entropy

MDPI

Article

An Optimal WSN Node Coverage Based on Enhanced Archimedes Optimization Algorithm

Thi-Kien Dao [1], Shu-Chuan Chu [2], Trong-The Nguyen [1,3,4,*], Trinh-Dong Nguyen [3,4] and Vinh-Tiep Nguyen [3,4]

[1] Fujian Provincial Key Laboratory of Big Data Mining and Applications, Fujian University of Technology, Fuzhou 350118, China; jvnkien@gmail.com
[2] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; scchu0803@sdust.edu.cn
[3] University of Information Technology, Ho Chi Minh City 700000, Vietnam; dongnt@uit.edu.vn (T.-D.N.); tiepnv@uit.edu.vn (V.-T.N.)
[4] Vietnam National University, Ho Chi Minh City 700000, Vietnam
* Correspondence: thent@uit.edu.vn

Abstract: Node coverage is one of the crucial metrics for wireless sensor networks' (WSNs') quality of service, directly affecting the target monitoring area's monitoring capacity. Pursuit of the optimal node coverage encounters increasing difficulties because of the limited computational power of individual nodes, the scale of the network, and the operating environment's complexity and constant change. This paper proposes a solution to the optimal node coverage of unbalanced WSN distribution during random deployment based on an enhanced Archimedes optimization algorithm (EAOA). The best findings for network coverage from several sub-areas are combined using the EAOA. In order to address the shortcomings of the original Archimedes optimization algorithm (AOA) in handling complicated scenarios, we suggest an EAOA based on the AOA by adapting its equations with reverse learning and multidirection techniques. The obtained results from testing the benchmark function and the optimal WSN node coverage of the EAOA are compared with the other algorithms in the literature. The results show that the EAOA algorithm performs effectively, increasing the feasible range and convergence speed.

Keywords: coverage optimization; enhanced Archimedes optimization algorithm; wireless sensor network; optimization approach

Citation: Dao, T.-K.; Chu, S.-C.; Nguyen, T.-T.; Nguyen, T.-D.; Nguyen, V.-T. An Optimal WSN Node Coverage Based on Enhanced Archimedes Optimization Algorithm. *Entropy* **2022**, *24*, 1018. https://doi.org/10.3390/e24081018

Academic Editor: Ernestina Menasalvas

Received: 14 June 2022
Accepted: 19 July 2022
Published: 23 July 2022

1. Introduction

Wireless sensor networks (WSNs) are mainly composed of several autonomous devices called sensor nodes implemented for specific purposes and scattered in wide areas [1,2]. As wireless communication technology has improved and time has passed [3], WSNs have become more common in the information field [4]. They are utilized in various crucial fields, including the military, intelligent transportation, urban planning, industrial and agricultural automation, and environmental monitoring [5]. The sensor node's job is to send captured information to the base station (BS) or the destination node by sensing and collecting ambient data, including sound vibration, pressure, temperature, and light intensity, among other things [6].

Due to their ease of implementation, cheap maintenance costs, and high flexibility, WSNs have successfully replaced wired networks and been embraced in the industrial field in recent years [7]. However, due to the nature of wireless communication, interference and conflict are invariably present during data transmission [8], and data packets may be lost or delayed past their planned deadline [9].

One of the most fundamental difficulties in WSNs is coverage, which is a critical metric for evaluating coverage optimization efforts. Because coverage affects the monitoring capa-

bility of the target monitoring area, it substantially impacts WSNs' quality of service [10]. A node coverage optimization technique has been developed to increase the coverage of wireless sensor nodes in a big-data environment, considering the characteristics of large wireless sensor networks with limited node computing capabilities [11,12]. However, wireless sensor networks' operating environment is complex and changing, and sensor energy is limited and cannot be supplemented [13].

Deployment of sensor nodes that is both sensible and effective reduces network expenses and energy consumption [14]. All WSN coverage applications try to deploy a minimal number of sensor nodes to monitor a defined target region of interest to improve coverage efficiency. Sensor nodes are typically placed randomly in the target monitoring region, resulting in an uneven distribution of nodes and limited coverage [15]. As a result, the network coverage control problem is the central research problem in wireless sensor networks [16]. Adopting an effective and acceptable network coverage control technique is beneficial to optimizing sensor node deployment to increase wireless sensor network performance. Sensor nodes are randomly placed around the monitoring region [17]. Strategically positioning sensor nodes in the monitoring zone is crucial to increasing WSN node coverage. For large-scale sensor node deployment challenges, logical and efficient deployment of WSNs has been demonstrated to be an NP-hard problem, and finding the best solution remains challenging [18]. Multiple nodes must be deployed to meet the monitoring needs, resulting in significant network redundancy coverage issues, the repeated transmission of vast amounts of data in the network, and a rise in the number of network nodes [19].

The metaheuristic algorithm is one of the promising approaches being examined as a solution for dealing with WSN node coverage in this scenario [20]. Metaheuristic algorithms can identify near-optimal solutions in a fair amount of time with limited nodes and computational resources, making them a convenient approach to the WSN coverage optimization problem [21]. Approximation optimization techniques with solutions that can tackle high-dimensional optimization problems effectively are known as metaheuristic algorithms [22]. Natural phenomena, such as human behaviors, physical sensations, animal swarm behaviors, and evolutionary concepts, are frequently used to inspire metaheuristic algorithms [23]. The metaheuristic optimization algorithms are widely used in a variety of fields, including technology, health, society, and finance, and are especially good at meeting time deadlines [20]. They are usually fairly easy to implement, having few parameters, being relatively simple to understand, and powerful, including selecting for biological nature, natural social swarm behavior, and autocatalytic physical phenomena, e.g., simulated annealing (SA) [24], genetic algorithms (GAs) [25,26], particle swarm optimization (PSO) [27], cat swarm optimization (CSO) [28], parallel PSO (PPSO) [29], ant colony optimization (ACO) [30], artificial bee colony (ABC) [31], bat algorithms (BA) [32,33], moth–flame optimization (MFO) [34,35], whale optimization algorithm (WOA) [36], flower pollination algorithm (FPA) [37,38], sine–cosine algorithm (SCA) [39,40], etc.

A new metaheuristic optimization method based on suggested physical laws is the Archimedes optimization algorithm (AOA) [41], which is mimicked by the location update technique that uses object collisions for processing optimization equations. The optimization is carried out by modeling Archimedes' buoyancy principle process: following a crash, the object progressively assumes neutral buoyancy. The AOA has advantages and the potential to optimize various engineering problems because of its fewer parameters, making it more easily understandable in programming. However, there are specific problems with the AOA algorithm approach to particular issues, such as the solution's slow convergence time and poor quality.

This paper suggests an enhanced Archimedean algorithm (EAOA) for the global optimization problems and node coverage optimization in WSN deployment. The difficulties of WSN nodes' uneven distribution and low coverage in the random deployment of WSN monitoring applications are approached based on the EAOA. The entire WSN monitoring area can be divided into multiple sub-areas, and then node optimization coverage can be implemented in each sub-area based on evaluating the objective function values. The mod-

eled objective function is calculated by the all-nodes coverage area ratio of the probability of the deployed surface 2D WSN monitoring area of the network. We implemented the EAOA by adapting its updating equations using reverse learning and multidirection strategies to overcome the limitations of its original approach. The following item list briefly highlights the contributions of this paper's innovations:

- Offering strategies for enhancing the AOA to prevent the original algorithm's drawbacks in dealing with complex situations, evaluating the recommended method's performance by using the CEC2017 test suite, and comparing the proposed method's results with the other algorithms in the literature.
- Establishing the objective function of the optimal WSN node coverage issues in applying the EAOA and AOA for the first time, and analyzing and discussing the results of the experiment in comparison with swarm intelligence optimization algorithms.

The paper's remaining parts are organized as follows: Section 2 describes the WSN node coverage model as a statement problem, and reviews the AOA algorithm as related work. Section 3 presents the proposed EAOA, and evaluates its performance under the test suite. Section 4 offers the EAOA for tackling the node coverage issues by applying the EAOA algorithm and analyzing the simulation results. The conclusions are presented in Section 5.

2. System Definition

This section presents the WSN node coverage model as the problem statement, and the original algorithm—called the Archimedes optimization algorithm (AOA)—as a recent metaheuristic optimization algorithm. The subsections are reviewed as follows.

2.1. WSN Node Coverage Model

The coverage optimization problem is the desired location of each deployed node, with a fixed sensing radius for each sensor. Each node needs to be deployed with a limited sensing radius, and each sensor can only sense and find within its sensing radius. Detection within its sensing radius is a workable solution to the coverage problem. Assuming that the WSN is deployed in a two-dimensional (2D) monitoring area of $W \times L$ m^2, with M nodes set up randomly [15,42], then if S is a set of nodes denoted as $S = \{S_1, S_2, \ldots, S_i, \ldots, S_M, i = 1, 2, \ldots, \text{and } M\}$, the coordinates of each node S_i can be represented as (x_i, y_i). A sensor node's sensing range is a circle, with the center of the sensing radius R_s as its radius. The model of a two-dimensional WSN monitoring area network is assumed as follows:

- The sensing radius of each sensor node is R_s, and the communication radius is R_c, both measured in meters, with $R_c \geq 2R_s$.
- The sensor nodes can normally communicate, have sufficient energy, and can access time and data information.
- The sensor nodes have the same parameters, structure, and communication capabilities.
- The sensor nodes can move freely and update their location information in time.

Let T be a set of target monitoring points, $T = \{T_1, T_2, \ldots, T_j, \ldots, T_n\}$, $j = 1, 2, \ldots, n$; the T_j coordinate is (x_j, y_j) in the two-dimensional WSN monitoring area. If the distance between the target monitoring point T_j and any sensor node is less than or equal to the sensing radius R_s, then T_j is covered by the sensor nodes. With the sensor node S_i and goal monitoring point T_j, the Euclidean distance is defined as follows:

$$d(S_i, T_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \quad (1)$$

where $d(S_i, T_j)$ is the distance from node $S_i(x_i, y_i)$ to node $T_j(x_j, y_j)$. The node sensing model is set on the sensing radius if R_s is greater than or equal to $d(S_i, T_j)$—the probability

p that the target is set to 1; otherwise, it is set to 0. The probability formula is given as follows:

$$p(S_i, T_j) = \begin{cases} 1, & R_s \geq d(S_i, T_j) \\ 0, & R_s < d(S_i, T_j) \end{cases}, \quad (2)$$

where $p(S_i, T_j)$ is the probability between the sensor node S_i and goal monitoring point T_j. The sensor nodes can work cooperatively by affecting the neighbor nodes of the deployed two-dimensional WSN monitoring area. Whenever any target monitoring point can be covered by more than one sensor simultaneously, the probability of monitoring the target point T_j is given by the following formula:

$$P(S, T_j) = 1 - \prod_{i=1}^{M} (1 - p(S_i, T_j)), \quad (3)$$

The ratio of the total area covered by all sensor nodes in the monitoring area to that area's overall size is known as the coverage rate. Accordingly, the probability ratio to the network's surface 2D WSN monitoring area is used to calculate the coverage ratio.

$$Cov_R = \frac{\sum_{j=1}^{M} P(S, T_j)}{W \times L}, \quad (4)$$

where Cov_R is the WSN nodes' coverage ratio in the target point reaching area, $P(S, T_j)$ is the probability of the target point reaching sensed node monitoring, and $W \times L$ is the deployed area of the desired surface 2D network.

2.2. Archimedes Optimization Algorithm (AOA)

The AOA is a recent metaheuristic optimization algorithm based on Archimedes' buoyancy principle's physical principles [41]. The position of its object is updated by imitating the process of the object gradually exhibiting neutral buoyancy following a collision. The AOA algorithm provides the individual population by immersing objects with volume, density, and acceleration properties. The items can determine their position in the fluid based on these attributes. The characteristics and places of the object are randomly initialized at the start of the process. The AOA updates the object's volume, density, and acceleration during processing optimization. The object's position is updated based on its individual qualities. Initialization, updating object properties, updating the object's status, and evaluation are the significant processing steps of the AOA.

Initialization of the position and attributes of the object is conducted as follows:

$$X_i = lb_i + rand() \cdot (ub_i - lb_i), \quad (5)$$

where X_i is a candidate solution vector i-th of the object population size N, $i = 1, 2, \ldots, N$; the boundaries lb_i and ub_i are the upper and lower boundaries, respectively; and the variable $rand()$ is a d-dimensional vector generated randomly between [0, 1]. The variables of acceleration, volume, and density of the i-th object are noted as ac_i, vo_i, and de_i, respectively; $vo_i = rand()$, $de_i = rand()$, and $acc_i = lb_i + rand() \cdot (ub_i - lb_i)$. The position and attributes of the optimal object—such as X_{best}, de_{best}, vo_{best}, and ac_{best}—are the selected objects with the best fitness values according to the evaluation of each object.

Updating object properties phase: During the iteration, the volume and density of the object are updated according to the following formula:

$$vo_i^{t+1} = vo_i^t + rand \cdot (vo_{best} - vo_i^t), \quad (6)$$

$$de_i^{t+1} = de_i^t + rand \cdot \left(de_{best} - de_i^t\right), \quad (7)$$

where vo_i^{t+1} and de_i^{t+1} denote the volume and density of the i-th object in the $t+1$ iteration, respectively. The simulated collisions between objects in the AOA are mimicked for the

optimization process; as time goes on with iterations, the algorithm gradually reaches equilibrium. A transform variable is used as a simulation of the process to realize the algorithm's transformation from searching exploration to exploitation, as follows:

$$TF = \exp\left(\frac{t - t_{max}}{t_{max}}\right) \tag{8}$$

where TF is the transition transform variable, while t_{max} and t are the maximum number of iterations and the current number of iterations, respectively. TF gradually increases to 1 over time. $TF \leq 0.5$, meaning that one second of the iteration is in the exploration phase. The update acceleration of object attributes is related to the collision objects.

$$\mathrm{ac}_i^{t+1} = \begin{cases} \frac{de_{mr} + vo_{mr} \cdot ac_{mr}}{de_i^{t+1} \cdot vo_i^{t+1}}, & if\ TF \leq 1/2 \\ \frac{de_{best} + vol_{best} \cdot ac_{best}}{de_i^{t+1} \cdot vo_i^{t+1}}, & otherwise \end{cases} \tag{9}$$

where de_{mr}, vo_{mr}, and ac_{mr} are the density, volume, and acceleration of random material (mr), respectively. If $TF \leq 0.5$, there is a collision between objects, and the acceleration updates the formula of object i in iteration t; otherwise, there is no collision between objects. The normalization strategy for the acceleration can be updated as follows:

$$ac_{i,norm}^{t+1} = ur \cdot \frac{ac_i^{t+1} - \min(ac)}{max(ac) - \min(ac)} + lr, \tag{10}$$

where $ac_{i,norm}^{t+1}$ represents the normalized acceleration of the i-th object in the $t+1$ iteration, while ur and lr are the normalized ranges, which are set to 0.8 and 0.2, respectively.

