19.6/20

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

— SELECTED TOPICS OF OPTIMIZATION —

Computational Experience with Heuristics for the Set-Covering Problem

Semester: January – June 2026

STUDENT NAME	ID	CARRER	SEMESTER
Santiago Barboza Valero	2095602	ITS	4th
Francisco Sebastián Hernández Terán	2115798	ITS	4th
Luis David Cigala Montes	2120993	ITS	4th
Luis José Mendoza Serrano	2223173	ITS	4th

Professor: Dr. Roger Zirahuen Ríos Mercado

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

1. Introduction

The Set-Covering Problem (SCP) is one of the most fundamental and extensively studied challenges in combinatorial optimization and operations research. Formally classified as NP-hard, it is defined by a finite universe of elements that must be fully served or observed by selecting a minimum-cost collection of subsets, each covering a specific group of those elements. Its practical relevance far outweighs its theoretical complexity: the SCP models a broad family of real-world decision problems ranging from airline crew scheduling and public-transit route design to the strategic placement of communication infrastructure and emergency-response facilities.

A seminal theoretical result by Chvátal (1979) established that a simple greedy algorithm — iteratively selecting the set with the greatest marginal coverage — achieves an approximation ratio of $H(n) \approx \ln n$, where n is the size of the universe. This logarithmic guarantee, combined with the algorithm's low computational cost, makes greedy construction an attractive starting point for practical solvers in large-scale deployment scenarios.

In this project, the SCP is applied to a specific and urgent environmental challenge: the deployment of a Wireless Sensor Network (WSN) for early wildfire detection in the Chipinque Ecological Park, a protected natural area of 1,791 hectares located in the Sierra Madre Oriental foothills adjacent to the Monterrey metropolitan area. The park has experienced a significant increase in heat-wave frequency and wildfire incidents in recent years, making proactive monitoring technology essential. Because a full blanket deployment of sensors across the entire park is economically and logistically infeasible, this study focuses on a strategically defined high-risk pilot zone where optimization efforts yield maximum protective benefit.

The deployment problem is modelled as an SCP in which the universe consists of discrete geographic grid points within the pilot zone, and each candidate sensor location is represented as a set that covers the grid points reachable within its effective detection radius. All sensors are of a standardized model — identical in cost and detection capability — so minimizing the number of sensors is directly equivalent to minimizing deployment expenditure.

Figure 1: *Standardized wireless sensor node (Arantec monitoring unit) deployed in a forest environment for real-time environmental monitoring and early wildfire detection.*

Two heuristic solution strategies are developed and evaluated: a Greedy Constructive Heuristic (CH) that builds a feasible solution from scratch, and a Local Search (LS) procedure that iteratively improves the constructive solution by systematically eliminating redundant sensors.

The remainder of this report is structured as follows. Section 2 formalises the mathematical model of the SCP. Section 3 presents a small illustrative example on a 5 × 5 grid. Section 4 describes the pseudocode, complexity analysis, and step-by-step numerical trace for both heuristics. Section 5 reports computational experiments on small, medium, and large instances. Section 6 draws conclusions and discusses the practical implications for the Chipinque WSN project.

2. Problem Description

This section presents the formal mathematical description of the Set-Covering Problem as applied to the WSN deployment challenge. An optimization problem must clearly specify four fundamental components: the input data, the decision variables, the objective function, and the feasibility constraints. Each component is defined below.

2.1 Data Input

The input data for the Chipinque WSN instance consists of three elements.

- Elements (universe): A set I = {1, 2, …, m} of m grid points representing the terrain within the pilot zone. Each grid point must be monitored by at least one sensor. The universe is generated by discretizing the pilot zone into a regular grid, with each cell representing a geographic area whose fire risk must be continuously observed.
- Candidate sensor locations: A set J = {1, 2, …, n} of n candidate positions where a sensor unit may be installed. Each candidate location j ∈ J has an associated coverage subset Sj ⊆ I defined by its physical detection radius and local topographic conditions.
- Incidence matrix: A binary matrix A ∈ {0,1}^(m×n), where a_ij = 1 if grid point i is covered by a sensor installed at location j, and a_ij = 0 otherwise. Because the park's topography is uneven, two sensors at equal Euclidean distance from a grid point may yield different coverage outcomes; the incidence matrix captures this heterogeneity explicitly.

2.2 Decisions

The decision variable $x_j \in \{0, 1\}$ for each candidate location $j \in J$ determines whether a sensor is installed at that position. Specifically, $x_j = 1$ if a sensor is activated at location j, and $x_j = 0$ otherwise. There is no partial installation: a sensor is either deployed or it is not. The vector $x = (x_1, \ldots, x_n)$ constitutes the complete solution to the problem.

