International Journal of Computational Intelligence Systems
Vol. **12(2)**, 2019, *pp.* 676–696
DOI: https://doi.org/10.2991/ijcis.d.190614.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

A Real-Coded Optimal Sensor Deployment Scheme for Wireless Sensor Networks Based on the Social Spider Optimization Algorithm

Fernando Fausto*, Erik Cuevas, Oscar Maciel-Castillo, Bernardo Morales-Castañeda

Departamento de Electrónica, Universidad de Guadalajara, CUCEI, Boulevard Marcelino García Barragán, No. 1421, Guadalajara, Jalisco, 44430, México

ARTICLE INFO

Article History
Received 25 Jan 2019
Accepted 22 May 2019

Keywords
Wireless sensor networks
Optimal sensor deployment
Social Spider Optimization
Metaheuristics

ABSTRACT

Wireless sensor networks (WSNs) involves a set of wireless sensor nodes located within a region of interest (ROI) to acquire and/or transmit specific information from their surroundings. A common problem in the operation of WSNs is sensor coverage, which is related to the distribution of sensor nodes within their ROI. Several approaches have been proposed to solve this problem; however, most of these methods consider a simplified arrangement scheme based on sensor placement over a set of fixed discrete locations defined by a grid. This fact severally limits the ability of these methods to find potentially better solutions as they are conditioned to select a limited number of candidate solutions. In this paper, a real-coded sensor deployment approach based on the Social Spider Optimization (SSO) algorithm is proposed to solve the problem of optimal sensor deployment (OSD) in WSNs. The performance of our proposed approach (referred in this paper as real-coded SSO [R-SSO]) was also compared against other metaheuristics-based methods used in the literature. Experimental results demonstrate its ability to solve the problem of OSD in terms of accuracy and robustness.

1. INTRODUCTION

The development of wireless communication technologies has led to a rapid increase in the use of wireless sensor networks (WSNs) for a vast number of applications, ranging from civilian to military [1]. WSNs consists of a set of wireless sensor devices (or sensor nodes) located within a specific region of interest (ROI) with the intention of acquiring or transmitting information from their surroundings [2–7]. Each sensor node is capable of sensing and processing data (see Figure 1). Furthermore, sensor nodes also have communication capabilities, which allow them to transmit and share information with other sensing units within the WSN. Nowadays, WSNs are relevant for many application domains, which include vehicular tracking, seismic activity observation, forest monitoring, target detection, and surveillance, among others [1].

Sensor coverage is one of the most studied problems related to the operation of WSN. In sensor coverage, the idea is to find the spatial configuration of sensor nodes within a given ROI so that the WSN reaches the best possible coverage of the ROI. In the design of a WSN, for practical reasons, it is common to distribute sensor nodes randomly.

However, this method does not guarantee sufficient coverage of the ROI, especially when sensors nodes adopt a configuration with small concentrations in specific locations of the service area. Since the coverage performance of a WSN depends on the spatial arrangement of the sensor nodes, techniques to properly distribute sensor nodes had been extensively studied and developed in the last few years; as a result, there are plenty of works on the literature devoted to this subject [8–16].

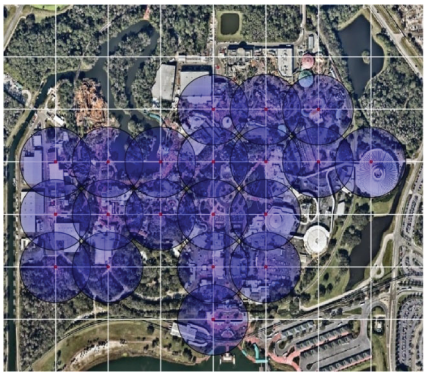

Figure 1 | Wireless Sensor Network composed of 20 sensor nodes (red dots), deployed for the monitoring of some specific data within several regions of interest in an amusement park. The coverage radius of each sensor (blue circles) is shown for illustrative purposes.

Recently, due to interesting results, the use of metaheuristic optimization techniques have received increasing attention as an alternative to solve the optimal sensor deployment (OSD) for WSNs.

