International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2025/26 Volume 12, Issue 3, May – June (2025)

RESEARCH ARTICLE

KIPSO: Kalman-Inertia Particle Swarm Optimization Approach for Wireless Sensor Nodes Deployment

Ameer A. Al-Shammaa
Information Technology Research and Development Center, University of Kufa, Al Najaf Governorate, Iraq.
✉ ameer.alshammaa@uokufa.edu.iq

Hadeel N. Saad
Department of Computer, Faculty of Education for Women, University of Kufa, Najaf Governorate, Iraq.
hadeel.jrew@uokufa.edu.iq

Abbas G. Jreou
Technology Incubator Center, Al-Mustaqbal University, Babylon, Iraq.
abbas.jreou@uomus.edu.iq

Received: 03 April 2025 / Revised: 07 June 2025 / Accepted: 13 June 2025 / Published: 30 June 2025

Abstract – Random deployment of wireless sensor nodes within a monitoring area introduces several challenges, including blind spot formation and coverage redundancy. Given that coverage efficiency fundamentally determines the performance efficacy of a Wireless Sensor Network (WSN), formulating an optimization algorithm that effectively balances the conflicting objectives of exploration and exploitation presents a significant challenge. Particle Swarm Optimization (PSO) and its variants have been widely investigated for this purpose, with considerable focus directed at optimizing convergence control parameters. To address the convergence challenge innovatively, this work proposes the Kalman-Inertia Particle Swarm Optimization (KIPSO) algorithm. KIPSO leverages the Kalman filter estimation accuracy to dynamically adjust particle inertia weights during the PSO process. This adaptation enhances both the convergence rate and the precision of the solutions obtained. The experimental and analytical work performed in this study demonstrates that by dynamically adjusting inertia, KIPSO is capable of avoiding local minima and improving node placement, resulting in better coverage and reduced interference compared to methods such as PSO and VFPSO. This leads to enhanced overall network coverage, reduced energy consumption, and prolonged network lifetime.

Index Terms – Coverage Optimization, KIPSO, Wireless Sensor Network, PSO, VFPSO, Wireless Nodes Deployment.

1. INTRODUCTION

Wireless sensor networks (WSNs) find widespread application in both civil and military surveillance, such as agricultural monitoring, smart city infrastructure, and industrial observation systems [1]. In most deployments, randomly distributed sensor nodes within the monitoring area form a wireless sensor network by cooperatively transmitting data [2]. Sensor nodes perform data acquisition, analysis, and transmission from monitored areas to a central station, enabling informed decision-making [3, 4]. Nevertheless, WSNs face significant challenges in optimizing deployment, minimizing data transmission, and extending network lifetime. These difficulties stem from inherent node limitations and frequently harsh operating environments [2, 5]. Such constraints directly impact sensing accuracy, energy consumption, and overall network longevity [6]. Consequently, efficient management of energy resources, data traffic, and node coverage is imperative to ensure reliable, sustained WSN operation. A typical traditional monitoring method fails to address the dynamic and complex nature of real-world environments, which requires sophisticated algorithms to provide optimal coverage for the monitoring area over a long time [7]. Consequently, significant research effort has been directed towards overcoming coverage issues through the proposal of diverse heuristic approaches [8–10].

Among these approaches, Particle Swarm Optimization (PSO) [10] was applied to determine optimal coverage configurations for randomly deployed sensor nodes within a designated monitoring area. Although the PSO algorithm is characterized by flexibility and simplicity of implementation [11], it suffers from several shortcomings. As an example, it is necessary to repeat its iteration process several times to find the best solution [12]. Consequently, it fails to provide optimal solutions within a reasonable amount of time, especially when it is applied to critical applications which require a quick response, such as gas leak detection and fire detection.

Thus, this research aims to address the challenges inherent in PSO for wireless sensor node placement by proposing an

RESEARCH ARTICLE

innovative enhancement: the Kalman-Inertia Particle Swarm Optimization (KIPSO) algorithm. KIPSO integrates the accurate predictive capabilities of the Kalman filter with PSO optimization strengths and an inertia weight mechanism. This synthesis yields a more efficient and resilient deployment strategy than the standard PSO. Specifically, the Kalman filter enhances prediction accuracy by dynamically adjusting node positions using real-time data, while PSO effectively explores the solution space to identify optimum configurations. The inertia weight further optimizes the search process by balancing exploration and exploitation, promoting convergence toward global optima. Accordingly, the proposed hybrid method simultaneously improves placement accuracy and efficiency and enables a robust, adaptive network capable of preserving optimal functionality when responding to environmental dynamics. Ultimately, these enhancements contribute to improved coverage, energy efficiency, and network longevity. This paper subsequently examines the implementation, performance, and benefits of KIPSO for wireless sensor networks.

The remainder of this work is structured in six sections: Section 2 reviews relevant literature. Section 3 details the standard Particle Swarm Optimization (PSO) algorithm. Section 4 outlines the Kalman filter's principles and operational workflow. Section 5 delineates the proposed KIPSO methodology. Section 6 comparatively assesses KIPSO, PSO, and VFPSO performance. Finally, Section 7 concludes with key findings, contributions, and possible future works.

2. RELATED WORK

Numerous investigations have examined challenges inherent in sensor node deployment. The majority of proposed approaches to WSN coverage challenges emphasize dual objectives: coverage expansion and operational longevity enhancement, as the networks are typically deployed in remote regions that are difficult to reach for repairs in the event of issues. For example, evolutionary methods have been developed to address coverage limitations arising from stochastic sensor node deployment in surveillance regions. An ant colony algorithm [9], an artificial fish swarm algorithm [8], a firefly algorithm [13], a bacterial foraging algorithm [14], and a particle swarm optimization (PSO) algorithm [15] are examples of evolutionary methods. The principle of operation of these algorithms is based on the behavior of what is known as swarm intelligence. The core objective of swarm-based techniques is to resolve deployment inaccuracies through precise node localization. This framework counteracts the spatial imbalance of stochastically distributed sensors within monitoring environments. Furthermore, the swarm intelligence methodology is greatly beneficial in solving other deployment problems, including searching for places that are not covered by sensor nodes due to the random deployment of network nodes to cover them and places where more than one sensor node is deployed and overlaps. For example, the artificial fish swarm algorithm presented in [16] is primarily designed to enhance the efficiency of coverage optimization by simulating fish behavior, where the group of fish follows the fish that find a place with enough food. Similarly, Ke-qing Li [17] implemented the Artificial Fish Swarm Algorithm (AFSA) to optimize coverage efficiency in WSN deployments. Despite this, there is one disadvantage in that the latter algorithm performs a blind search, making it unsuitable for time-critical applications. There has been an extension of the ant colony algorithm proposed by Jingwen Tian et al. [18], as they attempted to improve the coverage optimization in the old version of the algorithm; however, their method experienced some problems, such as stagnation phenomena and slow coverage rates.

Owing to its implementation simplicity, robustness, and computational efficiency, the Particle Swarm Optimization (PSO) algorithm demonstrates superior performance in solving both static and dynamic optimization problems compared to alternative evolutionary approaches [19]. However, as the search space dimensionality increases, the computational burden of the PSO algorithm escalates proportionally when seeking optimal solutions within the solution domain [12]. This leads to higher energy consumption and a longer time to find the best solutions for coverage optimization, which decreases the lifetime of the network. Furthermore, parameter calibration for optimal performance presents significant challenges, as particle population density and solution space dimensionality vary substantially across applications [20].