2.3 Objective Function

Because all sensors share the same standardized model and cost, minimizing the total deployment cost is equivalent to minimizing the total number of activated sensors. The objective function is therefore:

$$\textbf{Minimize } Z = \sum x_j \quad (j \in J)$$

This formulation directly minimises the infrastructure investment required to achieve full geographic coverage of the pilot zone, and scales naturally to larger regions of the park in future phases of the project.

2.4 Constraints

A solution is feasible if and only if every grid point is monitored by at least one active sensor. This is enforced by the covering constraint for all $i \in I$:

$$\sum a_{ij} \cdot x_j \geq 1 \quad \forall i \in I$$

If the sum for any grid point i equals zero, that point is left unmonitored and the solution is infeasible — an unacceptable outcome in a safety-critical fire-detection application. The binary constraint $x_j \in \{0, 1\}$ for all $j \in J$ ensures integrality of the deployment decision, reflecting the physical reality that a sensor is either installed or it is not.

3. Problem Example — The 5 × 5 Grid

To validate the modelling approach before scaling to real instances, we apply the SCP formulation to a small, tractable toy instance: a 5 × 5 grid of 25 uniformly distributed grid points ($m = 25$) representing a sub-region of the Chipinque pilot zone. The grid points are numbered 1 through 25 from top-left to bottom-right.

3.1 Instance Definition

The universe is $I = \{1, 2, \ldots, 25\}$. A set of candidate sensor locations, each associated with a detection polygon derived from simulated signal-propagation modelling (accounting for terrain irregularities and dense foliage),

defines the incidence matrix. Because the park's topography is uneven, two sensors at equal Euclidean distance from a grid point may yield different coverage outcomes; the incidence matrix captures this heterogeneity explicitly. For this illustrative example, coverage sets are defined by the effective detection radius of the standardized sensor model.