Updating the objects' position is conducted as follows: If $TF \leq 1/2$ (exploration phase), the position update formula of object i at the $t+1$ iteration is helpful to search from global to local and converge in the region where the optimal solution exists; otherwise, it is a searching exploitation phase for the positional updating. When the object is far from the best position, the acceleration value is enormous, and the object is in the exploration phase. When the acceleration value is small, the object is close to the optimal solution. The exploitation phase can be described as follows:

$$X_i^{t+1} = X_i^t + C_1 \cdot rand \cdot ac_{i,norm}^{t+1} \cdot d \cdot \left(X_{rand} - X_i^t\right) \tag{11}$$

where C_1 is a constant that is set to 2, and d is the density factor that decreases over time, i.e., $d = \exp\left(\frac{t - t_{max}}{t_{max}}\right) - \left(\frac{t}{t_{max}}\right)$. The acceleration changes from big to small, indicating the algorithm's transition from exploration to exploitation, respectively, which helps the object approach the optimal global solution.

$$X_i^{t+1} = X_{best}^t + F \cdot C_2 \cdot rand \cdot ac_{i,norm}^{t+1} \cdot d \cdot \left(T \cdot X_{best} - X_i^t\right) \tag{12}$$

where C_2 represents the constant t; T is a variable proportional to the transfer operator—the percentage used to attain the best position—$T = C_3 \times TF$; and F is the direction of motion, and its expression is as follows:

$$F = \begin{cases} +1, & if\ P \leq 0.5 \\ -1, & if\ P > 0.5 \end{cases} \tag{13}$$

where P is set to $2 \cdot rand - C_4$.

Evaluating objective function involves computing the fitness values for the objective function after updating the object's position each iteration time. The model with objective function is used for fitness value evaluation by evaluating each object that is recorded with the best fitness value found in each position, e.g., X_{best}, de_{best}, vo_{best}, and ac_{best} are updated for the next iterations or generations.

3. Enhanced Archimedes Optimization Algorithm

In order to enhance the population of diverse objects, an enhanced version of the Archimedes optimization algorithm (EAOA) based on the opposing learning and diversity guiding techniques is presented in this section. The suggested processes are offered first, followed by a detailed presentation of the evaluation and discussion findings.

3.1. Enhanced Archimedes Optimization Algorithm

The AOA is a new metaheuristic algorithm with several advantages, including ease of understanding and implementation, along with local search capability. Still, it has drawbacks, such as jumping out of the optimal local operation, slow convergence, or vulnerability to local optima when dealing with complex problems, such as optimal WSN node coverage issues.

A multiverse-directing strategy: In the original expression in Equation (13), the direction of motion F has just two motion directions. For complicated problems, the space may have more scales in terms of motion in space. We can exploit this to increase the number of search directions in complex spaces. A variable of the direction guiding factor G is used as an equivalent to the direction value. An alternative formula of motion direction can be expressed as follows:

$$F_{new} = \begin{cases} +G \cdot rand(), \; if \; P \leq 0.5 \\ -tG \cdot rand(), \; otherwise \end{cases}, \tag{14}$$

where F_{new} is an alternative direction guiding factor, and $rand()$ is a random number $\in$ [0, 1] for making the different search values of directions.

Opposite direction strategy: The original and reversed solutions are sorted fitness values based on objective function issues to convert objects in a seeking, forward-exploiting procedure in the optimization problem space. The agents in the optimization space can swiftly converge to the task of the ideal solution by identifying new objects with the best fitness ratings by using direct vetting or other optimization strategies to establish new things in the solution space. A new solution set can be generated by applying reverse learning with a specific rate to join the original for further optimization.

Let $S(x_1, \; x_2, \; \ldots, \; x_i, \; \ldots, \; x_D)$, and $S'(x'_1, \; x'_2, \; \ldots, \; x'_D)$ be solutions of forwarding and corresponding inverse sets, where $x_i \in [a_i, b_i]$, $(i = 1, \; 2, \; \ldots, \; d)$. A range $[a, b]$ of the opposite solution set can be expressed as $x'_i = a_i + b_i - x_i$. The same idea of the opposite learning applied to a new solution is as follows:

$$S'_{inew} = S \cdot \beta_r, \tag{15}$$

where β_r is a variable as an adjustment coefficient for generating and affecting a new solution object set. A portion of the worst solution—e.g., about 15% of sorting values of evaluation object positions—is eliminated to be used for generating a new object set in dimension d of the solution space. The adjustment coefficient is calculated as follows:

$$\beta_r = R_{istar} \cdot \frac{rand(\beta, \gamma)}{D}, \tag{16}$$

where $rand()$ is a random function in the range from β to γ. In the experiment, β can be set to -0.5 and γ set to 0.5. D is the dimension of problem space, while R_{istar} is the distance between the ideal solution and the one that is closest to optimal. The adjustment coefficient can be applied to the exploiting search of the algorithm for generating and affecting a new solution object set merged into Equation (17).

The strategies and equations of reverse learning β_r and multiverse-directing F_{new} can be hybridized into updated formulas for generating new solutions. An update of the position of the objects is conducted as follows:

$$x_i^{t+1} = \begin{cases} x_i^t + \beta_r \cdot C_1 \cdot rand \cdot ac_{i,norm}^{t+1} \cdot d \cdot \left(x_{rand} - x_i^t\right), & if \; TF \leq 0.5 \\ x_{best}^t + F_{new} \cdot C_2 \cdot rand \cdot ac_{i,norm}^{t+1} \cdot d \cdot \left(T \times x_{best} - x_i^t\right), & otherwise \end{cases} \tag{17}$$

Algorithm 1 depicts the pseudocode of the enhanced Archimedes optimization algorithm (EAOA).

Algorithm 1 A pseudocode of the EAOA.

Input: N_P: The population size, D: dimensions, T: the Max_iter, C_1, C_2, C_3, C_4: variables, and *ub*, *lb*: upper and lower boundaries.
Output: The global best optimal solution.
Initialization: Initializing the locations, *vol.*, *de.*, and *acc.* of each object in the population of Equation (8); obtaining each object's position by calculating the objective function, and the best object in the population is selected; the iteration t is set to 1.
While $t < T$ **do**
    For $i = 1 : N_p$ **do**
        Updating *vol.*, and *de.*, of the object by Equations (6) and (7).
        Updating TF- transfer impactor and *d-de.*, variables are by Equation (8).
            If $TF \leq 1/2$ **then**
                Updating *acc.* the object acceleration by Equation (10).
                Updating the local solution by Equation (11).
            Else
Updating the object accelerations by Equations (9) and (10).
Updating global solution position by Equation (17).
End-if
        End-for
End-while
Evaluating each object with the positions and
    Selecting the best object of the whole population.
Recording the best global outcome of the optimal object.
t-iteration is set to $t + 1$
Output: The best object optimization of the whole population size.

3.2. Experimental Results for Global Optimization

The suggested algorithm's potential performance needs to be tested and verified with the benchmark functions. The CEC 2017 [43] test suite has 29 different test functions to evaluate the EAOA algorithm. There are various types of complexity and dimension functions in the test suite, e.g., f1~f3: unimodal, f4~f10: multimodal, f11~f20: hybrid, and f21~f29: compound test functions. The achieved results of the EAOA are compared not only with the original AOA [41], but also with other selected popular algorithms in the literature, e.g., genetic algorithms (GAs) [25], simulated annealing (SA) [24], particle swarm optimization (PSO) [27], moth–flame optimization (MFO) [34], improved MFO (IMFO) [35], flower pollination algorithm (FPA) [37], sine–cosine algorithm (SCA) [39], enhanced SCA (ESCA) [40], parallel PSO (PPSO) [29], and parallel bat algorithm (PBA) [33]. An expression of $\Delta f = f_i - f_i^*$ is a different error value between the function minimum value f_i^* and the obtained result value f_i of the i-th function. The fundamental conditions are set for all algorithms to ensure that the experiment is fair, e.g., the population size is set to 40; the maximum number of iterations is set to 1000; the number of dimensions is set to 30; the solution range of all of the test functions is set to $[-100, 100]$; and the number of runs is set to 25. Table 1 lists the basic parameters of each algorithm.

The obtained outcomes of the proposed EAOA approach can be verified by several cases—such as the affected strategies with the original algorithm—and compared with the other algorithms. First, the outcomes of several implemented tactics are contrasted with those of the original AOA algorithm. The findings from the EAOA are then contrasted with those from other algorithms. Table A1 compares the affected strategies in applying the EAOA with the original AOA algorithm, and verifies the impact of the suggested techniques used in the EAOA compared to the original AOA algorithm. The data values of the mean outcomes of 25 runs show the best obtained global optimal results, as well as the data on runtime and CPU execution. It can be seen that in some cases strategies 1 and 2 are better than the original algorithm. In most test function cases, the combined strategies 1

and 2 in the proposed EAOA can produce better results than the AOA, and the runtime is not much longer than that of the AOA.

Table 1. Algorithm settings for parameters and variables.

Algorithms	Setting Parameters
EAOA	$C_1 = 2.1,\ C_2 = 5.6,\ C_3 = 1.95,\ C_4 = 0.65$
AOA [41]	$C_1 = 2.1,\ C_2 = 5.6,\ C_3 = 1.95,\ C_4 = 0.65$
GA [25]	$R_{mu} = 0.1,\ R_{cr} = 0.9$
SA [24]	$P = 0.6,\ \alpha = 0.8,\ \tau = 0.05,\ S_N = 14.41$
PSO [27]	$V_{max} = 10,\ V_{min} = -10,\ \omega = 0.9\ to\ 0.4,\ c_1 = c_2 = 1.49455$
PPSO [29]	$G = 2,\ R = 10,\ V_{max} = 10,\ V_{min} = -10,\ \omega = 0.9\ to\ 0.4,\ c_1 = c_2 = 1.49465$
PBA [33]	$G = 2,\ R = 10,\ A_0 = 0.7,\ r_0 = 0.15,\ \alpha = 0.25,\ \gamma = 0.16$
FPA [37]	$P_{switch} = 0.65,\ \lambda = 1.5,\ s_0 = 0.1$
MFO [34]	$a = -1,\ b = 1$
IMFO [35]	$a = -1,\ b = 1,\ \omega = 0.9\ to\ 0.4$
WOA [36]	$a = 2$ to $0,\ b = 1,\ l = [-1, 1]$
SCA [39]	$r_1,\ r_3, = rand(0, 2),\ r_2\ \in [0,\ 2\pi],\ r_4 = rand(0, 1)$
ESCA [40]	$r_1,\ r_3, = rand(0, 2),\ r_2\ \in [0,\ 2\pi],\ r_4 = rand(0, 1),\ \omega = 0.9\ to\ 0.5$

Moreover, the obtained results from the EAOA were also further evaluated to verify the proposed approach's performance in the presentation. The findings of the EAOA compared with the other algorithms—e.g., GA [25], PSO [27], BA [32], PPSO [29], MFO [34], and WOA [36] algorithms—are presented in Tables A2–A5 and Figure A1. The data values in Tables A2–A4 are the Mean, Best, and Std.—a standard deviation that measures variables for analyzing the algorithm's performance. The values of Mean, Best, and Std. are for assessing the search capability, quality, and resilience of the algorithm, respectively. Tables A2–A4 compare the results of the proposed EAOA with the other popular metaheuristic algorithms in the literature, e.g., the GA [25], PSO [27], BA [32], PPSO [29], MFO, [34], and WOA [36] algorithms. The highlighted data values in each row of Tables A2–A4 are the best in each pair comparing the EAOA-obtained results with the others for testing functions with a suitable format and layout. The symbols Win, Loss, and Draw at the end of each table provide a brief statistical summary. It can be seen that the proposed EAOA algorithm has the highest number of 'Wins'. This means that the EAOA produces better results than the other algorithms, and that the EAOA has an excellent optimization performance.

Figure A1 compares the convergence outcome curves of the EAOA with the ESCA [40], IMFO [35], AOA [41], PPSO [29], WOA [36], and PBA [33] algorithms for the selected functions. The Y coordinate axis represents the average of 25 runs of the best output of the algorithms thus far. The X coordinate shows the iteration in the generation of searching methods. It can be seen from the figure that the EAOA performance curve shows a faster convergence rate than the other algorithms.

Furthermore, for another view of the evaluation results of the proposed approach, we applied the Wilcoxon signed-rank technique for ranking the outcomes. This test compares the pairwise algorithms' results between the EAOA and other enhanced methods—e.g., PBA, WOA, PPSO, AOA, IFMO, and ESCA algorithms—under the Wilcoxon signed-rank test. Table A5 lists the results of comparison of the pairwise algorithms' results between the EAOA and other algorithms when applying the Wilcoxon signed-rank test. The bold-highlighted results in Table A5 are the outcomes with $p < 0.05$. It can be seen that most values have $p < 0.05$, indicating that the optimization results of the EAOA are significantly different from those of the other algorithms. The average ranking value is 2.25204, and the

lowest output of the EAOA is superior to that of the other algorithms. In general, it can be seen that the proposed EAOA can compete with some of the other popular algorithms.

4. Optimal WSN Node Coverage Based on EAOA

This section demonstrates how the EAOA algorithm can be used to deploy a WSN with the best node coverage possible, followed by a subsection covering the majority of the processing stages, analysis, and discussion of the findings.

4.1. Optimal Node Coverage Strategy

The feasible solution to the optimal node coverage problem is the deployment of each node with a limited sensing radius, where each sensor can only sense and find within its sensing radius. The finding within its sensing radius is a workable solution to the coverage optimization problem. Assuming that the sensing radius of all nodes is the same, and the sensing radius of the node $r \leq R$, any point in the monitoring area is covered if it is located within the sensing radius of at least one sensor node. The monitoring area is divided into the coverage area and the blind spot. Any point in the coverage area is covered by at least one sensor node, and the blind spot complements the coverage area. Some applications need to monitor events with high accuracy. Any point in the coverage area must be at least within the sensing radius of M nodes simultaneously; otherwise, it will be regarded as a blind spot, which we call M double-coverage. The location-seeking process of nodes is abstracted as the process of implementing varied movement behaviors of the object group toward food or a specific site.