A number of researchers have extensively studied methodological constraints in original PSO frameworks to mitigate performance deficiencies. For example, the inertia weight mechanism has been combined with the standard PSO to achieve optimal optimization within a limited number of iterations [21]. In [22], another solution for improving PSO is presented, in which random coefficients are generated to be used in the PSO algorithm to improve coverage. Whereas, in [23–29], different strategies and methods are suggested to adapt to the complexity and variation of the search area in terms of size and conditions. A new optimization algorithm (called Virtual Force-Directed Particle Swarm (VFPSO)) based on combining two algorithms (Virtual Force (VF) and particle swarm optimization (PSO)) to improve WSNs deployment has been proposed in [30]. VFPSO incorporates the VF algorithm [31] to enhance PSO-based coverage optimization across stationary and mobile network deployments. As part of the VF process, a balanced mixing of attractive and repellent forces is employed to calculate the speed and direction of virtual motions for sensors. Based on these computations, the velocity vector of every particle is dynamically updated within the solution space. Within the

RESEARCH ARTICLE

VFPSO framework, particle speeds are recalibrated through integration of VF-derived calculations and PSO-generated local/global optimal solutions accumulated during optimization. Although the algorithm proposed in [30] decreases the sensors' energy consumption, hence extending the network lifetime, it is unable to deal with multiple failures that occur in different parts of the network. Wu et al. [32] introduce an extended VFPSO algorithm that integrates Dempster-Shafer theory, enhancing network coverage by 19.34% over conventional VFPSO. Nevertheless, the framework retains fixed prediction parameters incapable of adapting to network dynamics. Li and Cao [33] developed an Adaptive Binary PSO (ABPSO) incorporating adaptive learning and genetic mutation, reducing active nodes by 36% while maintaining over 90% coverage. Although ABPSO enhances convergence, it lacks predictive parameter adjustment based on optimization state trends. Liu et al. [34] proposed HICPSO with linearly decreasing inertia and compressed search space, achieving 3–5% improvements in convergence and localization. However, they provided no methodological guidance for parameter tuning to optimize results. Bhargavi et al. [35] integrated Delaunay Triangulation with PSO to improve coverage and energy efficiency, though this introduced computational complexity and communication range limitations. Amer et al. [36] combined comprehensive learning and Fick's law within PSO, yielding 66.5% higher connectivity and 16.56% greater coverage. Nevertheless, their multi-algorithm approach proved incompatible with resource-constrained nodes and prioritized scalability/connectivity while neglecting critical energy consumption metrics. Siamantas and Kandris [37] employed meta-optimized and manually tuned PSO for k-coverage and 1-connectivity, but static parameters restricted adaptability and comparative analysis.

Collectively, existing hybrid algorithms address deployment scalability and connectivity but incur high parametric dimensionality. To resolve persistent challenges in dynamic exploration-exploitation balance and environmental adaptability, this research advances the Kalman-Inertia PSO (KIPSO) framework, achieving optimal resource utilization, accelerated convergence, and consistent coverage-connectivity without complex hybridization constraints. KIPSO's efficacy will be comparatively validated against foundational reference algorithms: Particle Swarm Optimization (PSO) and Virtual Force PSO (VFPSO).

3. STANDARD PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) [10] is a population-based computational intelligence algorithm inspired by emergent collective intelligence in biological systems, notably avian flocking and piscine schooling behaviors. This evolutionary algorithm has been effectively applied to optimize coverage configurations for stochastically deployed sensor nodes within monitoring areas. It seeks a better solution from the multiple solutions generated randomly during its initial iterations by searching a space containing many particles. A PSO iteration process involves two functions: velocity and position updates. In Figure 1, each step of the PSO process is described, from initialization to iterative updates of position and velocity, addressing the problem of wireless sensor nodes coverage optimization. Furthermore, this figure illustrates the iterative nature of the PSO process, emphasizing the use of particle positions and velocities to explore the search space for optimal wireless sensor node configurations.

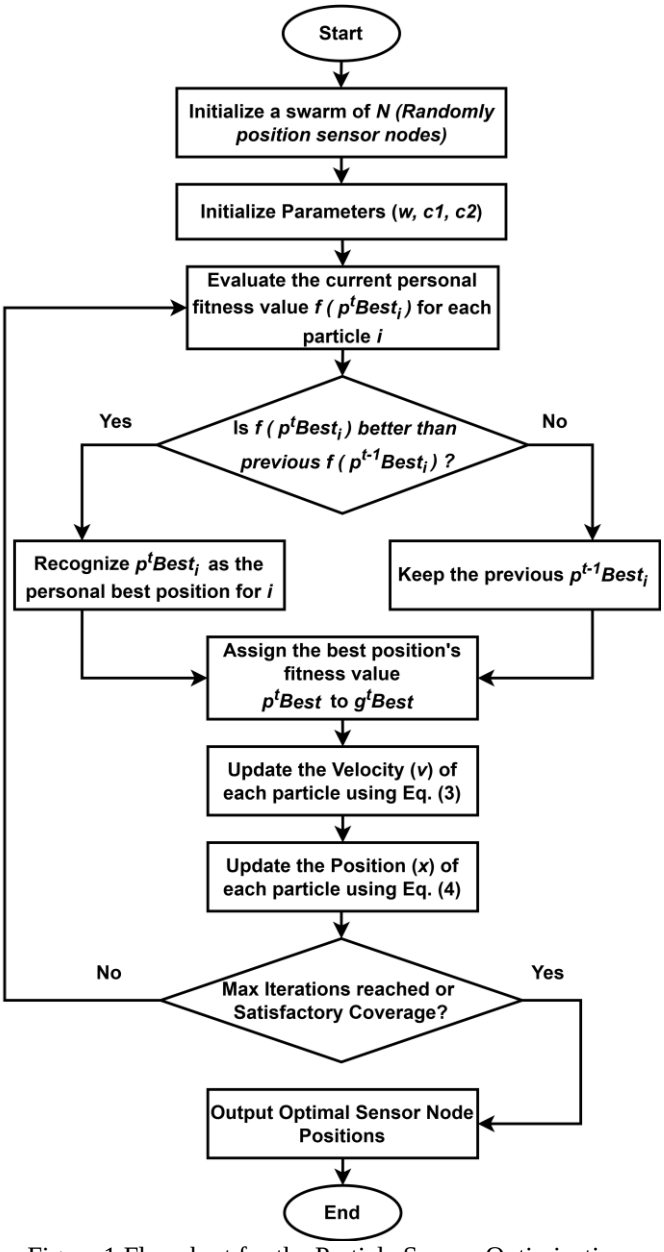


Figure 1 Flowchart for the Particle Swarm Optimization Operation

RESEARCH ARTICLE

As shown in Figure 1, the PSO has eight phases, which can be discussed in more detail as follows:

3.1. Initialize Particle Swarm

The PSO work starts with initialization, where several parameters are initialized (see Figure 1). As part of initialization, both the initial pBest positions for individual particles and the swarm-wide gBest position (optimal among all particles) are determined.

3.2. Calculate Coverage Fitness for Each Particle

The second phase of the PSO process involves computing coverage fitness values for each particle using a predefined objective function (e.g., total area covered by sensor nodes or reduction of nodal overlap). Furthermore, the objective function may incorporate single or multiple constraints during fitness evaluation, such as energy consumption, deployment costs, and communication range limitations.

3.3. Update Personal Best (pBest)

During this phase, each particle's current position (p^tBest) is evaluated against its prior personal best position (p^{i-1}Best) within the solution space. If the current position yields superior coverage (higher fitness value), the personal best is updated to the new position (p^iBest), as shown in equation (1).

$$p^i\text{Best} = \arg\ \min_{x_j \in N_i} f(x_j) \quad (1)$$

Where N_i denote the topological neighborhood of particle i, f (x_j) represents the fitness value of neighborhood particle $x_j \in N_i$, and p^iBest signify the optimal position within N_i.

3.4. Update Global Best (gBest)

This phase computes the global optimum (gBest) for all particles. The gBest parameter is then iteratively refined per equation (2) using Phase 3 results.

$$\text{gBest} = \arg\ \min_{i\in\{1,2,\ldots,N\}} f(x_i) \quad (2)$$

Where N denotes the swarm population size, $f(x_i)$ represents the fitness value of particle i^{th} position x_i, and arg min identifies the position yielding the minimum fitness value (or maximum, depending on the optimization problem).

