International Journal of Engineering Research and Development

DOI: 10.29137/ijerad.1520344

Volume:17 Issue:02 Pages:299-308 July/2025

Research Article

Localization in Wireless Sensor Networks Using Metaheuristic Algorithms

Buğra Hatipoğlu* [1], **Tolga Eren** [1], **Murat Lüy** [1]

[1]Kırıkkale University, Faculty of Engineering and Natural Sciences, Department of Electrical and Electronics Engineering, 71451, Yahşihan, Kırıkkale, Türkiye

Received: 25/07/2024 **Accepted**: 21/12/2024 **Published Online:** 15/07/2025
Final Version: 01/07/2025

Abstract

Wireless sensor networks are utilized in a wide range of applications, where accurate determination of node positions is critical for network performance and energy efficiency. Metaheuristic algorithms, which have replaced traditional methods, offer significant advantages by providing more effective and faster solutions. In this study, Particle Swarm Optimization (PSO), Whale Optimization Algorithm (WOA), Monarch Butterfly Optimization Algorithm (MBOA), and Coati Optimization Algorithm (COA) were used to determine the positions of nodes in wireless sensor networks. The parameters of the proposed algorithms were determined using grid-search hyperparameter optimization. With the obtained optimal parameter values, the average error values of the metaheuristic algorithms were compared, and the results were observed. The results allow for evaluating the performance of the used algorithms and selecting the most suitable method.

Keywords

Wireless sensor networks, Coati optimization algorithm, Localization, Hyperparameter optimization

* *Corresponding Author: bugrahatipoglu@kku.edu.tr*

1. Introduction

Wireless Sensor Networks (WSNs) are systems composed of numerous sensor nodes that can communicate wirelessly with each other and are typically used to monitor various environmental and physical conditions and transmit this information to a central point. These networks are widely used in many fields such as health monitoring, environmental control, smart agriculture, military applications, and industrial automation. Determining the optimal positions of the sensor nodes is of great importance for the overall performance and energy efficiency of the network. Proper positioning enhances the accuracy of data collection while reducing energy consumption, thereby extending the network's lifespan.

Traditional localization methods often have certain limitations, such as high computational costs and low accuracy rates. Therefore, in recent years, metaheuristic algorithms have been increasingly used to provide effective and efficient solutions for determining node positions in WSNs. Metaheuristic algorithms are techniques inspired by biological and natural processes and are designed to offer fast and accurate solutions to complex optimization problems.

There are numerous studies in the literature on localization in WSNs. In their study, (Rajakumar et al., 2017) estimated node positions in WSNs using Particle Swarm Optimization, Modified Bat Algorithm, and Grey Wolf Optimizer by employing varying numbers of reference nodes within a 300x300 unit area. When comparing simulation results based on parameters such as computation time, percentage of localized nodes, and minimum localization error, the Grey Wolf Optimizer stood out in terms of performance. (Kanoosh et al., 2019) compared various metaheuristic algorithms with the Salp Swarm Algorithm. They conducted simulations to determine the positions of different numbers of nodes using different numbers of reference nodes within a 100x100 area. The most successful results were obtained using 25 target nodes and 10 reference nodes, achieving an error rate of 0.451. (Gou et al., 2021) developed a node localization model for WSNs based on an enhanced Whale Optimization Algorithm. Simulation results demonstrated that this algorithm outperformed the original RSSI algorithm, HPSO, and WOA-QT under the same conditions. (W. Wang et al., 2019) proposed an equal-arc triangle placement algorithm based on Received Signal Strength Indicator (RSSI), aiming to improve measurement accuracy and the placement of beacon nodes. Compared to the traditional triangular placement algorithm, the equal-arc configuration increased the area coverage ratio by 23%. (Mohar et al., 2021) conducted studies to enhance the coverage area of sensor nodes in order to improve simulation performance. By optimizing node distribution using the Bat Algorithm, they aimed to maximize the coverage area relative to the total area. Results showed that the Bat Optimization Algorithm achieved higher coverage values compared to other algorithms.

In this study, Particle Swarm Optimization (PSO), Whale Optimization Algorithm (WOA), Monarch Butterfly Optimization (MBO), and Coatí Optimization Algorithm (COA) were used to determine node positions in Wireless Sensor Networks (WSNs). The simulations were carried out using the Python programming language. The aim of the study is to compare the performance of these algorithms and identify the most suitable method. In this context, the parameters of the proposed algorithms were determined using grid search hyperparameter optimization, and the results were observed by comparing the average error values of the algorithms based on the obtained optimal parameter settings.

2. Localization in Wireless Sensor Networks

Localization in Wireless Sensor Networks (WSNs) is used to determine the positions of sensor nodes whose locations are unknown. The accuracy of localization depends on parameters such as the distribution of sensors within the area, the search method, and the transmission radius. The position of a new sensor located at the intersection point of three known sensors can be determined. However, the sensor to be localized must be within the transmission range of the other sensors. Figure 1 illustrates an example of localization in WSNs.

