19.5/20

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

COMPUTATIONAL EXPERIENCE WITH HEURISTICS FOR THE MAXIMAL COVERING LOCATION PROBLEM

GROUP: 003

TUESDAY M4 – M6

AUTUMN 2025

TEAM B

MATRÍCULA	NOMBRE
2029675	Adriana Concepción Aguirre Dorantes
1959360	Jonathan Francisco Galván Villanueva
1433732	José Carlos Rodríguez Rivera

Introduction

Facility location decisions represent critical strategic choices with far-reaching implications across public and private sectors. The Maximal Covering Location Problem (MCLP), first introduced by Church and ReVelle [1] in 1974, has emerged as one of the most influential models in location science due to its practical relevance and mathematical elegance. MCLP addresses the fundamental challenge of optimally positioning a limited number of facilities to maximize service coverage to demand populations under distance or time constraints.

The enduring significance of MCLP stems from its widespread applications across numerous domains. In public health, it guides the placement of emergency medical services and vaccination centers to maximize population access within critical response times [2]. In urban planning, MCLP informs the location of public facilities like libraries, schools, and fire stations to ensure equitable service distribution [3]. Retail corporations employ MCLP to strategically position stores and distribution centers to maximize market reach while respecting budget constraints [4].

The classical MCLP formulation seeks to locate p facilities on a network such that the covered demand weight is maximized, where coverage is defined by a predetermined critical distance or time threshold. This coverage radius represents a fundamental parameter that directly influences solution characteristics and reflects real-world service standards, such as 8-minute response times for emergency services or 15-minute access thresholds for public amenities [5].

MCLP becomes particularly relevant in resource-constrained environments where budgetary limitations prevent universal coverage. Rather than attempting to cover all demand points, which may be economically infeasible, decision makers must make strategic trade-offs to maximize coverage within fixed budgets [6]. This constrained optimization framework has motivated extensive research into efficient solution methodologies, ranging from exact algorithms for small instances to heuristic and metaheuristic approaches for large-scale practical problems.

Over the decades, numerous extensions to the original MCLP have been developed, including probabilistic coverage models, multi-objective formulations, and dynamic problems. Despite these advancements, the core MCLP remains a cornerstone un location theory due to its computational challenge and practical utility. The problem's NP-hard nature [7] ensures that developing efficient heuristics approaches continues to be an active research area with significant practical implications.

This computational experience contributes to this ongoing research by conducting a comprehensive computational evaluation of constructive and local search heuristics for MCLP across multiple problem scales, providing insights into algorithm performance and practical implementation guidelines.

Problem description

The Maximal Covering Location Problem can be formally described as follows:

Data (input information):

- A set of demand points I, each with an associated population to be served a_i.
- A set of potential facility locations J.
- A coverage radius or maximum service distance S.
- A maximum number of facilities to be located p.

Decisions (variables):

- Decide which facility locations $j \epsilon J$ will be located.
- Decide which demand points are covered by the chosen facilities.

Optimization objective:

- Maximize the number of people served or covered within the desired service distance.

Constraints (feasible solution conditions):

- At most *p* facilities can be located.
- A demand point is considered covered if it is within distance *S* of at least one located facility.

Mathematical Model:

$$Maximize\ Z = \sum_{i \epsilon I} a_i * y_i$$

Subject to:

$$\sum_{j \epsilon N_i} x_j \geq y_i \qquad \forall i \epsilon I$$

$$\sum_{j \epsilon J} x_j = p$$

$$x_j \epsilon \{0,1\} \qquad \forall j \epsilon J$$

$$y_i \epsilon \{0,1\} \qquad \forall i \epsilon I$$

Where:

I = denotes the set of demand nodes.

J = denotes the set of facility sites.

S = maximum service distance.

d_{ij} = the shortest distance from node i to node j.

$N_i = \{ j \in J \mid d_{ij} \leq S\}$ = set of facility sites that can "cover" demand node i.

a_i = population to be served at demand node i.

p = number of facilities to be located.

x_j = 1 if a facility is located at site j, 0 otherwise.

y_i = 1 if demand node i is covered, 0 otherwise.