The purpose of WSN coverage optimization utilizing the EAOA approach is to optimize the coverage of the target monitoring area by using a limited number of sensor nodes and optimizing their deployment locations. Let $F(x)$ be the objective function of the WSN nodes' coverage optimization; the coverage ratio, which is the maximum ratio of probability to the network's deployed surface 2D WSN monitoring area, is used to determine the objective function for the optimization problem. The maxima are as follows according to Equation (4):

$$F(x) = Maximize\ Cov_R = \frac{\sum_{j=1}^{M} P(S, T_j)}{W \times L}, \tag{18}$$

where Cov_R and $P(S, T_j)$ are the coverage ratio of the WSN nodes and the probability of the target point reaching $W \times L$ in the sensed node of the 2D monitoring network's deployed area, respectively. Each individual object in the algorithm represents a coverage distribution, and the specific steps of the algorithm scheme for the coverage optimization are listed as follows:

Step 1: Input parameters such as a number of nodes M, perception radius R_s, area of region $W \times L$, etc.
Step 2: Set the parameters of population size N, the maximum number of iterations *max_Iter*, the density factor, and prey attraction, and randomly initialize the object's positions using Equations (5)–(7).
Step 3: Enhance the initializing population—the parameters of Equations (8)–(10), (14), and (15)—and calculate the objective function for initial coverage according to Equation (18).
Step 4: Update the position of objects and the strategy according to Equation (17), and then compare them to select the best fitness value according to the objective function value.
Step 5: Calculate the individual values of objects and retain the optimal solution of the global best.
Step 6: Determine whether the end condition is reached; if yes, proceed to the next step; otherwise, return to Step 4.
Step 7: The program ends and outputs the optimal fitness value and the object's best location, representing the node's optimal coverage rate outputs.

4.2. Analysis and Discussion of Results

The scenarios assuming that the WSN's sensor nodes are deployed in a square monitoring area of $W \times L$ can be set to scenario areas, e.g., 40 m × 40 m, 80 m × 80 m, 100 m × 100 m, and 160 m × 160 m. Table 2 lists the experimental parameters of the WSN node deployment areas. The sensing radius of sensor nodes R_s is set to 10 m. The communication radius R_c is set to 20 m. The number of sensor nodes is denoted by M, consisting of 20, 40, 50, and 60 sensor nodes. *Iter* indicates the number of iterations, which may be set to 500, 1000, or 1500.

Table 2. The parameter settings for the desired WSN node deployment areas.

Description	Parameters	Values
Desired deployment areas	$W \times L$	40 m × 40 m, 80 m × 80 m, 100 m × 100 m, 160 m × 160 m
Sensing radius	*Rs*	15 m
Communication radius	*Rc*	20 m
Number of sensor nodes	*M*	20, 40, 50, 60
Number of iterations	*Iter*	500, 1000, 1500

The optimal results of the EAOA were compared with the other selected schemes—i.e., the SSA [44], PSO [45], GWO [46], SCA [47], and AOA [48]—for the coverage optimization of WSN node deployment to verify the adequate performance of the algorithm. Figure 1 displays a graphical diagram of the nodes' initialization with the EAOA for the statistical coverage optimization scheme with different numbers of sensor nodes: (a) 20, (b) 40, (c) 50, and (d) 60.

Table 3 compares the proposed EAOA approach to other strategies—i.e., the SSA, PSO, GWO, SCA, and AOA algorithms—in terms of percentage coverage rate, running time, convergence iterations, and monitoring area size. It can be seen that the EAOA scheme produces the best global solution in the coverage areas, with a high coverage rate, coverage of the node's whole area, and a faster runtime than the other approaches.

Table 3. Comparison of the proposed EAOA method with the other techniques used—i.e., the SAA, PSO, GWO, SCA, and AOA algorithms—in terms of percentage coverage rate, running time, iterations to convergence, and monitoring area size.

Approach	Factor Variables	40 m × 40 m	80 m × 80 m	100 m × 100 m	160 m × 160 m
SSA	Coverage rate (%)	78%	74%	77%	74%
	Consumed execution time (s)	3.09	6.91	7.38	9.34
	No. of iterations to convergence	145	256	234	844
	WSN node numbers	20	40	50	60
PSO	Coverage rate (%)	79%	77%	79%	76%
	Consumed execution time (s)	2.78	6.22	6.65	8.41
	No. of iterations to convergence	396	343	578	754
	WSN node numbers	20	40	50	60
GWO	Coverage rate (%)	80%	80%	84%	78%
	Consumed execution time (s)	3.06	6.84	7.31	9.25
	No. of iterations to convergence	334	44	544	755
	WSN node numbers	20	40	50	60

Table 3. *Cont.*

Approach	Factor Variables	40 m × 40 m	80 m × 80 m	100 m × 100 m	160 m × 160 m
CSA	Coverage rate (%)	78%	79%	82%	78%
	Consumed execution time (s)	2.92	6.29	7.23	9.22
	No. of iterations to convergence	445	555	665	876
	No. of mobile nodes	20	40	50	60
AOA	Coverage rate (%)	80%	79%	80%	79%
	Consumed execution time (s)	3.12	6.98	7.46	9.44
	No. of iterations to convergence	665	333	563	954
	WSN node numbers	20	40	50	60
EAOA	Coverage rate (%)	80%	82%	87%	80%
	Consumed execution time (s)	2.75	6.15	6.57	8.31
	No. of iterations to convergence	135	503	556	765
	WSN node numbers	20	40	50	60

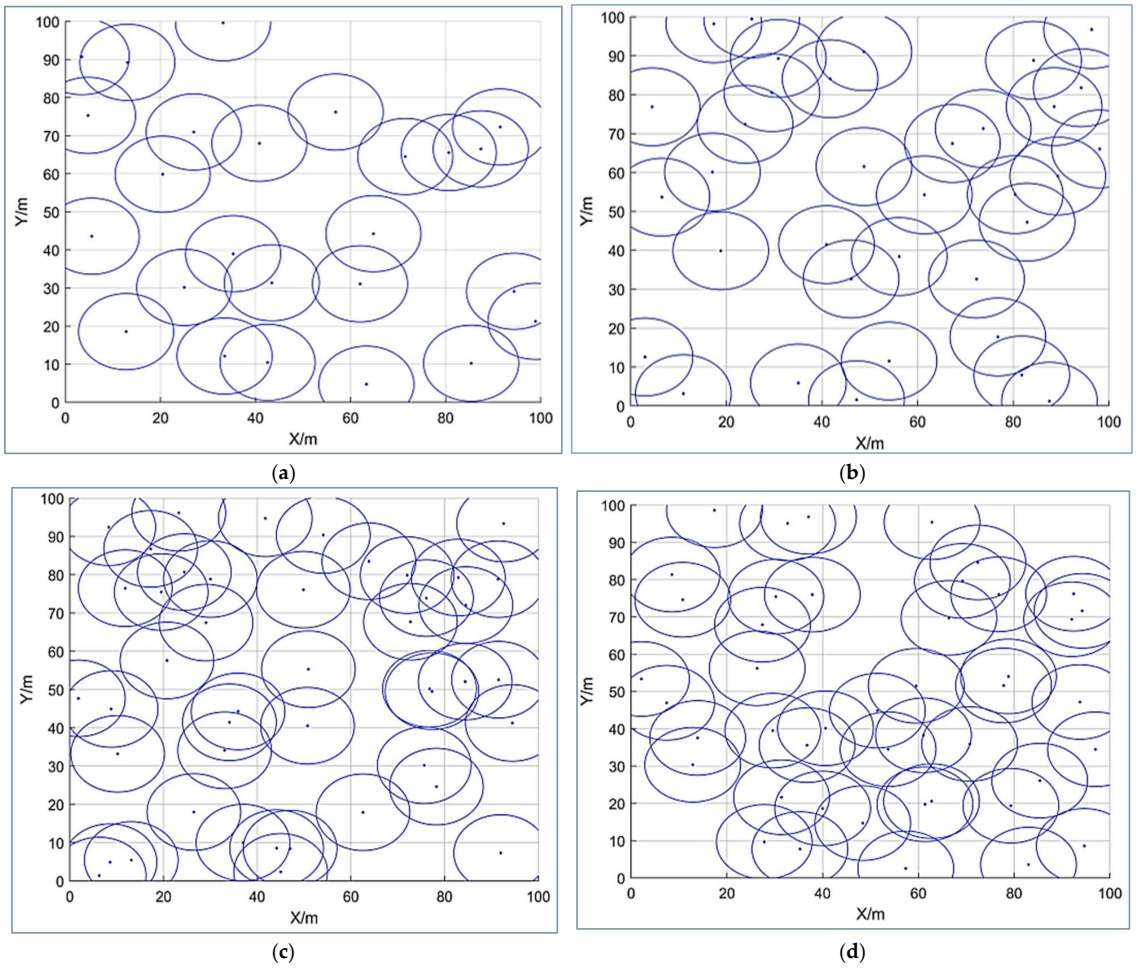
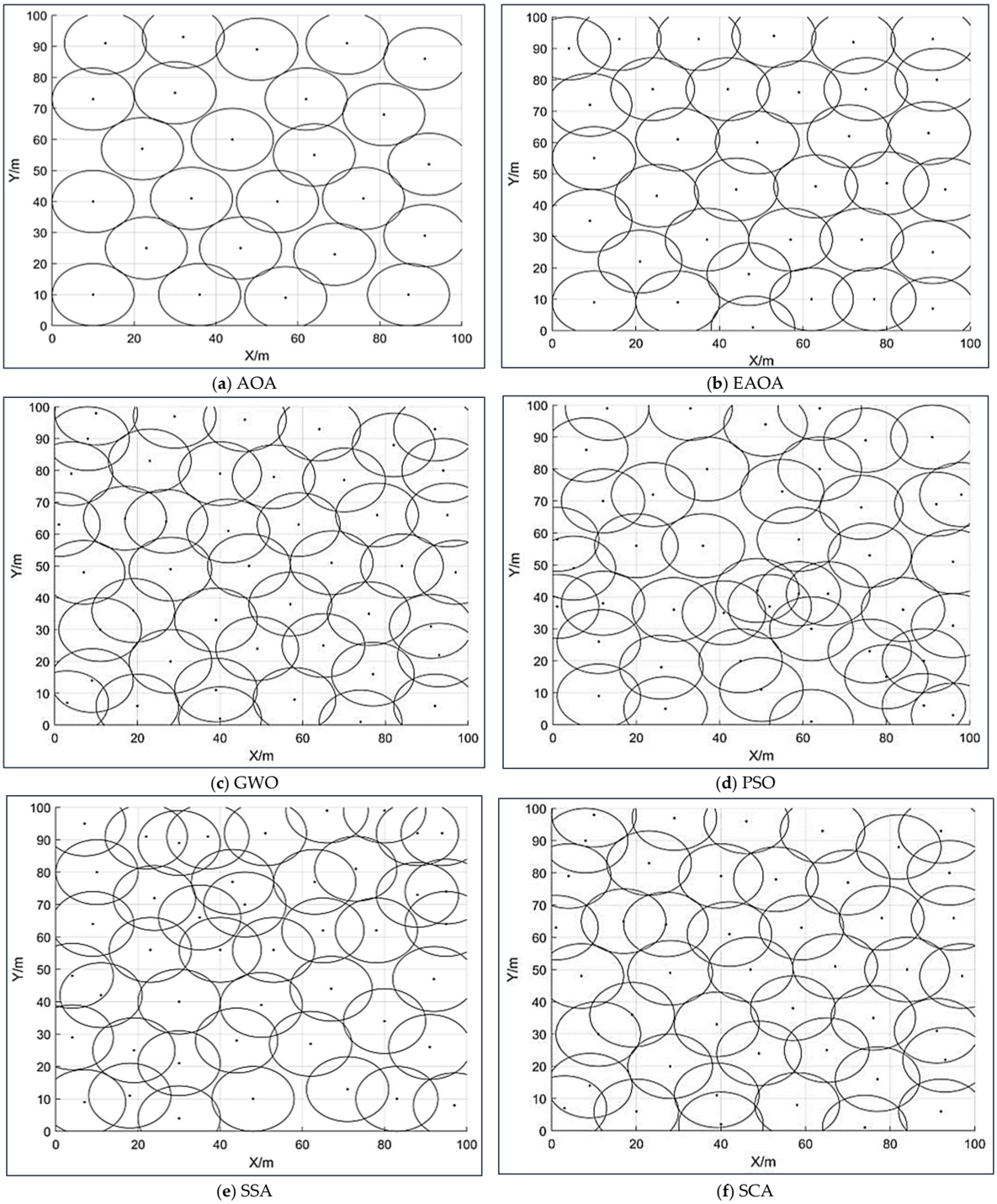


Figure 1. The graphical initialization of the EAOA with the statistical node coverage optimization scheme for different numbers of sensor nodes: (**a**) 20, (**b**) 40, (**c**) 50, and (**d**) 60.

Figure 2 indicates the graphical coverage of six different metaheuristic algorithms—i.e., the AOA, SSA, PSO, GWO, SCA, and EAOA approaches—for the WSN node area deployment scenarios for optimal coverage rates, with the same density and environmental

setting conditions. Because the EAOA algorithm can avoid premature phenomena, its coverage rate is reasonably high, with less overlap. It can better alter the node configuration than the other competitors for the monitoring area's network coverage. The graphics show the differences in the distribution of coverage; the differences are so small that the graphics look very similar, with the graph of node coverage distribution showing seemingly identical results. Furthermore, Figures 3 and 4 show that the convergence curves of the proposed EAOA approach can provide higher percentages of statistical coverage than the other methods used.