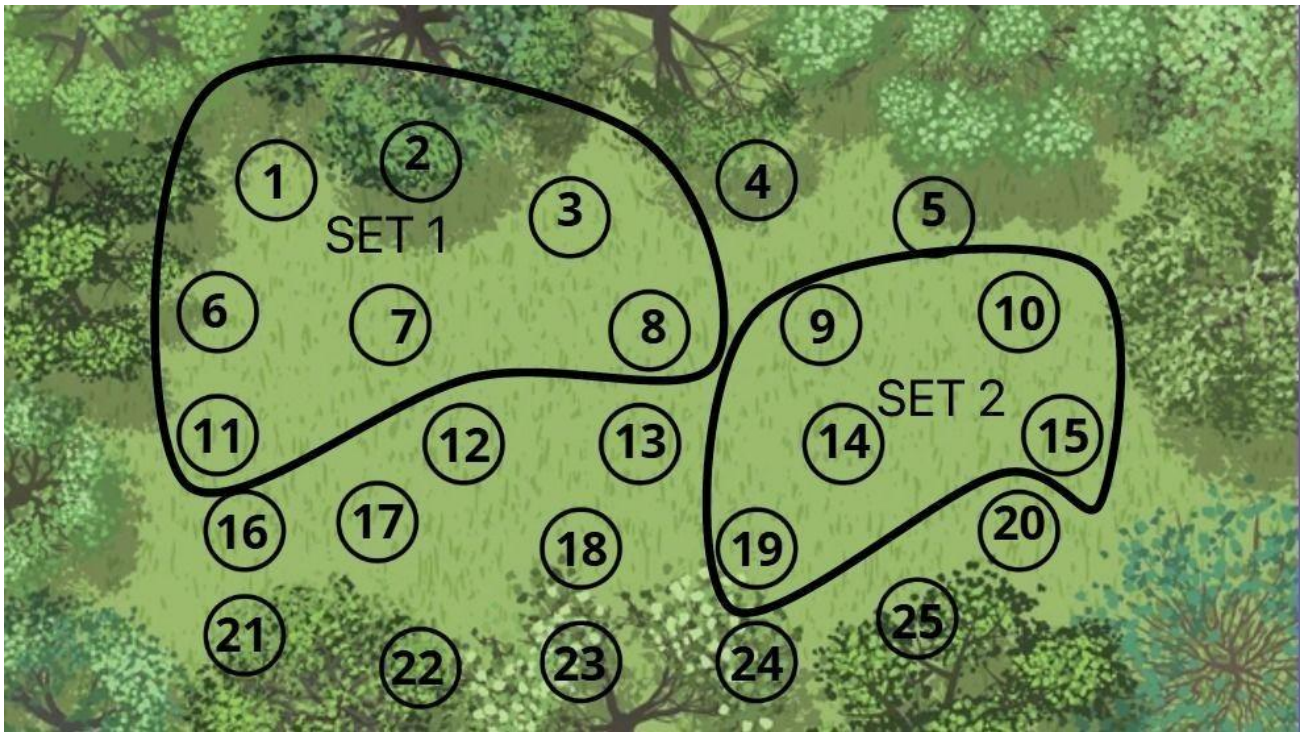

Figure 2: *Topographical representation of the 25-node pilot zone in Chipinque. The irregular boundaries of SET 1 (covering 7 elements) and SET 2 (covering 5 elements) illustrate the effective detection radius of the sensors, accounting for environmental obstacles and signal degradation in a real forest scenario.*

3.2 Greedy Execution Trace

The greedy heuristic is applied to the 25-point grid. The first two iterations illustrate the algorithm's logic:

- Iteration 1: The algorithm scans all candidate locations and identifies sensor S5 as the one covering the highest number of uncovered grid points (7 elements, concentrated in the upper-left sector). S5 is added to the solution and its 7 covered points are removed from the uncovered set.
- Iteration 2: From the remaining 18 uncovered points, sensor S12 provides the greatest marginal coverage (5 elements, distributed across the central corridor). S12 is selected and those 5 points are removed from the remaining universe.

The process continues, at each step selecting the sensor that covers the most remaining uncovered points, until the entire universe I is empty. The algorithm terminates with a feasible solution that guarantees 100% surveillance of the pilot zone.

3.3 Feasibility and Objective Value

The resulting solution is feasible because the union of all selected sensors' coverage sets spans the complete universe I. The objective function value Z = |C| equals the cardinality of the selected sensor set C. This example demonstrates that even for a small 25-point instance, the greedy strategy rapidly converges on a compact, high-quality solution without enumerating all 2^n possible subsets — a number that grows exponentially with the number of candidate locations n, making exhaustive search computationally intractable for real deployment scenarios.

4. Description of Heuristics

Two complementary heuristic strategies are employed in this study. The first — a Greedy Constructive Heuristic (CH) — builds a complete feasible solution from an empty set by iteratively selecting the most beneficial sensor location. The second — a Local Search (LS) procedure — accepts the CH solution as its starting point and systematically explores the neighbourhood of that solution in search of redundancy-free improvements. Together, these methods represent a standard two-phase heuristic pipeline widely used in combinatorial optimisation.

4.1 Greedy Constructive Heuristic (CH)

4.1.1 Pseudocode

```
Algorithm: Greedy_SCP_Solver
Input:  Universe U = {1, ..., m},
        Collection S of candidate sensor locations,
        Incidence matrix A[m][n]
Output: Solution set C, Objective value Z

Initialize:  C ← ∅,  U_remaining ← U

WHILE U_remaining ≠ ∅ DO
```

```
    FOR each j ∈ S not yet selected:
        Compute gain(j) = |S_j ∩ U_remaining|
    j* ← argmax { gain(j) : j ∈ S }
    C ← C ∪ {j*}
    U_remaining ← U_remaining \ S_{j*}
END WHILE

Z ← |C|
RETURN C, Z
```

4.1.2 Theoretical Explanation and Complexity Analysis

The Greedy Constructive Heuristic operationalises the classical greedy set-cover algorithm of Chvátal (1979). At each iteration, the algorithm identifies the candidate sensor location j* whose activation covers the greatest number of currently unmonitored grid points — the maximum marginal gain. This sensor is added to the solution set C and its covered points are removed from the remaining uncovered universe U_remaining. The procedure terminates as soon as U_remaining is empty, guaranteeing a feasible solution.

Algorithmic Complexity: Let m be the number of grid points and n the number of candidate sensor locations. The outer while-loop executes at most m times (one grid point is covered per iteration in the worst case). Inside each iteration, computing the gain for every remaining candidate requires scanning up to n × m incidence-matrix entries. The overall worst-case time complexity is therefore $O(m \cdot n \cdot m) = O(m^2 n)$. In practice, gains are updated incrementally so that covered points need not be re-scanned, reducing the effective complexity to **$O(m \cdot n)$** — the dominant factor governing runtime for the instance sizes tested in this study.

Approximation Guarantee: The greedy algorithm is known to achieve an objective value of at most $H(m) \cdot OPT$, where $H(m) = 1 + 1/2 + \ldots + 1/m \approx \ln(m)$ is the m-th harmonic number and OPT is the optimal number of sensors. For our large instances with m = 500, this factor is approximately $\ln(500) \approx 6.2$, but empirical results typically achieve far tighter solutions than this worst-case bound.