Problem example

Population growth and industrialization in Nuevo León have significantly increased the generation of municipal and industrial solid waste. However, a large share of recycled materials (PET, cardboard, metals) does not reach collection centers because of the distance residents must travel to dispose of them properly.

In this context there is a clear need to decide where to locate a limited number of recycling collection centers so that the maximum possible amount of recyclable waste si captured within a maximum acceptable service distance.

We consider a set of municipalities *J*, each associated with an estimated amount pf recyclable waste generated per day (wj). The set of potential sites *I* coincides with these municipalities.

The goal is to open exactly *p* recycling centers in such a way that the total recyclable waste covered is maximized, where a municipality is considered covered if it lies in a maximum distance *S* from at least one open site.

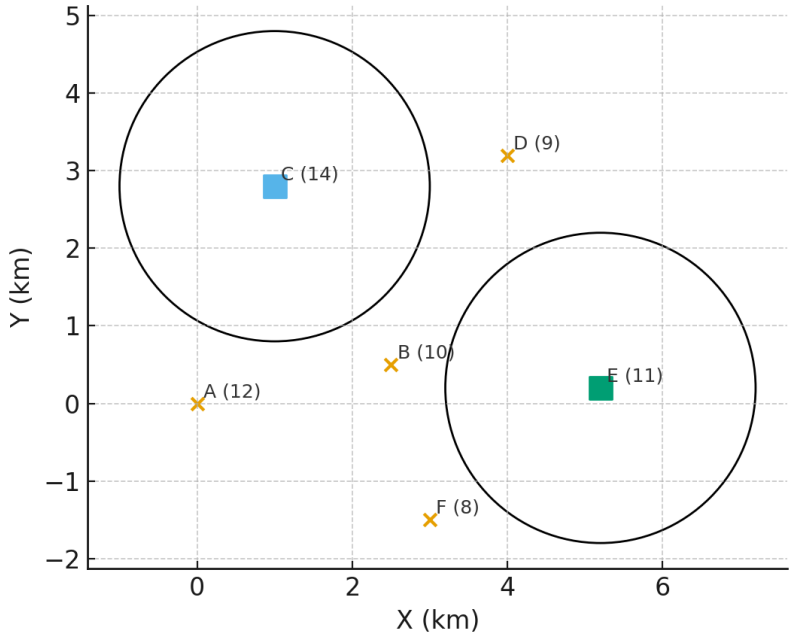
Small instance:

- Demand nodes (also candidate sites): A–F
- Weight = recyclable tons/day (objective weight)
- Parameters: p = 2 centers, S = 2 km service radius

node	x	y	weight
A	0	0	12
B	2.5	0.5	10
C	1	2.8	14

D	4	3.2	9
E	5.2	0.2	11
F	3	-1.5	8

The image shows the feasible solution where we open at C and E; each circle is the 2 Km service radius.



- **Chosen open sites (feasible): {C, E}**
- **A demand node *j* is considered covered if at least one facility *i* in {C, E} is located within the service distance *S* of *j*.**
- **Objective value is equal to the sum of weights of covered nodes.**

Check coverage

node	weight	Covered by C or E?	Contribution
A	12	No	12
B	10	No	10
C	14	Yes	14
D	9	No	9
E	11	Yes	11
F	8	No	0

Z = 14 + 11 = 25

Description of Heuristics

Constructive Heuristic: Greedy Adding

The Greedy Adding heuristic represents a myopic approach that sequentially selects facilities offering the maximum marginal coverage improvement. This method belongs to the class of greedy algorithms known for their computational efficiency and reasonable solution quality.

Function: Greedy_Adding(C, P, W)

Inputs:

- C: Coverage matrix (numpy array) where $C[i, j] = 1$ if demand point i is covered by candidate site j, and 0 otherwise.
- P: Number of facilities ($p\_facilities$) to select.
- W: Demand weights where $W[i]$ is the weight (demand of demand point i.

