

FIXED-SET LEARNING FOR CLUSTER-HEAD SELECTION IN MULTI-HOP WIRELESS SENSOR NETWORKS

Raouf Ouanis Lakehal Ayat¹ and Salim Bouamama²

(Received: 16-Feb.-2026, Revised: 7-Apr.-2026, Accepted: 30-Apr.-2026)

ABSTRACT

Wireless Sensor Networks (WSNs) have remained an active research field in both military and civilian domains, driven by the expanding diversity of their applications. In recent years, there has been a progressive shift toward integrating Artificial Intelligence to address the persistent challenge of energy optimization in WSNs. We introduce a novel adaptation of a Fixed Set Search (FSS) mechanism to WSNs. FSS adds a learning phase to the well-known GRASP metaheuristic. FSS-WSN approach guides the Base Station (BS) in a centralized multi-hop environment to select the optimal cluster-heads, thereby maximizing the global utility of the network. We evaluated our approach under documented fairness conditions, against a wide range of established baselines including classical clustering protocols (LEACH, HEED, SEP), widely used swarm optimizers (PSO, GWO, ABC), the recent multi-hop hybrid EEM-LEACH-ABC, and recent SO-GJO-family variants (SO, GJO, EMO-GJO, and ESO-GJO). The results demonstrate a statistically significant improvement (paired Wilcoxon test with Holm correction) over the best baseline regarding two key metrics—the number of delivered reports and the CPU time required for decision-making. These results suggest that our approach is a strong, practical option for many WSN use cases.

KEYWORDS

Wireless sensor networks, Cluster-head selection, Multi-hop, Fixed Set Search, Metaheuristics, Energy efficiency.

1. INTRODUCTION

Energy remains a major limitation in most Wireless Sensor Networks (WSNs), particularly in remote or safety-critical deployments where replacing batteries is impractical. On many platforms, radio communication accounts for the largest share of energy consumption. Consequently, both network lifetime and the volume of data successfully delivered to the Base Station (BS) are closely tied to the way in which measurements are aggregated and forwarded [1] [2] [3].

WSN-based IoT architectures have been deployed in diverse safety-critical domains, including gas-leakage detection [4], smart-grid monitoring [5], and industrial-process control [1]. In all such settings, energy-efficient clustering directly impacts system reliability and data availability.

To address this issue, clustering has become a common approach. Sensor nodes transmit short-range reports to a cluster head (CH), which aggregates the data and forwards it to the BS, thereby reducing the number of long-distance transmissions. To avoid premature energy depletion of a small subset of nodes, most clustering protocols periodically rotate the CH role, as in LEACH [6] and HEED [7]. In networks with heterogeneous initial energy levels, CH selection is often weighted in favor of higher-energy nodes, as in SEP [8].

Beyond extending network lifetime, many WSN applications place strong emphasis on the timely and reliable delivery of decision-relevant information. In scenarios, such as industrial monitoring, emergency response, or perimeter surveillance, the objective is not only to keep the network operational, but also to maximize the amount of useful data delivered under strict time and energy constraints.

Things get more complicated when multi-hop forwarding enters the picture, because coverage and routing can no longer be treated separately. The set of CHs chosen in a given round does not just decide which sensor nodes are covered—it also shapes the relay chain that funnels data back to the BS. In practice, CHs that sit near the sink often end up relaying traffic for distant clusters on top of their own, which drains them faster and creates localized energy bottlenecks [9] [2]. Most existing formulations gloss over this coupling, either by using rough distance-based surrogates or by folding feasibility into penalty terms that the optimizer can partially ignore.

1. R. O. Lakehal Ayat is with Department of Computer Science, University of M'sila, Algeria. Email: raouf.lakehalayat@univ-msila.dz
2. S. Bouamama is with Department of Computer Science, University of Setif 1-Ferhat Abbas, Sétif 19000, Algeria. Email: salim.bouamama@univ-setif.dz

1.1 Related Work

Over the past two decades, a large number of protocols and algorithms have been proposed for cluster-head selection and clustered routing in WSNs. Early work focused on lightweight distributed heuristics, while more recent studies have increasingly turned to optimization-based formulations. We review the most relevant contributions below, grouped by theme.

- **Energy-sensitive clustering protocols:** A handful of distributed protocols still set the reference point for CH election. LEACH [6] introduced probabilistic self-election with round-by-round role rotation and local data aggregation—a design that remains surprisingly competitive given its simplicity. HEED [7] took a slightly different route, iterating over both residual energy and intra-cluster communication cost; the result, in our experience, is noticeably more balanced cluster sizes. For networks where batteries are not identical, SEP [8] provides a natural fix by scaling each node's election weight with its initial energy. A more recent entry is EEM-LEACH-ABC [10], in which Zhang et al. use an Artificial Bee Colony optimizer to automatically set the CH ratio and an energy-blending parameter within a LEACH-like framework. While their numbers look favorable, the study covers only one network layout under fixed settings and does not include replicated experiments, which makes it difficult to draw firm conclusions. It is worth stressing that every protocol in this group is distributed by design; we therefore treat them as reference baselines and not as direct alternatives to the centralized, multi-hop optimization that we develop.
- **Metaheuristic-based CH selection:** CH selection is often framed as an optimization problem, typically involving a weighted combination of energy, distance, and load balance [11], on the face of it, a sensible idea, and a long list of continuous-space metaheuristics have been applied to it: PSO [12], GWO [13], ABC [14], SO [15], GJO [16], among others. Some of these are growing more sophisticated. Gupta et al., for instance, fold relay-load balancing into a GSO-based clustering framework [17] and tackle path planning for heterogeneous WSNs separately [18]. All of this sounds promising, yet the numbers are hard to trust across studies: change in the continuous-to-discrete decoding rule or tweak coverage feasibility checking, and the rankings can shift substantially [19].
- **Augmented metaheuristic frameworks:** A few recent studies have tried to patch the encoding and feasibility problems by bolting domain-specific modules onto existing metaheuristics. Mazumder et al. recast CH selection as a multi-objective energy-aware task and attacked it with a Golden Jackal Optimizer variant which they call EMO-GJO [20]—an interesting formulation, though the evaluation stays within a single simulator setup. Wang et al. [21] went further: their ESO-GJO couples the Snake Optimizer with GJO, so that CH placement and multi-hop routing are decided in one pass. Both papers show energy and lifetime gains, but since each team ran its own simulator with its own parameter conventions, putting their numbers side by side with anything else is not straightforward.

Structural limitations of continuous-space approaches: Strip away the algorithmic labels and every metaheuristic reviewed above—plain or augmented—does the same thing: search a continuous $[0,1]^N$ space, then snap the result to a binary CH vector with a threshold or top-k rule. That two-step detour has real consequences. Huge swaths of the continuous space collapse onto the same CH set, so the optimizer wastes iterations exploring what is effectively one solution. Nudge a single coordinate by a small amount and a node can flip in or out of the CH set, leaving the fitness surface jagged in ways the search operators were never designed for. Worse, the memory each algorithm carries—PSO's velocities and personal bests, the alpha-beta-delta hierarchy in GWO, ABC's food-source vectors—lives entirely in continuous coordinates. None of it records which nodes tend to show up in good CH configurations.

Table 1 summarizes these structural differences. Other memory-driven paradigms fare little better: ACO pheromone trails and Tabu Search lists [22] [23] track individual moves, not recurring sub-sets of promising nodes.

1.2 Gap and Contributions

In the round-based centralized setting that most metaheuristic studies adopt, the BS picks a CH set at the start of each round and broadcasts the assignment to all nodes. As discussed above and summarized in Table 1, every existing optimizer works in a continuous space and must decode its solutions into binary CH vectors—a process that severs the link between the algorithm's internal memory and the discrete structure of the problem that it is actually solving. What is missing, in particular, is a mechanism

that identifies which specific nodes keep showing up in good CH configurations during a given round and feeds that information back into the search.

Table 1. How FSS-WSN differs structurally from continuous-space CH selection methods.

Feature	PSO, GWO, ABC, SO, GJO variants	FSS-WSN (this work)
Search space	Real-valued $[0,1]^N$	Combinatorial (CH subsets built directly)
Decoding	A threshold or top- k step is always needed to obtain a binary CH vector	Skipped entirely: solutions are already discrete
What the memory captures	Continuous coordinates: velocity & personal bests (PSO), pack hierarchy (GWO), food-source positions (ABC) none tied to specific nodes	Which individual nodes recur in high-quality CH configurations
Exploitation style	Population converges toward a global-best real-valued vector	Later GRASP iterations are biased by the fixed set of promising CHs
Feasibility	Usually enforced <i>via</i> penalty terms; <i>post-hoc</i> repair added in some variants	Hard two-stage repair (coverage then connectivity), with a regularization term penalizing solutions that rely heavily on it
Per-round CPU*	1.1–2.1 s (varies across algorithms)	0.07–0.09 s

*AMD Ryzen 5 7600X (6C/12T), Python 3.12, $N=100$; see Subsection 4.2.