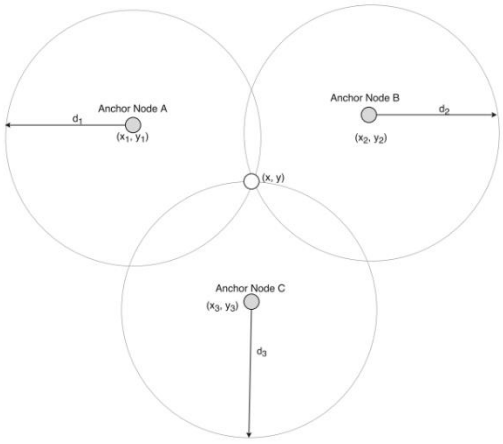

Figure 1. Localization in Wireless Sensor Networks (Arora & Singh, 2017)

In the localization process of WSNs, it is assumed that there are N reference nodes and M target nodes. The positions of the reference nodes are denoted as (xi, yi), the estimated positions as (Xj, Yj), and the actual positions of the target nodes as (x, y). The distance between the target nodes and the reference nodes is represented as di.

$$d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2} \tag{1}$$

Equation 1 presents the actual distance formula. By adding a noise value *(n)* to the actual distance, the estimated (or approximate) distance is calculated. Equation 2 shows the approximate distance formula.

$$d_{in} = d_i + n \tag{2}$$

In WSNs, the objective is to minimize the difference between the estimated positions and the actual positions as much as possible. The objective function is given in Equation 3.

$$f(x, y) = \frac{1}{N}(\sum_{i=1}^{N} \sqrt{(x - x_i)^2 + (y - y_i)^2} - d_{in})^2 \tag{3}$$

When the maximum number of iterations is reached, the error of the position estimates is calculated. The error is expressed as the mean squared error between the estimated node coordinates and the actual node coordinates, as shown in Equation 4 (Sekhar et al., 2021).

$$E = \frac{1}{N}\sum_{i=1}^{N} \sqrt{(x_i - X_i)^2 + (y_i - Y_i)^2} \tag{4}$$

3. Metaheuristic Algorithms

Metaheuristic algorithms are heuristic methods used to find optimal or near-optimal solutions in complex and large search spaces. These algorithms are not specific to a particular problem class and can be applied to various optimization problems, making them highly generalizable. Unlike local search techniques, metaheuristic algorithms have the potential to find the global optimum and are utilized across a wide range of applications.

Metaheuristic algorithms are developed by drawing inspiration from the behaviors of living organisms and natural processes. For example, Genetic Algorithms (GA) are based on the principles of natural selection and genetic evolution, while Ant Colony Optimization (ACO) models how ants find the shortest path by using the pheromone trails they leave during food searching.

3.1. Particle Swarm Optimization Algorithm

Particle Swarm Optimization (PSO) is a population-based metaheuristic algorithm inspired by the behaviors of natural communities. Originally proposed by James Kennedy and Russell Eberhart in 1995, PSO draws inspiration from the collective movements of natural groups such as bird flocks and fish schools (Kennedy & Eberhart, 1995).

In PSO, each candidate solution is called a particle, and these particles move within a multidimensional search space. Each particle updates its position and velocity in an attempt to find a better solution in the search space. Particles move by considering both their own best position (pBest) and the best position found by the entire swarm (gBest). This mechanism allows particles to benefit from both individual experience and collective knowledge.

Each particle i exists in a d-dimensional solution space. The parameter $\vec{x}_i$ represents the position vector of particle i in the d-dimensional solution space. $\vec{v}_i$ is the velocity vector of the particle. $\vec{p}_i$ denotes the best position particle i has achieved so far, while $\vec{g}$ represents the best position found by any particle in the entire swarm. In the PSO algorithm, at each iteration, the velocities and positions of the particles are updated. The velocity update equation is given in Equation 5.

$$\vec{v}_i(t+1) = w \cdot \vec{v}_i(t) + c_1 \cdot r_1(\vec{p}_i - \vec{x}_i(t)) + c_2 \cdot r_2 \cdot (\vec{g}_i - \vec{x}_i(t)) \tag{5}$$

In this equation, the parameter www is the inertia weight, which represents the tendency of a particle to maintain its previous velocity. The parameters c_1 and c_2 are acceleration coefficients; c_1 is the coefficient derived from the particle's individual experience, while c_2 is the coefficient derived from the experiences of other particles. r_1 and r_2 are random values between 0 and 1. The position update equation is given in Equation 6.

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \quad (6)$$

After the positions and velocities of the particles are updated according to the equations in Equation 5 and Equation 6, the fitness value of each particle is calculated, and the personal best $\vec{p}_i$ and the global best $\vec{g}_i$ values are updated. If the fitness value at a particle's current position is better than its own $\vec{p}_i$, this condition is checked, and if the current position is better, the $\vec{p}_i$ value is updated. Then, each particle's $\vec{p}_i$ value is compared with the swarm's best position $\vec{g}_i$. If any $\vec{p}_i$ is better than $\vec{g}_i$, the global best $\vec{g}_i$ is updated accordingly.

3.2. Whale Optimization Algorithm

The Whale Optimization Algorithm (WOA) is a nature-inspired metaheuristic optimization algorithm developed by Mirjalili and Lewis in 2016. This algorithm is designed based on the hunting behavior of humpback whales. Humpback whales hunt by encircling fish schools and performing spiral movements around them. This behavior forms the foundation of the whale optimization algorithm (Mirjalili & Lewis, 2016).