3.5. Update Particle Velocity

During this phase, particle velocities are updated according to equation (3):

$$v_i^{t+1} = w.v_i^t + c_1.r_1.(\text{pBest}_i - x_i) + c_2.r_2\ .(\text{gBest} - x_i) \quad (3)$$

where:

- v_i^{t+1}: Updated velocity for particle i.
- w: Inertia weight (modulates the exploration-exploitation trade-off).
- c_1, c_2: Cognitive and social acceleration coefficients.
- r_1, r_2: Uniformly distributed random variables ∈ [0, 1].
- pBest_i: Personal best position of particle i.
- gBest: Global best position.
- x_i: Current location vector of particle i.

3.6. Update Particle Position

This phase employs equation (4) to iteratively reposition particles (sensor nodes) within the search domain using velocity vectors computed in Phase 5.

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (4)$$

3.7. Check Stopping Criteria

At this stage, the algorithm evaluates whether predefined stopping criteria are satisfied. The optimization process advances to subsequent phases upon meeting any of the following conditions:

1) Attainment of the maximum iteration count.
2) Solution convergence (insignificant improvement in global best fitness across successive iterations).
3) Achievement of a predetermined coverage threshold.

Should termination criteria remain unsatisfied, the process reverts to Phase 2 and iterates until the stopping criteria are met.

3.8. Output optimal sensor node position

In the final phase, the global best position (gBest), representing the optimal position for each sensor node, is determined. The resulting gBest achieves either coverage maximization or coverage gap minimization.

4. KALMAN FILTER

The Kalman filter [38] operates as a recursive estimation algorithm for dynamic system states, employing prediction-correction mechanics to forecast future system states and refine them through incoming measurements. Within wireless sensor networks (WSNs), this technique optimizes node coverage in monitoring regions, particularly under mobile node deployment or dynamic environmental conditions. Kalman filtering quantifies uncertainties via two noise parameters: process noise (system model inaccuracies) and measurement noise (sensor observation errors). These uncertainties play a major role in determining how the filter estimates a system's state. Uncertainty in process noise is associated with the system's evolution and the evolution of the inertial weight over time. To correct the overfitting issue and

RESEARCH ARTICLE

enhance PSO performance, it is important to take into consideration the adaptive or dynamic nature of the system. As an example, in low dynamic optimization problems whose search space and convergence patterns are somewhat predictable, the process noise should be relatively low to allow the Kalman filter to adjust near the current inertia weight in response to the changing environment of the PSO algorithm and the smooth version of the inertia weights. Conversely, low process noise will cause the Kalman filter to make more aggressive adjustments to the inertia weight when the PSO is stuck, resulting in slow convergence and poor deployment. On the other hand, measurement noise corresponds to observational uncertainty within the system, inherently linked to performance indicators such as the iteration-specific optimal fitness values generated by the PSO approach. In the deployment process, the fitness values are relatively unstable and can reach local minima, which results in poor best fitness function values (high noise in observations). In this case, the measurement noise can be set high, allowing the Kalman filter to be untrusting of PSO's performance feedback. Optimal sensor node deployment and PSO inertia weight adaptation thus necessitate careful calibration of Kalman filter parameters: process noise (Q) and measurement noise (R). The Kalman filter methodology comprises four sequential steps for wireless sensor node deployment optimization, detailed as follows:

4.1. Initialization

At this step, a number of parameters must be initialized in the vector ($\hat{X}_k$) for each sensor node at a time (k) as shown in equation (5). Node position (x_k, y_k), node velocity (v_x, v_y), coverage radius, covariance matrix (P_k), process noise (Q), and measurement noise (R) are examples of the parameters being initialized at this phase.

$$\hat{X}_k = [x_k, y_k, v_x, v_y] \quad (5)$$

4.2. Prediction

During the prediction phase, Kalman filters forecast a node's current state and associated uncertainty by extrapolating from its prior state and governing dynamics (e.g., linear or nonlinear motion models). These predictions are implemented via equations (6) and (7).

$$\hat{X}_k = F.\hat{X}_{k-1} + B.u_{k-1} \quad (6)$$

$$P_k = F.P_{k-1}.F^T + Q \quad (7)$$

Define:

- F: State transition model.
- $\hat{X}_k$: Predicted state estimate at timestep k.
- B: Control input model
- u_k: Control vector
- Q: Process noise covariance matrix
- P: Estimate the covariance matrix

4.3. Measurement

The Kalman filter utilizes information measured (Z_k) and collected by wireless sensor nodes in an area of interest to update its prediction. This measured information includes distances or signal strength between the node and its neighbors or a fixed base station (sink node) in the environment.

4.4. Update

This step involves updating the state estimate of the sensor node by incorporating the new measurements obtained in Section (4.3) using equations (8), (9), and (10).

$$K_k = P_k.H^T.(H.P_k.H^T + R)^{-1} \quad (8)$$

$$\hat{X}_k = \hat{X}_{k-1} + K_k.(Z_k - H.\hat{X}_{k-1}) \quad (9)$$

$$P_k = (1 - K_k.H).P_{k-1} \quad (10)$$

Define:

- K_k: Kalman gain (governing the relative weighting between prediction and measurement).
- P_k: State covariance matrix (quantifying uncertainty in the state estimate).
- R: Measurement noise covariance matrix.
- H_k: Observation model matrix.
- Z_k: Measurement vector at timestep k.
- I: Identity matrix.

5. PROPOSED ALGORITHM

5.1. KIPSO Algorithm Process

The proposed Kalman-Inertia Particle Swarm Optimization (KIPSO) algorithm dynamically adapts the inertia weight through synergistic integration of standard PSO and Kalman filter principles. KIPSO's formulation emerges from critical limitations observed in existing PSO-based deployment approaches for Wireless Sensor Networks (WSNs), notably their susceptibility to premature convergence, reliance on static parameters, and inability to respond dynamically to emergent swarm behavior. Furthermore, the PSO method lacks an effective feedback mechanism to address stochastic fluctuations in optimization performance. While prior solutions, such as linearly decreasing inertia in equation (11) and randomized inertia updates in equation (12), exist, they

RESEARCH ARTICLE

remain fundamentally unresponsive to real-time swarm performance metrics across iterations.

$$w_k = w_{max} - \frac{k}{k_{max}}(w_{max} - w_{min}) \quad (11)$$

$$w_k \sim U(w_{max}, w_{min}) \quad (12)$$

Within the proposed methodology, the inertia weight is modeled as a dynamic state variable. Its optimal value undergoes recursive estimation via a Kalman filter, which conceptualizes inertia as a system state while utilizing the swarm's best function value as its measurement input. This filter minimizes the expected squared estimation error, as shown in equation (13), through covariance matrices representing: a) inertia change uncertainty (Q) and b) swarm performance fluctuations (R).

$$E\left[\left(w_k - \widehat{W}_k\right)^2\right] \quad (13)$$

This recursive estimation framework and lightweight computation facilitate integration with iterative metaheuristics like PSO. This enables adaptive inertia adjustment derived from comparative analysis between the current global best fitness and historical optimum fitness over the preceding N iterations using the PSO algorithm. When fitness improvement surpasses a defined threshold, optimization triggers inertia weight reduction. Otherwise, subsequent inertia weights are estimated via the Kalman filter. The proposed algorithm improves the PSO's ability to balance exploration and exploitation. To obtain greater accuracy from the Kalman filter, it is important to select the initialization parameters of the filter precisely at the beginning of the KIPSO process, such as previous inertia weights. For a balanced approach, the Exponential Moving Average (*EMA*) of past inertia weights can be used to incorporate both responsiveness and stability. Primary control parameters essential to the proposed algorithm's operation are cataloged in Table 1. Moreover, the KIPSO algorithm executes its optimization through five principal phases, with the core computational process operationalized during the Inertia Weight Adaptation step via equations (14), (15), (16), (17), and (18). The complete procedural workflow is formally delineated in Algorithm 1.

