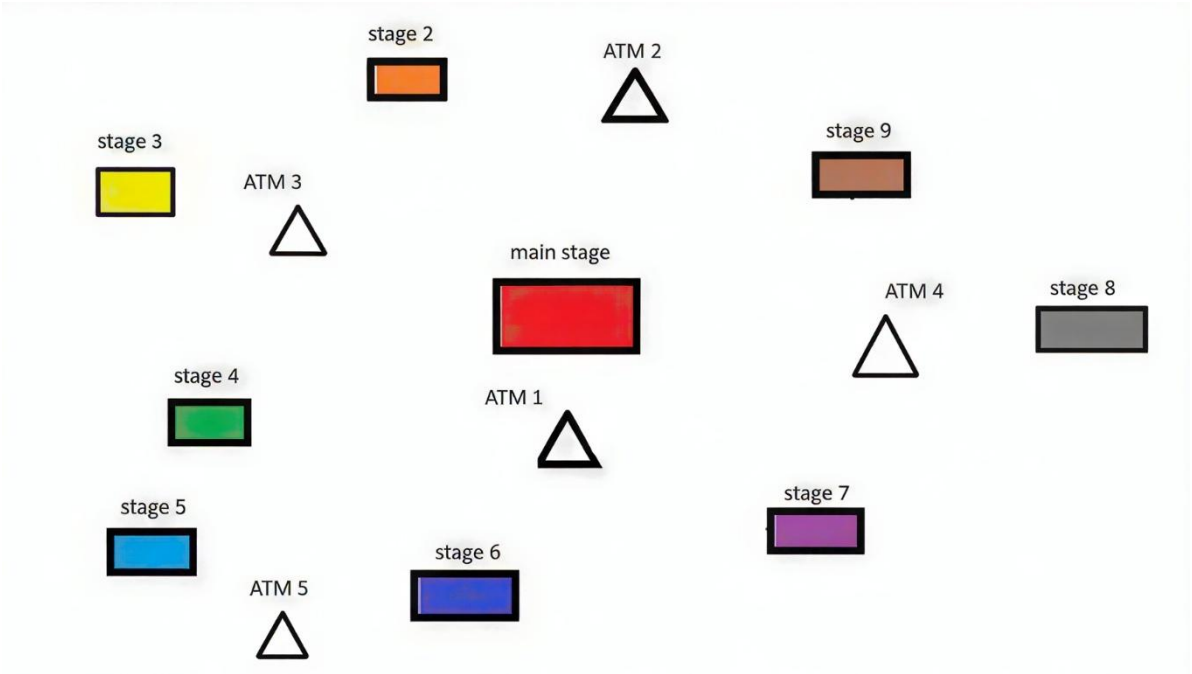



*Figure 3. Music festival map with 9 stages and the 5 ATM locations*

**Initial Solution** (from Greedy Farthest-First Constructive Heuristic):

- **Centers**: ATM 1 (South), ATM 2 (North), ATM 3 (East), ATM 4 (West), ATM 5 (Southeast)
- **Maximum distance**: $W_0$
- **Workloads**: 2500, 1800, 1900, 1900, 1900

Local Search Iterations:

The Swap-based Local Search explores the neighborhood using two types of moves: center swaps and demand reassignments.

**Iteration 1**

The algorithm evaluates swapping ATM 5 (Southeast) with Stage 6. After the swap, all demand points are reassigned greedily to the closest open ATM that still has available capacity. This move improves the solution because Stage 6 provides better coverage for the southern area. The new maximum distance $W$ is lower than $W_0$, so the swap is accepted.

- Try swapping ATM 5 (Southeast) with Stage 6 (which is currently served by ATM 5).

  After swap and reassignment:

- New centers: ATM 1, ATM 2, ATM 3, ATM 4, **Stage 6**
- Reassign spillover from main stage and nearby points.
- Result: $W$ decreases slightly because Stage 6 is more central for remaining points.