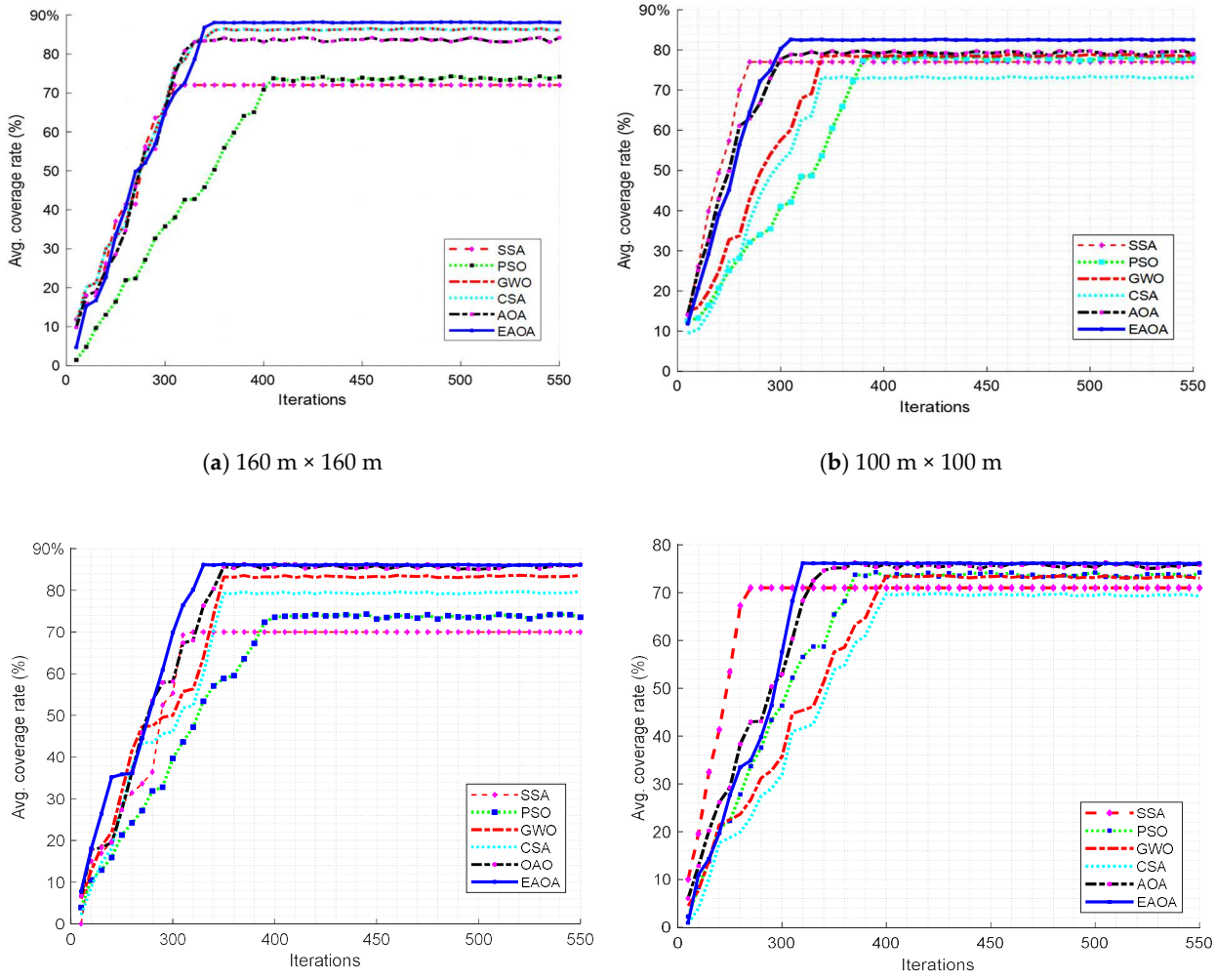


(**a**) AOA (**b**) EAOA

(**c**) GWO (**d**) PSO

(**e**) SSA (**f**) SCA

Figure 2. The graphical coverage of six different metaheuristic algorithms for the WSN node area deployment. (**a**) AOA, (**b**) EAOA, (**c**) GWO, (**d**) PSO, (**e**) SSA, (**f**) SCA algorithms.



(**a**) 160 m × 160 m

(**b**) 100 m × 100 m

(**c**) 80 m × 80 m

(**d**) 40 m × 40 m

Figure 3. Comparison of the optimal coverage rates of the EAOA with the other schemes in different-sized WSN monitoring node area deployment scenarios. (**a**) 160 m × 160 m, (**b**) 100 m × 100, (**c**) 80 m × 80 m, and (**d**) 40 m × 40 m.

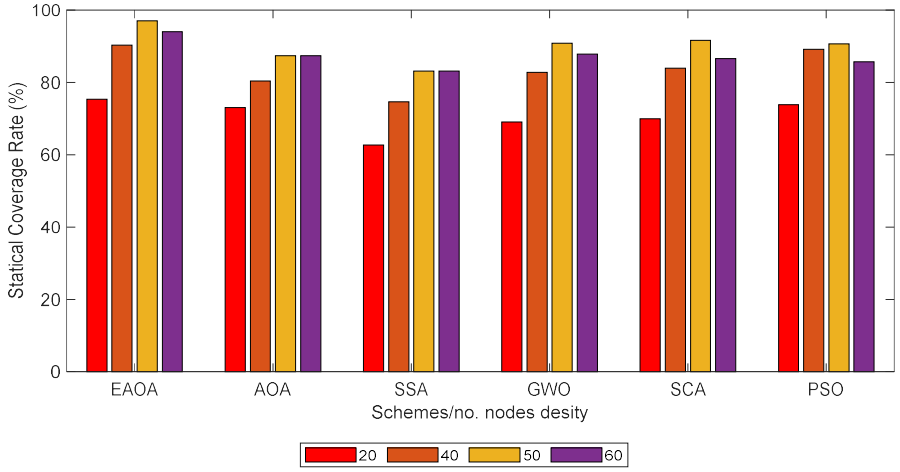


Figure 4. Comparison of the EAOA optimization coverage rates for various sensor node counts deployed in the 2D monitoring of a 100 m × 100 m area.

Figure 3 indicates four different sizes of WSN monitoring node area deployment scenarios of the metaheuristic approaches for optimal coverage rates. The convergence curves of the proposed EAOA approach can provide higher percentages of statistical coverage than the other methods used.

Figure 4 shows the coverage rate of the EAOA compared against the SSA, PSO, GWO, SCA, and AOA algorithms for statistical sensor node count deployment for the 2D monitoring of different areas. It can be seen the EAOA algorithm produces a coverage rate that is reasonably high in the monitoring area's network coverage. The results show that the EAOA approach provides a reasonably high coverage rate, with less overlap and better alteration of the sensor nodes' configuration, compared to the average coverage rate under the same test conditions.

5. Conclusions

This paper suggests an enhanced Archimedes optimization algorithm (EAOA) to solve the wireless sensor network (WSN) nodes' uneven distribution and low coverage issues in random deployment. Each divided sub-area of the monitoring area of the entire WSN was subjected to node coverage optimization based on the EAOA. The objective function of the optimal node coverage was modeled mathematically by calculating the distance between nodes by measuring each sensor node's sensing radius and its communication capability in the deployed WSN. The optimization results of multiple sub-areas were fused, combining the sub-areas' coverage with the complete network node coverage via a mapping mechanism. The updated equations of the EAOA were modified with reverse learning and multidirection strategies to avoid the original drawbacks of the AOA, e.g., slow convergence speed and ease of falling into local extrema whenever dealing with complicated situations. The compared results of the optimal findings on the selected benchmark functions and the WSN node coverage show that the proposed EAOA makes the optimal solution effective for both coverage and benchmark problems. The suggested algorithm will be applied in future works to address WSN node localization [49,50] and optimal WSN deployment [51,52].

Author Contributions: Conceptualization, T.-K.D. and V.-T.N.; methodology, T.-T.N.; software, T.-T.N.; validation, T.-K.D., V.-T.N. and T.-T.N.; formal analysis, T.-T.N.; investigation, S.-C.C.; resources, T.-K.D.; data curation, S.-C.C.; writing—original draft preparation, T.-D.N.; writing—review and editing, T.-D.N.; visualization, V.-T.N.; supervision, T.-T.N.; project administration, V.-T.N.; funding acquisition, V.-T.N. All authors have read and agreed to the published version of the manuscript.

Funding: This study was partially supported by the VNUHCM-University of Information Technology's Scientific Research Support Fund.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Verifying the impact of the suggested techniques used in the EAOA in comparison with the original AOA algorithm.

Fun Test	Original		Suggested Strategy 01		Suggested Strategy 02		Suggested Strategies 01 and 02	
	AOA		Multidirection		Opposite Learning		EAOA	
	Mean	CPU Runtime (s)	Mean	CPU Runtime (s)	Mean	CPU Runtime (s)	Mean	CPU Runtime (s)
f1	2.95×10^{-1}	37.93	1.86×10^{-1}	36.10	1.91×10^{-1}	34.30	$\mathbf{1.71 \times 10^{-1}}$	38.52
f2	$2.71 \times 10^{+1}$	32.76	$1.94 \times 10^{+1}$	34.02	$\mathbf{1.61 \times 10^{+1}}$	34.12	$1.65 \times 10^{+1}$	38.32
f3	3.66×10^{-1}	45.34	2.58×10^{-1}	47.09	$\mathbf{6.52 \times 10^{-2}}$	47.23	2.44×10^{-1}	53.04
f4	3.02×10^{-1}	44.16	1.45×10^{-1}	45.86	4.59×10^{-2}	46.00	$\mathbf{1.27 \times 10^{-2}}$	52.12
f5	7.99×10^{-2}	40.32	7.84×10^{-3}	42.81	1.38×10^{-2}	42.00	$\mathbf{5.38 \times 10^{-3}}$	48.03
f6	5.58×10^{-1}	85.44	2.11×10^{-1}	88.73	$\mathbf{6.21 \times 10^{-2}}$	89.00	1.92×10^{-1}	98.89
f7	2.21×10^{-1}	203.52	$\mathbf{1.07 \times 10^{-1}}$	221.31	2.44×10^{-1}	212.10	1.26×10^{-1}	237.18
f8	6.32×10^{0}	117.12	$\mathbf{6.61 \times 10^{-1}}$	121.41	1.97×10^{0}	122.00	7.25×10^{-1}	136.23
f9	7.20×10^{0}	229.60	4.82×10^{0}	234.61	4.25×10^{0}	235.010	$\mathbf{4.36 \times 10^{0}}$	251.72
f10	2.25×10^{0}	224.61	2.63×10^{-1}	233.42	2.05×10^{-1}	234.10	$\mathbf{2.10 \times 10^{-2}}$	263.69
f11	$4.95 \times 10^{+3}$	274.65	$\mathbf{1.77 \times 10^{+3}}$	275.31	$8.09 \times 10^{+3}$	278.01	$1.06 \times 10^{+3}$	278.59
f12	$1.66 \times 10^{+2}$	229.44	$3.65 \times 10^{+1}$	238.28	$8.09 \times 10^{+1}$	239.00	$2.29 \times 10^{+1}$	268.40
f13	$3.58 \times 10^{+1}$	120.01	2.87×10^{0}	124.61	$3.30 \times 10^{+1}$	125.10	$\mathbf{1.53 \times 10^{0}}$	140.28
f14	$2.96 \times 10^{+1}$	96.26	1.62×10^{0}	100.71	$1.09 \times 10^{+1}$	101.10	$\mathbf{1.26 \times 10^{0}}$	113.41
f15	2.05×10^{0}	221.76	7.88×10^{-1}	231.31	4.74×10^{-1}	231.10	7.27×10^{-1}	259.42
f16	4.73×10^{-1}	126.72	1.85×10^{-1}	131.61	2.59×10^{-1}	132.01	$\mathbf{1.30 \times 10^{-1}}$	148.34
f17	$4.04 \times 10^{+2}$	223.69	$5.63 \times 10^{+1}$	232.31	$5.53 \times 10^{+2}$	233.10	$7.90 \times 10^{+1}$	262.71
f18	$2.49 \times 10^{+2}$	100.81	3.70×10^{-1}	104.35	$1.46 \times 10^{+1}$	105.10	$\mathbf{1.09 \times 10^{0}}$	117.92
f19	4.06×10^{-1}	206.40	$\mathbf{3.24 \times 10^{-1}}$	214.36	3.79×10^{-1}	215.00	3.86×10^{-1}	241.45
f20	5.87×10^{-1}	298.56	4.11×10^{-1}	310.07	4.34×10^{-2}	311.00	$\mathbf{3.98 \times 10^{-2}}$	349.25
f21	6.51×10^{-1}	327.36	2.25×10^{-1}	339.98	8.29×10^{-2}	341.00	2.10×10^{-1}	384.15
f22	8.94×10^{-1}	312.96	$\mathbf{6.22 \times 10^{-1}}$	325.76	7.03×10^{-1}	326.00	6.34×10^{-1}	367.09
f23	1.02×10	303.36	7.63×10^{-1}	315.05	$\mathbf{5.72 \times 10^{-2}}$	316.00	7.59×10^{-2}	354.87
f24	7.38×10^{-1}	282.24	6.63×10^{-1}	294.32	4.75×10^{-1}	294.00	$\mathbf{4.12 \times 10^{-1}}$	331.25
f25	3.28×10^{0}	206.40	$\mathbf{7.26 \times 10^{-1}}$	215.36	1.51×10^{0}	215.00	7.74×10^{-1}	243.15
f26	8.53×10^{-1}	253.44	8.03×10^{-1}	263.22	$\mathbf{2.78 \times 10^{-2}}$	264.00	7.78×10^{-1}	297.17
f27	7.28×10^{-1}	273.44	7.74×10^{-1}	265.45	5.19×10^{-1}	284.00	$\mathbf{7.41 \times 10^{-2}}$	295.92
f28	2.37×10^{0}	225.60	1.09×10^{0}	234.30	$\mathbf{3.34 \times 10^{-1}}$	235.00	9.39×10^{-1}	263.91
f29	$2.15 \times 10^{+3}$	221.76	$8.37 \times 10^{+1}$	230.31	$3.14 \times 10^{+2}$	231.12	$\mathbf{4.67 \times 10^{+1}}$	259.82
Avg.	1.72×10^{-1}	198.01	6.88×10^{-1}	199.91	3.15×10^{-1}	199.47	$\mathbf{4.45 \times 10^{-2}}$	219.12

The bold data values in each row of the Table are the best ones in each pair compared with the EAOA approach.

Table A2. The performance presentation of the EAOA, SA, and GA for the CEC 2017 test suite with each paired comparison.