Algoritma, çözüm uzayında rastgele bir popülasyon oluşturularak başlar. Popülasyon sayısı ve iterasyon sayısı belirlenir. Balinalar hedef avın (en iyi çözümün) etrafını sarar. Matematiksel olarak, balinanın mevcut konumu ve avın konumu arasındaki mesafeyi azaltmaya yönelik Eş 7. ve Eş 8'deki denklemleri kullanır.

$$\vec{D} = \left|\vec{C}.\vec{X}_{best}(t) - \vec{X}(t)\right| \quad (7)$$

$$\vec{X}(t+1) = \vec{X}_{best}(t) - \vec{A} \cdot \vec{D} \quad (8)$$

Here, $\vec{X}_{best}(t)$ represents the position of the best solution, and $\vec{X}(t)$ denotes the position of the current whale. The vectors $\vec{A}$ and $\vec{C}$ are parameters that control the search process and take random values. The calculations of these parameters are shown in Equations 9 and 10.

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \quad (9)$$

$$\vec{C} = 2 \cdot \vec{r} \quad (10)$$

In the equation, the parameter $\vec{a}$ decreases linearly from 2 to 0. The parameter $\vec{r}$ is a random vector with values between 0 and 1. Subsequently, whales capture their prey by following a spiral trajectory and creating bubble nets. This behavior is modeled using spiral equations. Equation 11 shows the corresponding equation used.

$$\vec{D}' \cdot e^{bl} \cdot \cos(2\pi l) \cdot \vec{X}_{best}(t) \;\; eğer\; \alpha \geq 0.5 \quad (11)$$

$\vec{D}'$ represents the distance between the whale and its prey. The parameter b is a constant that defines the shape of the spiral. The parameter l is a random number between -1 and 1. Whales move randomly while searching for potential prey. This phase occurs when $A \geq 1$.

$$\vec{X}(t+1) = \vec{X}_{rand}(t) - \vec{A} \cdot \vec{D} \quad (12)$$

The parameter $\vec{X}_{rand}(t)$ represents the position of a randomly selected whale at time t. If $A < 1$, whales update their positions based on a probability value α. The equation dependent on α is given in Equation 13.

$$\vec{X}(t+1) = \begin{cases} \vec{X}_{best}(t) - \vec{A} \cdot \vec{D} \;\; eğer\; \alpha < 0.5 \\ \vec{D}' \cdot e^{bl} \cdot \cos(2\pi l) \cdot \vec{X}_{best}(t) \;\; eğer\; \alpha \geq 0.5 \end{cases} \quad (13)$$

3.3. Monarch Butterfly Optimization Algorithm

The Monarch Butterfly Optimization Algorithm (MBO) is a nature-inspired metaheuristic algorithm that mimics the migratory behavior of monarch butterflies (G. G. Wang et al., 2019). Monarch butterflies migrate over long distances across North America. This migration occurs between two main groups: the migrating group (G1) and the resident group (G2). It is assumed that the butterflies in the population belong to one of these two groups, and the total number of butterflies remains constant. For each butterfly, an offspring is generated through a migration operator regardless of whether the parents are in region 1 or region 2. Since the population size must remain constant, either the parent or the offspring is eliminated based on the fitness function. Butterflies selected according to the fitness function are transferred to the next generation and are not altered by the migration operator. The MBO algorithm imitates this

migratory behavior by dividing the solutions (butterflies) into two subgroups and facilitating information exchange between these groups at each iteration. The equation for butterflies in G1 is given in Equation 14.

$$x_i^{t+1} = x_i^t + F.c.(x_{best}^t - x_i^t) + \Delta x \tag{14}$$

In this equation, x_i^t represents the position of the i butterfly at iteration t. x_{best}^t is the position of the best solution (butterfly) at iteration t. F is a parameter that controls the position updates. Δx represents the displacement caused by migration. The equation for the butterflies in group G2 is given in Equation 15.

$$x_j^{t+1} = x_j^t + rand.a.\left(x_{best}^t - x_j^t\right) \tag{15}$$

In this equation, x_j^t represents the position of the j butterfly at iteration t. The rand parameter is a randomly generated value between 0 and 1. The migration model determines whether butterflies migrate from G1 to G2 or vice versa. The migration probability p governs the decision of the butterfly to relocate. If rand < p, the butterfly migrates between G1 and G2. In each iteration, the fitness value of the butterflies is calculated. The fitness is evaluated according to the defined fitness function, and each butterfly's fitness value and best position are updated accordingly.

3.4. Coati Optimization Algorithm

Coati Optimization Algorithm (COA) is a nature-inspired metaheuristic optimization algorithm, developed by taking inspiration particularly from the social behaviors of coatis. Coatis are social and curious animals that live in the tropical regions of South America. This algorithm mimics the behavior of coatis moving in groups and searching for food (Dehghani et al., 2023). Coatis move in groups, and within each group, a leader coati is selected. The leader coati represents the best solution in the group, and other coatis follow this leader. In each group, the coati that finds the best solution is chosen as the leader. The leader coati is determined based on the fitness function. The follower coatis follow the leader coati and move around the best solution found by it. The positions of the follower coatis are updated according to the position of the leader coati. The equation for the leader coati is given in Equation 16.

$$x_{leader}^{t+1} = x_{leader}^t + F.(x_{best}^t - x_{leader}^t) \tag{16}$$

In Equation 16, the parameter x_{leader}^t represents the position of the leader coati at iteration t, while x_{best}^t denotes the position of the best coati (solution) at the same iteration. F is a parameter that controls the position updates. The equation for the follower coati is given in Equation 17.

$$x_i^{t+1} = x_i^t + rand.(x_{lider}^t - x_i^t) \tag{17}$$

In Equation 17, x_i^t represents the position of the i follower coati at iteration t, while rand is a randomly generated value between 0 and 1.