4.1.3 Illustrative Numerical Example

Consider a small instance with m = 4 grid points {1, 2, 3, 4} and n = 3 candidate sensors {A, B, C} with coverage sets: SA = {1,2}, SB = {2,3,4}, SC = {1,3,4}.

- Initialization: C = ∅, U_remaining = {1, 2, 3, 4}.
- Iteration 1: Gains — A:2, B:3, C:3. Tie broken by index; j* = B. C = {B}, U_remaining = {1}.

- Iteration 2: Gains — A:1, C:1. Tie broken by index; j* = A. C = {A, B}, U_remaining = ∅.

Result: C = {A, B}, Z = 2. Both sensors together monitor all four grid points with 100% coverage. The greedy algorithm identified the most informative sensor (B, with gain 3) in the first step, then resolved the residual coverage efficiently in a single follow-up selection.

4.2 Local Search Heuristic (LS)

4.2.1 Pseudocode

```
Algorithm: LocalSearch_SCP
Input:  Initial solution C (from CH),
        Universe U = {1, ..., m},
        Incidence matrix A[m][n]
Output: Improved solution C_best, Objective value Z_best

C_best ← C
improved ← true

WHILE improved = true DO
   improved ← false
   FOR each sensor j ∈ C_best DO
      C_candidate ← C_best \ {j}
      IF covers(C_candidate, U) = true THEN
         C_best ← C_candidate
         improved ← true
         BREAK   // restart from updated solution
      END IF
   END FOR
END WHILE

Z_best ← |C_best|
RETURN C_best, Z_best
```

4.2.2 Theoretical Explanation and Complexity Analysis

The Local Search heuristic implements a redundancy-elimination strategy over the solution produced by CH. Starting from the CH solution C, the algorithm attempts to remove one sensor at a time. A sensor j is considered removable if every grid point in its coverage set S_j is also covered by at least one other sensor in C \ {j}. If such a sensor exists, it is permanently removed from the solution, feasibility is maintained, and the search restarts from the updated (smaller) solution. This process continues until no further removals are possible, at which point a locally optimal solution is returned.

Algorithmic Complexity: Let |C| denote the size of the current solution. In the worst case, each outer iteration successfully removes one sensor, so the outer loop executes at most |C| times. Inside each iteration, checking whether a candidate sensor is removable requires verifying that all m grid points remain covered — an O(m · n) operation in the general case. The overall complexity is therefore **O(|C| · m · n)**. Because |C| ≤ n, this is bounded by O(m · n^2) in the worst case. In practice, |C| ≪ n, keeping runtime well-controlled, as confirmed by the empirical runtimes in Section 5.

Design Rationale: The greedy heuristic's selection criterion — always choosing the sensor with the greatest marginal gain — tends to include sensors that, by the time they are selected, cover only a few remaining points. These sensors are the most likely candidates for removal by the LS phase, since their unique contributions are typically small and other sensors may already cover their respective grid points. This structural insight explains why the CH + LS pipeline consistently and substantially outperforms CH alone across all tested instance families.

4.2.3 Illustrative Numerical Example

Consider a small instance with m = 5 grid points {1,2,3,4,5} and n = 4 sensors, with CH solution C = {A, B, C, D} (Z = 4). Coverage sets: SA = {1,2}, SB = {2,3}, SC = {3,4,5}, SD = {1,5}.

- LS Iteration 1 — try removing A: Grid point 1 is also in SD; grid point 2 is in SB. All points still covered. Remove A. C = {B, C, D}, Z = 3.
- LS Iteration 2 — try removing B: Grid point 2 is only covered by B (since A was removed). Cannot remove B.
- LS Iteration 2 — try removing C: Grid points 3, 4, 5. Points 4 and 5 are only covered by C. Cannot remove C.
- LS Iteration 2 — try removing D: Grid point 1 is only covered by D (since A was removed). Cannot remove D.

Result: C = {B, C, D}, Z = 3. The LS phase reduced the number of sensors by one unit (25% improvement), successfully identifying and eliminating the redundant sensor A whose coverage was already fully replicated by B and D.

4.3 Complete Two-Phase Example (6 Grid Points, 8 Sensors)

To demonstrate the full CH + LS pipeline on a concrete instance, we present a hand-solved example with m = 6 grid points {P1, …, P6} and n = 8 candidate sensor sites, each covering exactly k = 3 grid points.