$$w_{new} = 0.5 \times w_{previous} \quad (14)$$

$$w_{ema} = \alpha \times w_{last} + (1 - \alpha) \times w_{ema,prev} \quad (15)$$

$$k = \frac{P_{previous}}{(P_{previous} + R)} \quad (16)$$

$$w_{new} = w_{previous} + k \times \left(gBest - w_{previous}\right) \quad (17)$$

$$P_{new} = (1 - k) \times P_{previous} + Q \quad (18)$$

Table 1 Primary Control Parameters Essential to the Proposed Algorithm's Operation

Parameters	Explanation
ε	Threshold for Evolution: determines if the fitness improvement is significant enough.
α	The smoothing factor (e.g., 0.7)
Q	Process Noise Variance: Controls uncertainty in the Kalman filter process model.
w	Inertia weight
$w_{initial}$	Initial estimate of inertia weight
w_{ema_prev}	Previous EMA of inertia weights.
R	Measurement Noise Variance: controls noise in the measurement used by the Kalman filter.
P	Initial Covariance: The starting value for covariance in the Kalman filter.
$max_{iterations}$	Maximum iterations
iter	Iteration counter
S	Swarm size
x	Position of particle
v	Velocity for particle
c_1, c_2	Cognitive and social acceleration coefficients.
r_1, r_2	Uniformly distributed random variables ∈ [0, 1].
pBesti	Personal best position of particle i.
gBesti	Global best position.

Output: the final results of x_i, gBest, fiteness(gBest)

Input: Initialization step: set parameters:

- Optimization objective function f(x), where x =(x_1, x_2, …….. , x_d)^T ∈ R^d
- Stochastic initialization of particle positions (x_i) and velocities (v_i)
- Initialize personal best positions ($pBest_i$) and global best position (gBest) (at t = 0)
- S, $max_{iterations}$, c_1, c_2, w, Q, R, $w_{initial}$, and P
- Set iteration counter (iter = 0)

RESEARCH ARTICLE

While Criterion do

While iter < $max_{iterations}$ do

Evaluate Fitness:

for all particles i in the swarm S do

Calculate fitness value $f(P^tBest_i)$ for each particle i.

if $f(P^tBest_i)$ better than previous $f(P^{t-1}Best_i)$

then

Update ($pBest_i$) with the new ($pBest_i$) and $f(P^tBest_i)$

else

Keep the previous $(P^{t-1}Best_i)$

endif

end for

Update (gBest) and f(gBest) based on ($pBest_i$).

Check Optimization Evolution:

Compare (g^tBest) with ($g^{t-1}Best$) over the last N iterations.

Adapt Inertia Weight:

if improvement > ε then

Consider optimization process to be evolving and update w using equation (11).

else

Optimization is stagnating do:

- Calculate EMA of past inertia weights using equation (15).
- Calculate Kalman Gain (k) using equation (16).
- Update Estimate using equation (17).
- Update Covariance using equation (18).

endif

Update Particle Velocities and Positions:

for all particles i in the swarm S do

Update velocity (v_i^{t+1}) using equation (3).

Update new positions (x_i^{t+1}) using equation (4).

endfor

Iteration Control:

Update iter = iter + 1.

if iter >= $max_{iterations}$ or no significant improvement is observed then

end while

else

Return back to Evaluate Fitness:

Endif

endwhile

endwhile

Algorithm 1 Pseudocode of the KIPSO Algorithm

5.2. Proposed System Assumptions

In this paper, all experimental work is conducted under the following assumptions:

5.2.1. Process Noise (Q) and Measurement Noise (R)

First, we investigate how variations in process noise covariance (*Q*) and measurement noise covariance (*R*) influence optimal sensor node placement within the target environment. This parametric analysis identifies *Q* and *R* values that maximize coverage optimization performance in the proposed algorithm. Consequently, identical noise parameters are applied in subsequent comparative evaluations of benchmark algorithms to ensure equitable performance assessment.

Table 2 Process Noise and Measurement Noise on Coverage Optimization Using KIPSO

	Low R = 0.2	Moderate low R = 0.4	Moderate high R = 0.6	High R = 0.8
Low Q = 0.2	2.045	2.045	2.045	1.992
Moderate low Q = 0.4	2.045	2.045	2.045	2.045
Moderate high Q = 0.6	2.045	2.04	2.045	2.045
High Q = 0.8	2.045	2.045	2.045	2.045

As Table 2 indicates, tested *Q* and *R* values ranged from 0.2 to 0.8. The results demonstrate that KIPSO achieves peak coverage optimization (1.992) when *R* = 0.8 and *Q* = 0.2.

5.2.2. Communication Range

Each sensor node in this research employs an IEEE 802.15.4-compliant communication module. As established in [39], this protocol supports practical indoor transmission ranges of 10-30 meters, extending to 75 meters outdoors. Consequently, this study adopts a 40-meter communication range, conforming to medium-range sensor standards for moderately dense or outdoor environments. This configuration achieves an optimal balance between coverage and energy efficiency in monitoring applications such as environmental sensing and precision agriculture.

International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2025/26 Volume 12, Issue 3, May – June (2025)

RESEARCH ARTICLE

5.2.3. Experimental Area Size

The total size of the experimental area (A_{total}) used in this research was (300 m × 300 m =90,000 m^2).

5.2.4. Covered Area

- With the help of equation (19), the size of the area covered by each node (A_{node}) can be calculated.

$$A_{node} = \pi \times r^2 = \pi \times 40^2 \approx 5026.55\ m^2 \quad (19)$$

Where r denotes the wireless transmission radius of a sensor node.

- The total covered area can be calculated theoretically using equation (20).

$$A_{covered} = N \times A_{node} \quad (20)$$

- Based on the output of equations (19) and (20), the coverage ratio can be calculated using equation (21).

$$Coverage\ Ratio = \frac{A_{covered}}{A_{total}} \quad (21)$$

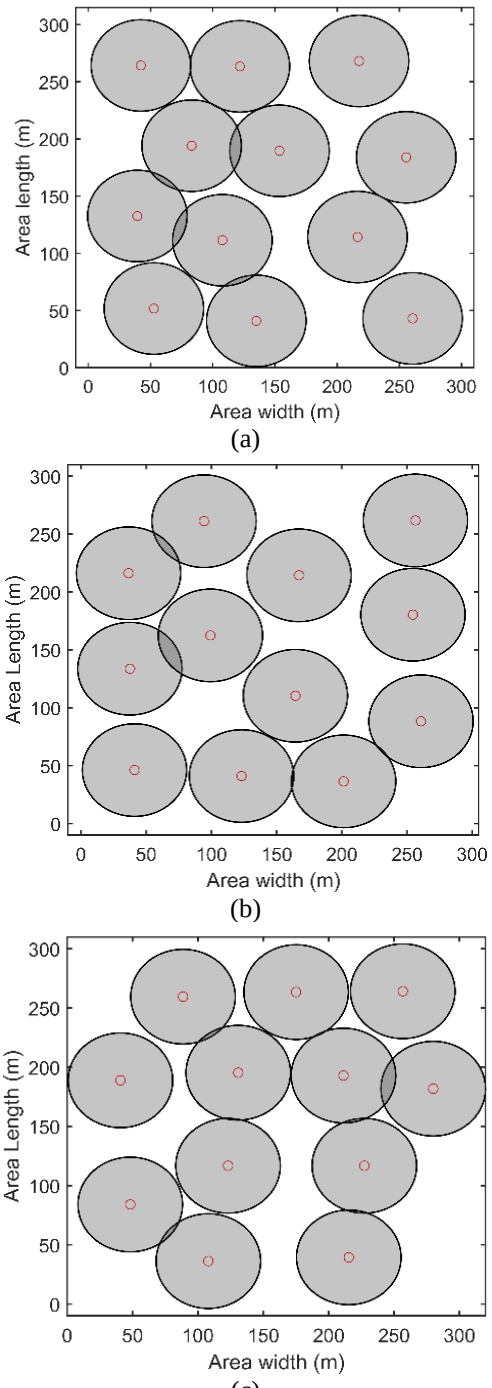
5.2.5. Number of Nodes (N)

This study employed varying node counts (N = 10-18) across sequential experiments to assess the algorithm's coverage optimization performance. Initial nodes are selected according to the required challenge level to demonstrate the optimization capability that can be used to improve search efficiency within the solution space. According to equation (21), starting with 10 nodes results in coverage of 60% of the monitoring area, which is an acceptable level for initialization. In contrast, using 18 nodes can achieve an almost 100% coverage rate.