We fill this gap by adapting Fixed Set Search (FSS) [24] [25], a hybrid metaheuristic designed for combinatorial problems, to centralized CH selection in multi-hop WSNs where routing is part of the decision. The main contributions are as follows:

- 1) **Native combinatorial search:** FSS-WSN works directly on CH sub-sets, without any continuous-to-binary conversion. Each candidate solution is built by a coverage-driven GRASP procedure that enforces the cluster radius during construction, avoiding the selection rules commonly used in continuous-space methods.
- 2) **Two-phase in-round search with component-level memory:** Each round is split into two phases. Phase I runs a diversified GRASP to build a pool of high-quality CH sets; from this pool, per-node selection frequencies are computed and a fixed set of consistently promising CH candidates is extracted. Phase II then biases new constructions toward nodes in the fixed set, provided that they still have sufficient energy (an energy-guard check prevents the algorithm from over-exploiting depleted nodes). Because the fixed set is rebuilt at every round, it naturally tracks the shifting energy landscape.
- 3) **Deterministic repair with regularization:** Feasibility is enforced through a two-stage deterministic procedure: the first stage ensures that every sensor is covered within the cluster radius, and the second ensures that every CH can reach the sink within its transmission range. No randomness is involved, so results are fully reproducible. A regularization term in the objective penalizes solutions that lean heavily on repair, pushing the search toward configurations that are feasible by construction.
- 4) **Uniform experimental comparison:** All 11 baselines—PSO, GWO, ABC, SO, GJO, EMO-GJO, ESO-GJO, LEACH, HEED, SEP, and EEM-LEACH-ABC are reimplemented within a single Python simulator that enforces the same energy model, repair procedure, multi-hop Dijkstra routing, and metric definitions. Every experiment is replicated over 30 paired random seeds and assessed with Wilcoxon signed-rank tests under Holm correction to control family-wise error.
- 5) **Best throughput with markedly lower CPU cost:** FSS-WSN achieves the highest cumulative throughput in all four tested scenarios (homogeneous/heterogeneous energy with center/corner BS), with Holm-corrected $p < 0.05$ over every baseline. Per-round computation is 10–25 times faster than the continuous-space alternatives, a margin relevant for deployment scenarios where CH assignments must be recomputed in near-real time.

1.3 Paper Organization

The remainder of this paper is organized as follows. Section 2 introduces the system model and the

routing-coupled problem formulation. Section 3 describes the proposed FSS-WSN method in detail. Sections 4 and 5 present the evaluation protocol and discuss the experimental results. Finally, Section 6 concludes the paper.

2. SYSTEM MODEL AND PROBLEM FORMULATION

2.1 Two-graph Abstraction

Notation: In this subsection, V_t denotes the set of alive nodes and $d(u, v)$ the Euclidean distance between any two nodes u and v . Edge sets are denoted by E_t^{cov} and E_t^{tx} .

To model clustering and routing at round t , we use two distinct graphs: one for cluster membership and one for multi-hop forwarding. This separation makes the execution rules deterministic, i.e., under identical conditions, the same decisions are obtained. As in classical WSN models, coverage is governed by a fixed clustering radius R_c , while data forwarding is restricted to a cluster-head (CH) backbone. Energy dissipation follows the first-order radio model [6] [26]. At each round, CH selection, routing, and constraint satisfaction are evaluated under a fixed decision budget.

We use two thresholds:

- R_c (Coverage radius): governs cluster membership (clustering feasibility).
- r_{tx} (Transmission radius): governs one-hop connectivity for routing.

Figure 1 illustrates this distinction:

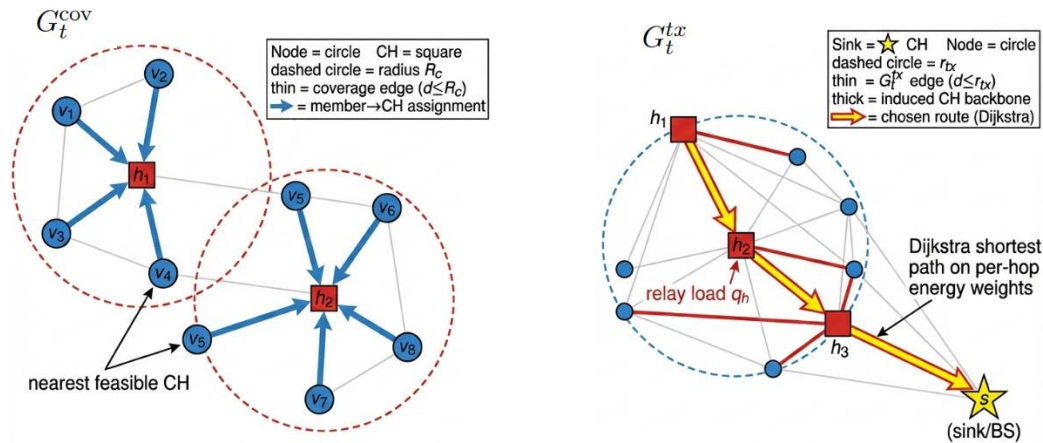


Figure 1. Graph abstractions used in this work.

In all experiments, we set $R_c = 25$ m and $r_{tx} = 50$ m (i.e., $r_{tx} = 2R_c$) unless stated otherwise. The clustering (coverage) graph is $G_t^{\text{cov}} = (V_t, E_t^{\text{cov}})$, where $(u, v) \in E_t^{\text{cov}}$ if and only if $d(u, v) \leq R_c$. The communication graph is $G_t^{\text{tx}} = (V_t, E_t^{\text{tx}})$, where $(u, v) \in E_t^{\text{tx}}$ if and only if $d(u, v) \leq r_{tx}$.

Keeping these constraints separate is important: full coverage does not guarantee balanced multi-hop forwarding. A design may still create relay bottlenecks (hotspot) and drain the nodes closest to the sink, this phenomenon is known as the 'energy hole' problem [27] especially when the sink placement is unfavorable (e.g., at the corner) [2] [9].

In round t , a candidate is a CH seed set denoted $\mathcal{H}_0 \subseteq V_t$. We enforce feasibility *via* the deterministic mapping to construct the repaired CH set $\mathcal{H}^+ = \text{Repair}_t(\mathcal{H}_0)$, and only \mathcal{H}^+ is deployed. The repair step guarantees a non-empty CH set, strict radius coverage.

$\text{Repair}_t(\cdot)$ proceeds deterministically in two stages:

First, in the coverage: if $\mathcal{H}_0 = \emptyset$, it selects one initial CH maximizing a fixed greedy score (residual-energy ratio, sink-centrality, local density; ties by smallest node index), then iteratively adds the node that covers the largest number of currently uncovered alive nodes under R_c (ties by smallest index) until (1) holds.

Second, in the connectivity (multi-hop): if no CH is within r_{tx} of the sink, it promotes the highest-energy sink-direct node as a CH, then connects remaining CHs by promoting intermediate nodes along a

deterministic BFS path on G_t^{tx} to a connected component containing a sink-direct CH (neighbors explored in increasing index order).

$$\forall v \in V_t, \exists h \in \mathcal{H}^+ \text{ s.t. } d(v, h) \leq R_c \quad (1)$$

and CH-to-sink reachability over the inter-CH backbone under r_{tx} . Each node then attaches to its nearest CH in \mathcal{H}^+ (using deterministic tie-breaking). This repair-first convention is standard in constrained optimization [19]; moreover, minimizing $|\mathcal{H}^+|$ subject to (1) reduces to the NP-complete minimum dominating-set problem [28], for which approximation algorithms with logarithmic ratios have been established [29].

2.2 Bounded Fitness Surrogate

A link (u, v) exists if and only if $d(u, v) \leq r_{tx}$, i.e., u and v can communicate in one hop. We weight each edge by the energy spent to forward one aggregated packet of length L over that hop using the first-order radio model [6] [26], which is widely adopted in the WSN literature for energy accounting. This choice makes the routing cost physically meaningful: it captures the strong dependence of transmission energy on distance and thus reflects the actual energy drain induced by multi-hop forwarding. Running Dijkstra on the graph $G^{mh}(\mathcal{H}^+)$ (the multi-hop backbone derived from the global communication graph G_t^{tx}) then yields, for each CH $h \in \mathcal{H}^+$, the minimum-energy delivery cost κ_h to the sink s .

Score normalization: All surrogate components with the symbol tilde (e.g., \tilde{C}_D, \tilde{C}_L) are normalized to lie in $[0,1]$ (lower is better). We use $\text{clip}_{[0,1]}(x)$ to truncate any value x to the interval $[0,1]$.

Bounded surrogate terms and component definitions We compute a bounded surrogate in the repaired clustering that combines: CH energy robustness C_E , member-to-CH distance \tilde{C}_D , cluster-size imbalance \tilde{C}_L , and a routing-aware term \tilde{C}_S^{mh} derived from $\{\kappa_h\}$. Specifically:

- C_E is the average CH energy depletion ratio $1 - E_{\text{res}}/E_{\text{init}}$ over $h \in \mathcal{H}^+$;
- \tilde{C}_D is the mean distance from each alive node to its selected (nearest) CH in \mathcal{H}^+ , normalized by the area diagonal;
- \tilde{C}_L is the normalized variance of the resulting cluster sizes $\{|C_h|\}_{h \in \mathcal{H}^+}$ (load imbalance);
- \tilde{C}_S^{mh} is the mean minimum-energy multi-hop delivery cost from each CH to the sink (Dijkstra on $G^{mh}(\mathcal{H}^+)$), normalized by a fixed constant.

From the Dijkstra next-hop structure, we also compute each CH relay load q_h (own plus relayed packets) and penalize relay-load dispersion *via* \tilde{C}_R , defined as the normalized variance of CH relay loads induced by the next-hop structure.