*Corresponding author. Email: abraham.fausto@academicos.udg.mx

Some approaches include schemes such as the Self-Deployment method based on the implementation of the Virtual Force Algorithm (VFA) [7]. This scheme was proposed to improve sensor field coverage via the reconfiguration of randomly placed sensing units into uniformly distributed node topologies. Similarly, in [17], a sensor deployment approach based in Particle Swarm Optimization (PSO) has been applied with the objective to maximize sensor coverage over a given ROI, while also considering the minimization of energy usage in cluster-based network topologies. Also, in [18], a multiobjective implementation of Genetic Algorithms (GAs) [18] has been employed to optimize the configuration of WSNs with regard to two competing objectives: sensor coverage and network's lifetime. Furthermore [19], proposes a sensor deployment scheme based on a recent swarm optimization algorithm known as Social Spider Optimization (SSO). In this scheme, a simple binary sensor coverage model is implemented as part of the primary optimization criterion, yielding to good results. Although the previous sensor deployment approaches have demonstrated to produce competitive results, these methods often model the deployment area as a discrete grid, which for most cases comprises an unrealistic assumption. Under such conditions, the lattice presents a finite number of positions (grid points) in which sensor nodes can be placed. Intuitively, such assumption limits the flexibility of the sensor distribution approach, as it restricts the deployment of candidate locations to those modeled in the grid.

In this paper, we propose an SSO-based approach for OSD in WSNs. In our proposed method, the SSO algorithm is applied with the intention of finding an optimal arrangement for a set of available sensor nodes (that is, a configuration that enables to maximize the area covered by the sensor network). However, different to previous works on the literature which consider only a set of candidate positions defined by a 2-D grid, our approach considers the deployment area to be defined as a continuous flat surface; as such, instead of only considering a set of finite positions as candidate for placing a sensor node, our approach is able to place sensing devices in virtually any real location within the ROI. Also, instead of calculating the number of grid points that are covered by the deployed WSN, the total covered area is estimated by considering the intersection between the individual coverage area of each sensor node. Under these considerations, our proposed approach is able to handle some technical challenges related to optimal WSN deployment: first of all, considering real candidate locations instead of a set of finite discrete locations gives our proposed method the flexibility to explore potentially better solutions to the sensor deployment problem, while also increasing its performance when placement restrictions are introduced; in second place, the fact that the actual covered area is considered for calculating the coverage area rate allows to reduce the uncertainty of the model, as this gives us better insight on the actual amount of the ROI that is being monitored by the sensor network under certain arrangement configurations.

Our proposed approach has been exhaustively tested and compared with other similar methods. The experimental results reported in this work are presented three stages: first, we present a comparison between discrete-coded and real-coded OSD schemes based SSO; furthermore, we include an exhaustive comparison against different real-coded OSD approaches based on other popular metaheuristic optimization techniques such as Moth-flame Optimization (MFO) algorithm [20], Firefly Algorithm (FA) [21], PSO [22], Artificial Bee Colony (ABC) [23], Whale Optimization Algorithm (WOA) [24], Crow Search Algorithm (CSA) [25], and Grey Wolf Optimizer (GWO) [26]; finally, a comparison of these techniques with regard to a particular case of OSD, which involves the deployment of a set of sensor nodes within several specific disjointed ROIs.

The rest of this paper is organized as follows: in Section 2, we discuss some approaches commonly applied for calculating the rate of coverage in WSNs. In Section 3, we review the SSO algorithm, highlighting its main characteristics and steps. In Section 4, we describe our proposed SSO-based OSD approach, emphasizing its differences to previously proposed similar methods. In Section 5, we present our comparative analysis and results for the three previously specified sets of experiments. Finally, in Section 6, we present our conclusions for this work.