Funs	GA			SA			EAOA		
	Mean	Best	Std.	Mean	Best	Std.	Mean	Best	Std.
f1	5.66×10^{-5}	1.46×10^{-5}	5.19×10^{-5}	7.16×10^{-5}	$\mathbf{1.25 \times 10^{-5}}$	2.38×10^{-5}	$\mathbf{1.29 \times 10^{-5}}$	2.71×10^{-5}	$\mathbf{1.11 \times 10^{-5}}$
f2	3.72×10^{-1}	$\mathbf{1.54 \times 10^{-1}}$	1.01×10^{-1}	$3.78 \times 10^{+1}$	$2.21 \times 10^{+1}$	$9.86 \times 10^{+1}$	$\mathbf{3.57 \times 10^{-1}}$	1.85×10^{-1}	$\mathbf{1.11 \times 10^{-1}}$
f3	2.57×10^{-1}	1.58×10^{-1}	$\mathbf{5.11 \times 10^{-2}}$	4.92×10^{-1}	2.66×10^{-1}	1.28×10^{-1}	$\mathbf{2.33 \times 10^{-1}}$	$\mathbf{1.44 \times 10^{-1}}$	5.52×10^{-2}

Table A2. *Cont.*

Funs	GA			SA			EAOA		
	Mean	**Best**	**Std.**	**Mean**	**Best**	**Std.**	**Mean**	**Best**	**Std.**
f4	2.31×10^{-1}	1.45×10^{-1}	4.84×10^{-2}	4.58×10^{-1}	3.02×10^{-1}	$\mathbf{4.34 \times 10^{-2}}$	$\mathbf{1.91 \times 10^{-1}}$	$\mathbf{1.11 \times 10^{-1}}$	4.59×10^{-2}
f5	3.90×10^{-2}	7.86×10^{-3}	1.68×10^{-2}	$\mathbf{1.12 \times 10^{-2}}$	7.99×10^{-2}	1.63×10^{-2}	2.57×10^{-2}	$\mathbf{5.38 \times 10^{-3}}$	$\mathbf{1.38 \times 10^{-2}}$
f6	3.28×10^{-1}	2.11×10^{-1}	7.81×10^{-2}	8.30×10^{-1}	5.58×10^{-1}	1.26×10^{-1}	$\mathbf{2.68 \times 10^{-1}}$	$\mathbf{1.92 \times 10^{-1}}$	$\mathbf{6.21 \times 10^{-2}}$
f7	1.95×10^{-1}	$\mathbf{1.07 \times 10^{-1}}$	3.69×10^{-2}	3.59×10^{-1}	2.21×10^{-1}	8.33×10^{-2}	$\mathbf{1.66 \times 10^{-1}}$	1.26×10^{-1}	$\mathbf{2.44 \times 10^{-2}}$
f8	3.43×10^{0}	$1.21 \times 10^{+1}$	$\mathbf{1.64 \times 10^{0}}$	$\mathbf{1.32 \times 10^{0}}$	6.32×10^{0}	4.29×10^{0}	3.23×10^{0}	$\mathbf{7.17 \times 10^{-1}}$	1.97×10^{0}
f9	$\mathbf{6.43 \times 10^{0}}$	4.81×10^{0}	1.19×10^{0}	8.79×10^{0}	7.20×10^{0}	$\mathbf{1.09 \times 10^{0}}$	6.53×10^{0}	$\mathbf{4.36 \times 10^{0}}$	1.25×10^{0}
f10	3.99×10^{-1}	$\mathbf{2.03 \times 10^{-1}}$	1.03×10^{-1}	5.33×10^{0}	1.20×10^{0}	4.24×10^{0}	$\mathbf{3.81 \times 10^{-1}}$	2.10×10^{-1}	$\mathbf{1.01 \times 10^{-1}}$
f11	$1.02 \times 10^{+4}$	$1.77 \times 10^{+3}$	$9.36 \times 10^{+3}$	$2.81 \times 10^{+5}$	$4.45 \times 10^{+4}$	$1.90 \times 10^{+5}$	$\mathbf{7.58 \times 10^{+3}}$	$\mathbf{1.06 \times 10^{+3}}$	$\mathbf{8.09 \times 10^{+3}}$
f12	$9.53 \times 10^{+2}$	$3.65 \times 10^{+1}$	$1.07 \times 10^{+2}$	$\mathbf{9.30 \times 10^{+2}}$	$1.66 \times 10^{+2}$	$1.10 \times 10^{+3}$	$9.47 \times 10^{+1}$	$\mathbf{2.29 \times 10^{+1}}$	$\mathbf{8.09 \times 10^{+1}}$
f13	$3.62 \times 10^{+1}$	9.87×10^{0}	$3.51 \times 10^{+1}$	$9.11 \times 10^{+3}$	$\mathbf{9.80 \times 10^{0}}$	$2.89 \times 10^{+2}$	$\mathbf{9.23 \times 10^{+1}}$	9.83×10^{0}	$\mathbf{3.30 \times 10^{+1}}$
f14	$1.21 \times 10^{+1}$	1.62×10^{0}	$\mathbf{7.68 \times 10^{0}}$	$1.66 \times 10^{+2}$	$2.96 \times 10^{+1}$	$1.15 \times 10^{+2}$	$\mathbf{1.05 \times 10^{+1}}$	$\mathbf{1.26 \times 10^{0}}$	$1.09 \times 10^{+1}$
f15	$\mathbf{1.57 \times 10^{0}}$	7.88×10^{-1}	4.83×10^{-1}	3.69×10^{0}	2.05×10^{0}	$\mathbf{4.63 \times 10^{-1}}$	1.66×10^{0}	$\mathbf{7.27 \times 10^{-1}}$	$4.74 \times 10{-}1$
f16	5.97×10^{-1}	1.85×10^{-1}	2.70×10^{-1}	1.30×10^{0}	4.73×10^{-1}	3.87×10^{-1}	$\mathbf{5.77 \times 10^{-1}}$	$\mathbf{1.30 \times 10^{-1}}$	$\mathbf{2.59 \times 10^{-1}}$
f17	$6.05 \times 10^{+2}$	$\mathbf{5.63 \times 10^{+1}}$	$7.15 \times 10^{+2}$	$1.01 \times 10^{+4}$	$4.04 \times 10^{+2}$	$1.60 \times 10^{+4}$	$\mathbf{4.81 \times 10^{+2}}$	$7.90 \times 10^{+1}$	$\mathbf{5.53 \times 10^{+2}}$
f18	$\mathbf{9.73 \times 10^{0}}$	$\mathbf{3.70 \times 10^{-1}}$	$1.74 \times 10^{+1}$	$2.11 \times 10^{+4}$	$2.49 \times 10^{+2}$	$1.87 \times 10^{+4}$	$1.10 \times 10^{+1}$	1.09×10^{0}	$\mathbf{1.46 \times 10^{+1}}$
f19	7.95×10^{-1}	$\mathbf{3.24 \times 10^{-1}}$	3.13×10^{-1}	$\mathbf{1.29 \times 10^{-1}}$	4.06×10^{-1}	3.70×10^{-1}	7.97×10^{-1}	3.86×10^{-1}	$\mathbf{2.79 \times 10^{-1}}$
f20	4.87×10^{-1}	4.11×10^{-1}	$\mathbf{3.48 \times 10^{-2}}$	7.39×10^{-1}	5.87×10^{-1}	8.77×10^{-2}	$\mathbf{4.80 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-1}}$	4.34×10^{-2}
f21	3.46×10^{-1}	2.25×10^{-1}	$\mathbf{6.95 \times 10^{-2}}$	8.38×10^{0}	6.51×10^{-1}	2.32×10^{0}	$\mathbf{2.95 \times 10^{-1}}$	$\mathbf{2.10 \times 10^{-1}}$	8.29×10^{-2}
f22	7.69×10^{-1}	$\mathbf{6.22 \times 10^{-1}}$	5.70×10^{-2}	1.21×10^{0}	9.94×10^{-1}	1.04×10^{-1}	$\mathbf{7.46 \times 10^{-1}}$	6.79×10^{-1}	$\mathbf{4.63 \times 10^{-2}}$
f23	8.64×10^{-1}	7.63×10^{-1}	6.44×10^{-2}	1.26×10^{0}	$\mathbf{1.02 \times 10^{-1}}$	1.34×10^{-1}	$\mathbf{8.49 \times 10^{-1}}$	7.59×10^{-1}	$\mathbf{5.72 \times 10^{-2}}$
f24	7.34×10^{-1}	6.63×10^{-1}	$\mathbf{3.85 \times 10^{-2}}$	8.77×10^{-1}	7.38×10^{-1}	7.48×10^{-2}	$\mathbf{7.00 \times 10^{-1}}$	$\mathbf{6.12 \times 10^{-1}}$	4.75×10^{-2}
f25	$\mathbf{2.79 \times 10^{0}}$	$\mathbf{7.26 \times 10^{-1}}$	1.62×10^{0}	8.04×10^{0}	3.28×10^{0}	1.63×10^{0}	3.45×10^{0}	7.74×10^{-1}	$\mathbf{1.51 \times 10^{0}}$
f26	8.44×10^{-1}	8.03×10^{-1}	2.31×10^{-2}	1.13×10^{0}	8.53×10^{-1}	1.82×10^{-1}	$\mathbf{8.29 \times 10^{-1}}$	$\mathbf{7.78 \times 10^{-1}}$	2.78×10^{-2}
f27	8.55×10^{-1}	7.74×10^{-1}	5.76×10^{-2}	1.02×10^{0}	8.28×10^{-1}	1.38×10^{-1}	$\mathbf{8.17 \times 10^{-1}}$	$\mathbf{7.41 \times 10^{-1}}$	$\mathbf{5.19 \times 10^{-2}}$
f28	1.74×10^{0}	1.09×10^{0}	3.76×10^{-1}	3.66×10^{0}	$\mathbf{2.37 \times 10^{-1}}$	6.31×10^{-1}	$\mathbf{1.49 \times 10^{0}}$	9.39×10^{-1}	$\mathbf{3.34 \times 10^{-1}}$
f29	$1.12 \times 10^{+3}$	$8.37 \times 10^{+1}$	$1.16 \times 10^{+3}$	$4.33 \times 10^{+4}$	$4.15 \times 10^{+3}$	$3.70 \times 10^{+4}$	$\mathbf{3.55 \times 10^{+2}}$	$\mathbf{4.89 \times 10^{+1}}$	$\mathbf{3.14 \times 10^{+2}}$
Win	5	9	7	6	5	5	20	18	19
Lose	21	18	20	22	22	22	9	11	10
Draw	3	4	4	3	2	4	0	0	0

The bold data values in each row of the Table are the best ones in each pair compared with the EAOA approach.

Table A3. The performance presentation of the EAOA, FPA, and PSO for the CEC 2017 test suite with each paired comparison.

Funs	FPA			PSO			EAOA		
	Mean	**Best**	**Std.**	**Mean**	**Best**	**Std.**	**Mean**	**Best**	**Std.**
f1	2.24×10^{-2}	1.19×10^{-2}	5.79×10^{-2}	$\mathbf{4.37 \times 10^{-2}}$	$\mathbf{2.25 \times 10^{-2}}$	$\mathbf{1.36 \times 10^{-2}}$	1.34×10^{-3}	2.83×10^{-2}	1.16×10^{-2}
f2	1.15×10^{0}	7.23×10^{-1}	$\mathbf{2.60 \times 10^{-1}}$	8.74×10^{-1}	5.18×10^{-1}	6.01×10^{-1}	$\mathbf{7.58 \times 10^{-1}}$	$\mathbf{3.92 \times 10^{-1}}$	4.36×10^{-1}
f3	$\mathbf{2.11 \times 10^{-1}}$	1.43×10^{-1}	3.47×10^{-1}	2.34×10^{-1}	$\mathbf{1.21 \times 10^{-1}}$	3.40×10^{-1}	2.43×10^{-1}	1.50×10^{-1}	$\mathbf{5.77 \times 10^{-2}}$
f4	3.45×10^{-1}	2.40×10^{-1}	5.46×10^{-2}	2.55×10^{-1}	1.63×10^{-1}	$\mathbf{3.90 \times 10^{-2}}$	$\mathbf{2.00 \times 10^{-1}}$	$\mathbf{1.16 \times 10^{-1}}$	4.80×10^{-2}
f5	6.68×10^{-2}	2.54×10^{-2}	1.98×10^{-2}	7.62×10^{-2}	4.46×10^{-2}	$\mathbf{1.20 \times 10^{-2}}$	$\mathbf{2.68 \times 10^{-2}}$	$\mathbf{5.63 \times 10^{-3}}$	1.44×10^{-2}
f6	4.30×10^{-1}	3.80×10^{-1}	$\mathbf{2.70 \times 10^{-2}}$	3.26×10^{-1}	2.50×10^{-1}	4.61×10^{-2}	$\mathbf{2.81 \times 10^{-1}}$	$\mathbf{2.01 \times 10^{-1}}$	6.49×10^{-2}
f7	2.59×10^{-1}	1.96×10^{-1}	3.70×10^{-2}	1.98×10^{-1}	1.36×10^{-1}	2.82×10^{-2}	$\mathbf{1.73 \times 10^{-1}}$	$\mathbf{1.32 \times 10^{-1}}$	$\mathbf{1.25 \times 10^{-1}}$
f8	4.69×10^{0}	$\mathbf{1.11 \times 10^{-1}}$	2.95×10^{0}	4.42×10^{0}	1.83×10^{0}	$\mathbf{1.59 \times 10^{0}}$	$\mathbf{3.37 \times 10^{0}}$	7.49×10^{-1}	2.06×10^{0}
f9	9.68×10^{0}	7.43×10^{0}	$\mathbf{1.17 \times 10^{0}}$	7.17×1^{00}	4.82×1^{00}	$\mathbf{5.09 \times 10^{0}}$	$\mathbf{6.82 \times 10^{0}}$	$\mathbf{4.56 \times 10^{0}}$	1.30×10^{0}
f10	4.07×10^{-1}	2.77×10^{-1}	6.08×10^{-2}	$\mathbf{3.14 \times 10^{-1}}$	2.25×10^{-1}	$\mathbf{4.61 \times 10^{-2}}$	3.98×10^{-1}	$\mathbf{2.19 \times 10^{-1}}$	1.06×10^{-1}
f11	$\mathbf{3.01 \times 10^{+1}}$	$\mathbf{3.25 \times 10^{+1}}$	$3.26 \times 10^{+1}$	$3.13 \times 10^{+1}$	$3.15 \times 10^{+1}$	$3.15 \times 10^{+1}$	$3.40 \times 10^{+1}$	$3.11 \times 10^{+1}$	$3.10 \times 1^{0+1}$
f12	$7.20 \times 10^{+2}$	$3.72 \times 10^{+2}$	$8.30 \times 10^{+2}$	$1.11 \times 10^{+2}$	$4.52 \times 10^{+1}$	$\mathbf{4.62 \times 10^{+1}}$	$\mathbf{9.90 \times 10^{+1}}$	$\mathbf{2.39 \times 10^{+1}}$	$8.46 \times 10^{+1}$
f13	$7.34 \times 10^{+1}$	8.89×10^{0}	$6.28 \times 10^{+1}$	$\mathbf{2.59 \times 10^{+1}}$	$\mathbf{5.45 \times 10^{-1}}$	$\mathbf{2.66 \times 10^{+1}}$	$3.06 \times 10^{+1}$	1.59×10^{0}	$3.44 \times 10^{+1}$