Output: S: Set of selected sites

Initialization:

- $N\_sites \leftarrow$ Number of potential sites (columns of C).
- $S \leftarrow$ Empty list of selected sites.
- $R \leftarrow [0, 1, \ldots, N\_sites - 1]$

Selection Loop

- **while** $[S] < P$):
  - $Best\_Site \leftarrow$ NULL
  - $Best\_Additional\_Coverage \leftarrow -1$
  - For each candidate site j in R
    - $Calculate\_Coverage(S, C$ $C, W)$
    - $S\,temp \leftarrow S \cup \{j\}$
    - $Additional\ Covereage \leftarrow$ $Mejor\ Cobertura = Adcinran$

Update Solution:

- **if** $Best\_Site \leqq$ NULL
  - Add $Best\_Site$ to S, $w1$, R
  - Remove $Mejor\_Sitio$ or a R

CalCulate_Coverage(S, C, W)

- Covered_Points = boolean vectors of demand points
- **for** each selected site $j = S$
  - **sum**$\{i$ in $Covered\_Points\}$ $W[i]$

Computational complexity: The algorithm exhibits O(p*m*n) time complexity, where p is the number of facilities, m is the number of demand points, and n is the number of potential sites.

Local Search: Swap Neighborhood

Local search serves as an improvement procedure to enhance solutions obtained from constructive heuristics. The swap neighborhood examines pairwise exchanges between selected and unselected facilities, employing a best-improvement strategy.

Function: Swap_Move(S_{current}, C, W)

Inputs:

S_{current}: The current solution (list of selected site indices)
C: Coverage Matrix
W: Demand Weights

Output: S_{best}: The best solution found after local search

Initialization:

$N\_sites \leftarrow$ Number of potential sites
$S_{\text{best}} \leftarrow S_{\text{current}}$ (Copy of the initial solution)
Coverage_best $\leftarrow$ Calculate_Coverage(S_{current}, C, W)
Improved $\leftarrow$ TRUE
Main Loop (While Improved isTRUE):
Improved = FALSE
For each i_j in [0,1,... |S_current| − 1] (
(position of the faciiity to remove in S'_{current},
For each j in [0,1,.., N_sites − 1]
if j is not in S_{current}:
$S_{\text{new}} \leftarrow S_{\text{current}}$ (Copy)
Replace the site at position i in S_{new} with
site j Coveragenew $\leftarrow$ Coverage_best
If *Coverage*_new> *Coverage*_new
S_best $\leftarrow S_{\text{new}}$ (Copy)
Improved $\leftarrow$ TRUE
Break the loop over i (an improvement was found)
If *Improved* is TRUE
$S_{\text{current}} = S_{\text{best}}$ (Restart the search from s_cur-)

The swap neighborhood size is O(p*(n-p)), making exhaustive exploration feasible for moderate-sized problems. The best-improvement strategy ensures convergence to local optimal.

Computational Work

To comprehensively evaluate heuristic performance, we generated three sets of problem instances with varying scales:

- SET_100: 20 instances with 100 demand points, 30 potential sites, 8 facilities

- SET_1000: 20 instances with 1000 demand points, 100 potential sites, 15 facilities
- SET_10000: 20 instances with 10000 demand points, 200 potential sites, 25 facilities

Coverage matrices were generated with controlled densities (12%, 8% and 5% respectively) to ensure challenging yet solvable instances that simulate real-world spatial distributions.

We evaluated algorithm performance using coverage percentage, computational time, improvement from local search and swap count.