4. Grid Search Hyperparameter Optimization

The selection of appropriate hyperparameters is critically important for improving the performance of machine learning and deep learning models. In particular, the success of metaheuristic algorithms largely depends on the correct choice of parameters. Hyperparameter optimization refers to a set of methods used to find the most suitable combination of hyperparameters in this process. Grid Search Hyperparameter Optimization is a widely used approach that offers a systematic search strategy for this purpose (Bergstra & Bengio, 2012).

Grid Search Hyperparameter Optimization is a method that systematically tries every possible combination within a specified hyperparameter range to find the set that provides the best performance. Grid Search is typically applied over a limited hyperparameter space and guarantees that all combinations are tested. First, the parameters to be optimized are identified. Then, the value ranges and step intervals for these parameters are determined. After testing all combinations, model performance is evaluated, and the parameters that yield the best performance are selected.

5. Simulation Study

In this study, a simulation for localization in wireless sensor networks was conducted. Four reference nodes are placed at the corners of a 100x100 unit area, and 50 sensors are randomly distributed within this area. The locations of the reference nodes are (0,0), (0,100), (100,0), and (100,100). The noise level is set to 0.5, representing the random error or uncertainty in the measurements. The positions of these sensors were estimated using metaheuristic optimization algorithms. Subsequently, the estimated locations were compared

with the actual positions, and the results were illustrated on a figure. Grid-search method was employed for hyperparameter optimization. The simulation was implemented in the Python programming environment.

The parameters used in the Particle Swarm Optimization method are given in Table 1.

Table 1. Particle Swarm Optimization Algorithm

Parameters	Values
Number of Sensor Nodes	50
Number of Iterations	100
Transmission Radius	22
Inertia Weight	1
Inertia Weight Damping Rate	0.95
Cognitive Coefficient (c1)	2
Social Coefficient (c2)	2.5
Noise Level	0.5

After hyperparameter optimization, the simulation was carried out using the selected parameters. The inertia weight is a factor that controls the influence of particles' previous velocities on their velocity updates. A high inertia weight encourages particles to maintain their current velocities, helping them explore broadly across the search space. Conversely, a low inertia weight reduces the particles' current velocities, enabling a more precise search around local solutions. The rate at which the inertia weight decreases in each iteration is called the inertia weight damping rate. This value gradually slows down the particles' velocities as iterations progress, allowing the algorithm to explore widely in the early stages and focus on narrower regions in later stages. This balance helps the algorithm to find better solutions. The cognitive coefficient (c1) determines the weight of the updates based on the particles' own best positions, while the social coefficient (c2) represents the weight that guides particles toward the best position found by the entire swarm. A higher social coefficient increases the tendency of particles to move towards the swarm's best solution. The average positional error was measured as 5.31 units. The error values along with the results comparing actual vs. predicted positions and neighborhood connectivity are presented in Figure 2.