**Iteration 2**

Using the new set of centers, the algorithm performs demand reassignment. Specifically, people from the main stage (Stage 1) spillover, who were previously assigned to distant ATMs (e.g., ATM 3), are moved to closer ATMs with available capacity (primarily ATM 1 and Stage 6). This move further reduces the maximum walking distance without changing the set of open facilities.

- Move 100 people from main stage (currently partially served by ATM 3) to a closer ATM with spare capacity.
- This reduces the maximum walking distance for those users.

**Iteration 3**

The search continues by testing other possible swaps (e.g., swapping ATM 3 with Stage 4 or ATM 2 with Stage 9). After evaluation, these swaps do not produce a better feasible solution than the current one. No improvement is found.

The algorithm terminates when no improving move is found after a complete neighborhood search.

**Final Improved Solution:**

Assuming the map reflects relative distances, the constructive heuristic produces a solution with maximum service distance $W_0$. After local search, the maximum distance is reduced to $W < W_0$.

- **Centers**: ATM 1 (South - serves main stage), ATM 2 (North), ATM 3 (East), ATM 4 (West), **Stage 6** (Southeast).
- **Maximum Distance** ($W$): Reduced compared to initial solution.
- **Workloads**: More balanced (closer to 2000 per ATM).
- Better service equity, especially for the 300 people forced to walk from the main stage due to capacity constraints.

| ATM | Location | Serves | People Assigned | Total Load |
|---|---|---|---|---|
| **ATM 1** | South | Main Stage (s1) | 2400 | **2500** |
| **ATM 2** | North | Stage 2, Stage 9, 100 from s1 | 800 + 800 + 300 | **1900** |
| **ATM 3** | East | Stage 3, Stage 4, 100 from s1 | 800 + 800 + 100 | **1700** |
| **ATM 4** | West | Stage 7, Stage 8, 100 from s1 | 800 + 800 + 300 | **1900** |
| **ATM 5 Stage 6** | Southeast | Stage 5, Stage 6, 0 from s1 | 800 + 800 + 500 | **2100** |