6. RESULTS AND DISCUSSIONS

This study employs rigorous performance analysis to contrast the efficacy of the proposed algorithm against reference implementations of PSO and VFPSO. Multiple performance metrics were analyzed to empirically validate the proposed algorithm's superiority over benchmark methods (PSO, VFPSO) in coverage optimization efficacy. These parameters tested here include optimization of the node deployment and uncovered areas based on the iteration number of the optimization algorithm for different numbers of nodes. All simulations were executed within the MATLAB® environment to evaluate the Kalman-Inertia Particle Swarm Optimization (KIPSO) algorithm's performance. In addition, the wireless nodes were assumed to be thrown randomly and non-uniformly throughout the experimental area, which has a dimension of 300 by 300 meters. Identical parameter configurations were maintained across all experimental trials to ensure equitable comparison between the proposed KIPSO framework and benchmark algorithms (PSO, VFPSO). Moreover, all algorithms have been assessed under the same number of iterations. Last but not least, the average of twenty runs was used to calculate all experimental results.

6.1. Deployment Evaluation



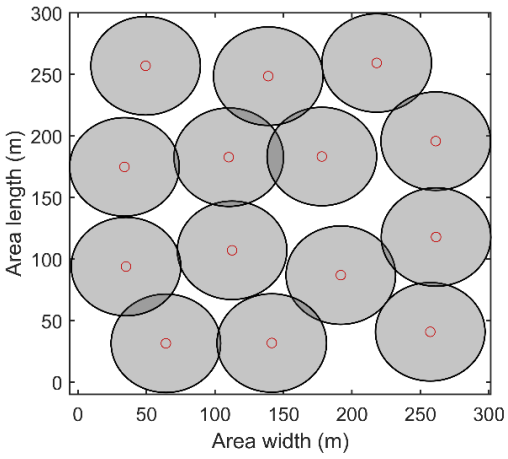
(a) (b) (c)

Figure 2 Deployment Optimization Evaluation of: (a) KIPSO, (b) VFPSO, and (c) PSO Using 12 Nodes

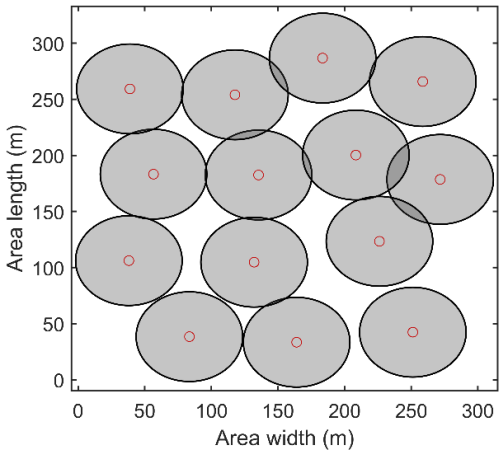
International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2025/26 Volume 12, Issue 3, May – June (2025)

RESEARCH ARTICLE

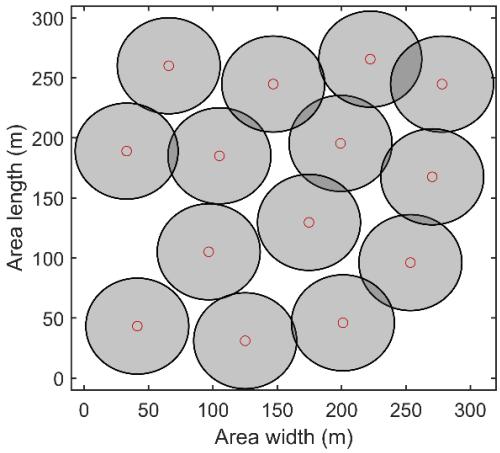
This experimental series benchmarks the proposed algorithm against established methods (standard PSO, VFPSO) for wireless sensor node deployment optimization within a 300 *m* × 300 *m* monitoring area (system assumptions detailed in Section 5.2). Each sub-experiment has been carried out with a different number of nodes to evaluate the deployment optimization of each algorithm under a variety of deployment scenarios. Within Figures 2, 3 and 4, the square perimeter demarcates the 300 *m* × 300 *m* operational zone designated for comprehensive sensor network coverage. Similarly, the small red circles represent the locations of the nodes within the monitored area, and the large grey circles indicate the ranges of coverage for each node within the region. The coverage range of all nodes employed in the area was assumed to be equal, which is 5026.55 m^2 (see Sections 5.2.3 and 5.2.4 for more details about node coverage calculation). The first scenario, as shown in Figure 2, involved the deployment starting with a challenging number of nodes (e.g., 60% area coverage, beginning with 12 nodes) to cover the experimental area using the three algorithms evaluated. An Initial number of nodes is selected according to the required challenge level that could showcase the optimization capability to improve search efficiency within the solution space. In this work, 60% of the covered area is an acceptable challenging initialization. Even though the number of nodes in this test is smaller than the number required to completely cover the area, the deployment optimization achieved by the proposed algorithm, shown in Figure 2a, outperformed the standard PSO and VFPSO algorithms in Figure 2b and Figure 2c. While node count was insufficient for complete area coverage, the proposed algorithm achieves enhanced deployment optimization (Figure 2a), exhibiting performance advantages over benchmark PSO and VFPSO configurations (Figures 2b and 2c). Under these conditions, the selected values for process noise (Q=0.2) and measurement noise (R=0.85) provide a crucial balance between stability and adaptability.



(a)



(b)



(c)

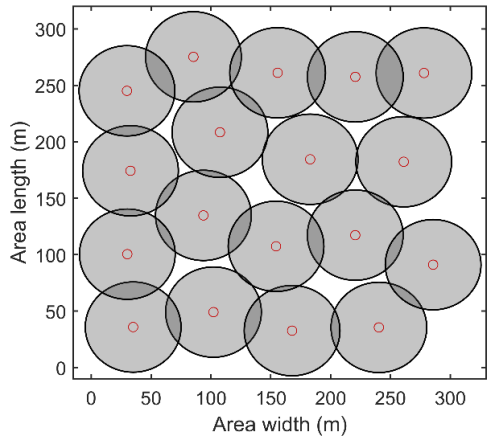
Figure 3 Deployment Optimization Evaluation of: (a) KIPSO, (b) VFPSO, and (c) PSO Using 14 Nodes

The smaller Q value ensures that inertia weight adjustments are cautious, minimizing abrupt changes that could destabilize the particle swarm when coverage is sparse. The higher R value helps the Kalman filter remain less sensitive to variations in fitness values, allowing the algorithm to adapt without overreacting to local fluctuations.

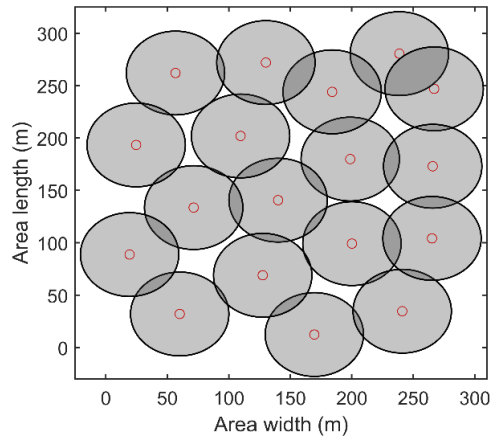
This balance allows the algorithm to explore effectively in underpopulated regions, gradually improving coverage while maintaining stability.