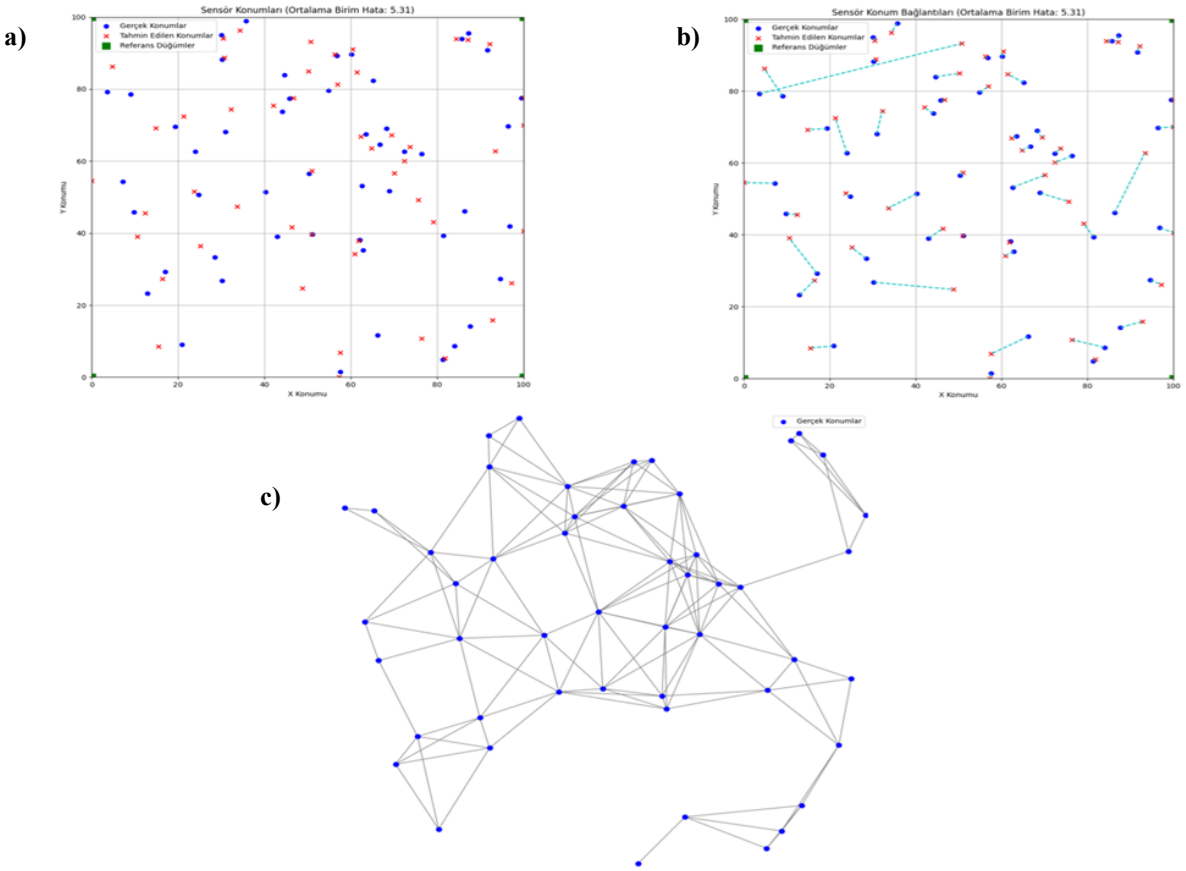

Figure 2. (a) Error Values for PSO; **(b)** Actual vs. Predicted Connections for PSO; **(c)** Actual Position Neighborhood Connections for PSO

Table 2. Whale Optimization Algorithm

Parameters	Values
Number of Sensor Nodes	50
Number of Iterations	100
Transmission Radius	22
Spiral Movement (b)	1.5
Control Parameter (a)	0.5
Population Size	30
Noise Level	0.5