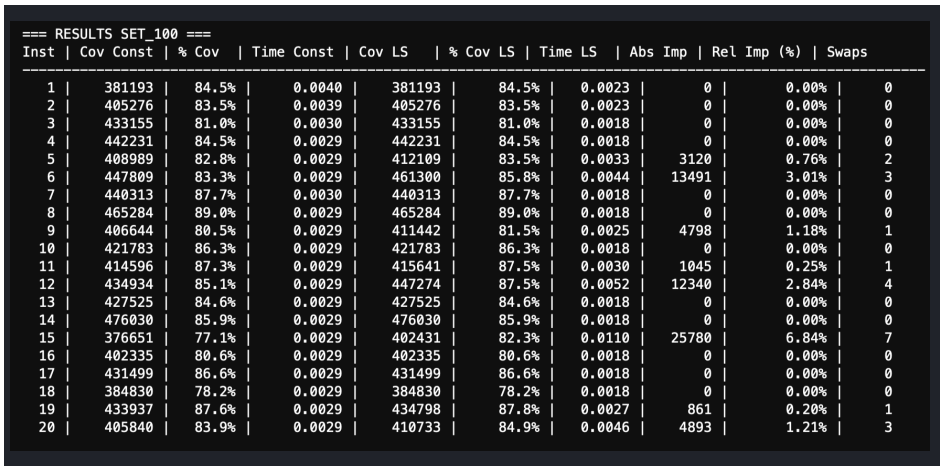
Implementation details:

All algorithms were implemented in Python 3.13.9 using NumPy for efficient matrix operations. Experiments were conducted on an Apple MacBook Air with Apple M4 chip and 24GB RAM. Each instance was solved independently to ensure statistical reliability.

Results by problem scale

Set 1(Small instances):

- Constructive coverage: 77.1% to 89.0%
- Average relative improvement: 0.81%
- Instances improved: 8/20 (40%)
- Average computational time: 0.003 seconds



```
=== RESULTS SET_100 ===
Inst | Cov Const | % Cov  | Time Const | Cov LS   | % Cov LS | Time LS  | Abs Imp | Rel Imp (%) | Swaps
-------------------------------------------------------------------------------------------------------
   1 |    381193 |  84.5% |     0.0040 |    381193 |    84.5% |   0.0023 |       0 |       0.00% |     0
   2 |    405276 |  83.5% |     0.0039 |    405276 |    83.5% |   0.0023 |       0 |       0.00% |     0
   3 |    433155 |  81.0% |     0.0030 |    433155 |    81.0% |   0.0018 |       0 |       0.00% |     0
   4 |    442231 |  84.5% |     0.0029 |    442231 |    84.5% |   0.0018 |       0 |       0.00% |     0
   5 |    408989 |  82.8% |     0.0029 |    412109 |    83.5% |   0.0033 |    3120 |       0.76% |     2
   6 |    447809 |  83.3% |     0.0029 |    461300 |    85.8% |   0.0044 |   13491 |       3.01% |     3
   7 |    440313 |  87.7% |     0.0030 |    440313 |    87.7% |   0.0018 |       0 |       0.00% |     0
   8 |    465284 |  89.0% |     0.0029 |    465284 |    89.0% |   0.0018 |       0 |       0.00% |     0
   9 |    406644 |  80.5% |     0.0029 |    411442 |    81.5% |   0.0025 |    4798 |       1.18% |     1
  10 |    421783 |  86.3% |     0.0029 |    421783 |    86.3% |   0.0018 |       0 |       0.00% |     0
  11 |    414596 |  87.3% |     0.0029 |    415641 |    87.5% |   0.0030 |    1045 |       0.25% |     1
  12 |    434934 |  85.1% |     0.0029 |    447274 |    87.5% |   0.0052 |   12340 |       2.84% |     4
  13 |    427525 |  84.6% |     0.0029 |    427525 |    84.6% |   0.0018 |       0 |       0.00% |     0
  14 |    476030 |  85.9% |     0.0029 |    476030 |    85.9% |   0.0018 |       0 |       0.00% |     0
  15 |    376651 |  77.1% |     0.0029 |    402431 |    82.3% |   0.0110 |   25780 |       6.84% |     7
  16 |    402335 |  80.6% |     0.0029 |    402335 |    80.6% |   0.0018 |       0 |       0.00% |     0
  17 |    431499 |  86.6% |     0.0029 |    431499 |    86.6% |   0.0018 |       0 |       0.00% |     0
  18 |    384830 |  78.2% |     0.0029 |    384830 |    78.2% |   0.0018 |       0 |       0.00% |     0
  19 |    433937 |  87.6% |     0.0029 |    434798 |    87.8% |   0.0027 |     861 |       0.20% |     1
  20 |    405840 |  83.9% |     0.0029 |    410733 |    84.9% |   0.0046 |    4893 |       1.21% |     3
```