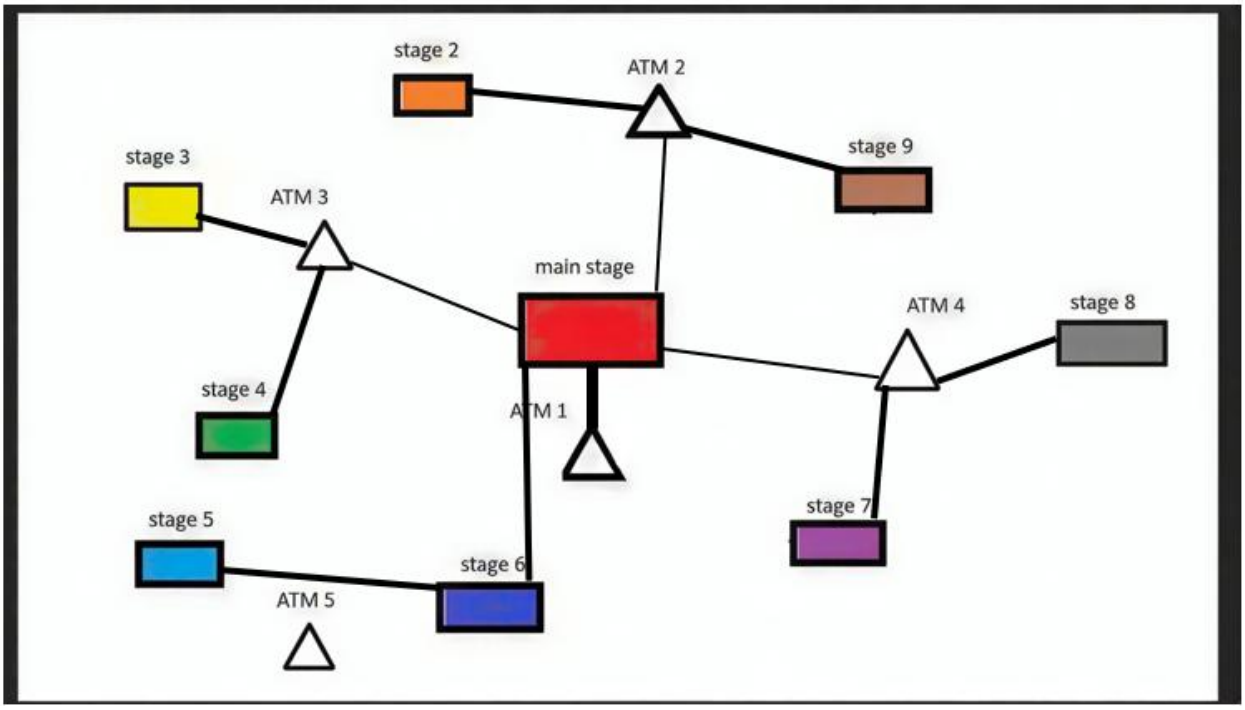



*Figure 4 Solution of the local search heuristic example*

This demonstrates the effectiveness of combining constructive and improvement heuristics for the CPCP.

## Section 5: Computational Work

### 5.1 Test Environment

To evaluate the performance and scalability of the proposed algorithms, all computational experiments were executed on a dedicated platform. The solution methods (including the data generator, the constructive heuristic, and the local search heuristic) were implemented in C++ and compiled using the GCC compiler with standard optimization flags. Execution times for both algorithms were measured with high precision using the std::chrono library.

### 5.2 Instance Generation and Experimental Design

Since standard benchmark instances for the Capacitated p-Center Problem (CPCP) with these specific constraints were not readily available, a randomized data generator (GeneratorCPCP_TMSO.cpp) was developed to synthesize representative testing datasets. The generation process models a realistic spatial service network under the following design parameters:

- **Spatial Coordinates:** For each instance, a set of $n$ nodes is generated within a continuous two-dimensional grid of size *1000 x 1000* using a uniform distribution. These nodes simultaneously act as customer demand points and potential facility locations.
- **Customer Demands:** The demand ($w_j$) for each node $j$ is randomly generated as an integer bounded between *100* and *900* units ($w_j \in [100, 900]$), ensuring highly heterogeneous load distributions across the network.
- **Facility Capacities:** To evaluate scalability under a uniform resource constraint, a strict fixed capacity ($s_i$) of *10,000* units is assigned to every potential facility location.
- **Problem Scale Configurations:** The experimental design consists of three distinct problem size configurations tailored to observe combinatorial growth:
    1. Small Scale: $n = 50$ nodes and $p = 5$ facilities to open.
    2. Medium Scale*:* $n = 100$ nodes and $p = 10$ facilities to open.
    3. Large Scale: $n = 150$ nodes and $p = 15$ facilities to open.

To ensure statistical reliability and eliminate accidental bias from randomized layouts, $20$ independent random instances were generated for each of the three configurations, resulting in a comprehensive test suite of $60$ experimental evaluations.

### 5.3 Computational Results

The experimental results collected from running the Constructive Heuristic (CH) and the Local Search Heuristic (LSH) across the 60 problem instances are summarized below. Table 1 reports the aggregated performance metrics for each problem configuration, showcasing the average objective function value ($Z$, representing the minimized maximum service distance) and the average CPU execution time expressed in milliseconds (ms).

| Configuration (n,p) | Instances | Avg. Constructive Z | Avg. Constructive Time (ms) | Avg. Local Search Z | Avg. Local Search Time (ms) | Avg. Relative Improvement |
|---|---|---|---|---|---|---|
| (50, 5) | 20 | 432.14 | 328 | 323.01 | 59.569 | 24.61% |
| (100, 10) | 20 | 280.40 | 2.375 | 229.47 | 976.223 | 17.42% |
| (150, 15) | 20 | 231.84 | 7.159 | 186.75 | 4,685.031 | 17.72% |