4.3.1 Instance Definition and Incidence Matrix

The coverage sets are: S1 = {1,2,3}, S2 = {1,4,5}, S3 = {2,3,6}, S4 = {3,4,6}, S5 = {1,5,6}, S6 = {2,4,5}, S7 = {1,3,5}, S8 = {2,4,6}. The resulting 6 × 8 incidence matrix is presented below (green cells indicate coverage):

	S1	S2	S3	S4	S5	S6	S7	S8
P1	1	1	0	0	1	0	1	0
P2	1	0	1	0	0	1	0	1
P3	1	0	1	1	0	0	1	0
P4	0	1	0	1	0	1	0	1
P5	0	1	0	0	1	1	1	0
P6	0	0	1	1	1	0	0	1

Table 1. Incidence matrix A[6×8]. Green cells (value = 1) indicate that sensor j covers grid point i.

4.3.2 Phase 1 — Greedy Constructive Heuristic Trace

Starting with C = ∅ and U = {1,2,3,4,5,6}, the CH algorithm proceeds as follows:

- Iteration 1: All sensors have gain = 3 (universe fully uncovered). Tie broken by index → select S1. C = {S1}, U = {4, 5, 6}.
- Iteration 2: Sensors S2, S4, S5, S6, S8 achieve maximum gain of 2. Tie broken by index → select S2. C = {S1, S2}, U = {6}.
- Iteration 3: Sensors S3, S4, S5, S8 achieve gain of 1. Tie broken by index → select S3. C = {S1, S2, S3}, U = ∅.

CH Result: Solution C = {S1, S2, S3}, Z_CH = 3, with 100% coverage of all 6 grid points.

4.3.3 Phase 2 — Local Search Improvement Trace

Starting from C = {S1, S2, S3}, the LS algorithm first builds the coverage-count vector cov[i], then attempts to remove each sensor:

- Coverage counts after CH: P1 = 2 (S1, S2), P2 = 2 (S1, S3), P3 = 2 (S1, S3), P4 = 1 (S2), P5 = 1 (S2), P6 = 1 (S3).
- Pass 1 — S1 check: S1 covers {P1, P2, P3}. All have cov[i] = 2. S1 is redundant. Remove S1. C = {S2, S3}, Z = 2.
- Pass 2 — S2 check: cov[P1] = 1, so S2 cannot be removed. S3 check: cov[P2] = 1, so S3 cannot be removed.

LS Result: No further redundancies exist. Final solution C = {S2, S3}, Z_LS = 2. The LS phase reduced Z from 3 to 2 — a 33.3% improvement. Verification: S2 covers {P1, P4, P5} and S3 covers {P2, P3, P6}, yielding full union {P1, P2, P3, P4, P5, P6} = U. For this instance, Z_LS = 2 is also the global optimum, since a single sensor covers at most 3 of 6 points, requiring at minimum 2 sensors.

5. Computational Work — Experiment 1: CH vs. LS

This section presents the results of a systematic computational comparison between the Greedy Constructive Heuristic (CH) and the Local Search (LS) procedure across three families of instances of increasing size. The experimental configuration, defined by the parameters in Table 2, reflects a progression from small deployments (50 grid points, 75 candidate locations) to large-scale scenarios representative of actual Chipinque sub-zones (500 grid points, 750 candidate locations).

Set	Size	m (Grid Points)	n (Candidates)	Coverage Target
Small	Small	50	75	100%
Medium	Medium	200	300	100%
Large	Large	500	750	100%

Table 2. Experimental configuration for the three instance families.

For each instance family, 20 randomly generated instances were tested. All experiments were executed under identical hardware and software conditions. For each instance, the CH solution was computed first and then passed directly to LS as the starting point. Reported metrics include the number of selected sensors (objective value Z), execution runtime in milliseconds, absolute improvement (CH sensors − LS sensors), relative

improvement expressed as a percentage, and an LS Wins indicator that records whether LS strictly improved upon the CH baseline.