Set 2 (Medium instances)

- Constructive coverage: 76.6% to 77.4%
- Average relative improvement: 0.29%
- Instances improved: 12/20 (60%)
- Average computational time: 0.12 seconds

```
=== RESULTS SET_1000 ===
Inst | Cov Const | % Cov  | Time Const | Cov LS   | % Cov LS | Time LS  | Abs Imp | Rel Imp (%) | Swaps
-------------------------------------------------------------------------------------------------------
   1 |   4005127 |   82.6% |     0.1215 |   4027286 |    83.1% |   0.1454 |   22159 |       0.55% |     3
   2 |   4136094 |   82.4% |     0.1217 |   4153770 |    82.8% |   0.1281 |   17676 |       0.43% |     3
   3 |   4165183 |   82.4% |     0.1205 |   4174647 |    82.6% |   0.1662 |    9464 |       0.23% |     1
   4 |   4059236 |   81.5% |     0.1214 |   4060536 |    81.5% |   0.1101 |    1300 |       0.03% |     1
   5 |   4054094 |   82.1% |     0.1211 |   4054094 |    82.1% |   0.0687 |       0 |       0.00% |     0
   6 |   4083274 |   81.3% |     0.1202 |   4083274 |    81.3% |   0.0682 |       0 |       0.00% |     0
   7 |   4134250 |   82.6% |     0.1203 |   4134250 |    82.6% |   0.0687 |       0 |       0.00% |     0
   8 |   4121540 |   81.7% |     0.1206 |   4161150 |    82.4% |   0.0826 |   39610 |       0.96% |     2
   9 |   4143994 |   82.8% |     0.1205 |   4147733 |    82.9% |   0.1067 |    3739 |       0.09% |     1
  10 |   4185270 |   81.5% |     0.1207 |   4185270 |    81.5% |   0.0689 |       0 |       0.00% |     0
  11 |   4310814 |   82.5% |     0.1225 |   4320759 |    82.7% |   0.1889 |    9945 |       0.23% |     2
  12 |   4278127 |   81.8% |     0.1222 |   4278127 |    81.8% |   0.0693 |       0 |       0.00% |     0
  13 |   4110852 |   82.6% |     0.1209 |   4110852 |    82.6% |   0.0685 |       0 |       0.00% |     0
  14 |   4059921 |   81.0% |     0.1210 |   4114414 |    82.1% |   0.2256 |   54493 |       1.34% |     3
  15 |   4104500 |   83.3% |     0.1208 |   4104500 |    83.3% |   0.0687 |       0 |       0.00% |     0
  16 |   4234104 |   82.5% |     0.1205 |   4234104 |    82.5% |   0.0689 |       0 |       0.00% |     0
  17 |   4190223 |   82.0% |     0.1214 |   4223984 |    82.6% |   0.0970 |   33761 |       0.81% |     1
  18 |   4070444 |   80.6% |     0.1214 |   4078168 |    80.7% |   0.2219 |    7724 |       0.19% |     2
  19 |   4184090 |   81.3% |     0.1210 |   4208985 |    81.8% |   0.1658 |   24895 |       0.59% |     3
  20 |   4019060 |   82.4% |     0.1201 |   4033877 |    82.7% |   0.1987 |   14817 |       0.37% |     3
```

Set 3 (Large Instances)