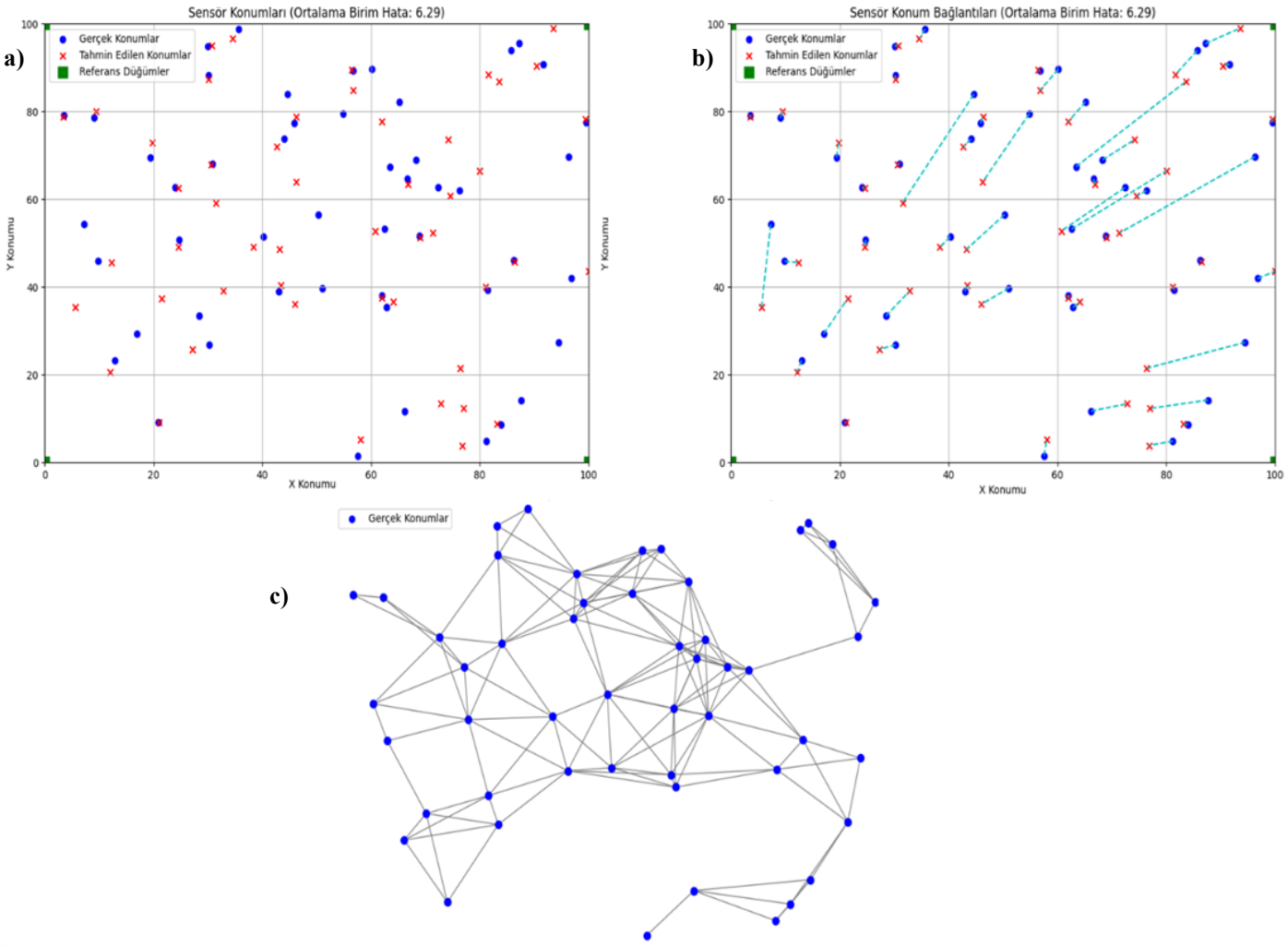

Figure 3. (a) Error Value for WOA; **(b)** Actual vs. Predicted Connection for WOA; **(c)** Actual Location Neighborhood Connection for WOA

The whale optimization parameters are provided in Table 2. The spiral movement parameter (b) is used in the WOA algorithm to model the behavior of humpback whales encircling and capturing their prey through spiral movements. This parameter controls the width or tightness of the spiral movement performed by the whale (solution candidate) to reach the target. A higher spiral movement value increases environmental exploration by allowing wider spiral motions. The control parameter (a) is a regulatory coefficient used to adjust the whales' movement and their distance from the prey. This value starts high and decreases linearly throughout the iterations. Such a decrease helps the algorithm explore a broader solution space in the early stages of the search and focus more on the solution in later stages. The average error value was measured as 6.29 units. The error values, real vs. predicted connections, and neighborhood connection results are presented in Figure 3.

Table 3. Monarch Butterfly Optimization Algorithm

Parameters	Values
Number of Sensor Nodes	50
Number of Iterations	100
Transmission Radius	22
Global Solution Update Coefficient (c)	0.5
Local Solution Update Coefficient (a)	0.1
Migration Probability (p)	0.5
Population Size	20
Noise Level	0.5

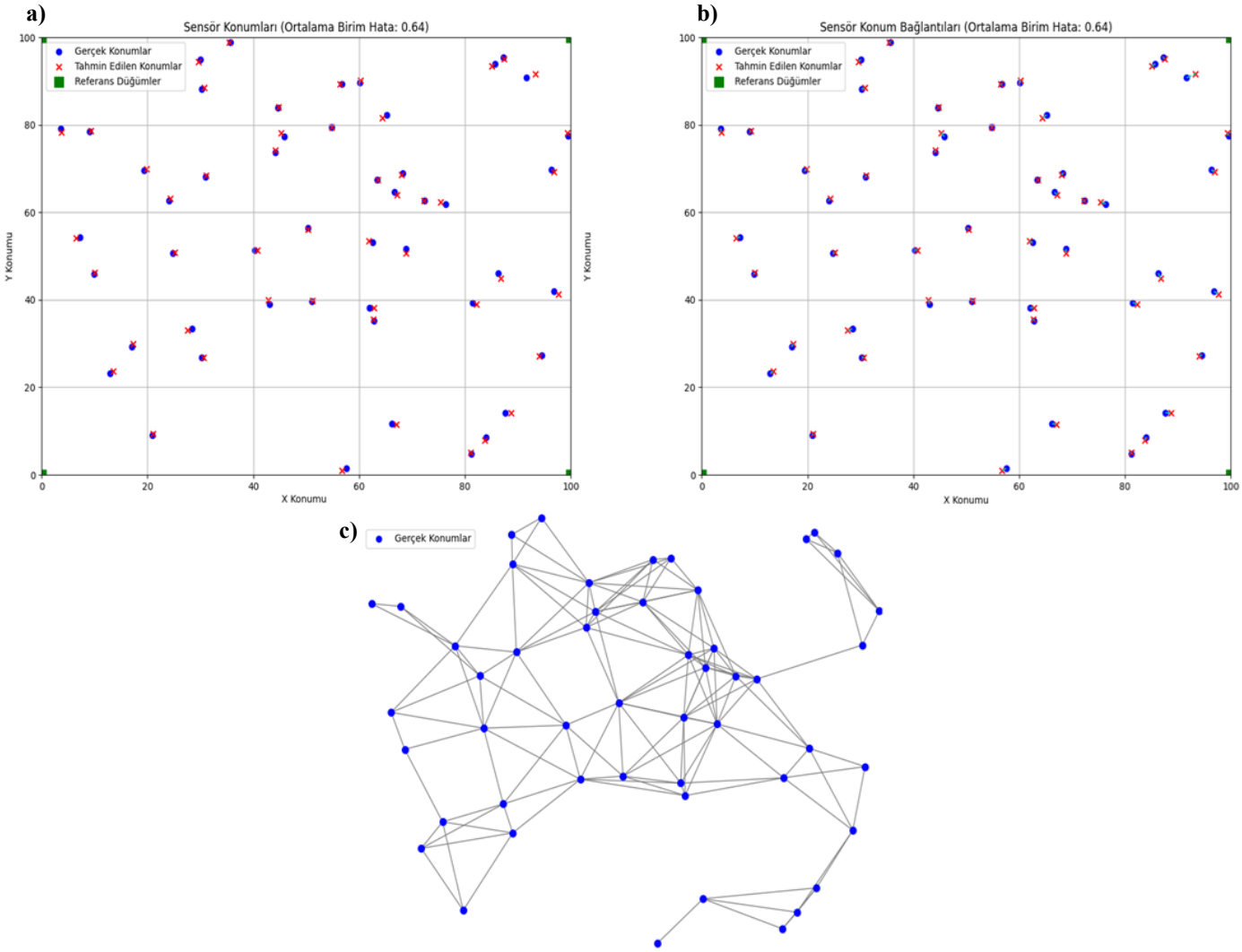

Figure 4. **(a)** Error Value for MBO; **(b)** Actual and Predicted Connections for MBO; **(c)** Actual Position Neighborhood Connections for MBO