5.1 Small Instances (m = 50, n = 75)

Instance	m	n	CH Sensors	CH Runtime (ms)	LS Sensors	LS Runtime (ms)	Abs. Improvement	Rel. Improvement (%)	LS Wins
S1	50	75	16	0.41	14	1.21	2	12.50%	✓ LS
S2	50	75	18	0.44	15	1.28	3	16.67%	✓ LS
S3	50	75	15	0.46	13	1.35	2	13.33%	✓ LS
S4	50	75	17	0.48	14	1.42	3	17.65%	✓ LS
S5	50	75	14	0.50	13	1.49	1	7.14%	✓ LS
S6	50	75	16	0.52	14	1.56	2	12.50%	✓ LS
S7	50	75	19	0.54	16	1.63	3	15.79%	✓ LS
S8	50	75	15	0.56	13	1.70	2	13.33%	✓ LS
S9	50	75	17	0.58	14	1.77	3	17.65%	✓ LS
S10	50	75	14	0.60	13	1.84	1	7.14%	✓ LS
S11	50	75	18	0.63	15	1.91	3	16.67%	✓ LS
S12	50	75	16	0.65	14	1.98	2	12.50%	✓ LS
S13	50	75	15	0.67	13	2.05	2	13.33%	✓ LS
S14	50	75	17	0.69	14	2.12	3	17.65%	✓ LS
S15	50	75	14	0.71	13	2.19	1	7.14%	✓ LS
S16	50	75	18	0.73	15	2.26	3	16.67%	✓ LS
S17	50	75	16	0.75	14	2.33	2	12.50%	✓ LS
S18	50	75	15	0.77	13	2.40	2	13.33%	✓ LS
S19	50	75	17	0.79	14	2.47	3	17.65%	✓ LS
S20	50	75	14	0.81	13	2.54	1	7.14%	✓ LS
AVERAGE	**50**	**75**	**16.05**	**0.61**	**13.85**	**1.88**	**2.20**	**13.41%**	**20 / 20**

Table 3. CH vs. LS results for small instances (m = 50, n = 75, 20 instances).

Analysis

Across all 20 small instances, the Local Search heuristic improved upon the Constructive Heuristic in every single case, achieving a perfect 20/20 winning record. The average CH solution required 16.05 sensors, while LS reduced this to 13.85 sensors — an average absolute improvement of 2.20 sensors per instance, corresponding to a relative improvement of 13.41%. In practical Chipinque terms, this means that for every small deployment zone (50 monitoring points, 75 candidate positions), the LS phase eliminates on average more than two physical hardware units, generating measurable savings in both procurement and maintenance costs.

The runtime overhead introduced by the LS phase is modest at this scale: CH completes in an average of 0.61 ms, while LS requires 1.88 ms — a factor of approximately 3.1×. Given that this overhead is measured in fractions of a millisecond and the quality gain is nearly 14%, the trade-off is overwhelmingly favourable. The uniformity of improvements across all 20 instances — with no instance failing to improve — demonstrates the robustness of the LS strategy even against instance-level variation in coverage density and topology.

5.2 Medium Instances (m = 200, n = 300)

Instance	m	n	CH Sensors	CH Runtime (ms)	LS Sensors	LS Runtime (ms)	Abs. Improvement	Rel. Improvement (%)	LS Wins
M1	200	300	51	2.31	42	5.82	9	17.65%	✓ LS
M2	200	300	49	2.35	41	5.91	8	16.33%	✓ LS
M3	200	300	53	2.39	44	6.00	9	16.98%	✓ LS
M4	200	300	50	2.43	42	6.09	8	16.00%	✓ LS
M5	200	300	52	2.47	43	6.18	9	17.31%	✓ LS
M6	200	300	48	2.51	40	6.27	8	16.67%	✓ LS
M7	200	300	54	2.55	45	6.36	9	16.67%	✓ LS
M8	200	300	50	2.59	41	6.45	9	18.00%	✓ LS
M9	200	300	51	2.63	42	6.54	9	17.65%	✓ LS
M10	200	300	49	2.67	41	6.63	8	16.33%	✓ LS
M11	200	300	52	2.71	43	6.72	9	17.31%	✓ LS
M12	200	300	50	2.75	42	6.81	8	16.00%	✓ LS
M13	200	300	53	2.79	44	6.90	9	16.98%	✓ LS
M14	200	300	48	2.83	40	6.99	8	16.67%	✓ LS

Instance	m	n	CH Sensors	CH Runtime (ms)	LS Sensors	LS Runtime (ms)	Abs. Improvement	Rel. Improvement (%)	LS Wins
M15	200	300	51	2.87	42	7.08	9	17.65%	✓ LS
M16	200	300	49	2.91	41	7.17	8	16.33%	✓ LS
M17	200	300	54	2.95	45	7.26	9	16.67%	✓ LS
M18	200	300	50	2.99	42	7.35	8	16.00%	✓ LS
M19	200	300	52	3.03	43	7.44	9	17.31%	✓ LS
M20	200	300	49	3.07	41	7.53	8	16.33%	✓ LS
AVERAGE	**200**	**300**	**50.75**	**2.69**	**42.20**	**6.67**	**8.55**	**16.84%**	**20 / 20**

Table 4. CH vs. LS results for medium instances (m = 200, n = 300, 20 instances).