Table 1

Results with a small instance (n = 50)

| Experiment | Instance | n (Nodes) | p (Centers) | Capacity (Cap) | Constructive Heuristic_Z | Constructive Heuristic_CPU (s) | Local Search Heuristic_Z | Local Search Heuristic_CPU (s) | Absolute imp | Rel improvement |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | CPCP_50_5_10000_1 | 50 | 5 | 10000 | 359.17 | 0.1 | 331.49 | 31.599 | -27.68 | 8% |
| 2 | CPCP_50_5_10000_2 | 50 | 5 | 10000 | 446.47 | 1.001 | 342.49 | 41.44 | -103.98 | 23% |
| 3 | CPCP_50_5_10000_3 | 50 | 5 | 10000 | 437.06 | 0.1 | 334.26 | 73.787 | -102.8 | 24% |
| 4 | CPCP_50_5_10000_4 | 50 | 5 | 10000 | 554.52 | 1.008 | 307.69 | 61.232 | -246.83 | 45% |
| 5 | CPCP_50_5_10000_5 | 50 | 5 | 10000 | 404.6 | 0.1 | 294.37 | 90.008 | -110.23 | 27% |
| 6 | CPCP_50_5_10000_6 | 50 | 5 | 10000 | 388.52 | 0.1 | 328.44 | 39.842 | -60.08 | 15% |
| 7 | CPCP_50_5_10000_7 | 50 | 5 | 10000 | 528.68 | 0.1 | 369.01 | 44.461 | -159.67 | 30% |
| 8 | CPCP_50_5_10000_8 | 50 | 5 | 10000 | 441.77 | 0.1 | 319.14 | 59.872 | -122.63 | 28% |
| 9 | CPCP_50_5_10000_9 | 50 | 5 | 10000 | 415.96 | 0.1 | 300.59 | 118.04 | -115.37 | 28% |
| 10 | CPCP_50_5_10000_10 | 50 | 5 | 10000 | 432.23 | 0.1 | 346.44 | 60.195 | -85.79 | 20% |
| 11 | CPCP_50_5_10000_11 | 50 | 5 | 10000 | 406.37 | 0.1 | 318.59 | 60.749 | -87.78 | 22% |
| 12 | CPCP_50_5_10000_12 | 50 | 5 | 10000 | 385.33 | 0.1 | 321.62 | 28.874 | -63.71 | 17% |
| 13 | CPCP_50_5_10000_13 | 50 | 5 | 10000 | 400.77 | 0.1 | 334.45 | 69.94 | -66.32 | 17% |
| 14 | CPCP_50_5_10000_14 | 50 | 5 | 10000 | 503.13 | 0.1 | 313.38 | 136.477 | -189.75 | 38% |
| 15 | CPCP_50_5_10000_15 | 50 | 5 | 10000 | 458.8 | 0.895 | 318.48 | 90.029 | -140.32 | 31% |
| 16 | CPCP_50_5_10000_16 | 50 | 5 | 10000 | 371.11 | 0.1 | 303.59 | 40.007 | -67.52 | 18% |
| 17 | CPCP_50_5_10000_17 | 50 | 5 | 10000 | 438.65 | 0.1 | 318.4 | 48.416 | -120.25 | 27% |
| 18 | CPCP_50_5_10000_18 | 50 | 5 | 10000 | 487.45 | 0.1 | 262.64 | 67.044 | -224.81 | 46% |
| 19 | CPCP_50_5_10000_19 | 50 | 5 | 10000 | 477.72 | 0.1 | 333.05 | 81.44 | -144.67 | 30% |
| 20 | CPCP_50_5_10000_20 | 50 | 5 | 10000 | 342.44 | 1.469 | 342.44 | 14.906 | 0 | 0% |