Base objective and anti-hotspot component:

We first define:

$$F_0 = w_1 C_E + w_2 \left(\frac{\tilde{C}_D + \tilde{C}_S^{mh}}{2} \right) + w_3 \tilde{C}_L, \quad w_1 + w_2 + w_3 = 1. \quad (2)$$

and then incorporate the anti-hotspot term as:

$$F_{\text{base}} = (1 - w_R) F_0 + w_R \tilde{C}_R, \quad w_R \in [0,1]. \quad (3)$$

Repair-dependence regularizer and final fitness: While repair mechanisms guarantee feasibility, naive repair can bias metaheuristics toward 'easy-to-repair' regions of the search space—a phenomenon discussed in constrained optimization literature [19] [30]. To address this bias, we penalize candidates that rely heavily on repair, we measure the fraction of CHs added by $\text{Repair}_t(\cdot)$, where $|\cdot|$ denotes set cardinality and " \setminus " denotes set difference:

$$P(\mathcal{H}_0) = \frac{|\mathcal{H}^+ \setminus \mathcal{H}_0|}{N_t} \in [0,1]. \quad (4)$$

We measure the per-round regularized objective:

$$\text{Fitness}_t(\mathcal{H}_0) = \text{clip}_{[0,1]}(F_{\text{base}}) + \lambda P(\mathcal{H}_0), \quad \lambda > 0. \quad (5)$$

The parameter $\lambda > 0$ weights the repair-dependence penalty and discourages seeds that rely on $\text{Repair}_t(\cdot)$. The BS keeps the candidate with minimum fitness, applies $\text{Repair}_t(\cdot)$, and deploys the repaired CH set.

3. PROPOSED FSS-WSN ALGORITHM

3.1 Algorithm Description

At the beginning of each round t , the Base Station (BS) computes a deployable clustered configuration using **FSS-WSN**. Let $V_t = \{i \in V: E_i^{\text{res}}(t) > 0\}$ denote the alive nodes at round start; $E_i^{\text{res}}(t)$ denotes residual energy (should not be confused with the edge sets E_t^{cov} , E_t^{tx} , or E^{mh}). The decision variable is a *seed* set of prospective cluster heads (CHs), $\mathcal{H} \subseteq V_t$; the configuration ultimately deployed in the network is obtained by selecting the seed that minimizes the regularized surrogate fitness (Eq. (5)).

All seeds are evaluated under a fixed deployment convention. Given \mathcal{H} , we first apply the deterministic feasibility mapping $\mathcal{H}^+ = \text{Repair}_t(\mathcal{H})$ (Eq. (4)). This mapping enforces three hard requirements: $\mathcal{H}^+ \neq \emptyset$; strict R_c coverage for member attachment; and reachability of each CH to the sink on the inter-CH backbone under r_{tx} .

Once \mathcal{H}^+ is obtained, every alive node attaches to its nearest CH, and multi-hop forwarding costs are computed by running Dijkstra on the induced CH backbone with energy-based hop weights. The deployed round output is therefore $\mathcal{H}_t^{*+} = \text{Repair}_t(\mathcal{H}_t^*)$, where \mathcal{H}_t^* is the best seed under $\text{Fitness}_t(\cdot)$.

FSS-WSN follows a two-phase routine. Phase I populates a small elite pool using GRASP-style randomized greedy constructions (optionally followed by bounded swaps). Phase II extracts from that pool a compact set of recurrent, energy-safe CH nodes and uses this set as a warm start for additional constructions [24] [25] [31] [32].

Algorithm 1 summarizes the round-level procedure. (*Unless stated otherwise, swap refinement is disabled in experiments $L_{\text{max}} = 0$*).

The algorithm follows a two-pass pattern:

- **Phase I (diversified GRASP seeds):** Starting from \emptyset , we build seed CH sets using GRASP until all alive nodes are covered within R_c or the construction reaches K_{cap} (a practical cap to control effort and avoid oversized CH sets when coverage is hard). K_{cap} is used only during construction and should not be confused with $K_{\text{max}}^{\text{route}}$, which is a deterministic per-round bound used solely to normalize the routing-cost term.

At each step, GRASP ranks candidates by a greedy key, forms a restricted candidate list controlled by γ , samples one CH from that list, and continues [31] [32]. If enabled, Local-Search-Swap refines the seed by swapping a CH with a nearby non-CH node using a first-improvement rule, up to L_{max} accepted improvements while scanning at most k_{nn} neighbors per move. Each seed is evaluated through the deterministic pipeline (repair, assignment, routing) and may enter the elite pool $\mathcal{P}_{\text{elite}}$, which keeps at most B seeds ranked by $\text{Fitness}_t(\cdot)$; near-duplicates can be discarded using a Jaccard similarity threshold δ .

- **Phase II (fixed-set biased constructions):** From $\mathcal{P}_{\text{elite}}$, we learn a fixed set \mathcal{F}_t by counting how often each node appears in elite seeds. We retain nodes above frequency threshold τ and filter out low-energy nodes using threshold θ . We then rerun the same GRASP construction, starting from $\mathcal{H} \leftarrow \mathcal{F}_t$ when $\mathcal{F}_t \neq \emptyset$ (otherwise from \emptyset), with the same optional swap refinement and elite updates as in Phase I.

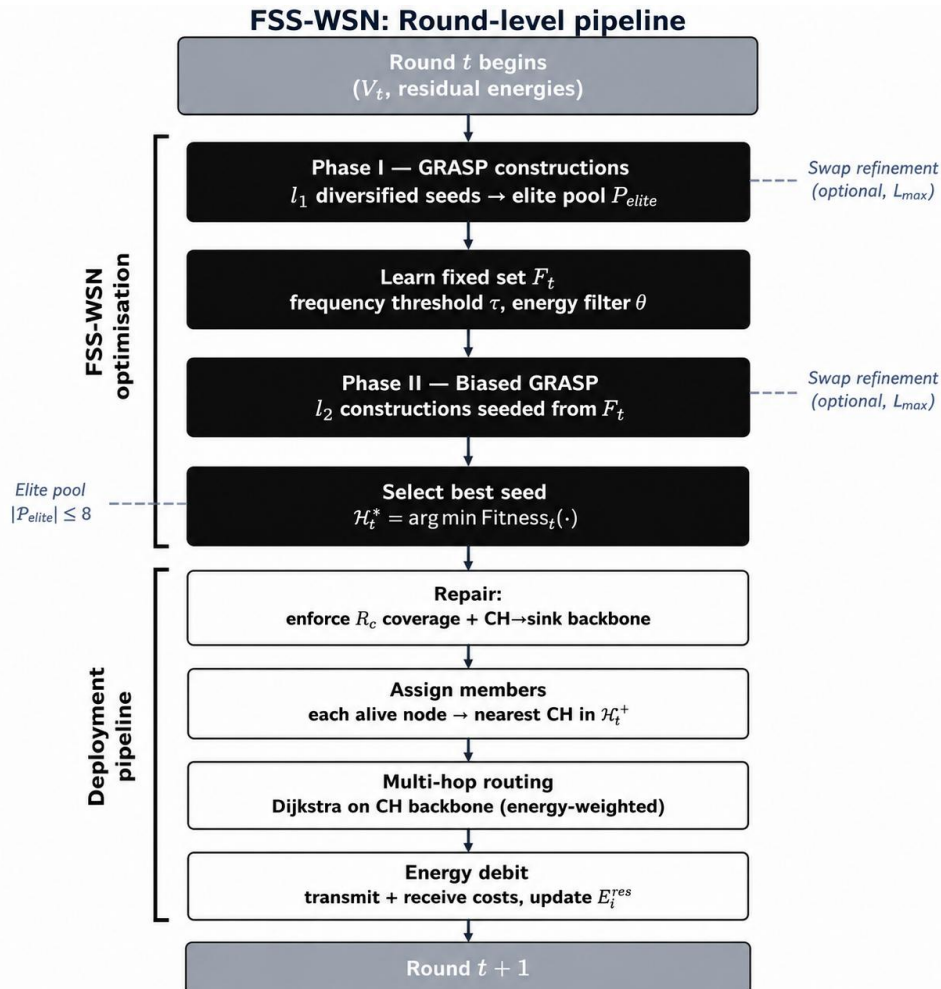
Finally, we select the best elite seed \mathcal{H}_t^* (minimum $\text{Fitness}_t(\cdot)$) and deploy the repaired configuration $\mathcal{H}_t^{*+} = \text{Repair}_t(\mathcal{H}_t^*)$. Figure 2 presents an FSS-WSN round-level pipeline.