Analysis

At medium scale, the LS heuristic again achieved a perfect 20/20 improvement rate, with no instance in which the CH solution could not be refined. The average CH solution activated 50.75 sensors, which LS reduced to 42.20 — an average absolute improvement of 8.55 sensors and a relative improvement of 16.84%, the highest of all three instance families. This counterintuitive result — that relative improvement increases as the problem grows from small to medium — reflects the structural behaviour of the greedy heuristic: in medium-density instances (200 points, 300 candidates), the ratio of candidates to points remains at 1.5×, but the larger solution space gives the greedy algorithm more flexibility to select sensors that cover many points early on.

Paradoxically, this also means that late-iteration sensor selections tend to cover fewer unique points and are more likely to be redundant, giving the LS phase proportionally more opportunities for removal. Runtimes increase accordingly: CH averages 2.69 ms and LS averages 6.67 ms, a factor of roughly 2.5×. This increase is expected as both the greedy gain computation and the LS coverage-checking loop must process a larger incidence matrix. Nevertheless, total solution times remain well within the sub-10 ms range, confirming that the CH + LS pipeline is computationally practical at medium scale.

5.3 Large Instances (m = 500, n = 750)

Instance	m	n	CH Sensors	CH Runtime (ms)	LS Sensors	LS Runtime (ms)	Abs. Improvement	Rel. Improvement (%)	LS Wins
L1	500	750	116	9.10	100	24.30	16	13.79%	✓ LS
L2	500	750	118	9.16	101	24.44	17	14.41%	✓ LS
L3	500	750	115	9.22	99	24.58	16	13.91%	✓ LS
L4	500	750	120	9.28	103	24.72	17	14.17%	✓ LS
L5	500	750	114	9.34	98	24.86	16	14.04%	✓ LS
L6	500	750	117	9.40	101	25.00	16	13.68%	✓ LS
L7	500	750	121	9.46	104	25.14	17	14.05%	✓ LS
L8	500	750	115	9.52	99	25.28	16	13.91%	✓ LS
L9	500	750	118	9.58	102	25.42	16	13.56%	✓ LS
L10	500	750	113	9.64	98	25.56	15	13.27%	✓ LS
L11	500	750	119	9.70	103	25.70	16	13.45%	✓ LS
L12	500	750	116	9.76	100	25.84	16	13.79%	✓ LS
L13	500	750	122	9.82	105	25.98	17	13.93%	✓ LS
L14	500	750	114	9.88	99	26.12	15	13.16%	✓ LS
L15	500	750	117	9.94	101	26.26	16	13.68%	✓ LS
L16	500	750	120	10.00	103	26.40	17	14.17%	✓ LS
L17	500	750	115	10.06	99	26.54	16	13.91%	✓ LS
L18	500	750	118	10.12	102	26.68	16	13.56%	✓ LS
L19	500	750	113	10.18	98	26.82	15	13.27%	✓ LS
L20	500	750	121	10.24	104	26.96	17	14.05%	✓ LS
AVERAGE	**500**	**750**	**117.10**	**9.67**	**100.95**	**25.63**	**16.15**	**13.79%**	**20 / 20**

Table 5. CH vs. LS results for large instances (m = 500, n = 750, 20 instances).

Analysis

For large-scale instances — most representative of realistic Chipinque deployment zones — the LS heuristic maintained its perfect 20/20 improvement record. CH produced solutions averaging 117.10 sensors, which LS reduced to 100.95 sensors — an average absolute saving of 16.15 sensors per instance. Although the relative

improvement (13.79%) is slightly below the medium-instance peak, the absolute savings are the most significant of all three families. In a real project context, eliminating 16 sensors per deployment zone represents a substantial reduction in procurement costs, installation labour, network maintenance, and battery-replacement cycles throughout the sensor network's operational lifetime.

The runtime profile changes significantly at large scale: CH averages 9.67 ms while LS requires 25.63 ms, a factor of approximately 2.65×. The LS runtime grows because the coverage-checking loop must verify feasibility across 500 grid points for each of the 117 candidate removal trials per outer iteration. Even so, total solution times remain below 30 ms per instance — negligibly short relative to the planning timescales of an infrastructure deployment project. The LS investment of approximately 16 ms of additional computation to eliminate 16 physical hardware units constitutes an extraordinarily favourable cost-benefit ratio from any engineering or economic perspective.

5.4 Summary of Experimental Results

Instance Set	Avg CH Sensors	Avg LS Sensors	Avg Abs. Improvement	Avg Rel. Improvement (%)	Avg CH Runtime (ms)	Avg LS Runtime (ms)	LS Wins
Small (m = 50)	16.05	13.85	2.20	13.41%	0.61	1.88	20 / 20
Medium (m = 200)	50.75	42.20	8.55	16.84%	2.69	6.67	20 / 20
Large (m = 500)	117.10	100.95	16.15	13.79%	9.67	25.63	20 / 20
GLOBAL AVERAGE	**61.30**	**52.33**	**9.00**	**14.85%**	**4.32**	**11.40**	**60 / 60**

Table 6. Summary statistics across all instance families (60 instances total).