The parameters of the Monarch Butterfly Optimization (MBO) algorithm are provided in Table 3. The global solution update coefficient (c) controls the algorithm's tendency to move toward the best solution in the population. This coefficient facilitates reaching the global optimum by enabling the exploration of a wider solution space. A higher value allows the algorithm to focus more on the global solution. The local solution update coefficient (a) controls each butterfly's tendency to move toward the best solution in its local vicinity. This parameter helps the algorithm guide each butterfly closer to the optimal point within its own solution space. A lower value can enhance solution quality by enabling more precise local search. The migration probability (p) defines the likelihood of butterflies switching between different groups within the MBO algorithm. The average error value was measured as 0.64 units. The error values, actual and predicted connections, and neighborhood connection results are presented in Figure 4.

Table 4. Coati Optimization Algorithm

Parameters	Values
Number of Sensor Nodes	50
Number of Iterations	100
Transmission Radius	22
Number of Coatis in a Troop	10
Number of Troops	7
Migration Probability (p)	0.5
Population Size	20
Noise Level	0.5

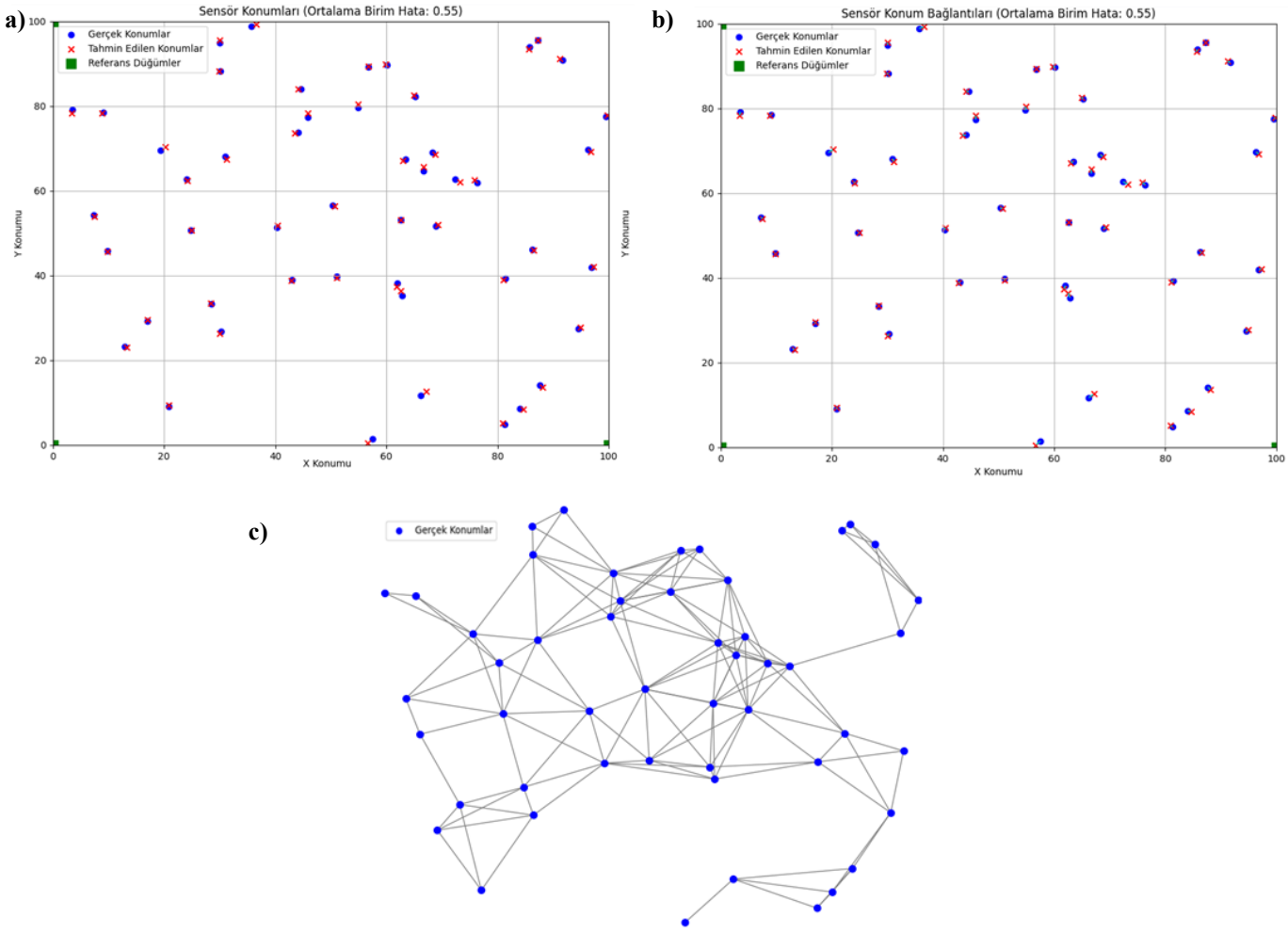

Figure 5. (a) Error Value for COA; **(b)** Actual and Predicted Connections for COA; **(c)** Actual Position Neighborhood Connections for COA