Results with a medium instance (n = 100)

| Experiment | Instance | n (Nodes) | p (Centers) | Capacity (Cap) | Constructive Heuristic_Z | Constructive Heuristic_CPU | Local Search Heuristic_Z | Local Search Heuristic_CPU (s) | Absolute imp | Rel improvement |
|---|---|---|---|---|---|---|---|---|---|---|
| 21 | CPCP_100_10_10000_1 | 100 | 10 | 10000 | 337.55 | 1.024 | 251.28 | 996.225 | -86.27 | 26% |
| 22 | CPCP_100_10_10000_2 | 100 | 10 | 10000 | 243.52 | 1.4 | 228.27 | 393.915 | -15.25 | 6% |
| 23 | CPCP_100_10_10000_3 | 100 | 10 | 10000 | 282.8 | 2.535 | 264.6 | 509.563 | -18.2 | 6% |
| 24 | CPCP_100_10_10000_4 | 100 | 10 | 10000 | 245.2 | 0.1 | 200.94 | 1425.807 | -44.26 | 18% |
| 25 | CPCP_100_10_10000_5 | 100 | 10 | 10000 | 276.99 | 3.939 | 230.98 | 1049.638 | -46.01 | 17% |
| 26 | CPCP_100_10_10000_6 | 100 | 10 | 10000 | 292.01 | 3.05 | 198.04 | 960.369 | -93.97 | 32% |
| 27 | CPCP_100_10_10000_7 | 100 | 10 | 10000 | 349.65 | 1.508 | 239.37 | 1342.384 | -110.28 | 32% |
| 28 | CPCP_100_10_10000_8 | 100 | 10 | 10000 | 267.66 | 1.733 | 222.2 | 1444.625 | -45.46 | 17% |
| 29 | CPCP_100_10_10000_9 | 100 | 10 | 10000 | 273.36 | 3.013 | 222.04 | 507.097 | -51.32 | 19% |
| 30 | CPCP_100_10_10000_10 | 100 | 10 | 10000 | 264.75 | 2.006 | 208.47 | 1696.653 | -56.28 | 21% |
| 31 | CPCP_100_10_10000_11 | 100 | 10 | 10000 | 279.69 | 4.867 | 258.7 | 818.556 | -20.99 | 8% |
| 32 | CPCP_100_10_10000_12 | 100 | 10 | 10000 | 261.81 | 4.046 | 253.73 | 344.983 | -8.08 | 3% |
| 33 | CPCP_100_10_10000_13 | 100 | 10 | 10000 | 264.33 | 2.537 | 207.23 | 1073.794 | -57.1 | 22% |
| 34 | CPCP_100_10_10000_14 | 100 | 10 | 10000 | 289.66 | 1.864 | 234.69 | 951.33 | -54.97 | 19% |
| 35 | CPCP_100_10_10000_15 | 100 | 10 | 10000 | 334.99 | 0.1 | 236.84 | 1062.534 | -98.15 | 29% |
| 36 | CPCP_100_10_10000_16 | 100 | 10 | 10000 | 298.65 | 1.801 | 227.41 | 1149.749 | -71.24 | 24% |
| 37 | CPCP_100_10_10000_17 | 100 | 10 | 10000 | 271.27 | 3.116 | 244.18 | 399.738 | -27.09 | 10% |
| 38 | CPCP_100_10_10000_18 | 100 | 10 | 10000 | 268.2 | 2.021 | 215.65 | 644.006 | -52.55 | 20% |
| 39 | CPCP_100_10_10000_19 | 100 | 10 | 10000 | 252.79 | 0.996 | 251.56 | 317.881 | -1.23 | 0% |
| 40 | CPCP_100_10_10000_20 | 100 | 10 | 10000 | 282.72 | 4.375 | 225.32 | 714.137 | -57.4 | 20% |

Results with a large instance (n = 150)