2. OSD FOR WSNs

WSNs consist of multiple sensors distributed through a specified sensing area with the purpose of acquiring some physical or environmental data. The main problem regarding the deployment of WSNs is the optimal placement of sensor nodes along a specified service area so that the resulting sensor network achieve sufficient coverage (i.e., that each location within the service area is monitored by at least one sensor unit). One of the most common models applied for OSD in WSNs is represented by the binary sensor model (BSM). According to this scheme, the coverage probability for any given position (x, y) within the deployment area is set to 1 if said position is at least partially covered by a sensor node; otherwise, its coverage probability is set as 0. In this sense, a sensor node s_i with location (x_i, y_i) within the deployment area is assumed to cover a specified location (x, y) if it is located within the sensor's coverage radius r_s [12,19]. In other words, the probability that a position (x, y) on the deployment area is covered by a sensor node s_i may be given by the following expression:

$$P(x, y, s_i) = \begin{cases} 1 & \text{if } \sqrt{(x - x_i)^2 + (y - y_i)^2} \leq r_s \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

For simplicity, most sensor deployment schemes assume that the experimental environment is represented by a 2-D grid $G(x, y)$, comprised of $M \times N$ discrete positions; if we consider that WSNs are constructed by deploying a set of sensor nodes $S = \{s_1, s_2, \ldots, s_{N_{\text{nodes}}}\}$ (with N_{nodes} denoting the total number of sensor nodes to be deployed), then the coverage probability for any location (x, y) within such an experimental environment, may be given as follows:

$$P(x, y, S) = 1 - \prod_{i=1}^{N_{\text{nodes}}} \left(1 - P(x, y, s_i)\right). \tag{2}$$

By considering the previous, the total area covered by the set of deployed sensor nodes may be given by adding up the coverage probabilities for all locations within the modeled 2-D grid; this is

$$A_{\text{cov}}(S) = \sum_{x=1}^{m} \sum_{y=1}^{n} P(x, y, S). \tag{3}$$

Finally, the deployment area's coverage rate is calculated by

$$R_{\text{cov}}(S) = \frac{A_{\text{cov}}(S)}{A_{\text{total}}}, \tag{4}$$

where $A_{\text{cov}}(S)$ denotes the area covered by the set of sensor nodes S (as given by Equation (3)), whereas $A_{\text{total}} = M \times N$ stands for the total area (number of locations) that comprise the experimental environment.

3. SOCIAL SPIDER OPTIMIZATION

The SSO algorithm is a swarm intelligence approach proposed by Cuevas *et al.* in 2013. As its name implies, the SSO approach draws inspiration on the collective behaviors manifested by certain species of spiders known for living in social units or colonies [27–30]. In nature, social spider colonies are mainly composed by two components: its members, which may be further distinguished by their gender (male or female), and a communal web which serves as a medium for interaction and communication among its inhabitants. Depending on their gender and particular characteristics, each member of the colony is assigned to cooperate in several activities, including building and maintaining the communal web, capturing prey, mating, and so on. Another interesting trait related to Social Spiders lies on their capacity to perceive the vibrations transmitted through the communal web. Each member on the colony can translate these vibrations into important information related to their surroundings, including the location and size of both, neighboring members and prey trapped in the communal web.

3.1. Main Operators of the SSO Algorithm

In the SSO approach, search agents are modeled as individual spiders, whose positions $\mathbf{s} = [s_1, s_2, \ldots, s_d]$ within a d-dimensional solution space (also referred as the communal web), each denote a candidate solution for a given optimization problem. During its initialization step, SSO randomly generates an initial population of spiders $\mathbf{S} = \left\{\mathbf{s}_1, \mathbf{s}_2, \ldots \mathbf{s}_{N_{\text{pop}}}\right\}$ (with $\mathbf{s} = \left[s_{i,1}, s_{i,2}, \ldots, s_{i,d}\right]$ and N_{pop} denoting the total population size) within the feasible solution space. The elements $s_{i,j}$ corresponding to each solution $\mathbf{s}_i \in \mathbf{S}$ are initialized by considering the following:

$$s_{i,j} = x_j^{\text{low}} + \text{rand} \cdot \left(x_j^{\text{high}} - x_j^{\text{low}}\right) \quad i = 1, 2, \ldots, N_{\text{pop}};\ \ j = 1, 2, \ldots, d, \tag{5}$$

where x_j^{low} and x_j^{high} represent the decision space's lower and upper bounds, respectively, while rand $(0, 1)$ stands for random number drawn from within the uniformly distributed interval [0, 1].