Table A3. *Cont.*

Funs	FPA			PSO			EAOA		
	Mean	**Best**	**Std.**	**Mean**	**Best**	**Std.**	**Mean**	**Best**	**Std.**
f14	$3.03 \times 10^{+2}$	$8.32 \times 10^{+1}$	$2.17 \times 10^{+2}$	$4.57 \times 10^{+1}$	$1.92 \times 10^{+1}$	$2.71 \times 10^{+1}$	$\mathbf{1.10 \times 10^{+1}}$	$\mathbf{1.32 \times 10^{0}}$	$\mathbf{1.14 \times 10^{+1}}$
f15	2.01×10^{0}	1.09×10^{0}	$\mathbf{3.70 \times 10^{-1}}$	2.01×10^{0}	1.26×10^{0}	4.51×10^{-1}	$\mathbf{1.73 \times 10^{0}}$	$\mathbf{7.59 \times 10^{-1}}$	4.95×10^{-1}
f16	7.50×10^{-1}	2.41×10^{-1}	$\mathbf{2.38 \times 10^{-1}}$	6.40×10^{-1}	1.93×10^{-1}	2.80×10^{-1}	$\mathbf{6.03 \times 10^{-1}}$	$\mathbf{1.36 \times 10^{-1}}$	2.70×10^{-1}
f17	$\mathbf{9.05 \times 10^{+1}}$	$5.47 \times 10^{+1}$	$1.23 \times 10^{+2}$	$9.08 \times 10^{+1}$	$\mathbf{1.09 \times 10^{+1}}$	$\mathbf{8.55 \times 10^{+1}}$	$1.02 \times 10^{+2}$	$1.68 \times 10^{+1}$	$1.17 \times 10^{+2}$
f18	$1.50 \times 10^{+2}$	$4.30 \times 10^{+1}$	$1.32 \times 10^{+2}$	$2.42 \times 10^{+1}$	1.64×10^{0}	$2.35 \times 10^{+1}$	$\mathbf{1.38 \times 10^{0}}$	$\mathbf{1.37 \times 10^{-1}}$	$\mathbf{1.84 \times 10^{0}}$
f19	7.85×10^{-1}	4.36×10^{-1}	2.39×10^{-1}	$\mathbf{7.69 \times 10^{-1}}$	5.01×10^{-1}	$\mathbf{1.78 \times 10^{-1}}$	8.33×10^{-1}	$\mathbf{4.03 \times 10^{-1}}$	2.91×10^{-1}
f20	6.30×10^{-1}	5.39×10^{-1}	4.59×10^{-2}	5.71×10^{-1}	4.96×10^{-1}	$\mathbf{4.14 \times 10^{-2}}$	$\mathbf{5.02 \times 10^{-1}}$	$\mathbf{4.16 \times 10^{-1}}$	4.53×10^{-2}
f21	1.45×10^{0}	2.33×10^{-1}	3.18×10^{0}	7.42×10^{-1}	$\mathbf{2.04 \times 10^{-1}}$	2.02×10^{0}	$\mathbf{3.08 \times 10^{-1}}$	2.20×10^{-1}	$\mathbf{8.66 \times 10^{-2}}$
f22	1.03×10^{0}	$\mathbf{7.09 \times 10^{-1}}$	9.12×10^{-1}	9.97×10^{-1}	8.73×10^{-1}	9.76×10^{-1}	$\mathbf{7.80 \times 10^{-1}}$	7.10×10^{-1}	$\mathbf{4.83 \times 10^{-2}}$
f23	1.09×10^{0}	9.27×10^{-1}	7.47×10^{-2}	1.05×10^{0}	8.69×10^{-1}	8.62×10^{-2}	$\mathbf{8.87 \times 10^{-1}}$	$\mathbf{7.93 \times 10^{-1}}$	$\mathbf{5.98 \times 10^{-2}}$
f24	7.17×10^{-1}	6.52×10^{-1}	$\mathbf{3.44 \times 10^{-2}}$	$\mathbf{7.17 \times 10^{-1}}$	6.61×10^{-1}	3.69×10^{-2}	7.32×10^{-1}	$\mathbf{6.39 \times 10^{-1}}$	4.96×10^{-2}
f25	3.91×10^{0}	6.47×10^{-1}	2.82×10^{0}	$\mathbf{3.26 \times 10^{0}}$	$\mathbf{5.36 \times 10^{-1}}$	2.80×10^{0}	3.60×10^{0}	8.08×10^{-1}	$\mathbf{1.57 \times 10^{0}}$
f26	9.56×10^{-1}	8.58×10^{-1}	8.74×10^{-2}	9.93×10^{-1}	8.44×10^{-1}	7.07×10^{-2}	$\mathbf{8.66 \times 10^{-1}}$	$\mathbf{8.12 \times 10^{-1}}$	$\mathbf{2.90 \times 10^{-2}}$
f27	7.98×10^{-1}	7.24×10^{-1}	$\mathbf{3.70 \times 10^{-2}}$	$\mathbf{7.86 \times 10^{-1}}$	$\mathbf{6.96 \times 10^{-1}}$	4.20×10^{-2}	8.54×10^{-1}	7.74×10^{-1}	5.43×10^{-2}
f28	2.03×10^{0}	1.31×10^{0}	4.27×10^{-1}	2.38×10^{0}	1.47×10^{0}	4.32×10^{-1}	$\mathbf{1.56 \times 10^{0}}$	$\mathbf{9.81 \times 10^{-1}}$	$\mathbf{3.49 \times 10^{-1}}$
f29	$5.47 \times 10^{+3}$	$1.89 \times 10^{+3}$	$2.54 \times 10^{+3}$	$2.89 \times 10^{+3}$	$4.93 \times 10^{+2}$	$1.89 \times 10^{+3}$	$\mathbf{3.71 \times 10^{+2}}$	$\mathbf{1.01 \times 10^{+2}}$	$\mathbf{3.28 \times 10^{+2}}$
Win	5	5	6	7	7	10	18	18	13
Lose	23	23	21	21	21	12	11	10	16
Draw	3	3	2	1	1	1	0	1	0

The bold data values in each row of the Table are the best ones in each pair compared with the EAOA approach.

Table A4. The performance presentation of the EAOA, MFO, and SCA for the CEC 2017 test suite with each paired comparison.

Funs	MFO			SCA			EAOA		
	Mean	**Best**	**Std.**	**Mean**	**Best**	**Std.**	**Mean**	**Best**	**Std.**
f1	4.60×10^{-1}	2.99×10^{-1}	7.04×10^{-1}	2.41×10^{-1}	$\mathbf{1.23 \times 10^{-1}}$	4.80×10^{-1}	$\mathbf{1.34 \times 10^{-1}}$	2.83×10^{-1}	$\mathbf{1.16 \times 10^{-1}}$
f2	$2.33 \times 10^{+2}$	$9.80 \times 10^{+1}$	$1.30 \times 10^{+2}$	$\mathbf{1.41 \times 10^{+1}}$	$9.03 \times 10^{+1}$	$2.65 \times 10^{+1}$	$3.73 \times 10^{+1}$	$\mathbf{1.93 \times 10^{+1}}$	$\mathbf{1.16 \times 10^{+1}}$
f3	6.57×10^{0}	4.83×10^{0}	1.17×10^{0}	2.97×10^{-1}	$\mathbf{1.36 \times 10^{-1}}$	8.95×10^{-1}	$\mathbf{2.43 \times 10^{-1}}$	1.50×10^{-1}	$\mathbf{5.77 \times 10^{-2}}$
f4	6.23×10^{-1}	5.46×10^{-1}	4.52×10^{-2}	5.24×10^{-1}	4.54×10^{-1}	$\mathbf{3.69 \times 10^{-2}}$	$\mathbf{2.00 \times 10^{-1}}$	$\mathbf{1.16 \times 10^{-1}}$	4.80×10^{-2}
f5	1.38×10^{-1}	1.21×10^{-1}	$\mathbf{1.43 \times 10^{-2}}$	$\mathbf{6.61 \times 10^{-2}}$	6.50×10^{-2}	1.54×10^{-2}	6.68×10^{-2}	$\mathbf{5.63 \times 10^{-3}}$	1.44×10^{-2}
f6	1.79×10^{0}	$\mathbf{1.34 \times 10^{-1}}$	1.66×10^{0}	9.48×10^{-1}	7.89×10^{-1}	9.52×10^{-2}	$\mathbf{2.81 \times 10^{-1}}$	2.01×10^{-1}	$\mathbf{6.49 \times 10^{-2}}$
f7	5.83×10^{-1}	5.16×10^{-1}	2.83×10^{-2}	4.86×10^{-1}	3.81×10^{-1}	$\mathbf{2.13 \times 10^{-2}}$	$\mathbf{1.73 \times 10^{-1}}$	$\mathbf{1.32 \times 10^{-1}}$	2.54×10^{-2}
f8	$\mathbf{2.38 \times 10^{-1}}$	1.78×10^{0}	3.28×10^{0}	$1.17 \times 10^{+1}$	5.84×10^{0}	3.05×10^{0}	3.37×10^{0}	$\mathbf{7.49 \times 10^{-1}}$	$\mathbf{2.06 \times 10^{0}}$
f9	9.99×10^{0}	9.31×10^{0}	$\mathbf{3.36 \times 10^{-1}}$	$1.23 \times 10^{+1}$	$1.05 \times 10^{+1}$	6.06×10^{-1}	$\mathbf{6.82 \times 10^{0}}$	$\mathbf{4.56 \times 10^{0}}$	1.30×10^{0}
f10	7.53×10^{0}	3.80×10^{0}	1.73×10^{0}	$\mathbf{3.00 \times 10^{-1}}$	$\mathbf{1.14 \times 10^{-1}}$	9.63×10^{-1}	3.98×10^{-1}	2.19×10^{-1}	$\mathbf{1.06 \times 10^{-1}}$
f11	$2.90 \times 10^{+6}$	$1.62 \times 10^{+6}$	$5.79 \times 10^{+5}$	$1.78 \times 10^{+6}$	$9.66 \times 10^{+5}$	$\mathbf{5.17 \times 10^{+5}}$	$\mathbf{7.92 \times 10^{+5}}$	$\mathbf{1.10 \times 10^{+5}}$	$8.45 \times 10^{+5}$
f12	$7.12 \times 10^{+5}$	$3.18 \times 10^{+5}$	$2.50 \times 10^{+5}$	$3.12 \times 10^{+5}$	$\mathbf{8.18 \times 10^{+4}}$	$2.17 \times 10^{+5}$	$\mathbf{9.90 \times 10^{+4}}$	$8.39 \times 10^{+4}$	$\mathbf{8.46 \times 10^{+4}}$
f13	$2.14 \times 10^{+2}$	$\mathbf{2.04 \times 10^{+1}}$	$9.37 \times 10^{+1}$	$7.63 \times 10^{+2}$	$2.93 \times 10^{+1}$	$5.99 \times 10^{+2}$	$\mathbf{3.06 \times 10^{+1}}$	$2.59 \times 10^{+1}$	$\mathbf{3.44 \times 10^{+1}}$
f14	$1.93 \times 10^{+4}$	$1.54 \times 10^{+3}$	$9.89 \times 10^{+3}$	$7.40 \times 10^{+3}$	$9.60 \times 10^{+2}$	$4.59 \times 10^{+3}$	$\mathbf{1.10 \times 10^{+1}}$	$\mathbf{1.32 \times 10^{0}}$	$\mathbf{1.14 \times 10^{+1}}$
f15	3.58×10^{0}	$\mathbf{3.14 \times 10^{-1}}$	2.28×10^{0}	$\mathbf{3.76 \times 10^{-1}}$	2.59×10^{0}	4.23×10^{0}	1.73×10^{0}	7.59×10^{-1}	$\mathbf{4.95 \times 10^{-1}}$
f16	1.42×10^{0}	1.09×10^{0}	$\mathbf{1.39 \times 10^{-1}}$	1.43×10^{0}	6.64×10^{-1}	3.42×10^{-1}	$\mathbf{6.03 \times 10^{-1}}$	$\mathbf{1.36 \times 10^{-1}}$	2.70×10^{-1}
f17	$\mathbf{3.40 \times 10^{+2}}$	$8.93 \times 10^{+2}$	$1.31 \times 10^{+3}$	$9.63 \times 10^{+3}$	$1.88 \times 10^{+3}$	$7.49 \times 10^{+3}$	$5.03 \times 10^{+2}$	$\mathbf{8.25 \times 10^{+1}}$	$\mathbf{5.78 \times 10^{+2}}$
f18	$5.66 \times 10^{+4}$	$2.32 \times 10^{+4}$	$2.43 \times 10^{+4}$	$1.32 \times 10^{+4}$	$3.22 \times 10^{+3}$	$7.60 \times 10^{+3}$	$\mathbf{1.15 \times 10^{+1}}$	$\mathbf{1.14 \times 10^{0}}$	$\mathbf{1.53 \times 10^{+1}}$
f19	1.17×10^{0}	9.15×10^{-1}	$\mathbf{1.15 \times 10^{-1}}$	1.17×10^{0}	6.21×10^{-1}	2.66×10^{-1}	$\mathbf{8.33 \times 10^{-1}}$	$\mathbf{4.03 \times 10^{-1}}$	2.91×10^{-1}
f20	$\mathbf{8.92 \times 10^{-2}}$	8.28×10^{-2}	$\mathbf{9.98 \times 10^{-1}}$	8.06×10^{-1}	7.10×10^{-1}	4.72×10^{-2}	5.02×10^{-1}	$\mathbf{4.16 \times 10^{-2}}$	4.53×10^{-2}
f21	9.34×10^{0}	7.19×10^{0}	1.04×10^{0}	3.19×10^{0}	$\mathbf{1.95 \times 10^{-1}}$	5.98×10^{-1}	$\mathbf{3.08 \times 10^{-1}}$	2.20×10^{-1}	$\mathbf{8.66 \times 10^{-2}}$
f22	1.28×10^{0}	$\mathbf{1.22 \times 10^{-1}}$	4.01×10^{-1}	1.14×10^{0}	1.05×10^{0}	$\mathbf{4.84 \times 10^{-2}}$	$\mathbf{7.80 \times 10^{-1}}$	$\mathbf{7.10 \times 10^{-1}}$	4.83×10^{-1}

Table A4. *Cont.*

Funs	MFO			SCA			EAOA		
	Mean	**Best**	**Std.**	**Mean**	**Best**	**Std.**	**Mean**	**Best**	**Std.**
f23	1.38×10^{0}	1.27×10^{0}	6.06×10^{-2}	1.26×10^{0}	1.13×10^{0}	$\mathbf{5.20 \times 10^{-2}}$	$\mathbf{8.87 \times 10^{-1}}$	$\mathbf{7.93 \times 10^{-1}}$	5.98×10^{-2}
f24	3.78×10^{0}	2.27×10^{0}	5.41×10^{-1}	$\mathbf{1.72 \times 10^{-1}}$	1.28×10^{-1}	$\mathbf{2.58 \times 10^{-1}}$	7.32×10^{-1}	$\mathbf{6.39 \times 10^{-2}}$	4.96×10^{-1}
f25	8.23×10^{0}	5.94×10^{0}	$\mathbf{9.24 \times 10^{-1}}$	6.57×10^{0}	3.22×10^{0}	1.84×10^{0}	$\mathbf{3.60 \times 10^{0}}$	$\mathbf{8.08 \times 10^{-1}}$	1.57×10^{0}
f26	1.20×10^{0}	1.13×10^{0}	3.57×10^{-1}	$\mathbf{1.20 \times 10^{-1}}$	8.15×10^{-1}	1.74×10^{-1}	8.66×10^{-1}	$\mathbf{8.12 \times 10^{-2}}$	2.90×10^{-1}
f27	3.39×10^{0}	$\mathbf{2.15 \times 10^{-1}}$	6.11×10^{-1}	2.06×10^{0}	4.31×10^{-1}	8.16×10^{-1}	$\mathbf{8.54 \times 10^{-1}}$	7.74×10^{-1}	$\mathbf{5.43 \times 10^{-1}}$
f28	3.20×10^{0}	2.76×10^{0}	$\mathbf{2.16 \times 10^{-1}}$	3.17×10^{0}	1.92×10^{0}	4.95×10^{-1}	$\mathbf{1.56 \times 10^{0}}$	$\mathbf{9.81 \times 10^{-1}}$	3.49×10^{-1}
f29	$8.17 \times 10^{+4}$	$5.16 \times 10^{+4}$	$2.12 \times 10^{+4}$	$4.90 \times 10^{+4}$	$1.70 \times 10^{+4}$	$2.05 \times 10^{+4}$	$\mathbf{3.71 \times 10^{+2}}$	$\mathbf{1.51 \times 10^{+2}}$	$\mathbf{3.28 \times 10^{+2}}$
Win	4	5	6	7	6	6	21	20	17
Lose	23	21	21	21	22	23	8	9	14
Draw	2	3	2	1	1	0	0	0	0

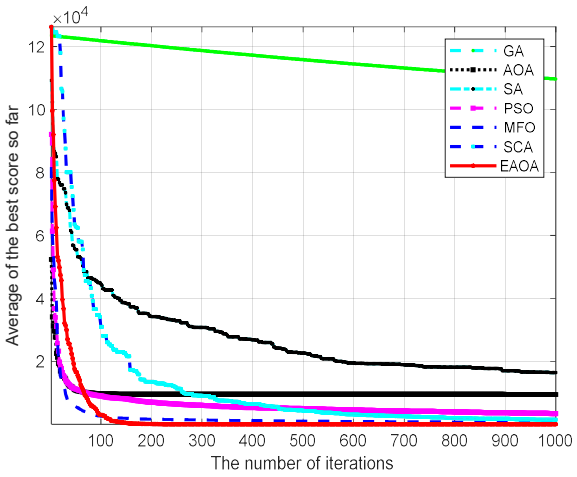
The bold data values in each row of the Table are the best ones in each pair compared with the EAOA approach.