Algorithm 1 FSS-WSN (round t at the BS)

```

1: Input: Alive nodes  $V_t$  with their energies  $E(t)$ 
2: Input: Parameters  $(B, \gamma, K_{\text{cap}}, L_{\text{max}}, k_{\text{nn}}, \tau, \theta, \delta, I_1, I_2)$ 
3: Output: Deployed CH set  $\mathcal{H}_t^{*+}$ 
4:  $\mathcal{P}_{\text{elite}} \leftarrow \emptyset$ 
5: for  $iter = 1$  to  $I_1$  do
6:    $\mathcal{H} \leftarrow \text{GRASPCONSTRUCT}(V_t, E(t), \gamma, K_{\text{cap}})$  ▷ Phase I: populate the elite pool
7:   if  $L_{\text{max}} > 0$  then
8:      $\mathcal{H} \leftarrow \text{LOCALSEARCHSWAP}(\mathcal{H}, L_{\text{max}}, k_{\text{nn}})$ 
9:   end if
10:   $\text{UPDATEELITE}(\mathcal{P}_{\text{elite}}, \mathcal{H}, B, \delta)$ 
11: end for
12:  $\mathcal{F}_t \leftarrow \text{LEARNFIXEDSET}(\mathcal{P}_{\text{elite}}, \tau, \theta)$ 
13: for  $iter = 1$  to  $I_2$  do
14:    $\mathcal{H} \leftarrow \mathcal{F}_t$  if  $\mathcal{F}_t \neq \emptyset$  else  $\emptyset$  ▷ Extract recurrent components
15:    $\mathcal{H} \leftarrow \text{GRASPCOMPLETE}(\mathcal{H}, V_t, E(t), \gamma, K_{\text{cap}})$  ▷ Phase II: construct with  $\mathcal{F}_t$  as a bias
16:   if  $L_{\text{max}} > 0$  then
17:      $\mathcal{H} \leftarrow \text{LOCALSEARCHSWAP}(\mathcal{H}, L_{\text{max}}, k_{\text{nn}})$ 
18:   end if
19:    $\text{UPDATEELITE}(\mathcal{P}_{\text{elite}}, \mathcal{H}, B, \delta)$ 
20: end for
21:  $\mathcal{H}_t^* \leftarrow \arg \min_{\mathcal{H} \in \mathcal{P}_{\text{elite}}} \text{Fitness}_t(\mathcal{H})$ 
22: return  $\text{Repair}_t(\mathcal{H}_t^*)$ 

```



3.2 Computational Complexity

Let $N = |V_t|$ and $K = |\mathcal{H}^+|$. One fitness evaluation is dominated by nearest-CH assignment, $O(NK)$, and shortest-path computation on the CH backbone (Dijkstra), $O((K + |E^{mh}|)\log K)$, hence:

$$T_{\text{eval}} = O(NK + (K + |E^{mh}|)\log K). \quad (6)$$

Across a round, we perform $I_1 + I_2$ constructions. Because construction length is capped by K_{cap} and swap refinement (when used) is bounded by L_{max} accepted moves (each scanning k_{nn} candidates), the number of evaluations per construction is bounded by a constant $C_{\text{LS}}(K_{\text{cap}}, L_{\text{max}}, k_{\text{nn}})$. Therefore,

$$T_{\text{round}} = O((I_1 + I_2) C_{\text{LS}} T_{\text{eval}}), \quad (7)$$

up to lower-order costs for elite maintenance and fixed-set learning. In the recommended configuration, $L_{\text{max}} = 0$, so swap refinement is disabled and C_{LS} simplifies accordingly.

4. EXPERIMENTAL SETUP AND CONFIGURATION

All methods are evaluated within the same simulator core and under the same execution rules: strict clustering feasibility under R_c , routing on the CH-induced backbone, and the deterministic feasibility mapping $\text{Repair}_t(\cdot)$. Each run is simulated for up to $R_{\text{max}} = 2500$ rounds, or until no node remains alive.

4.1 Compared Methods and Literature Positioning

FSS-WSN is compared against 11 baselines under identical conditions:

- **Metaheuristic optimizers (centralized):** PSO [12], GWO [13], ABC [14], SO [15], GJO [16], EMO-GJO [20], ESO-GJO [21].
- **Distributed protocols:** LEACH [6], HEED [7], SEP [8], EEM-LEACH-ABC [10].

Centralized optimizers (including FSS-WSN) run at the BS, which knows node positions *a priori* and collects one residual-energy scalar ($l_{\text{ctrl}} = 200$ bits) per alive node per round. The resulting control-plane overhead is ≈ 1.25 mJ/round (centre BS) to ≈ 1.98 mJ/round (corner BS), i.e. $< 4\%$ to $< 7\%$ of the per-round energy budget (≈ 31.7 mJ), borne identically by all eight centralized methods.

Table 2 confirms that all methods share the standard LEACH radio constants; EEM-LEACH-ABC originally uses different parameters and was re-implemented under ours.

Table 2. Simulation parameters vs. literature (\equiv : identical to the previous).

Method	Field (m)	N	E_0 (J)	E_{elec}	l (bits)	BS position
LEACH	100^2	100	0.5	50	4000	(50,175)
HEED	100^2	100	0.5	\equiv	\equiv	center
SEP	100^2	100	0.5	\equiv	\equiv	(50,50)
PSO-CH	100^2	100	0.5	\equiv	\equiv	center
GWO-CH	100^2	100	0.5	\equiv	\equiv	center
ABC-CH	100^2	100	0.5	\equiv	\equiv	center
SO-CH	100^2	100	0.5	\equiv	\equiv	center
GJO-CH	100^2	100	0.5	\equiv	\equiv	center
EMO-GJO	100^2	100	0.5	\equiv	\equiv	center
ESO-GJO	100^2	100	0.5	\equiv	\equiv	center
EEM-LEACH-ABC	250^2	150	0.5	55	4400	(100,250)
This work	100^2	100	0.5	50	4000	center & corner

Common constants shared by all rows except EEM-ABC: $\epsilon_{fs} = 10$ pJ/bit/m², $\epsilon_{mp} = 0.0013$ pJ/bit/m⁴, $E_{DA} = 5$ nJ/bit, $d_0 = 87$ m, $l_{\text{ctrl}} = 200$ bits.

4.2 Simulation Platform and Energy Model

The evaluation platform is a custom Python 3.12.3 simulator (NumPy 1.26.4) implementing the first-order radio energy model. All twelve methods execute within the same simulation core—identical energy accounting, Dijkstra-based multi-hop routing, and packet-level forwarding—so that differences in delivered utility reflect differences in the CH-selection strategy only.

The simulator operates at the network layer; MAC-layer contention and PHY-layer impairments are abstracted out, in line with all the referenced baselines. Experiments run on Windows 10 Pro, AMD Ryzen 5 7600X (6C/12T), 32 GB RAM. Source code: <https://github.com/RaoufOuanis/wsn-fss-simulation>.

Energy model: Fixed packet sizes: $l = 4000$ bits, $l_{\text{ctrl}} = 200$ bits; $E_{\text{elec}} = 50$ nJ/bit, $\epsilon_{fs} = 10$ pJ/bit/m², $\epsilon_{mp} = 0.0013$ pJ/bit/m⁴, $d_0 = 87$ m, $E_{DA} = 5$ nJ/bit. Member-to-CH links are single-hop under R_c ; backbone forwarding uses Dijkstra under r_{tx} . A hop the endpoint of which lacks energy is silently dropped.

Topology: $N = 100$ nodes, uniform 100×100 m field, $R_c = 25$ m. Two energy profiles and two BS positions (center/corner) yield the four configurations of Table 3.

Table 3. Network scenarios and topological parameters.

Parameter	S1 (Homogeneous)	S2 (Heterogeneous)
Field ($L \times L$)	100 × 100 m	100 × 100 m
N	100	100
Energy (J)	$E_0 = 0.5$	$E_0 = 0.5$ $E_{\text{adv}} = 1.0$
Percentage of advanced nodes (m)	–	20%
BS position	center / corner	center / corner
R_{max}	2500	2500

4.3 Algorithm Configuration

All iterative optimizers (FSS/PSO/GWO/ABC/SO/GJO/EMO-GJO/ESO-GJO) use the same per-round iteration budget, $n_{\text{iter}} = 60$. For population-based methods, we use `pop_size = 30` as a common setting. For FSS-WSN, the budget is split as implemented: $I_1 = \text{round}(0.60 n_{\text{iter}})$ and $I_2 = n_{\text{iter}} - I_1$. All hyper-parameters are summarized in Table 4.

Table 4. Algorithm configuration and hyper-parameters.

Algorithm	Parameter	Value
FSS-WSN	Iterations ($I_1 + I_2$)	36 + 24 = 60
	Elite pool size (B)	10
	RCL parameter (γ)	0.2
	Fixed-set thresholds (τ, θ)	0.6, 0.3
	Fitness weights ($w_1, w_2, w_3, w_R, \lambda$)	0.4, 0.4, 0.2, 0.05, 1.0
	Greedy-score weights ($\alpha_1, \alpha_2, \alpha_3$)	0.5, 0.3, 0.2
	Construction cap (K_{cap})*	20
	Swap refinement (L_{max}, k_{nn})	disabled (0, 0)
<i>All swarm methods below: $n_{\text{iter}} = 60$ and $\text{pop} = 30^*$, $k \in [1, 20]$</i>		
PSO-WSN	Inertia ω ; c_1, c_2	0.7; 1.5, 1.5
GWO-WSN	Control a	$2 \rightarrow 0$ (Linear decay)
ABC-WSN	Limit; pop	10; 20**
SO, GJO, EMO-GJO, ESO-GJO	Default settings from original papers	
LEACH	p_{opt}	0.05
HEED	$p_{\text{init}}, c_{\text{min}}, n_{\text{iter}}$	0.05, 0.02, 3
SEP	$p_{\text{opt}}, m, E_0, E_{\text{adv}}$	0.05, 0.2, 0.5, 1.0
EEM-LEACH-ABC	c_r, μ, epochs	0.10, 0.70, 5

* K_{cap} is a GRASP construction cap, whereas $K_{\text{max}}^{\text{route}}$ is a deterministic, per-round normalization bound used only inside the routing-cost term and is not a tunable hyperparameter. ** ABC uses $\text{pop} = 20$ food sources ($2 \times$ evaluations/iteration).

4.4 Evaluation Protocol

Every method produces a seed CH set \mathcal{H}_0 , which may be empty or infeasible. The deterministic mapping $\mathcal{H}^+ = \text{Repair}_t(\mathcal{H}_0)$ (Eq. (4)) is applied before any routing or energy debiting, enforcing a non-empty CH set, strict R_c coverage, and CH-to-sink reachability under r_{tx} . Members attach to their nearest CH (smallest-index tie-breaking); routing follows Dijkstra on the induced backbone (Sub-section 2.2).