The parameters of the Coati Optimization Algorithm (COA) are provided in Table 4. The number of coatis in a troop indicates the total number of coatis (solution candidates) within each group. This parameter determines the size of each troop; therefore, a higher number of coatis allows for the exploration of a larger solution space. The number of troops represents the total number of coati groups used in the algorithm. In the Coati Optimization Algorithm, coatis move in groups, searching for different solutions, and each group selects its own leader coati. A greater number of troops can help the algorithm explore a broader solution space. The migration probability defines the likelihood of coatis moving between different groups. The average error value was measured as 0.55 units. The error values, actual and predicted connections, and neighborhood connection results are presented in Figure 5.

6. Results and Discussion

In this study, the positions of target sensors in wireless sensor networks were estimated using metaheuristic algorithms. To compare the algorithms, 50 nodes were randomly distributed within a 100x100 unit area, and the target sensors were placed at the same positions for each algorithm. The sensor transmission radius (R) was set to 22 units. The optimization algorithms used were PSO, WOA, MBO, and COA. As seen in Table 1, the PSO method resulted in an average localization error of 5.31 units. In Table 2, the WOA algorithm was used for sensor position estimation, producing an average error of 6.29 units, making it the least successful among the evaluated metaheuristic algorithms. According to Table 3, the MBO algorithm achieved an average error of 0.64 units. Finally, as shown in Table 4, the Coati Optimization Algorithm produced the lowest error, with an average of 0.55 units. For all metaheuristic algorithms, Grid Search hyperparameter optimization was applied to select the most suitable parameter values specific to each algorithm. This approach enabled the algorithms to maintain lower error values. It is worth noting that reducing the step size in Grid Search can lead to lower average error results. However, this also increases the computational load and extends the time required for sensor localization. Therefore, it is crucial to strike a balance between performance and computational cost during hyperparameter tuning. Another way to enhance the success of the study would be to use adaptive learning-based algorithms or hybrid metaheuristic methods. Additionally, testing the algorithms under different node densities and transmission radius would be beneficial for evaluating their flexibility and generalization capabilities.

References

Arora, S., & Singh, S. (2017). Node Localization in Wireless Sensor Networks Using Butterfly Optimization Algorithm. Arabian Journal for Science and Engineering, 42(8), 3325–3335. https://doi.org/10.1007/s13369-017-2471-9

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(2), 281–305.

Dehghani, M., Montazeri, Z., Trojovská, E., & Trojovský, P. (2023). Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems. Knowledge-Based Systems, 259, 110011. https://doi.org/10.1016/J.KNOSYS.2022.110011

Gou, P., He, B., & Yu, Z. (2021). A Node Location Algorithm Based on Improved Whale Optimization in Wireless Sensor Networks. Wireless Communications and Mobile Computing, 2021, 1–17. https://doi.org/10.1155/2021/7523938

Kanoosh, H. M., Houssein, E. H., & Selim, M. M. (2019). Salp Swarm Algorithm for Node Localization in Wireless Sensor Networks. Journal of Computer Networks and Communications, 2019, 1–12. https://doi.org/10.1155/2019/1028723

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks, 4, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

Mirjalili, S., & Lewis, A. (2016). The Whale Optimization Algorithm. Advances in Engineering Software, 95, 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008

Mohar, S. S., Goyal, S., & Kaur, R. (2021). Optimized Sensor Nodes Deployment in Wireless Sensor Network Using Bat Algorithm. Wireless Personal Communications, 116(4), 2835–2853. https://doi.org/10.1007/s11277-020-07823-z

Rajakumar, R., Amudhavel, J., Dhavachelvan, P., & Vengattaraman, T. (2017). GWO-LPWSN: Grey Wolf Optimization Algorithm for Node Localization Problem in Wireless Sensor Networks. Journal of Computer Networks and Communications, 2017. https://doi.org/10.1155/2017/7348141

Sekhar, P., Lydia, E. L., Elhoseny, M., Al-Akaidi, M., Selim, M. M., & Shankar, K. (2021). An effective metaheuristic based node localization technique for wireless sensor networks enabled indoor communication. Physical Communication, 48, 101411. https://doi.org/10.1016/j.phycom.2021.101411

Wang, G. G., Deb, S., & Cui, Z. (2019). Monarch butterfly optimization. Neural Computing and Applications, 31(7), 1995–2014. https://doi.org/10.1007/S00521-015-1923-Y/TABLES/7

Wang, W., Liu, X., Li, M., Wang, Z., & Wang, C. (2019). Optimizing Node Localization in Wireless Sensor Networks Based on Received Signal Strength Indicator. IEEE Access, 7, 73880–73889. https://doi.org/10.1109/ACCESS.2019.2920279