Table A5. Wilcoxon signed-rank results of the test pairs of the pairwise algorithms' results between the EAOA and other algorithms, i.e., PBA [33], WOA [36], PPSO [29], AOA [41], IFMO [35], and ESCA [40].

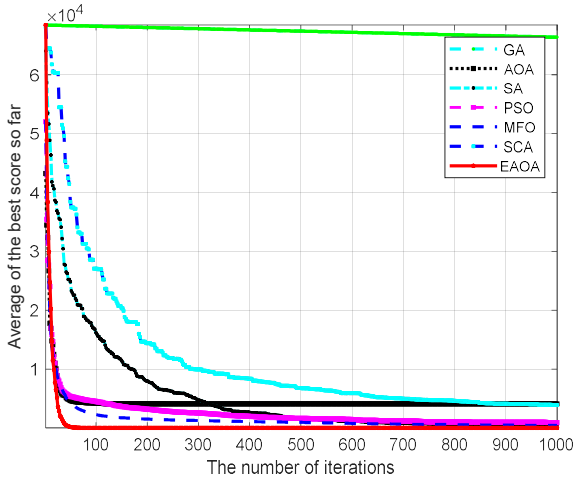
Funs	**PBA [33]**	**WOA [36]**	**PPSO [29]**	**AOA [41]**	**IFMO [35]**	**ESCA [40]**	**EAOA-Itself**
f1	$\mathbf{2.4018 \times 10^{-7}}$	1.4018×10^{-11}	1.7018×10^{-11}	1.1205×10^{-5}	6.9641×10^{-8}	2.5668×10^{-7}	~N/A
f2	1.4018×10^{-11}	1.4018×10^{-11}	1.4018×10^{-11}	2.2080×10^{-7}	7.1665×10^{-3}	1.3749×10^{-2}	~N/A
f3	1.4018×10^{-11}	1.4018×10^{-11}	8.5710×10^{-11}	1.8717×10^{-2}	1.8717×10^{-2}	4.6578×10^{-3}	~N/A
f4	1.4018×10^{-11}	1.4018×10^{-11}	1.4018×10^{-11}	4.8753×10^{-11}	1.5456×10^{-2}	2.7237×10^{-5}	~N/A
f5	1.4018×10^{-11}	1.5447×10^{-11}	1.4018×10^{-11}	1.8376×10^{-9}	5.3326×10^{-5}	4.0332×10^{-11}	~N/A
f6	1.4018×10^{-11}	1.4018×10^{-11}	1.4018×10^{-11}	4.4659×10^{-10}	6.5678×10^{-4}	9.8637×10^{-4}	~N/A
f7	1.4018×10^{-11}	1.4018×10^{-11}	1.7018×10^{-11}	9.4096×10^{-11}	1.6922×10^{-3}	6.2370×10^{-4}	~N/A
f8	1.4018×10^{-11}	3.6674×10^{-11}	1.8745×10^{-11}	$\mathbf{8.0856 \times 10^{-2}}$	$\mathbf{1.0244 \times 10^{-1}}$	1.2706×10^{-2}	~N/A
f9	4.8753×10^{-11}	1.4018×10^{-11}	1.2873×10^{-8}	2.0041×10^{-9}	3.9045×10^{-1}	$\mathbf{2.6004 \times 10^{-1}}$	~N/A
f10	1.4018×10^{-11}	1.4018×10^{-11}	1.4018×10^{-11}	$\mathbf{5.8296 \times 10^{-1}}$	9.7754×10^{-1}	1.5366×10^{-3}	~N/A
f11	1.4018×10^{-11}	1.4018×10^{-11}	1.4018×10^{-11}	1.5439×10^{-9}	$\mathbf{1.4703 \times 10^{-1}}$	3.1620×10^{-6}	~N/A
f12	1.4018×10^{-11}	1.4018×10^{-11}	6.4699×10^{-11}	1.5447×10^{-11}	1.3749×10^{-2}	4.1212×10^{-2}	~N/A
f13	7.1071×10^{-11}	7.8055×10^{-11}	7.1071×10^{-11}	1.2847×10^{-4}	$\mathbf{8.7693 \times 10^{-1}}$	$\mathbf{8.2178 \times 10^{-1}}$	~N/A
f14	1.4018×10^{-11}	1.4018×10^{-11}	2.5021×10^{-11}	1.4018×10^{-11}	3.7194×10^{-2}	3.6588×10^{-9}	~N/A
f15	1.4018×10^{-11}	1.4018×10^{-11}	4.0332×10^{-11}	2.9096×10^{-2}	7.2487×10^{-1}	7.3779×10^{-2}	~N/A
f16	1.4018×10^{-11}	3.7291×10^{-10}	$\mathbf{6.1854 \times 10^{-1}}$	2.7082×10^{-2}	7.3779×10^{-2}	6.3217×10^{-1}	~N/A
f17	7.1071×10^{-11}	1.8745×10^{-11}	1.6408×10^{-10}	1.4729×10^{-6}	9.8877×10^{-1}	$\mathbf{7.2487 \times 10^{-1}}$	~N/A
f18	1.4018×10^{-11}	1.4018×10^{-11}	1.4018×10^{-11}	1.4018×10^{-11}	$\mathbf{6.1228 \times 10^{-1}}$	6.4699×10^{-11}	~N/A
f19	2.7567×10^{-6}	5.3326×10^{-5}	1.8183×10^{-6}	$\mathbf{4.8148 \times 10^{-1}}$	$\mathbf{8.9917 \times 10^{-1}}$	$\mathbf{4.6412 \times 10^{-1}}$	~N/A
f20	1.4018×10^{-11}	1.4018×10^{-11}	1.8745×10^{-11}	1.0328×10^{-10}	2.5189×10^{-2}	4.3218×10^{-7}	~N/A
f21	1.4018×10^{-11}	1.4018×10^{-11}	1.4018×10^{-11}	$\mathbf{3.3134 \times 10^{-1}}$	7.4745×10^{-3}	6.9641×10^{-8}	~N/A
f22	1.4018×10^{-11}	1.4018×10^{-11}	1.4018×10^{-11}	2.7539×10^{-11}	3.7194×10^{-2}	2.5021×10^{-11}	~N/A
f23	1.4018×10^{-11}	1.4018×10^{-11}	1.4018×10^{-11}	9.4096×10^{-11}	$\mathbf{2.2599 \times 10^{-1}}$	2.5970×10^{-9}	~N/A
f24	1.4018×10^{-11}	1.4018×10^{-11}	7.8055×10^{-11}	$\mathbf{2.0014 \times 10^{-1}}$	1.8717×10^{-2}	$\mathbf{1.7206 \times 10^{-1}}$	~N/A
f25	2.2729×10^{-11}	1.3989×10^{-7}	1.3643×10^{-10}	$\mathbf{4.4711 \times 10^{-1}}$	$\mathbf{8.2178 \times 10^{-1}}$	$\mathbf{3.1751 \times 10^{-1}}$	~N/A
f26	1.4018×10^{-11}	3.9843×10^{-9}	3.0304×10^{-11}	1.1106×10^{-7}	2.8074×10^{-2}	2.3679×10^{-10}	~N/A
f27	1.4018×10^{-11}	9.4096×10^{-11}	5.8443×10^{-10}	2.1213×10^{-5}	5.9218×10^{-4}	2.2408×10^{-6}	~N/A
f28	1.4018×10^{-11}	2.2729×10^{-11}	1.4018×10^{-11}	7.1825×10^{-5}	2.0181×10^{-2}	4.3379×10^{-9}	~N/A
f29	1.4018×10^{-11}	1.4018×10^{-11}	1.4018×10^{-11}	$1.4018 \times 10-^{11}$	6.2370×10^{-4}	7.8055×10^{-11}	~N/A
Avg.	6.5517	5.7241	5.6207	3.6207	2.5138	2.5483	2.25204
Rank	7	6	5	4	2	3	1

The bold data values in each row of the Table are the best ones in each pair compared with the EAOA approach.

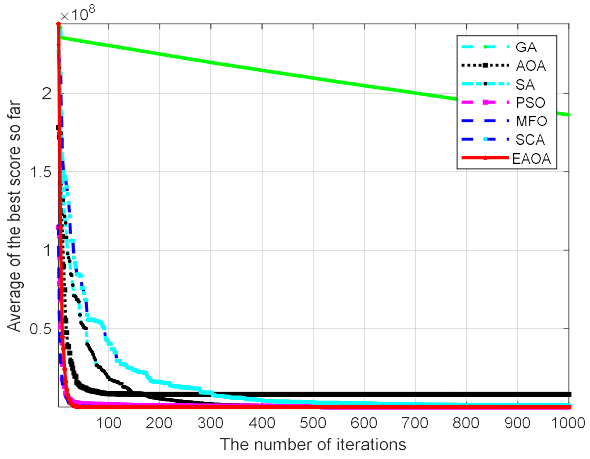
Appendix B



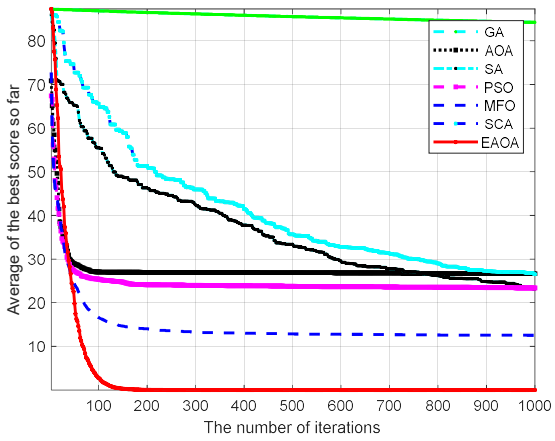
(**a**) f_1



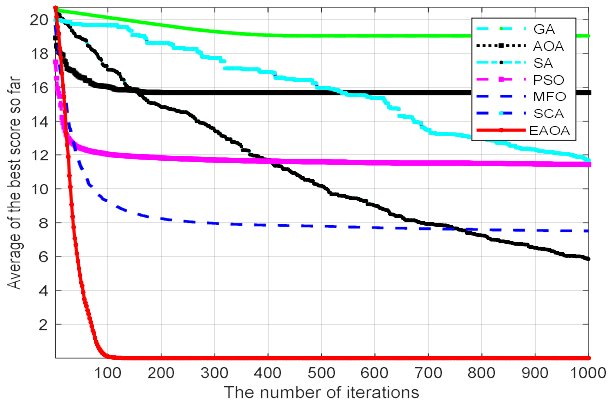
(**b**) f_2



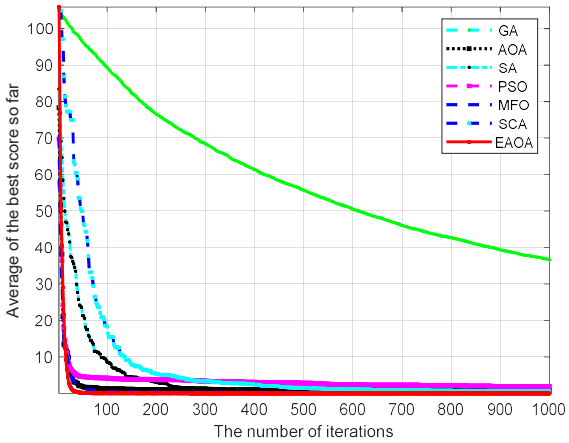
(**c**) f_3



(**d**) f_4

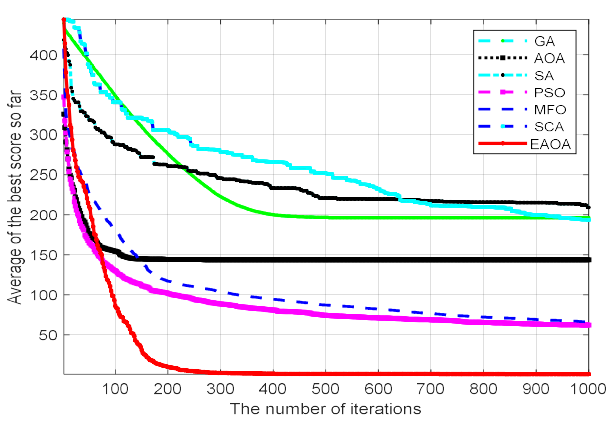


(**e**) f_5

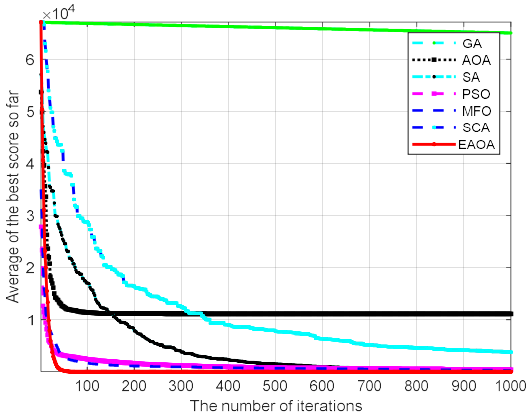


(**f**) f_6

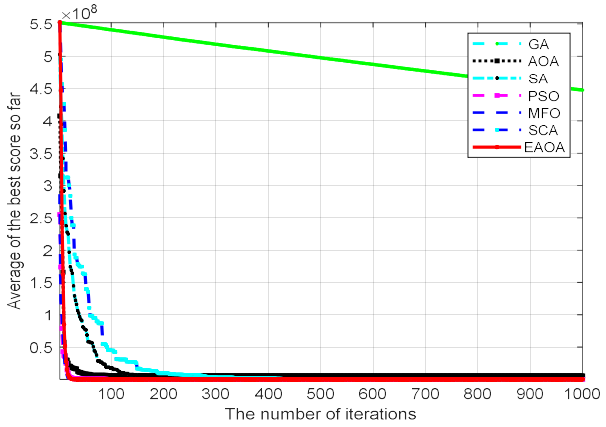
Figure A1. *Cont.*



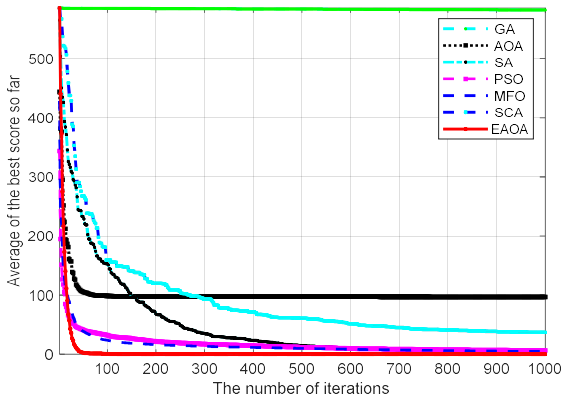
(**g**) f7



(**h**) f8



(**i**) f9



(**j**) f10

Figure A1. The EAOA's convergence output curves represented graphically and compared to those of the GA, AOA, SA, FPA, PSO, MFO, and SCA algorithms for the selected functions. The green line is the set background of the worst one; here, the green line is the GA method.