Protocol baselines (LEACH, HEED, SEP, EEM-LEACH-ABC) invoke repair once per round. Iterative optimizers invoke it at every fitness evaluation (Eq. (5)). The average number of CHs added by repair is reported for transparency.

Monte-Carlo protocol: Each configuration is evaluated over 30 paired seeds. A given seed fixes the node deployment—and, in S2, the advanced-node identities—across all methods; per-round randomness derives deterministically from the seed and round index. Results: $\mu \pm \sigma$; paired Wilcoxon signed-rank tests ($\alpha = 0.05$) with Holm correction.

Reported metrics: Wall-clock CPU time of the CH-selection routine (single machine); NFE for iterative methods only. Lifetime markers: FND, HND, LND, R_{last} . Primary endpoint: cumulative delivered throughput.

Ablation and sensitivity: Ablation results (\pm Phase II, \pm regularizer) and one-at-a-time sensitivity sweeps over $(\gamma, \tau, \theta, n_{\text{iter}})$ are reported in Sub-section 5.4 following standard metaheuristic validation practice [33] [34] [35] [36].

5. EXPERIMENTAL RESULTS

We evaluate FSS-WSN under the protocol described in Section 4: common simulator core, first-order energy accounting, strict clustering under R_c , multi-hop CH to sink forwarding, and deterministic hard-feasibility enforcement *via* $\text{Repair}_t(\cdot)$. All results are reported as mean $\mu \pm \sigma$ over 30 paired Monte-Carlo seeds. Paired Wilcoxon signed-rank tests are computed on matched seeds. Since multiple baselines are compared to FSS-WSN within a configuration, the Holm correction is applied within that configuration.

5.1 Protocol Recap and Reported Endpoints

All methods run for up to $R_{\text{max}} = 2500$ rounds in a centralized and round-based mode. For each seed, the same node deployment (and, in S2, the same advanced-node identities) is reused across methods. All methods -including classical protocols- are evaluated under the same convention of feasibility: the runner applies $\text{Repair}_t(\cdot)$ before routing and energy accounting, enforcing strict R_c coverage and CH \rightarrow sink reachability under r_{tx} . We report:

- **Primary endpoint:** is the cumulative useful delivered reports, denoted **Throughput**.
- **Service duration:** R_{last} and FND/HND/LND.
- **Traffic proxy:** CH \rightarrow sink (pkts).
- **Compute cost:** per-round CPU time as the main indicator; NFE only as an implementation-dependent proxy.

5.2 Primary Endpoint: Delivered Utility across Configurations

Table 5 summarizes **Throughput (reports)** across the four configurations (S1/S2 with centered/corner BS). The pattern is stable: **FSS-WSN achieves the highest delivered utility in all configurations**. A visual summary against the best baseline in each scenario is provided in Figure 3. Concretely:

- **S1 (homogeneous):** $119,533 \pm 1,190$ (center) and $97,064 \pm 1,610$ (corner), i.e., +4.57% vs. HEED (center) and +3.16% vs. GWO (corner).
- **S2 (heterogeneous):** $140,131 \pm 5,210$ (center) and $116,025 \pm 3,829$ (corner), i.e., +3.68% vs. HEED (center) and +2.40% vs. GWO (corner).

Positioning vs. SO/GJO-family variants (EMO-GJO, ESO-GJO): Under the unified evaluation protocol used in this study, FSS-WSN outperformed both EMO-GJO and ESO-GJO in all four configurations. In terms of delivered reports, FSS-WSN achieved gains of 8.17% in S1-center, 3.51%

in S1-corner, 5.88% in S2-center, and 5.08% in S2-corner relative to the best SO/GJO-family variant. These results suggest that part of the advantage reported for EMO-GJO and ESO-GJO in prior studies may depend on differences in feasibility handling and routing assumptions across simulation frameworks.

Table 5. Primary endpoint (delivered reports) across configurations (mean \pm std, 30 seeds).

Method	S1-center	S1-corner	S2-center	S2-corner
FSS-WSN	119,533 \pm 1,190	97,064 \pm 1,610	140,131 \pm 5,210	116,025 \pm 3,829
ESO-GJO	110,389 \pm 1,581	93,004 \pm 1,576	132,312 \pm 4,954	110,412 \pm 3,332
EMO-GJO	110,377 \pm 1,595	93,774 \pm 1,591	132,352 \pm 4,988	108,218 \pm 3,670
GJO	110,388 \pm 1,608	91,525 \pm 1,559	132,313 \pm 5,015	108,168 \pm 3,205
SO	110,500 \pm 1,603	91,398 \pm 1,511	132,294 \pm 5,010	107,508 \pm 3,026
PSO	110,433 \pm 1,626	93,484 \pm 1,558	132,019 \pm 4,967	109,888 \pm 3,431
GWO	110,417 \pm 1,596	94,092 \pm 1,335	131,991 \pm 4,973	113,309 \pm 3,597
ABC	110,372 \pm 1,622	92,623 \pm 1,543	132,423 \pm 4,988	110,144 \pm 3,504
HEED	114,306 \pm 1,734	87,358 \pm 1,366	135,153 \pm 5,676	107,175 \pm 3,488
LEACH	110,276 \pm 1,624	87,714 \pm 1,546	131,205 \pm 4,976	105,366 \pm 2,610
SEP	107,190 \pm 1,289	91,247 \pm 1,487	131,550 \pm 4,453	107,612 \pm 3,422
EEM-LEACH-ABC	109,658 \pm 895	90,471 \pm 1,603	131,421 \pm 4,605	109,348 \pm 3,999

All 44 FSS-WSN vs. baseline differences significant (paired Wilcoxon, $p_{\text{Holm}} < 0.001$, $r_b = 1.0$).

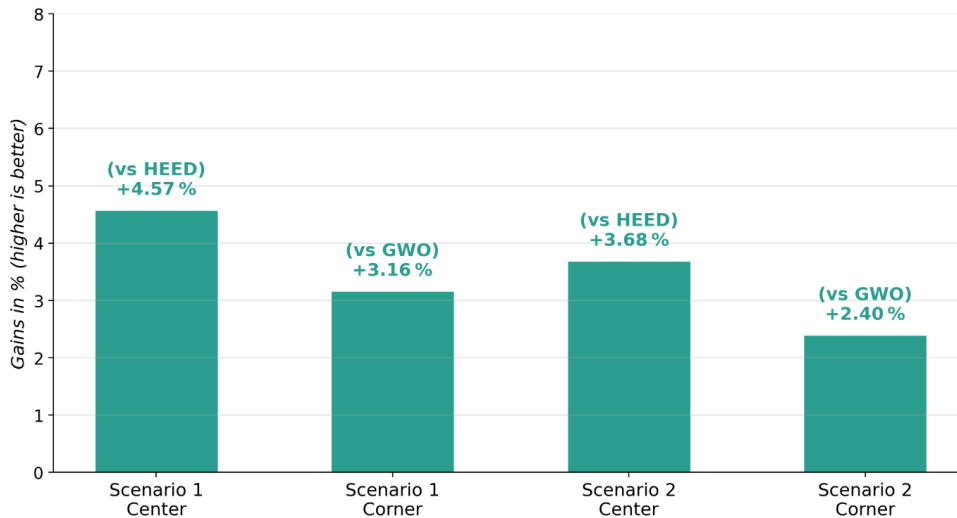


Figure 3. Primary endpoint summary (delivered reports): FSS-WSN compared to best baseline.

Positioning of EEM-LEACH-ABC: We added EEM-LEACH-ABC, as it is the most recent protocol in this problem family. Under our unified evaluation protocol, it ranked in the lower half across all configurations (11th in S1-center and S2-center, 10th in S1-corner, and 6th at best in S2-corner). Its throughput remained 6.11% to 9.00% below that of FSS-WSN, suggesting that ABC-based parameter tuning within a LEACH-style framework is not sufficient to match direct CH-set optimization in the present setting.

5.3 Service Duration, Robustness, and Computational Cost

Advantage persists under routing-coupled multi-hop: Two observations support the view that the throughput advantage is not merely an artifact of the experimental setup:

(i) *Lower repair dependence:* $\text{Repair}_t(\cdot)$ silently adds CHs or relays to force feasibility, so any method that relies heavily on repair is partly “cheating”—its throughput is artificially propped up by nodes that it did not choose. FSS-WSN barely triggers repair at all: 0.27 added CH/round in S1-center, as low as 0.004 in S1-corner, and similar numbers in S2. The EMO/ESO family, by contrast, needs ≈ 2.6 –3.5

insertions per round. This aligns perfectly with the regularizer’s goal: the optimizer has already learned to produce near-feasible solutions on its own.

(ii) *Lower relay hotspots in corner geometry*: Corner placement stretches multi-hop paths, and a few relay nodes inevitably become bottlenecks. What we found is that FSS-WSN nearly halves the worst-case relay load: mh_q_max drops from ≈ 22.4 to 12.83 (S1-corner) and 14.01 (S2-corner). Concretely, fewer packets pile up at any single relay, so fewer get dropped along the chain.

Service duration and tail behavior: Table 6 lists R_{last} for all methods. One thing that we noticed right away is that a longer R_{last} can be misleading: several baselines linger for dozens of extra rounds while barely delivering anything. Figure 4. Cumulative delivered reports over rounds (mean, 30 seeds). FSS-WSN plateaus highest in all scenarios despite shorter operational lifetime in some cases captures this well—FSS-WSN’s curve climbs faster, finishes higher, and then the network dies. HEED and GWO approach FSS-WSN’s final level in certain setups, but do not reach it; LEACH and SEP lag by a wider margin.