On the other hand, in the second scenario of this test, 14 nodes were used, which covered 70% of the area targeted. The KIPSO algorithm achieved higher performance than VFPSO and PSO in terms of high coverage with minimal interference, as demonstrated in Figure 3.

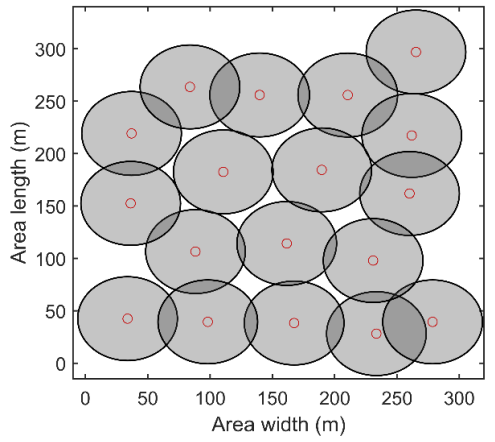
RESEARCH ARTICLE



(a)



(b)



(c)

Figure 4 Deployment Optimization Evaluation of: (a) KIPSO, (b) VFPSO, and (c) PSO Using 18 Nodes

However, when 18 nodes were used in the third deployment scenario (see Figure 4), the proposed algorithm achieved approximately full coverage (90% of the area covered). At the same time, PSO and VFPSO were unable to provide coverage in some places in the monitoring area when used for deployment optimization, even when there were enough nodes. These results empirically demonstrate the proposed algorithm's superiority over standard PSO and VFPSO in deriving optimal coverage configurations across diverse deployment scenarios. In these scenarios, the Kalman filter's-controlled inertia weight adjustments play a key role. The lower process noise ensures gradual and steady updates, promoting convergence to optimal node deployment configurations. Simultaneously, the higher measurement noise mitigates the impact of minor fitness variations, preventing the algorithm from prematurely locking onto suboptimal solutions. This approach enhances the algorithm's ability to fine-tune the deployment with precision, achieving effective coverage while maintaining robustness against overfitting. Finally, test results confirm that higher node density within the monitoring region improved coverage metrics (ratio and accuracy) when utilizing the KIPSO algorithm.

6.2. Coverage Optimization Evaluation

An assessment of the uncovered area from the monitoring area was conducted based on iterations of the KIPSO algorithm compared to the standard PSO and VFPSO using different node numbers. This methodological assessment validates KIPSO's coverage optimization performance against that of peer algorithms (PSO, VFPSO) through a rigorous comparative analysis. Each sub-test was conducted with a different number of nodes (N = 12, 14, 16, 18) deployed in a fixed-size area (300 m × 300 m). Each sub-test began with a random deployment of the nodes in the experimental area. KIPSO subsequently initiates sequential optimization iterations to refine sensor node positioning within the designated area, converging to deployment solutions that minimize coverage gaps while maximizing spatial efficiency. To verify that KIPSO outperformed PSO and VFPSO, a performance comparison was conducted, as illustrated in Figure 5. In all sub-figures within Figure 5, the uncovered area measurement is displayed on the y-axis. It represents the reciprocal of the covered-to-total monitoring zone ratio.

Based on the results shown in Figure 5, it appears that the KIPSO algorithm achieved the highest coverage optimization compared to PSO and VFPSO. Nevertheless, all three algorithms exhibit monotonic improvements in coverage efficacy with increasing iteration counts and node deployment density within the experimental region. As an example, the highest coverage optimization achieved by KIPSO at 10000 iterations and the number of nodes used (12, 14, 16, 18) was approximately 68,5%, 77.5%, 84.75%, and 90%, respectively. Furthermore, as demonstrated in Figure 5, all algorithms achieve about the same optimization performance at the beginning (less than 1000 iterations) as a result of using the

RESEARCH ARTICLE

same initial parameter values, including inertia weights (w), acceleration coefficients (c_1, c_2), particles' personal bests (pBests), global bests (gBests), etc., as explained in Sections 3 and 5. Consequently, none of the algorithms evaluated succeeded in finding the optimal position for nodes within the monitoring area. This occurs because the value of w is typically high in those iterations, resulting in oscillations around the optimal position of the nodes. However, as more iterations are performed, it becomes more apparent that each algorithm performs differently in terms of coverage performance.

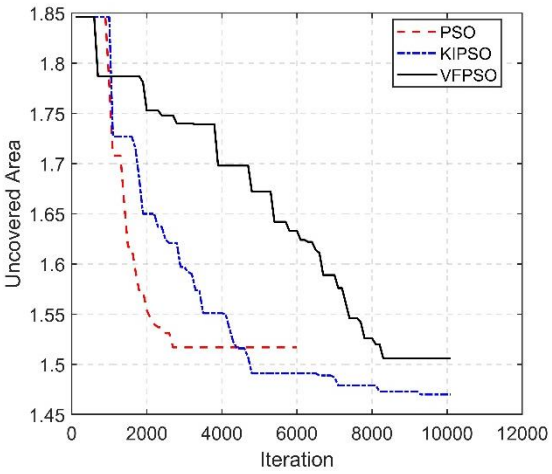
This is because each algorithm adapts its parameters, including w, according to its particular procedure in a dynamic manner. In conventional Particle Swarm Optimization (PSO), coverage gaps diminish progressively with increased iteration counts, irrespective of nodal density within the monitoring environment. This trend emerges as PSO dynamically adjusts its inertia weight (w) and fitness evaluation to converge toward optimal solutions. The empirical analysis of Figure 5 results is categorized into four distinct experimental scenarios.

6.2.1. Scenario 1: Measuring Uncovered Area Using 12 Nodes, Figure 5a.

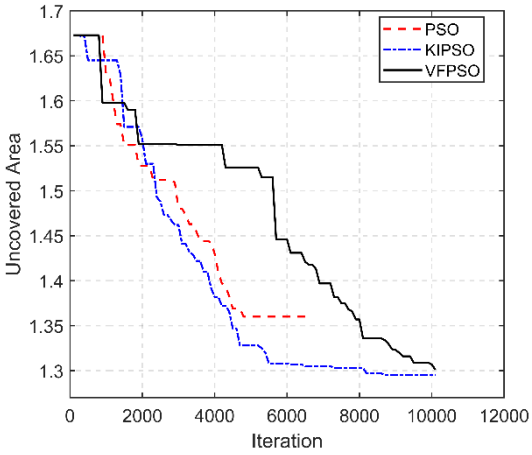
It is obvious that KIPSO achieves the smallest uncovered area with steady and fast convergence, VFPSO converges rapidly in the early iterations but plateaus prematurely, leaving a higher uncovered area, whereas PSO shows slower convergence and the largest uncovered area. With only 12 nodes, the optimization challenge is significant, requiring an algorithm that balances exploration (finding areas with high coverage potential) and exploitation (refining node position), then KIPSO benefits from its dynamic nature to balance this trade-off, achieving better final coverage. Early on, exploratory behavior is triggered by higher estimated uncertainty. The Kalman filter promotes controlled exploitation by stabilizing inertia as advancements mount. VFPSO sacrifices solution quality by prematurely stabilizing, leaving a higher uncovered area, and PSO without adaptive mechanisms, struggles to escape local minima, resulting in slow convergence.

6.2.2. Scenario 2: Measuring Uncovered Area Using 14 Nodes, Figure 5b.

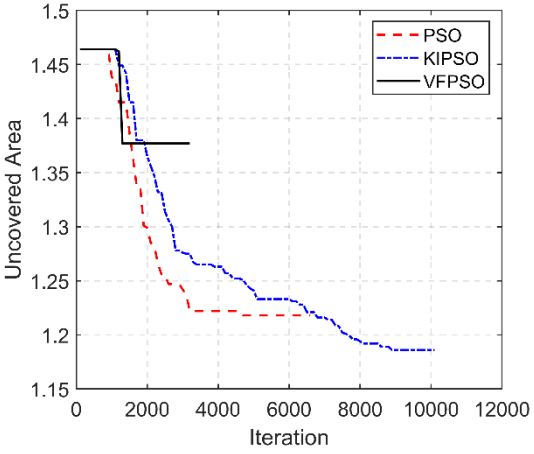
With 14 nodes, the optimization problem becomes slightly easier; however, the need for precise placement to minimize interference remains critical. KIPSO again achieves rapid early improvements by iteration 3000–4000, the inertia dynamically rises if improvement stalls, allowing particles to escape poor local optima, a feature absent in both PSO and VFPSO. VFPSO fast converges, but it trades accuracy for speed, leading to high uncovered areas, whereas PSO is still the least effective because of poor adaptability.