Furthermore, the SSO approach considers the gender of spiders (either male or female) as part of its search strategy. With that being said, the initial population $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots \mathbf{s}_N\}$ in SSO is further divided into two subsets: a set of N_f female spiders $\mathbf{F} = \left\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{N_f}\right\}$ and a set of N_m male spiders $\mathbf{F} = \left\{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_{N_m}\right\}$, and such that $\mathbf{S} = \mathbf{F} \cup \mathbf{M}$. Also, it is known that most Social Spider colonies have a predominance of female spiders within its population; in fact, some studies suggest that in most cases the number of female spiders can reach more than 70% of the total colony members. In the SSO, the number of female spiders N_f is randomly selected within a range of between 70% and 90% of the total population N_{pop}, while the remaining individuals are labeled as male spiders. As such, N_f and N_m are calculated as follows:

$$N_f = \text{floor}\left(N_{\text{pop}} \cdot \text{rand}\,(0.7, 0.9)\right), \tag{6}$$

$$N_m = N_{\text{pop}} - N_f, \tag{7}$$

where rand $(0.7, 0.9)$ denotes a random number from within the interval [0.7, 0.9], while floor $(\cdot)$ maps a real number to an integer number.

Furthermore, spiders within the communal web are assumed to be able to communicate with each other through a series of vibrations, emitted by each of them and transmitted through the silk threads which comprise said communal area. In the SSO approach, the vibration perceived by a given spider 'i' as a result of the information transmitted by a different spider 'j' is modeled as follows:

$$\text{Vib}_{i,j} = w_{\mathbf{s}_j} \cdot e^{-r_{i,j}^2}, \tag{8}$$

where $r_{i,j} = \|\mathbf{s}_i - \mathbf{s}_j\|$ denotes for the Euclidian distance between the spiders "i" and "j." Furthermore, $w_{\mathbf{s}_j}$ represents the weight corresponding to the j-th spider as given by the following expression:

$$w_{\mathbf{s}_j} = \frac{J\left(\mathbf{s}_j\right) - f_{\text{worst}}}{f_{\text{best}} - f_{\text{worst}}}, \tag{9}$$

where $J\left(\mathbf{s}_j\right)$ denotes the objective function evaluation value (fitness value) corresponding to the solution represented by the j-th spider $\left(\mathbf{s}_j\right)$, while f_{best} and f_{worst} each denote the best and worst fitness values from among all of the spiders within the communal web [27]. Also, while it is possible to compute the perceived vibrations by considering any pair of individuals, three special relationships are of particular interest in the SSO algorithm (see Figure 2):

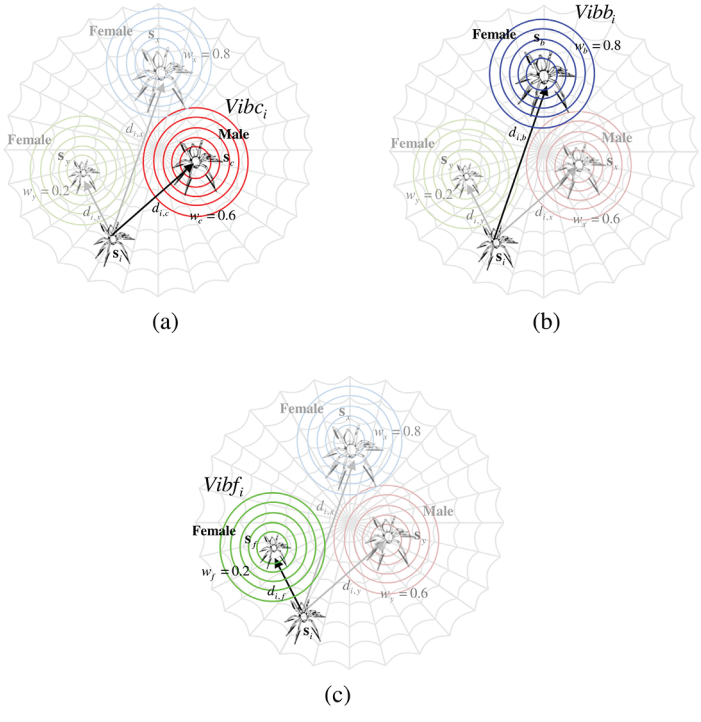

Figure 2 | Vibration models in the Social Spider Optimization (SSO) algorithm: (a) Vib_{c_i}, (b) Vib_{b_i} and (c) Vib_{f_i}.