Table 6. Operational lifetime R_{last} across configurations (mean \pm std, 30 paired seeds).

Method	S1-center	S1-corner	S2-center	S2-corner
FSS-WSN	1575 \pm 56	1400 \pm 78	2349 \pm 51	1560 \pm 62
ESO-GJO	1674 \pm 50 [‡]	1620 \pm 68 [‡]	2419 \pm 111 [‡]	1781 \pm 204 [‡]
EMO-GJO	1684 \pm 54 [†]	1590 \pm 65 [‡]	2440 \pm 105 [‡]	1772 \pm 219 [‡]
GJO	1685 \pm 52 [*]	1612 \pm 65 [‡]	2417 \pm 100 [‡]	1773 \pm 219 [‡]
SO	1684 \pm 48 [†]	1598 \pm 65 [‡]	2440 \pm 105 [‡]	1773 \pm 219 [‡]
PSO	1692 \pm 54 [‡]	1612 \pm 66 [‡]	2438 \pm 104 ^{ns}	1773 \pm 219 [‡]
GWO	1690 \pm 52 [†]	1490 \pm 61 [‡]	2438 \pm 102 ^{ns}	1773 \pm 219 [‡]
ABC	1692 \pm 53 ^{ns}	1612 \pm 67 [‡]	2440 \pm 105 [†]	1773 \pm 219 [‡]
HEED	1812 \pm 59 [‡]	1529 \pm 57 ^{ns}	2294 \pm 111 [*]	1605 \pm 218 [‡]
LEACH	1724 \pm 48 [‡]	1524 \pm 44 [‡]	2419 \pm 112 [†]	1802 \pm 240 [‡]
SEP	1513 \pm 70 [‡]	1603 \pm 58 [‡]	2440 \pm 105 [†]	1773 \pm 219 [‡]
EEM-LEACH-ABC	1797 \pm 120 [†]	1527 \pm 122 [‡]	2499 \pm 0 [†]	2358 \pm 195 [‡]

Significance markers are based on paired Wilcoxon signed-rank tests (Holm-corrected): [‡] $p_{Holm} < 0.001$; [†] $p_{Holm} < 0.01$; ^{*} $p_{Holm} < 0.05$; ^{ns} not significant. A positive rrb indicates that FSS-WSN has a longer R_{last} .

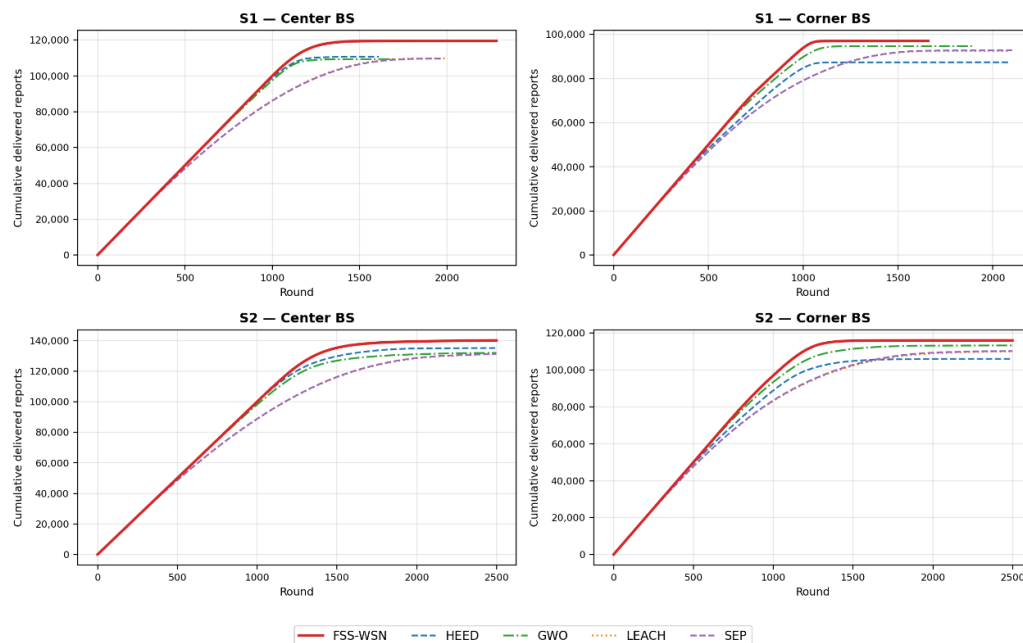


Figure 4. Cumulative delivered reports over rounds (mean, 30 seeds). FSS-WSN plateaus highest in all scenarios despite shorter operational lifetime in some cases.

Wilcoxon tests back this up: in corner configurations, FSS-WSN terminates significantly earlier than 10 of 11 baselines ($p_{\text{Holm}} < 0.001$). HEED is the sole exception—its aggressive clustering sometimes drains the network even faster than FSS-WSN does. The pattern is consistent: FSS-WSN pushes harder per round, finishes sooner, but accumulates more reports overall.

Practical significance of the throughput gain: The observed throughput improvement (2.4–4.6%) corresponds to approximately 2,700–5,200 additional reports delivered to the BS, which is equivalent to about 27–52 rounds of full service in a 100-node network. When combined with the 17–22× reduction in per-round CPU time relative to the fastest swarm baseline (Table 7), these results indicate that FSS-WSN offers a particularly favorable throughput–computation trade-off in the evaluated settings (Figure 5). This interpretation is supported by the statistical analysis, as all 44 pairwise throughput comparisons against FSS-WSN were significant ($r_{\text{tb}} = 1.0$, $p_{\text{Holm}} < 0.001$).

Computational cost: Table 7 and Figure 5 complete the picture: FSS-WSN is the cheapest iterative optimizer and the highest in delivered utility, in every configuration. The 17–22× CP advantage over the fastest swarm baseline stems in part from fitness memoization; the reported NFE counts include cache hits, so effective unique evaluations are substantially fewer. All 28 pairwise CPU-time comparisons are significant ($p_{\text{Holm}} < 0.001$, $r_{\text{tb}} = 1.0$). Protocol baselines (LEACH, HEED, SEP, EEM-LEACH-ABC) are lightweight by design and are not reported.

Table 7. Per-round computational overhead for iterative optimizers (CPU time in seconds).

Scenario	Method	CPU (s)	NFE
S1-center	FSS-WSN	0.085	600 ± 91
	SO	1.440 ± 0.047	1830 ± 0
	GJO	1.448 ± 0.047	1830 ± 0
	EMO–GJO	1.451 ± 0.053	1800 ± 0
	ESO–GJO	1.428 ± 0.077	1800 ± 0
	PSO	1.431 ± 0.077	1800 ± 0
	GWO	1.425 ± 0.067	1800 ± 0
	ABC	2.068 ± 0.087	2528 ± 4
S1-corner	FSS-WSN	0.076	754 ± 107
	SO	1.357 ± 0.032	1830 ± 0
	GJO	1.362 ± 0.032	1830 ± 0
	EMO–GJO	1.370 ± 0.026	1800 ± 0
	ESO–GJO	1.450 ± 0.033	1800 ± 0
	PSO	1.412 ± 0.039	1800 ± 0
	GWO	1.802 ± 0.143	1800 ± 0
	ABC	2.198 ± 0.261	2527 ± 5
S2-center	FSS-WSN	0.0885	711 ± 89
	SO	1.982 ± 0.072	1830 ± 0
	GJO	2.159 ± 0.071	1830 ± 0
	EMO–GJO	3.029 ± 0.074	1800 ± 0
	ESO–GJO	2.981 ± 0.052	1800 ± 0
	PSO	2.050 ± 0.056	1800 ± 0
	GWO	2.088 ± 0.070	1800 ± 0
	ABC	3.152 ± 0.091	2529 ± 5
S2-corner	FSS-WSN	0.067	860 ± 124
	SO	1.112 ± 0.180	1830 ± 0
	GJO	1.190 ± 0.267	1830 ± 0
	EMO–GJO	1.103 ± 0.209	1800 ± 0
	ESO–GJO	1.323 ± 0.276	1800 ± 0
	PSO	1.370 ± 0.271	1800 ± 0
	GWO	1.215 ± 0.282	1800 ± 0
	ABC	1.695 ± 0.367	2531 ± 6

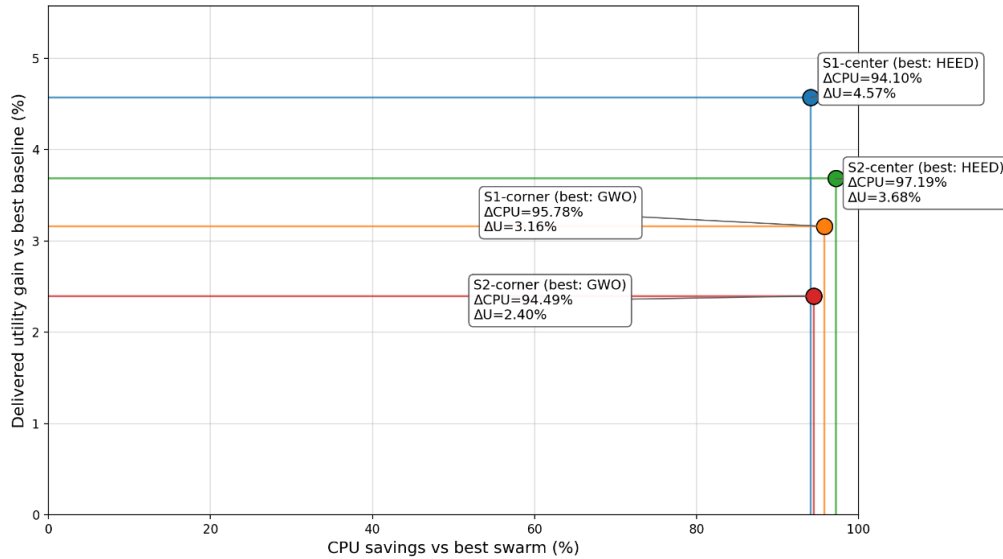


Figure 5. CPU–utility trade-off of FSS-WSN vs. the strongest iterative competitor per scenario.