| Experiment | Instance | n (Nodes) | p (Centers) | Capacity (Cap) | Constructive Heuristic_Z | Constructive Heuristic_CPU | Local Search Heuristic_Z | Local Search Heuristic_CPU (s) | Absolute imp | Rel improvement |
|---|---|---|---|---|---|---|---|---|---|---|
| 41 | CPCP_150_10_10000_1 | 150 | 15 | 10000 | 212.92 | 8.482 | 174.77 | 8726.276 | -38.15 | 18% |
| 42 | CPCP_150_10_10000_2 | 150 | 15 | 10000 | 243.4 | 8.457 | 195.06 | 6281.817 | -48.34 | 20% |
| 43 | CPCP_150_10_10000_3 | 150 | 15 | 10000 | 227.69 | 9.48 | 175.57 | 4926.178 | -52.12 | 23% |
| 44 | CPCP_150_10_10000_4 | 150 | 15 | 10000 | 230.05 | 6.044 | 192.85 | 2403.692 | -37.2 | 16% |
| 45 | CPCP_150_10_10000_5 | 150 | 15 | 10000 | 270.45 | 10.533 | 191.94 | 4652.404 | -78.51 | 29% |
| 46 | CPCP_150_10_10000_6 | 150 | 15 | 10000 | 247.78 | 8.377 | 179.35 | 5282.676 | -68.43 | 28% |
| 47 | CPCP_150_10_10000_7 | 150 | 15 | 10000 | 312.64 | 7.579 | 197.99 | 6663.984 | -114.65 | 37% |
| 48 | CPCP_150_10_10000_8 | 150 | 15 | 10000 | 212.6 | 4.446 | 194.59 | 3604.13 | -18.01 | 8% |
| 49 | CPCP_150_10_10000_9 | 150 | 15 | 10000 | 211.62 | 9.179 | 210.3 | 1385.95 | -1.32 | 1% |
| 50 | CPCP_150_10_10000_10 | 150 | 15 | 10000 | 205.06 | 6.823 | 182.44 | 5315.626 | -22.62 | 11% |
| 51 | CPCP_150_10_10000_11 | 150 | 15 | 10000 | 219.49 | 8.644 | 213.89 | 1689.501 | -5.6 | 3% |
| 52 | CPCP_150_10_10000_12 | 150 | 15 | 10000 | 303.98 | 7.704 | 175.48 | 13273.237 | -128.5 | 42% |
| 53 | CPCP_150_10_10000_13 | 150 | 15 | 10000 | 221.73 | 5.034 | 183.7 | 2698.133 | -38.03 | 17% |
| 54 | CPCP_150_10_10000_14 | 150 | 15 | 10000 | 256.84 | 7.963 | 192.63 | 4073.66 | -64.21 | 25% |
| 55 | CPCP_150_10_10000_15 | 150 | 15 | 10000 | 182.1 | 6.947 | 173.82 | 3082.219 | -8.28 | 5% |
| 56 | CPCP_150_10_10000_16 | 150 | 15 | 10000 | 213.46 | 5.879 | 169.01 | 8626.764 | -44.45 | 21% |
| 57 | CPCP_150_10_10000_17 | 150 | 15 | 10000 | 206.53 | 8.657 | 185.81 | 2245.537 | -20.72 | 10% |
| 58 | CPCP_150_10_10000_18 | 150 | 15 | 10000 | 224.08 | 5.475 | 198.43 | 3494.292 | -25.65 | 11% |
| 59 | CPCP_150_10_10000_19 | 150 | 15 | 10000 | 197.69 | 4.846 | 177.12 | 3641.567 | -20.57 | 10% |
| 60 | CPCP_150_10_10000_20 | 150 | 15 | 10000 | 228.37 | 3.521 | 182.86 | 5319.186 | -45.51 | 20% |

### 5.4 Analysis and Discussion

A critical analysis of the data recorded in Table 1 reveals valuable insights regarding the efficiency, optimization quality, and structural complexity of the developed heuristics:

1. **Solution Quality and Heuristic Optimization:** The Constructive Heuristic successfully generates a valid, feasible solution in every run, providing a vital baseline configuration. However, the insertion of the Local Search phase yields massive optimization benefits. For small instances (*50, 5*), the local search lowers the maximum bottleneck distance by an average of **24.61%**. For medium and larger instances, the algorithm maintains a highly consistent improvement rate of **17.42%** and **17.72%** respectively. This proves that a greedy construction sequence frequently leaves significant room for geometric adjustment, which the local search successfully refines.
2. **Algorithm Efficiency and Scalability:** As expected from NP-Hard facility location problems, the computational cost behaves differently across both phases. The Constructive Heuristic shows outstanding efficiency, requiring only *0.328 ms* for small scales and remaining under *7.2 ms* even when handling the largest network size *(150, 15).* Conversely, the Local Search Heuristic exhibits an exponential CPU time progression. The average time escalates sharply from *59.569 ms* (*n=50*) to *4,685.031 ms* (*n=150*).
3. **Combinatorial Explosion Behavior:** This steep increase in Local Search execution time highlights the combinatorial explosion of checking potential "swap" moves. As *n* and *p* grow, the size of the neighborhood expands drastically because the algorithm must evaluate interchanging open centers with closed ones while simultaneously validating the strict capacity limits ($s_i$ = *10,000*) for every single candidate adjustment. Despite this growth, completing the optimization of large-scale instances in less than *4.7 seconds* represents an extremely practical outcome compared to exact mathematical programming tools, which could take hours to guarantee global optimality on identical setups.