(a)

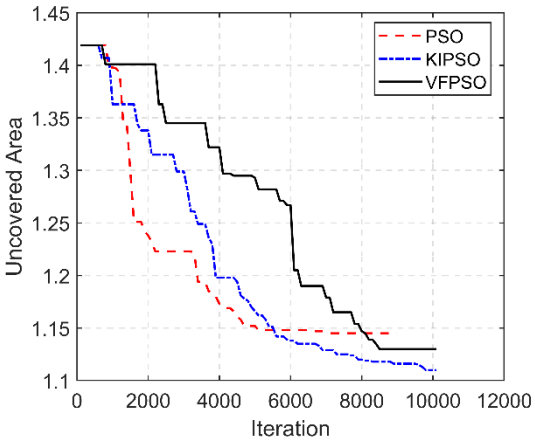


(b)



(c)

RESEARCH ARTICLE



(d)

Figure 5 Measuring Uncovered Area Versus Iterations Using Different Node Counts: (a) 12, (b) 14, (c) 16, and (d) 18 Nodes

6.2.3. Scenario 3: Measuring Uncovered Area Using 16 Nodes, Figure 5c.

With 16 nodes, there's more flexibility in achieving better coverage, but interference between nodes becomes a more significant factor. It is shown that KIPSO dynamically adjusts inertia to avoid local minima and refine node placement, achieving better coverage and minimizing interference. VFPSO converges very fast, but cannot adapt as dynamically as KIPSO, leading to suboptimal performance and struggle in local minima because the problem is in a transitional state, where the coverage is constrained and the interference becomes significant. Again, PSO lacks the mechanisms to adapt, resulting in the slowest and least effective performance.

6.2.4. Scenario 4: Measuring Uncovered Area Using 18 Nodes, Figure 5d.

In this scenario, the challenge shifts more towards managing node interference rather than just maximizing coverage. As shown in the figure, the Kalman filter estimates inertia based on cumulative trends and noise covariance, allowing it to slow convergence gracefully and avoid overshooting. VFPSO speed-first helps convergence but at the cost of fine-tuning node placement, leaving it less effective in handling interference and achieving optimal coverage, while PSO, suffers from both slow convergence and final coverage.

7. CONCLUSION

This research introduces the Kalman-Inertia Particle Swarm Optimization (KIPSO) algorithm, a novel methodology that optimizes stochastic wireless sensor network deployment by minimizing blind spot formation, signal interference, and coverage redundancy. Based on the Kalman filter estimation accuracy, the proposed algorithm dynamically estimates the inertia of particles in the PSO algorithm, improving convergence and optimization. The proposed algorithm demonstrates superior efficacy in balancing exploration-exploitation dynamics compared to conventional Particle Swarm Optimization. Based on the results obtained, KIPSO is found to improve node deployment efficiency compared to standard PSO and VFPSO, which leads to more reliable coverage, reduced energy consumption, and a longer life expectancy for the network.

A careful and studied selection of experimental settings contributed to achieving a crucial balance between stability and adaptability. This revealed that the KIPSO algorithm was able to explore effectively in underpopulated areas, gradually improving coverage while maintaining stability. KIPSO dynamically adjusts inertia to avoid local minima and refine node placement, achieving better coverage and minimizing interference, compared to PSO and VFPSO algorithms. Empirical results confirm that elevated nodal density within the experimental region substantially enhances both coverage ratio and localization accuracy under KIPSO deployment. KIPSO uses a Kalman filter to optimize the entire monitoring area as one search region. Future extensions of KIPSO can be accomplished by dividing the optimization area into regions and utilizing localized Kalman filters for each region. As a result, node placement can be made more accurate by concentrating on smaller subproblems while maintaining overall efficiency.

REFERENCES

[1] Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks : a survey," Comput. networks, vol. 38, no. 4, pp. 393–422, 2002.

[2] W. Dargie and C. Poellabauer, Fundamentals of Wireless Sensor Networks: Theory and Practice, 1st ed., no. January. West Sussex, PO19 8SQ, United Kingdom: John Wiley and Sons, 2010.

[3] C. Buratti, A. Conti, D. Dardari, and R. Verdone, "An overview on wireless sensor networks technology and evolution," Sensors, vol. 9, no. 9, pp. 6869–6896, 2009.

[4] A. A. Al-Shammaa and A. J. Stocker, "Distributed clusters classification algorithm for indoor wireless sensor networks using pre-defined knowledge-based database," in SAS 2019 - 2019 IEEE Sensors Applications Symposium, Conference Proceedings, 2019, pp. 1–6, doi: 10.1109/SAS.2019.8706099.

[5] A. A. Al-Shammaa and A. J. Stocker, "Discovering neighbour nodes based on signal strength using Waspmote nodes," in Proceedings of IEEE Sensors, 2019, vol. 2019-Octob, pp. 1–4, doi: 10.1109/SENSORS43011.2019.8956706.

[6] L. M. Borges, F. J. Velez, and A. S. Lebres, "Survey on the characterization and classification of wireless sensor network applications," IEEE Commun. Surv. Tutorials, vol. 16, no. 4, pp. 1860–1890, 2014.

[7] J. Liang, L. Wang, and Q. Ji, "A Particle Swarm Optimization Algorithm for Deployment of Sensor Nodes in WSN Network," J. Electr. Comput. Eng., vol. 2022, no. 1, pp. 1–12, 2022, doi: 10.1155/2022/1270029.

[8] Y. Wang, H. Liao, and H. Hu, "Wireless sensor network deployment using an optimized artificial fish swarm algorithm," in Proceedings - 2012 International Conference on Computer Science and Electronics Engineering, ICCSEE 2012, 2012, vol. 2, pp. 90–94, doi:

RESEARCH ARTICLE

10.1109/ICCSEE.2012.453.
[9] X. Liu and D. He, "Ant colony optimization with greedy migration mechanism for node deployment in wireless sensor networks," J. Netw. Comput. Appl., vol. 39, no. 1, pp. 310–318, 2014, doi: 10.1016/j.jnca.2013.07.010.
[10] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in ICNN'95-international conference on neural networks, 1995, pp. 1942–1948, doi: 10.1007/978-3-031-17922-8_4.
[11] Xin-She Yang, "Chapter 7 - Particle Swarm Optimization," in Nature-Inspired Optimization Algorithms, Elsevier, 2014, pp. 99–110.
[12] X. Wang, S. Wang, and J. J. Ma, "An improved co-evolutionary particle swarm optimization for wireless sensor networks with dynamic deployment," Sensors, vol. 7, no. 3, pp. 354–370, 2007, doi: 10.3390/s7030354.
[13] X. Lu, W. Cheng, Q. He, J. Yang, and X. Xie, "Coverage optimization based on improved firefly algorithm for mobile wireless sensor networks," in 2018 IEEE 4th International Conference on Computer and Communications, ICCC 2018, 2018, pp. 899–903, doi: 10.1109/CompComm.2018.8780713.
[14] F. Xue, Y. Cai, and Z. Cui, "Bacterial foraging optimization algorithm for coverage problem in wireless sensor network.," Sens. Lett., vol. 12, no. 1, pp. 160–163, 2014.
[15] K. S. Low, H. A. Nguyen, and H. Guo, "A particle swarm optimization approach for the localization of a wireless sensor network," in IEEE International Symposium on Industrial Electronics, 2008, pp. 1820–1825, doi: 10.1109/ISIE.2008.4677205.
[16] R. WANG and G. LIU, "Wireless sensor networks deployment based on fish-swarm optimization algorithm," J. Vib. Shock, vol. 28, no. 2, pp. 8–11, 2009.
[17] L. Ke-qing, "Coverage optimization of wireless sensor networks based on artificial fish swarm algorithm.," Appl. Res. Comput., vol. 20, no. 2, pp. 554–556, 2013.
[18] J. Tian, M. Gao, and G. Ge, "Wireless sensor network node optimal coverage based on improved genetic algorithm and binary ant colony algorithm," Eurasip J. Wirel. Commun. Netw., vol. 2016, no. 1, pp. 2–11, 2016, doi: 10.1186/s13638-016-0605-5.
[19] J. Izquierdo, I. Montalvo, R. Pérez, and V. S. Fuertes, "Design optimization of wastewater collection networks by PSO," Comput. Math. with Appl., vol. 56, no. 3, pp. 777–784, 2008, doi: 10.1016/j.camwa.2008.02.007.
[20] W. Guo, G. C. Chen, and J. S. Yu, "The Kalman Particle Swarm Optimization algorithm and its application in soft-sensor of acrylonitrile yield," in Proceedings of 2005 International Conference on Neural Networks and Brain Proceedings, ICNNB'05, 2005, vol. 1, pp. 124–127, doi: 10.1109/icnnb.2005.1614581.
[21] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360), 1998, pp. 69–73, doi: 10.1109/ICEMI.2007.4350772.
[22] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," IEEE Trans. Evol. Comput., vol. 6, no. 1, pp. 58–73, 2002, doi: 10.1016/0921-8734(92)90002-7.
[23] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," IEEE Trans. Evol. Comput., vol. 8, no. 3, pp. 240–255, 2004, doi: 10.1109/TEVC.2004.826071.
[24] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," IEEE Trans. Syst. Man, Cybern. Part B, vol. 39, no. 6, pp. 1362–1381, 2009, doi: 10.25103/jestr.062.37.
[25] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," Appl. Soft Comput. J., vol. 11, no. 4, pp. 3658–3670, 2011, doi: 10.1016/j.asoc.2011.01.037.
[26] Z. D. Zaharis et al., "Exponential Log-Periodic Antenna Design Using Improved Particle Swarm Optimization with Velocity Mutation," IEEE Trans. Magn., vol. 53, no. 6, pp. 2015–2018, 2017, doi: 10.1109/TMAG.2017.2660061.
[27] X. Yan, F. He, N. Hou, and H. Ai, "An efficient particle swarm optimization for large-scale hardware/software co-design system," Int. J. Coop. Inf. Syst., vol. 27, no. 1, pp. 1–28, 2018.
[28] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," IEEE Trans. Syst. Man, Cybern. Part B Cybern., vol. 42, no. 3, pp. 627–646, 2012, doi: 10.1109/TSMCB.2011.2171946.
[29] M. R. Tanweer, S. Suresh, and N. Sundararajan, "Self regulating particle swarm optimization algorithm," Inf. Sci. (Ny)., vol. 294, no. September, pp. 182–202, 2015, doi: 10.1016/j.ins.2014.09.053.
[30] X. Wang, S. Wang, and D. Bi, "Virtual force-directed particle swarm optimization for dynamic deployment in wireless sensor networks," in Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues: Third International Conference on Intelligent Computing, ICIC 2007 Qingdao, China, August 21-24, 2007, vol. 3, pp. 292–303, doi: 10.1007/978-3-540-74171-8_29.
[31] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," in IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428), San Francisco, CA, USA, 2003, vol. 2, pp. 1293–1303, doi: 10.1109/infcom.2003.1208965.
[32] L. Wu, J. Qu, H. Shi, and P. Li, "Node Deployment Optimization for Wireless Sensor Networks Based on Virtual Force-Directed Particle Swarm Optimization Algorithm and Evidence Theory," Entropy, vol. 24, no. 11, pp. 1–15, 2022, doi: 10.3390/e24111637.
[33] Y. Li and J. Cao, "Adaptive Binary Particle Swarm Optimization for WSN Node Optimal Deployment Algorithm," IECE Trans. Internet Things, vol. 1, no. 1, pp. 1–8, 2024, doi: 10.62762/tiot.2023.564457.
[34] X. Liu, K. Zhang, X. Zhang, G. Fiumara, and P. De Meo, "A hybrid improved compressed particle swarm optimization WSN node location algorithm," Phys. Commun., vol. 67, no. December, p. 102490, 2024, doi: https://doi.org/10.1016/j.phycom.2024.102490.
[35] K. V. N. A. Bhargavi, G. P. S. Varma, I. Hemalatha, and R. Dilli, "An Enhanced Particle Swarm Optimization-Based Node Deployment and Coverage in Sensor Networks," Sensors, vol. 24, no. 19, pp. 1–22, 2024, doi: 10.3390/s24196238.
[36] D. A. Amer, S. A. Soliman, A. F. Hassan, and A. A. Zamel, Enhancing connectivity and coverage in wireless sensor networks: a hybrid comprehensive learning-Fick's algorithm with particle swarm optimization for router node placement, vol. 36, no. 34. Springer London, 2024.
[37] G. Siamantas and D. Kandris, "Particle Swarm Optimization for k-Coverage and 1-Connectivity in Wireless Sensor Networks," Electron., vol. 13, no. 23, 2024, doi: 10.3390/electronics13234841.
[38] R. E. Kalman, "A new approach to linear filtering and prediction problems," J. Basic Eng. Trans. ASME, vol. 82, no. 1, pp. 35–45, 1960, doi: 10.1115/1.3662552.
[39] M. Petrova, J. Riihijärvi, P. Mähönen, and S. Labella, "Performance study of IEEE 802.15.4 using measurements and simulations," in IEEE Wireless Communications and Networking Conference, WCNC, 2006, vol. 1, pp. 487–492, doi: 10.1109/wcnc.2006.1683512.

Authors

Ameer A. Al-Shammaa received a PhD degree in Computer Engineering (2020) from the School of Engineering - Aerospace & Computational Engineering (ACE) Research Group at the University of Leicester, Leicester, UK.

Currently, he is a Director Assistant at the Information Technology Research and Development Center of the University of Kufa, Al-Najaf, Iraq. Also, he is a lecturer in the Computer Science Department at the Faculty of Education for

RESEARCH ARTICLE

Women at the University of Kufa, Iraq. Mr. AL-SHAMMAA is a member of the International Union of Radio Science (URSI)-Belgium (2019), the International Association of Engineers-Hong Kong (2012) and the International Association of Computer Science and Information Technology (IACSIT)-Singapore (2012). He was also awarded a student travel grant for his paper presented at the 2019 IEEE Sensor Applications Symposium in France. His research interests include improving event reliability in wireless sensor networks, wireless communications, the Internet of Things (IoT), VANETs, and networking algorithms design.

Hadeel N. Saad is a full professor at the Faculty of Education for Women, University of Kufa, Al-Najaf, Iraq. She received her Ph.D. degree from the Iraqi Commission for Computers and Informatics (ICCI), Informatics Institute for Higher Education, Iraq, in 2007. Her current research interests include the technology and application of the WSN.

Abbas G. Jreou received an MSc. Degree in Cybersecurity from the College of Information Technology, Department of Information Security, 2023. He is currently employed at Al-Mustaqbal University, Babylon, 51001, Iraq, in the Technology Incubator Division. His research interests are in the fields of computer security, hacking, networks, and programming.

How to cite this article:

Ameer A. Al-Shammaa, Hadeel N. Saad, Abbas G. Jreou, "KIPSO: Kalman-Inertia Particle Swarm Optimization Approach for Wireless Sensor Nodes Deployment", International Journal of Computer Networks and Applications (IJCNA), 12(3), PP: 418-431, 2025, DOI: 10.22247/ijcna/2025/26.