5.4 Ablation Study

To quantify the contribution of the two main components added on top of Phase I GRASP, we run 30-seed ablation experiments on the four test settings. Two algorithmic variants are considered: (1) without Phase II, which disables fixed-set intensification and relies on Phase I alone; and (2) without regularizer, which sets $\lambda = 0$ and removes the repair-dependence penalty from the objective. All other parameters remain as in Table 4, and the same paired seeds are used throughout. Table 8 reports relative changes with respect to the full FSS-WSN configuration. For reference, on S1-center (the setting used for detailed reporting), the absolute throughput values are: $119,533 \pm 1,190$ (full FSS-WSN), $118,534 \pm 1,190$ (without Phase II), and $118,859 \pm 1,198$ (without regularizer).

Table 8. Ablation study: relative changes with respect to full FSS-WSN (30 paired seeds).

	S1-center	S1-corner	S2-center	S2-corner
Δ Throughput (%)				
without Phase II	-0.84^{ns}	-1.18^{\ddagger}	-1.01^{ns}	-1.07^*
without regularizer	-0.56^{\ddagger}	-0.80^{ns}	-0.20^{\ddagger}	-1.02^{ns}
Δ Per-round CPU time (%)				
without Phase II	-36	-42	-33	-41
without regularizer	-5	+1	-1	-3

Table 8 reports throughput changes in % and CPU changes as relative per-round time; statistical significance is assessed *via* paired Wilcoxon tests ($\ddagger p < 0.001$; $* p < 0.05$; $^{\text{ns}}$ not significant). Removing Phase II consistently lowers mean throughput across the four test settings, although the effect remains modest in magnitude. This indicates that, at $N = 100$, the marginal contribution of Phase II over a Phase-I-only variant is small, but consistent. We therefore keep Phase II in the reference configuration: it delivers the best mean throughput in all cases and provides a principled intensification mechanism, at the cost of roughly one-third of the per-round CPU time (33%–42% reduction when disabled). Removing the repair-dependence regularizer also leads to small, but consistently negative, throughput changes, with negligible runtime impact ($|\Delta| \leq 5\%$). We retain it, because it biases the search away from solutions the quality of which depends heavily on repair; in that sense, the regularizer acts primarily as a robustness-oriented term rather than as a direct throughput booster.

One-at-a-time sensitivity sweeps: Figure 6 reports the throughput variation when each of the four main hyper-parameters (γ , τ , θ , n_{iter}) is varied individually while the others are kept at the defaults of Table 4. Each bar shows the percentage throughput change relative to the default value (marked with $*$), averaged over 5 paired seeds on S1-center (2 500 rounds). All variations remain below $\pm 1\%$ in throughput (τ , θ , and n_{iter} stay within $\pm 0.05\%$; only γ reaches $\approx 0.9\%$), confirming that the selected

configuration is not a local optimum artifact and that FSS-WSN performance is robust to moderate hyperparameter perturbations.

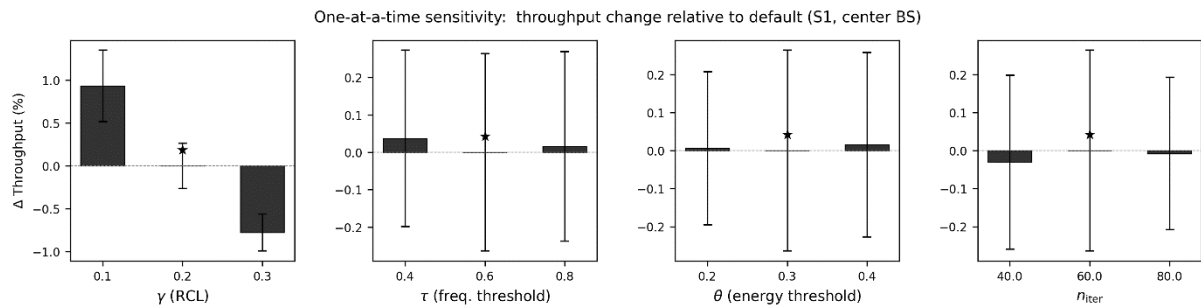


Figure 6. One-at-a-time sensitivity: throughput change relative to default (S1, center BS, 5 seeds, 2500 rounds). All deviations are below 1%; the \star marks the retained configuration.

5.5 Scalability $N = 200$

To assess how FSS-WSN scales beyond the 100-node configuration, we ran experiments with $N = 200$ homogeneous nodes in the same 100×100 m area (S1, center BS, $E_0 = 0.5$ J, 10 seeds, $R_{max} = 2000$). Table 9 shows that FSS-WSN maintains the highest throughput (238,707, +8.7% vs. HEED) and the most stable FND (-2.8% degradation vs. -15% to -20% for LEACH/SEP). Per-round CPU time scales from ~ 0.085 s ($N = 100$) to ~ 0.20 s ($N = 200$), consistent with the complexity of Eq. (7), all FSS-WSN vs. baseline differences are significant (paired Wilcoxon, $p = 0.002$, $r_{rb} = 1.0$).

Table 9. Scalability: $N = 100$ (30 seeds) vs. $N = 200$ (10 seeds), S1 center BS.

	Throughput		FND (rounds)	
	$N = 100$	$N = 200$	$N = 100$	$N = 200$
FSS-WSN	119,533	238,707	975	947 (-2.8%)
HEED	114,306	219,706	812	805 (-0.9%)
LEACH	110,276	216,869	201	172 (-14.7%)
SEP	107,190	216,413	200	161 (-19.7%)

5.6 Scope and Threats to Validity

These results concern a configuration that assumes a centralized round-based decision, with multi-hop CH to sink routing, under a strict deterministic feasibility convention for all methods. Interpretation is delimited by:

- **Routing abstraction and MAC/PHY gap:** The simulator operates at the network layer: Dijkstra provides optimal multi-hop relay paths, but MAC-layer contention (e.g., CSMA/CA back-off, duty cycling) and PHY-layer effects (fading, interference, packet-error rate) are abstracted out. This is the standard evaluation framework for all twelve compared methods.

Three arguments bound the impact of this abstraction on the *relative* comparison (FSS-WSN vs. baselines): (i) MAC losses are topology-dependent and round-dependent, but all methods produce comparable cluster counts (6–12 CHs/round) and comparable traffic patterns, so that the expected MAC loss ratio is similar across methods; (ii) the FSS-WSN advantage stems from *better CH placement* (lower relay hotspots, lower repair dependence), which reduces spatial congestion—a feature that would *improve* rather than degrade under a contention-based MAC; (iii) prior comparative studies using full-stack simulators for LEACH-family protocols report that relative algorithm rankings are preserved even though absolute packet-delivery ratios decrease. Nonetheless, full-stack validation (ns-3 or Cooja with IEEE 802.15.4 MAC) remains an explicit direction for future work.

- **Feasibility convention:** Deterministic repair enforces strict radius coverage and backbone reachability; alternative constraint handling may change absolute metric scales.

- **Bounded surrogate:** Normalization makes the optimization more stable across rounds under a fixed budget, but it can also “compress” differences among weak candidates. Therefore, our conclusions rely primarily on end-to-end system metrics, notably Throughput.
- **Central observability:** The BS is assumed to know all node positions (fixed, from pre-deployment survey) and to receive a single residual-energy scalar from each alive node at every round. As quantified in Subsection 4.1, this control-plane overhead represents less than 4 % (centre BS) to 7 % (corner BS) of the total network energy per round, and is borne identically by all eight centralized optimizers.
- **Compute-cost dependence:** CPU time is platform-dependent; we report it on a fixed platform and complement it with NFE as a secondary proxy.

Within this scope and the paired Monte-Carlo protocol, FSS-WSN shows a consistent delivered-utility advantage across the four configurations.

6. CONCLUSION

This work presented FSS-WSN and demonstrated its usefulness for Wireless Sensor Networks. By adding a learning mechanism to the well-known GRASP metaheuristic, the proposed approach naturally fits a centralized decision-making setting for selecting cluster heads. It improves overall network utility through a more guided search that is faster and less costly. We enforced strict feasibility through a deterministic repair procedure, since our goal was to implement a realistic and deployable approach.

The simulations, which covered both favorable and unfavorable configurations, and included diverse and representative baselines (including classical protocols, population-based metaheuristics, and more recent optimizers), demonstrated across all settings (four scenarios) that FSS-WSN consistently achieved the highest “cumulative throughput” until the network becomes inactive, with gains ranging from 2.4% to 4.57% over the best baseline. BS-side CPU time was dramatically lower than that of the strongest iterative competitors, with reductions ranging from 94.1% to 97.19% depending on the scenarios.

The results also indicate that the approach is particularly effective in scenarios with a high risk of bottlenecks. However, compared to some protocol baselines, FSS-WSN can be less favorable in terms of LND (Last Node Die). Overall, this positions FSS-WSN as a strong choice for use cases where the priority is to deliver as much useful information as possible, with a quick decision time (e.g., industrial monitoring, emergency response, sensitive perimeter surveillance, ...etc.), and where the survival of the very last node is less meaningful if it does not correspond to end-to-end delivery.