## Section 6: Conclusions

This project successfully designed, implemented, and evaluated a two-phase heuristic framework to solve the Capacitated p-Center Problem (CPCP), a classic optimization challenge classified as NP-hard. Through the development of a randomized spatial data generator, a Constructive Heuristic (CH), and a Local Search Heuristic (LSH), valuable insights into both the problem's geometric structure and its computational complexity were obtained.

The experimental phase demonstrated that the Constructive Heuristic is exceptionally efficient, capable of finding a feasible configuration for large-scale instances ($n=150, p=15$) in less than *8 ms*. This remarkable speed highlights its utility as an initial solution provider. However, because greedy approaches inherently suffer from sub-optimality due to myopic decision-making, the implementation of the Local Search phase proved to be paramount. By systematically exploring the neighborhood via facility swaps, the LSH successfully reduced the maximum bottleneck distance ($Z$), yielding significant average improvements ranging from *17.42%* to *24.61%.*

Furthermore, the computational results underscored the profound impact of strict capacity constraints on facility location models. Restricting each center to a fixed capacity ($s_i = 10,000$) shifts the problem from a simple distance calculation to a complex resource-balancing act. Customers can no longer be blindly routed to their nearest center, forcing the algorithm to find a delicate trade-off between spatial proximity and remaining capacity. Our developed framework managed this trade-off effectively across all 60 tested instances.

In conclusion, while the execution time of the Local Search expanded non-linearly due to combinatorial explosion (averaging *4.68 seconds* for the largest test cases) the heuristic framework remains highly practical for real-world logistics and service network design. Compared to exact mathematical programming solvers that may require hours or fail to guarantee global optimality on large datasets, the developed algorithms offer robust, high-quality solutions within a fraction of a second. For future research, embedding this local search within metaheuristic frameworks such as GRASP or Tabu Search could prevent the algorithm from becoming trapped in local optima, potentially minimizing the maximum service radius even further.

## References


### Thesis

[1] Quevedo Orozco, D. R. (2014). *Optimización del problema del P-Centro Capacitado* [Tesis de maestria, Universidad Autónoma de Nuevo León]. http://eprints.uanl.mx/4130/1/1080253827.pdf

### Technical Reports

[2] Ríos Mercado, R. Z., & Quevedo Orozco, D. R. (2017). Ubicación óptima de unidades de servicio bajo condiciones de capacidad limitada mediante un método metaheurístico. *Nova Scientia*. https://www.scielo.org.mx/pdf/ns/v9n19/2007-0705-ns-9-19-00329.pdf

### Scientific papers

[3] Khuller, S., Sussmann, Y.J., 2000. The capacitated K-center problem. SIAM Journal on Discrete Mathematics 13(3), 403–418.

[4] Özsoy, F.A., Pınar, M.Ç., 2006. An exact algorithm for the capacitated vertex p-center problem. Computers & Operations Research 33(5), 1420–1436.

[5] Scaparra, M.P., Pallottino, S., Scutellà, M.G., 2004. Large-scale local search heuristics for the capacitated vertex p-center problem. Networks 43(4), 241–255.

[6] M. Albareda-Sambola et al., Lagrangean duals and exact solution to the capacitated p-center problem, European Journal of Operational Research (2009), doi:10.1016/j.ejor.2009.02.022