- Constructive coverage: 76.6% to 77.4%
- Average relative improvement: 0.20%
- Instances improved: 16/20 (80%)
- Average computational time for constructive heuristic: 5.3 seconds
- Average computational time for local search: 11.2 seconds

```
=== RESULTS SET_10000 ===
Inst | Cov Const | % Cov  | Time Const | Cov LS   | % Cov LS | Time LS  | Abs Imp | Rel Imp (%) | Swaps
-------------------------------------------------------------------------------------------------------
   1 |  38951211 |   76.8% |     5.3919 |  39048502 |    77.0% |   9.5882 |   97291 |       0.25% |     4
   2 |  39164963 |   77.2% |     5.3490 |  39243696 |    77.4% |  14.2555 |   78733 |       0.20% |     5
   3 |  38488894 |   76.6% |     5.3332 |  38671206 |    76.9% |  12.3156 |  182312 |       0.47% |     6
   4 |  38889897 |   77.1% |     5.3086 |  39069428 |    77.5% |  24.2917 |  179531 |       0.46% |     8
   5 |  38990465 |   77.2% |     5.2848 |  38990465 |    77.2% |   2.8748 |       0 |       0.00% |     0
   6 |  39003756 |   77.2% |     5.2293 |  39031457 |    77.3% |   7.0727 |   27701 |       0.07% |     2
   7 |  39183779 |   76.9% |     5.3696 |  39222305 |    77.0% |  10.8945 |   38526 |       0.10% |     4
   8 |  38926712 |   77.4% |     5.3681 |  38931232 |    77.4% |   7.5699 |    4520 |       0.01% |     2
   9 |  38468565 |   76.9% |     5.3847 |  38567817 |    77.1% |  21.7868 |   99252 |       0.26% |     8
  10 |  38595611 |   77.0% |     5.3396 |  38683451 |    77.2% |   9.7999 |   87840 |       0.23% |     4
  11 |  39005163 |   77.0% |     5.2528 |  39049843 |    77.0% |   5.1425 |   44680 |       0.11% |     2
  12 |  38853525 |   76.9% |     5.2300 |  38911700 |    77.1% |   6.2231 |   58175 |       0.15% |     2
  13 |  39123165 |   77.0% |     5.2250 |  39181473 |    77.2% |   6.4397 |   58308 |       0.15% |     1
  14 |  38970618 |   76.9% |     5.2193 |  38970618 |    76.9% |   2.8854 |       0 |       0.00% |     0
  15 |  39010888 |   77.1% |     5.2320 |  39165212 |    77.4% |  25.3550 |  154324 |       0.40% |    10
  16 |  38862671 |   76.9% |     5.2287 |  38984957 |    77.1% |  14.0125 |  122286 |       0.31% |     3
  17 |  39069353 |   77.3% |     5.2366 |  39196823 |    77.5% |  20.5815 |  127470 |       0.33% |     6
  18 |  38979237 |   77.1% |     5.2425 |  38979237 |    77.1% |   2.8916 |       0 |       0.00% |     0
  19 |  38825937 |   76.9% |     5.2450 |  38825937 |    76.9% |   2.8748 |       0 |       0.00% |     0
  20 |  38920869 |   77.0% |     5.2632 |  39077258 |    77.3% |  14.7044 |  156389 |       0.40% |     5
```

Conclusions

This computational study demonstrates that simple, yet well-designed heuristics can achieve high-quality solutions for the Maximal Covering Location Problem. The Greedy Adding heuristic proved exceptionally effective, achieving 77-89% coverage across problem scales with minimal computational requirements. Local search with swap neighborhoods provided valuable refinements.

References

[1] C. R. R.L. Church, The maximal covering location problem, Baltimore, Maryland: Papers of the Regional Science Association, 1974.

[2] D. M. &. S. D. Current J., «Discrete network location models In Facility location: Applications and theory,» Springer, 2002, pp. 81-118.

[3] M. S. Daskin, Network and discrete location: Models, algorithms and applications, John Wiley & Sons, 1995.

[4] R. A. N. H. N. H. M. &. G. M. Farahani, Covering problems in facility location: A review., Computers & Industrial Engineering, 2012.

[5] C. &. E. H. ReVelle, Location analysis: A synthesis and survey, European Journal of Operational Research, 2005.

[6] V. &. S. D. Marianov, Location problems in the public sector. In Facility location: Applications and theory, Springer, 2002.

[7] N. &. S. K. J. Megiddo, On the complexity of some common geometric location problems, SIAM Journal on Computing, 1984.

[8] S. D. M.H. Fazel Zarandi, «Science Direct,» 2011. [En línea]. Available: https://www.sciencedirect.com/science/article/pii/S1026309811002100. [Último acceso: 4 Septiembre 2025].