1. The vibrations Vib_{c_i} perceived by $\mathbf{s}_i$ as a result of the information transmitted by the individual $\mathbf{s}_{c_i}$ that is both, the nearest member to $\mathbf{s}_i$ and heavier than it:

$$\text{Vib}_{c_i} = w_{c_i} \cdot e^{-r_{i,c_i}^2}. \tag{10}$$

2. The vibrations Vib_{b_i} perceived by the i-th spider ($\mathbf{s}_i$) as a result of the information transmitted by the heaviest (best) individual within the whole population ($\mathbf{s}_b$):

$$\text{Vib}_{b_i} = w_{b_i} \cdot e^{-r_{i,b}^2}. \tag{11}$$

3. The vibrations Vib_{f_i} perceived by $\mathbf{s}_i$ as a result of the information transmitted by its nearest female individual $\mathbf{s}_{f_i}$:

$$\text{Vib}_{f_i} = w_{f_i} \cdot e^{-r_{i,f_i}^2}. \tag{12}$$

Also, depending on their gender, spiders are assumed to be able to manifest several different behaviors. In the case of female spiders, for example, an attraction or dislike toward other members of the colony (independent of their gender) may be manifested. In the SSO approach, such a phenomenon is modeled as either an attraction or repulsion movement toward other prominent individuals within the communal web. With that being said, at each iteration "k," the position of a given female spider $\mathbf{f}_i^k$ is updated by applying the following movement rules:

$$\mathbf{f}_i^{k+1} = \begin{cases} \mathbf{f}_i^k + \alpha \cdot \text{Vib}_{c_i}\left(\mathbf{s}_{c_i}^k - \mathbf{f}_i^k\right) + \beta \cdot \text{Vib}_{b_i}\left(\mathbf{s}_b^k - \mathbf{f}_i^k\right) \\ \quad + \delta \cdot (\text{rand} - 1/2) & \text{if } P > P_f \\ \mathbf{f}_i^k - \alpha \cdot \text{Vib}_{c_i}\left(\mathbf{s}_{c_i}^k - \mathbf{f}_i^k\right) - \beta \cdot \text{Vib}_{b_i}\left(\mathbf{s}_b^k - \mathbf{f}_i^k\right) \\ \quad + \delta \cdot (\text{rand} - 1/2) & \text{if } P \leq P_f, \end{cases} \tag{13}$$

where $\mathbf{s}_{c_i}^k$ denotes the position of the nearest best (nearest heavier) member to the spider "i," while $\mathbf{s}_b^k$ stand for the position of the best (heaviest) spider within the communal web. Furthermore, α, β, δ, rand and P each denote a random number drawn from within the uniformly distributed interval $[0, 1]$. Finally, P_f stands for a probability threshold used to define the kind of movement the female spider will perform (either an attraction or a repulsion) [27].

On the other hand, male spiders, which are said to manifest an exclusive attraction toward female individuals within the communal web, may be further identified as either dominant or nondominant male spiders. Typically, dominant male spiders have more prominent characteristics (i.e., a greater size or weight) in comparison to nondominant male spiders. Furthermore, while dominant male spiders are usually attracted toward their closest female spider in the communal web, nondominant male spiders tend to concentrate toward the center of the male population as a strategy to take advantage of resources that are wasted by dominant males. In SSO, these male-characteristic behaviors are modeled by first considering the median weight from within the group of male spiders. At each iteration "k," the weight of each male spider is compared to such median value, and then, an appropriate movement rule is applied to update the position $\mathbf{m}_i^k$ of each of such individuals, as illustrated as follows:

$$\mathbf{m}_i^{k+1} = \begin{cases} \mathbf{m}_i^k + \alpha \cdot \text{Vib}_{f_i}\left(\mathbf{s}_{f_i}^k - \mathbf{m}_i^k\right) \\ \quad + \delta \cdot (\text{rand} - 1/2) & \text{if } w_{\mathbf{m}_i}^k > M(\mathbf{w}_\mathbf{m}) \\ \mathbf{m}_i^k + \alpha \cdot \left(\dfrac{\sum_{h=1}^{N_m} \mathbf{m}_h^k \cdot w_{\mathbf{m}_h}^k}{\sum_{h=1}^{N_m} w_{\mathbf{m}_h}^k} - \mathbf{m}_i^k\right) & \text{if } w_{\mathbf{m}_i}^k \leq M(\mathbf{w}_\mathbf{m}), \end{cases} \tag{14}$$

where $w_{\mathbf{m}_i}^k$ denotes the weight corresponding to the i-th male spider, whereas $M(\mathbf{w}_\mathbf{m})$ stand for the median weight value (intermediate weight) from among the set of weights off all male spider weights (sorted in descending order). Furthermore, $\mathbf{s}_{f_i}^k$ represents the position of the nearest female spider to the i-th male individual, while the values α, δ, and rand each denote a random number drawn from the uniformly distributed interval $[0, 1]$ [27].

Finally, the SSO approach employs a mating mechanism in which female spiders and dominant male spiders are used to construct new candidate solutions. For such a procedure, a dominant male spider (individual with a weight greater than the median weight value of the male population) is first selected, and then, a set of female spiders within a particular mating radius r are further selected to perform a mating operation. Said mating radius is given by the following equation:

$$r = \frac{\sum_{i=1}^{d}\left(x_j^{\text{high}} - x_j^{\text{low}}\right)}{2d}. \tag{15}$$

For the mating operation, a new individual $\text{s}^{\text{new}} = \left[s_1^{\text{new}}, s_2^{\text{new}}, \ldots, s_d^{\text{new}}\right]$ is formed by randomly choosing and combining elements (position information) from among each involved spider. In this case, individuals possessing heavier weights are more likely to influence the newly produced solution while those with lower weights tend to be of less relevance. With that being said, the SSO assigns an influence probability to each involved member as follows:

$$P_{\mathbf{s}_i} = \frac{w_{\mathbf{s}_i}}{\sum_{\mathbf{s}_j \in \mathbf{T}_i} w_{\mathbf{s}_j}}, \tag{16}$$

where $\mathbf{T}_i$ denote the set of individuals involved in the mating operation (the selected dominant male spider and the female spiders within its mating radius) and where $\mathbf{s}_i \in \mathbf{T}_i$.

Once these influence probabilities have been calculated, each of the elements s_j^{new} from the new spider s^{new} is assigned by randomly taking a corresponding element s_j from among the available mating candidates, this by applying the roulette selection method with regard to their respective influence probabilities. Once a new spider is formed by means of said mating operation, its weight is calculated by applying Equation (9), and then, it is compared against the worst spider in the colony (the individual with the lowest weight from among the entire population). If the new spider has a greater weight than said worst individual, then the worst individual is replaced by the new one; otherwise, the new spider is discarded and the population suffers no change. This solution generating procedure is done once for each dominant male individual in the colony. Also, it is worth noting that if there are no candidate female spiders within the

mating radius of a chosen dominant male spider, then the mating operation is canceled; thus, no new individual is generated in that instance [27].

3.2. Computational Procedure of the SSO Algorithm

In Figure 3, we present a flowchart illustrating the main steps of the SSO algorithm [27]. In general, the computational procedure of the SSO algorithm can be summarized as follows:

Step 1 Considering N_{pop} as the total number of d-dimensional spiders, define the number of male spiders N_m and female spiders N_f.

Step 2 Initialize the population of spiders (see Equations (6) and (7)).

Step 3 Calculate the weights for all spiders in the population (see Equation (9))

Step 4 Apply movement operator for female spiders (see Equation (13)).

Step 5 Apply movement operator for male spiders (see Equation (14)).

Step 6 Perform mating operation

Step 7 If the stop criterion is met, end the process; otherwise, return to **Step 3**.