References

1. Yick, J.; Mukherjee, B.; Ghosal, D. Wireless sensor network survey. *Comput. Netw.* **2008**, *52*, 2292–2330. [CrossRef]
2. Qiao, Y.; Dao, T.K.; Pan, J.S.; Chu, S.C.; Nguyen, T.T. Diversity teams in soccer league competition algorithm for wireless sensor network deployment problem. *Symmetry* **2020**, *12*, 445. [CrossRef]
3. Dao, T.K.; Yu, J.; Nguyen, T.T.; Ngo, T.G. A Hybrid Improved MVO and FNN for Identifying Collected Data Failure in Cluster Heads in WSN. *IEEE Access* **2020**, *8*, 124311–124322. [CrossRef]
4. Dao, T.K.; Nguyen, T.T.; Pan, J.S.; Qiao, Y.; Lai, Q. Identification Failure Data for Cluster Heads Aggregation in WSN Based on Improving Classification of SVM. *IEEE Access* **2020**, *8*, 61070–61084. [CrossRef]
5. Ali, A.; Ming, Y.; Chakraborty, S.; Iram, S. A comprehensive survey on real-time applications of WSN. *Future internet* **2017**, *9*, 77. [CrossRef]
6. Chu, S.C.; Dao, T.K.; Pan, J.S.; Nguyen, T.T. Identifying correctness data scheme for aggregating data in cluster heads of wireless sensor network based on naive Bayes classification. *Eurasip J. Wirel. Commun. Netw.* **2020**, 52. [CrossRef]
7. Nguyen, T.T.; Pan, J.S.; Dao, T.K. An Improved Flower Pollination Algorithm for Optimizing Layouts of Nodes in Wireless Sensor Network. *IEEE Access* **2019**, *7*, 75985–75998. [CrossRef]
8. Chai, Q.-W.; Chu, S.-C.; Pan, J.-S.; Zheng, W.-M. Applying Adaptive and Self Assessment Fish Migration Optimization on Localization of Wireless Sensor Network on 3-D Terrain. *J. Inf. Hiding Multim. Signal Process.* **2020**, *11*, 90–102.
9. Kulkarni, R.V.R.; Ferster, A.; Venayagamoorthy, G.K. Computational intelligence in wireless sensor networks: A survey. *IEEE Commun. Surv. Tutor.* **2011**, *13*, 68–96. [CrossRef]
10. Li, Z.; Chu, S.-C.; Pan, J.-S.; Hu, P.; Xue, X. A Mahalanobis Surrogate-Assisted Ant Lion Optimization and Its Application in 3D Coverage of Wireless Sensor Networks. *Entropy* **2022**, *24*, 586. [CrossRef]

11. Pan, J.-S.; Dao, T.-K.; Pan, T.-S.; Nguyen, T.-T.; Chu, S.-C.; Roddick, J.F. An improvement of flower pollination algorithm for node localization optimization in WSN. *J. Inf. Hiding Multimed. Signal. Process.* **2017**, *8*, 486–499.
12. Pan, J.-S.; Sun, X.-X.; Chu, S.-C.; Abraham, A.; Yan, B. Digital watermarking with improved SMS applied for QR code. *Eng. Appl. Artif. Intell.* **2021**, *97*, 104049. [CrossRef]
13. Nguyen, T.-T.; Pan, J.-S.; Dao, T.-K.; Sung, T.-W.; Ngo, T.-G. Pigeon-Inspired Optimization for Node Location in Wireless Sensor Network. In Proceedings of the International Conference on Engineering Research and Applications, Thai Nguyen, Vietnam, 1–2 December 2019; Volume 104, pp. 589–598.
14. Nguyen, T.-T.; Dao, T.-K.; Kao, H.-Y.; Horng, M.-F.; Shieh, C.-S. Hybrid Particle Swarm Optimization with Artificial Bee Colony Optimization for Topology Control Scheme in Wireless Sensor Networks. *J. Internet Technol.* **2017**, *18*, 743–752. [CrossRef]
15. Nguyen, T.-T.; Pan, J.-S.; Wu, T.-Y.; Dao, T.-K.; Nguyen, T.-D. Node Coverage Optimization Strategy Based on Ions Motion Optimization. *J. Netw. Intell.* **2019**, *4*, 2414–8105.
16. Pan, J.-S.; Nguyen, T.-T.; Dao, T.-K.; Pan, T.-S.; Chu, S.-C. Clustering Formation in Wireless Sensor Networks: A Survey. *J. Netw. Intell.* **2017**, *2*, 287–309.
17. Dao, T.K.; Pan, T.S.; Nguyen, T.T.; Chu, S.C. A compact Articial bee colony optimization for topology control scheme in wireless sensor networks. *J. Inf. Hiding Multimed. Signal Processing* **2015**, *6*, 297–310.
18. Othman, M.F.; Shazali, K. Wireless sensor network applications: A study in environment monitoring system. *Procedia Eng.* **2012**, *41*, 1204–1210. [CrossRef]
19. Liu, N.; Pan, J.S.; Nguyen, T.T. A bi-Population QUasi-Affine TRansformation Evolution algorithm for global optimization and its application to dynamic deployment in wireless sensor networks. *Eurasip J. Wirel. Commun. Netw.* **2019**, *2019*, 175. [CrossRef]
20. Mahdavi, S.; Shiri, M.E.; Rahnamayan, S. Metaheuristics in large-Scale global continues optimization: A survey. *Inf. Sci.* **2015**, *295*, 407–428. [CrossRef]
21. Shiva Prasad Yadav, S.G.; Chitra, A. Wireless Sensor Networks-Architectures, Protocols, Simulators and Applications: A Survey. *Int. J. Electron. Comput. Sci. Eng.* **2012**, *1*, 1941–1953.
22. Akyildiz, I.F.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. Wireless sensor networks: A survey. *Comput. Netw.* **2002**, *38*, 393–422. [CrossRef]
23. Pan, T.-S.; Dao, T.-K.; Nguyen, T.-T.; Chu, S.-C. Optimal base station locations in heterogeneous wireless sensor network based on hybrid particle swarm optimization with bat algorithm. *J. Comput.* **2014**, *25*, 14–25.
24. Van Laarhoven, P.J.M.; Aarts, E.H.L. Simulated annealing. In *Simulated Annealing: Theory and Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1987; pp. 7–15.
25. Whitley, D. A genetic algorithm tutorial. *Stat. Comput.* **1994**, *4*, 65–85. [CrossRef]
26. Srinivas, M.; Patnaik, L.M. Genetic Algorithms: A Survey. *Computer* **1994**, *27*, 17–26. [CrossRef]
27. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume 6, pp. 1942–1948.
28. Chu, S.A.; Tsai, P.W.; Pan, J.S. Cat swarm optimization. In Proceedings of the 9th Pacific Rim International Conference on Artificial Intelligence, Guilin, China, 7–11 August 2006; Volume 4099, pp. 854–858.
29. Chu, S.C.; Pan, J.-S. A parallel particle swarm optimization algorithm with communication strategies. *J. Inf. Sci. Eng.* **2005**, *21*, 9.
30. Dorigo, M.; Di Caro, G. Ant colony optimization: A new meta-Heuristic. In Proceedings of the 1999 Congress on Evolutionary Computation, CEC, Washington, DC, USA, 6–9 July 1999; Volume 2, pp. 1470–1477.
31. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
32. Yang, X.S. A new metaheuristic Bat-Inspired Algorithm. In *Studies in Computational Intelligence*; González, J., Pelta, D., Cruz, C., Terrazas, G., Krasnogor, N., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 284, pp. 65–74. ISBN 9783642125379.
33. Tsai, C.F.; Dao, T.K.; Yang, W.J.; Nguyen, T.T.; Pan, T.S. Parallelized bat algorithm with a communication strategy. In *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*; Ali, M., Pan, J.-S., Chen, S.-M., Horng, M.-F., Eds.; Springer International Publishing: Cham, Switzerland, 2014; Volume 8481, pp. 87–95.
34. Mirjalili, S. Moth-Flame optimization algorithm: A novel nature-Inspired heuristic paradigm. *Knowl.-Based Syst.* **2015**, *89*, 228–249. [CrossRef]
35. Nguyen, T.-T.; Wang, H.-J.; Dao, T.-K.; Pan, J.-S.; Ngo, T.-G.; Yu, J. A Scheme of Color Image Multithreshold Segmentation Based on Improved Moth-Flame Algorithm. *IEEE Access* **2020**, *8*, 174142–174159. [CrossRef]
36. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
37. Yang, X.S. Flower pollination algorithm for global optimization. In Proceedings of the Lecture Notes in Computer Science, Orléans, France, 3–7 September 2012; Volume 744, pp. 240–249.
38. Nguyen, T.T.; Shieh, C.S.; Horng, M.F.; Dao, T.K.; Ngo, T.G. Parallelized Flower Pollination Algorithm with a Communication Strategy. In Proceedings of the Proceedings-2015 IEEE International Conference on Knowledge and Systems Engineering, KSE, Ho Chi Minh City, Vietnam, 8–10 October 2015; pp. 103–107.
39. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
40. Chen, H.; Wang, M.; Zhao, X. A multi-strategy enhanced sine cosine algorithm for global optimization and constrained practical engineering problems. *Appl. Math. Comput.* **2020**, *369*, 124872. [CrossRef]

41. Hashim, F.A.; Hussain, K.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W. Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems. *Appl. Intell.* **2021**, *51*, 1531–1551. [CrossRef]
42. Tian, J.; Gao, M.; Ge, G. Wireless sensor network node optimal coverage based on improved genetic algorithm and binary ant colony algorithm. *EURASIP J. Wirel. Commun. Netw.* **2016**, 104. [CrossRef]
43. Wu, G.; Mallipeddi, R.; Suganthan, P.N. Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization; Technical Report 2017. 2017. Available online: https://www3.ntu.edu.sg/home/epnsugan/index_files/CEC2017/CEC2017.htm (accessed on 1 July 2021).
44. Chelliah, J.; Kader, N. Optimization for connectivity and coverage issue in target-based wireless sensor networks using an effective multiobjective hybrid tunicate and salp swarm optimizer. *Int. J. Commun. Syst.* **2021**, *34*, e4679. [CrossRef]
45. Ab Aziz, N.A.B.; Mohemmed, A.W.; Alias, M.Y. A wireless sensor network coverage optimization algorithm based on particle swarm optimization and Voronoi diagram. In Proceedings of the 2009 international conference on networking, sensing and control, Okayama, Japan, 26–29 March 2009; pp. 602–607.
46. Wang, Z.; Xie, H.; Hu, Z.; Li, D.; Wang, J.; Liang, W. Node coverage optimization algorithm for wireless sensor networks based on improved grey wolf optimizer. *J. Algorithms Comput. Technol.* **2019**, *13*, 1748302619889498. [CrossRef]
47. Fan, F.; Chu, S.-C.; Pan, J.-S.; Yang, Q.; Zhao, H. Parallel Sine Cosine Algorithm for the Dynamic Deployment in Wireless Sensor Networks. *J. Internet Technol.* **2021**, *22*, 499–512.
48. Hashim, F.A.; Houssein, E.H.; Hussain, K.; Mabrouk, M.S.; Al-Atabany, W. Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems. *Math. Comput. Simul.* **2022**, *192*, 84–110. [CrossRef]
49. Nguyen, T.-T.; Pan, J.-S.; Chu, S.-C.; Roddick, J.F.; Dao, T.-K. Optimization Localization in Wireless Sensor Network Based on Multi-Objective Firefly Algorithm. *J. Netw. Intell.* **2016**, *1*, 130–138.
50. Pan, J.-S.; Nguyen, T.-T.; Chu, S.-C.; Dao, T.-K.; Ngo, T.-G. Diversity enhanced ion motion optimization for localization in wireless sensor network. *J. Inf. Hiding Multimed. Signal Process.* **2019**, *10*, 221–229.
51. Nguyen, T.-T.; Yu, J.; Nguyen, T.-T.-T.; Dao, T.-K.; Ngo, T.-G. *A Solution to Sensor Node Localization Using Glow-Worm Swarm Optimization Hybridizing Positioning Model BT—Advances in Intelligent Information Hiding and Multimedia Signal Processing*; Pan, J.-S., Li, J., Ryu, K.H., Meng, Z., Klasnja-Milicevic, A., Eds.; Springer: Singapore, 2021; pp. 260–268.
52. Liu, N.; Pan, J.-S.; Wang, J.; Nguyen, T.-T. An adaptation multi-group quasi-affine transformation evolutionary algorithm for global optimization and its application in node localization in wireless sensor networks. *Sensors* **2019**, *19*, 4112. [CrossRef]