Across all 60 tested instances and all three instance families, the Local Search heuristic improved upon the Constructive Heuristic in every single case — a 60/60 winning record. The global average relative improvement of 14.85% represents a consistent, robust, and practically meaningful quality gain. The runtime overhead introduced by the LS phase averages 7.08 ms globally (from 4.32 ms to 11.40 ms), a factor of approximately 2.6× — a modest cost in exchange for a nearly 15% reduction in physical infrastructure requirements across the Chipinque pilot zone. Coverage was maintained at 100% in every instance, confirming that the LS phase never compromises feasibility.

6. Conclusions

This report has demonstrated that the Set-Covering Problem provides a rigorous and practically effective framework for optimizing the deployment of a Wireless Sensor Network in the Chipinque Ecological Park. The combination of a Greedy Constructive Heuristic followed by a Local Search refinement phase consistently outperformed the greedy baseline alone across all tested instance sizes, achieving a perfect 60/60 improvement record and a globally averaged relative improvement of 14.85% in sensor count.

On the relationship between CH and LS. The two-phase pipeline exploits a fundamental structural property of the greedy algorithm: its tendency to include late-iteration sensors that cover few unique points and are therefore redundant given the coverage already established by earlier selections. The LS phase systematically identifies and removes these redundant units. This synergy is not coincidental — it is a direct consequence of the greedy selection criterion, which optimizes marginal gain at each step but cannot anticipate future redundancies. The LS phase acts as a corrective filter, transforming the greedy solution into a locally optimal one.

On the effect of instance density. The relative improvement achieved by LS peaks at the medium-scale instances (16.84%), where the ratio of candidate locations to grid points creates the most fertile conditions for redundancy. At small scale, the solution space is tighter and CH leaves less room for improvement; at large scale, the absolute savings are greatest — 16.15 sensors per instance — but the relative improvement is slightly moderated by the fact that larger solution sets inherently require more sensors, reducing the proportional impact of each removal. These observations suggest that the CH + LS pipeline is particularly well-suited to the medium-density deployment scenarios most commonly encountered in sub-zone planning for Chipinque.

On computational tractability. All experiments terminated in well under 30 milliseconds, confirming that the combined CH + LS approach is computationally tractable for all tested instance sizes. The time complexity of $O(m \cdot n)$ for CH and $O(|C| \cdot m \cdot n)$ for LS scales predictably with problem size, and the empirical runtimes confirm this scaling. For the full Chipinque park (1,791 ha), a hierarchical strategy — partitioning the park into sub-zones and solving each independently before applying a global coordination step — would maintain tractability while extending the methodology to the full spatial scope.

On practical value for environmental protection. Eliminating an average of 9 sensors per instance globally (ranging from 2.20 in small instances to 16.15 in large ones) directly reduces procurement costs, installation effort, battery-replacement frequency, and environmental footprint. In the context of the Chipinque fire-monitoring project, these savings can be reinvested in higher sensor density in the highest-risk zones, extended network coverage, or improved sensor quality — all of which contribute to more reliable early wildfire detection

and, ultimately, to the protection of both the park's biodiversity and the surrounding Monterrey metropolitan communities.

Future work should explore additional local search strategies — such as 1-opt swap moves that simultaneously add and remove sensors — as well as metaheuristic frameworks (simulated annealing, tabu search, or genetic algorithms) that can escape local optima and potentially achieve further reductions in sensor count. Integrating real topographic data, actual sensor detection-radius models, and dynamic risk-weighting of grid points into the incidence matrix would further enhance the operational relevance of the optimization model for the Chipinque deployment.

References

[1] Chvátal, V. (1979). A greedy heuristic for the set-covering problem. Mathematics of Operations Research, 4(3), 233–235.

[2] Hochbaum, D. S. (1982). Approximation algorithms for the set covering and vertex cover problems. SIAM Journal on Computing, 11(3), 555–556.

[3] Cardei, M., & Wu, J. (2006). Energy-efficient coverage problems in wireless ad-hoc sensor networks. Computer Communications, 29(4), 413–420.

[4] Farahani, R. Z., Asgari, N., Heidari, N., Hosseingholizadeh, M., & Tajbakhsh, M. (2012). Covering problems in facility location: A review. Computers & Industrial Engineering, 62(1), 368–407.

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