Future work will focus on bringing the simulation even closer to real conditions by incorporating MAC/PHY constraints that are currently abstracted. Additionally, we will seek to better understand the utility–cost trade-off by using a fixed time limit per round when the BS has to decide under real deadlines.

REFERENCES

- [1] J. N. Al-Karaki and A. E. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 6–28, 2004.
- [2] K. Guleria and A. K. Verma, "Comprehensive Review for Energy Efficient Hierarchical Routing Protocols on Wireless Sensor Networks," *Wireless Network*, vol. 25, no. 4, pp. 1159–1183, 2019.
- [3] C. Nakas, D. Kandris and G. Visvardis, "Energy Efficient Routing in Wireless Sensor Networks: A Comprehensive Survey," *Algorithms*, vol. 13, no. 3, p. 72, 2020.
- [4] H. B. Salameh, M. Dhainat and E. Benkhelifa, "A Survey on Wireless Sensor Network-based IoT Designs for Gas Leakage Detection and Fire-Fighting Applications," *Jordanian Journal of Computers and Information Technology (JJCIT)*, vol. 5, no. 2, pp. 60–72, 2019.
- [5] L. Chhaya et al., "Wireless Sensor Network Based Smart Grid Communications: Cyber Attacks, Intrusion Detection System and Topology Control," *Electronics*, vol. 6, no. 1, p. 5, 2017.
- [6] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-efficient Communication Protocol for Wireless Microsensor Networks," *Proc. of the 33rd Annual Hawaii Int. Conf. on System Sciences (HICSS)*, DOI: 10.1109/HICSS.2000.926982, Maui, HI, USA, 2000.
- [7] O. Younis and S. Fahmy, "HEED: A Hybrid, Energy-efficient, Distributed Clustering Approach for Ad

- Hoc Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.
- [8] G. Smaragdakis, I. Matta and A. Bestavros, "SEP: A Stable Election Protocol for Clustered Heterogeneous Wireless Sensor Networks," *Proc. of the 2nd Int. Workshop on Sensor and Actor Network Protocols and Applications (SANPA)*, 2004.
- [9] S. Arjunan and S. Pothula, "A Survey on Unequal Clustering Protocols in Wireless Sensor Networks," *Journal of King Saud University - Computer and Information Sciences*, vol. 31, no. 3, pp. 304–317, 2019.
- [10] S. Zhang, X. Liu and M. Trik, "Energy Efficient Multi Hop Clustering Using Artificial Bee Colony Metaheuristic in WSN," *Scientific Reports*, vol. 15, p. 26803, 2025.
- [11] R. Sharma, V. Vashisht and U. Singh, "Metaheuristics-based Energy Efficient Clustering in WSNs: Challenges and Research Contributions," *IET Wireless Sensor Systems*, vol. 10, no. 5, pp. 253–264, 2020.
- [12] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proc. of Int. Conf. on Neural Networks (ICNN'95)*, DOI: 10.1109/ICNN.1995.488968, Perth, WA, Australia, 1995.
- [13] S. Mirjalili, S. M. Mirjalili and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.
- [14] D. Karaboga, "An Idea Based on Honey Bee Swarm for Numerical Optimization," [Online], Available: https://abc.erciyes.edu.tr/pub/tr06_2005.pdf, 2005.
- [15] F. A. Hashim et al., "A Novel Meta-heuristic Optimization Algorithm Inspired by Snake Movement Patterns," *Knowledge-based Systems*, vol. 242, p. 108320, 2022.
- [16] N. Chopra and M. M. Ansari, "Golden Jackal Optimization: A Novel Nature-Inspired Optimizer for Engineering Applications," *Expert Systems with Applications*, vol. 198, p. 116924, 2022.
- [17] N. Gupta, A. B. b. A. Hamid, A. B. B. Mahat and A. Kumar, "Machine-learning-enhanced Glowworm Swarm Optimization for Energy-efficient Multi-hop Routing in Wireless Sensor Networks," *Results in Control and Optimization*, vol. 22, p. 100667, 2026.
- [18] N. Gupta et al., "Analysis of Energy-efficient Smart Path Optimization Routing Protocol for Wireless Sensor Networks," *Results in Engineering*, vol. 28, p. 107456, 2025.
- [19] C. A. C. Coello, "Theoretical and Numerical Constraint-handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11–12, p. 1245–1287, 2002.
- [20] T. Mazumder, B. V. R. Reddy and A. Payal, "Energy Based Multi Objective Golden Jackal Optimization for Cluster Based Routing in Wireless Sensor Network," *Soft Computing*, vol. 28, no. 20, pp. 11927–11943, 2024.
- [21] Z. Wang, J. Duan and P. Xing, "Multi-hop Clustering and Routing Protocol Based on Enhanced Snake Optimizer and Golden Jackal Optimization in WSNs," *Sensors*, vol. 24, no. 4, p. 1348, 2024.
- [22] S. Okdem and D. Karaboga, "Routing in Wireless Sensor Networks Using an Ant Colony Optimization (ACO) Router Chip," *Sensors*, vol. 9, no. 2, pp. 909–921, 2009.
- [23] F. Glover, M. Laguna and R. Marti, "Principles and Strategies of Tabu Search," *Handbook of Approximation Algorithms and Metaheuristics: Methodologies and Traditional Applications*, 2nd Ed., T. F. Gonzalez, Ed., Chapman and Hall/CRC, pp. 573–597, 2018.
- [24] R. Jovanovic, M. Tuba and S. Voss, "Fixed Set Search Applied to the Traveling Salesman Problem," *Hybrid Metaheuristics*, vol. 11380, pp. 63–77, Springer, 2019.
- [25] R. Jovanovic and S. Voss, "Matheuristic Fixed Set Search Applied to the Multidimensional Knapsack Problem and the Knapsack Problem with Forfeit Sets," *OR Spectrum*, vol. 46, no. 4, pp. 1329–1365, 2024.
- [26] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd Ed., Prentice Hall PTR, 2002.
- [27] D. Ruan, J. Huang and X. Li, "Uneven Clustering Routing Algorithm Based on Energy and Distance for Wireless Sensor Networks," *Journal of Sensors*, vol. 2019, p. 8109616, 2019.
- [28] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979.
- [29] S. Guha and S. Khuller, "Approximation Algorithms for Connected Dominating Sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, 1998.
- [30] K. Deb, "An Efficient Constraint Handling Method for Genetic Algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2-4, pp. 311–338, 2000.
- [31] T. A. Feo and M. G. C. Resende, "Greedy Randomized Adaptive Search Procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.
- [32] M. G. C. Resende and C. C. Ribeiro, "Greedy Randomized Adaptive Search Procedures: Advances and Extensions," *Handbook of Metaheuristics, Part of the Book Series: Int. Series in Operations Research & Management Science*, vol. 272, pp. 169–220, Springer, 2018.
- [33] H. J. C. Barbosa, H. S. Bernardino and A. M. S. Barreto, "Using Performance Profiles to Analyze the Results of the 2006 CEC Constrained Optimization Competition," *Proc. of the IEEE Congress on Evolutionary Computation (CEC)*, DOI: 10.1109/CEC.2010.5586105, Barcelona, Spain, 2010.
- [34] T. Bartz-Beielstein et al., *Experimental Methods for the Analysis of Optimization Algorithms*, ISBN: 978-3-642-02537-2, Berlin: Springer, 2010.

- [35] T. Kadavy et al., "Impact of Boundary Control Methods on Bound-constrained Optimization Benchmarking," IEEE Transactions on Evolutionary Computation, vol. 26, no. 6, pp. 1207–1221, 2022.
- [36] M. Lopez-Ibañez et al., "The Irace Package: Iterated Racing for Automatic Algorithm Configuration," Operations Research Perspectives, vol. 3, pp. 43–58, 2016.

ملخص البحث:

لا تزال شبكات الاستشعار اللاسلكية مجالاً بحثياً نشطاً في كلِّ من القطاعين العسكري والمدني، مدفوعة بتزايد تطبيقاتها. وفي السنوات الأخيرة، شهدنا تحولاً تدريجياً نحو دمج الذكاء الاصطناعي لمواجهة التحدّي المستمر المتمثل في تحسين استهلاك الطاقة في هذه الشبكات.

في هذا البحث، نقدّم تعديلاً جديداً لآلية البحث في المجموعة الثابتة لتناسب شبكات الاستشعار اللاسلكية. وتضيف هذه الآلية مرحلة تعلّم إلى خوارزمية (GRASP) المعروفة. ويعمل النهج المستخدم على توجيه محطة القاعدة في بيئة مركزية متعدّدة القفزات لاختيار الرؤوس المثلى للمجموعات، الأمر الذي يعظّم الفائدة الإجمالية للشبكة.

وقد قُمنّا بتقييم نهجنا في ظلّ شروط عدالةٍ موثّقة، مقارنةً بمجموعة واسعة من المعايير الأساسية المعتمدة، بما في ذلك بروتوكولات التجميع اللاسلكية، ومجسّات السرب شائعة الاستخدام، والنموذج الهجين متعدّد القفزات.

وأظهرت النتائج تحسُّناً ذا دلالةٍ إحصائية مقارنةً بأفضل معيار أساسي فيما يتعلّق بعدد التقارير المسلمة ووقت وحدة المعالجة المركزية اللازم لاتخاذ القرار. وتُشير هذه النتائج إلى أنّ النهج الذي استخدمناه يمثل خياراً قوياً وعملياً للعديد من حالات استخدام شبكات الاستشعار اللاسلكية.



This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).