3.3. Computational Complexity of the SSO Algorithm

The SSO algorithm is comprised by several operators including: 1. Position update operators for both, female and male spiders; 2. Mating operator; and 3. Survival operator. Independent of their function within the SSO algorithm, the computational complexity added by any of these operators is related to the maximum number of iterations K that is chosen as a stop criterion for the SSO algorithm.

In the case of the movement operators modeled in SSO, female and male individuals (spiders) simulate movements that are different enough between them, and also have different time complexity. This computational complexity depends on both, the size of the female population N_f and the size of the male population N_m, as given by the following equation expressed in terms of the Big O notation [31]:

$$O(\text{moves}) = O\left(K \times \left(N_f^2 + \left(N_m \times N_f\right)\right)\right). \tag{17}$$

On the other hand, the mating operation in the SSO algorithm the computational complexity is usually not fixed. In the worst-case scenario, the time complexity depends on the population sizes of female spiders (N_f) and the number of dominant male individuals (which is approximately equal to $N_m/2$), as well as on the dimensionality of the problem d. With that being said, for this worst-case scenario, time complexity in Big O notation may be expressed as follows:

$$O\left(\text{mating}\right) = O\left(\frac{N_m}{2} \times N_f \times d \times K\right). \tag{18}$$

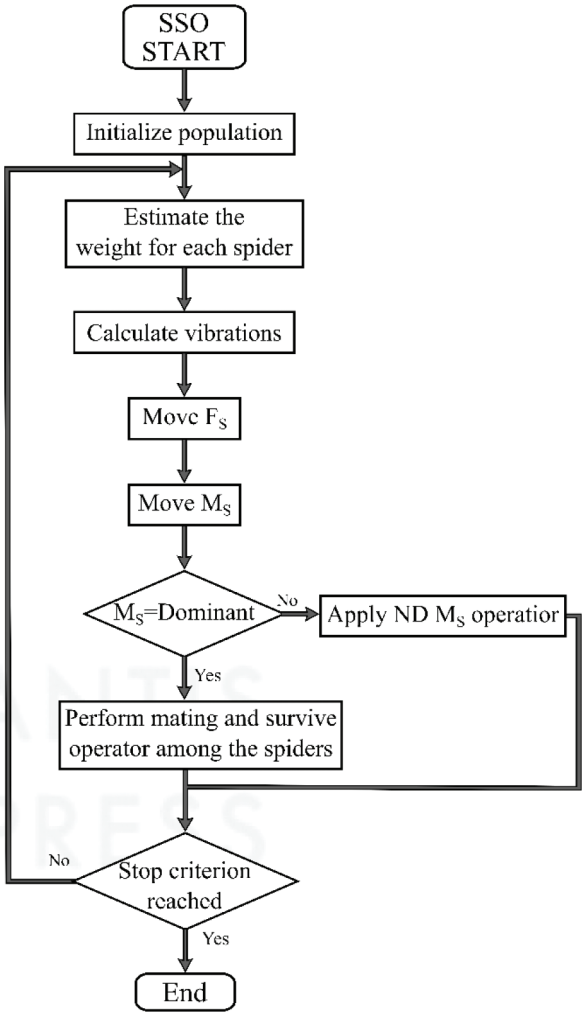

Figure 3 | Flowchart illustrating the computational procedure of the Social Spider Optimization (SSO) algorithm.

Finally, there is the survival operation, which depends only on the number of offspring spiders N_{new} (which is usually a small percentage of the total population) and the maximum number of iterations K. In the worst-case scenario, the number of newly generated individuals equals the number dominant male individuals in the spider population ($N_m/2$). As such, the time complexity for this operation is given at worst by:

$$O(\text{survival}) = O\left(\frac{N_m}{2} \times K\right). \tag{19}$$

Usually, the time complexity of an algorithm as a whole can be reduced to that of its most complex procedure, disregarding all other steps [31]. As the computational complexity for the movement operators is clearly the greatest, the time complexity of the SSO algorithm can then be approximated simply as:

$$O(\text{SSO}) \approx O(\text{moves}), \tag{20}$$

where $O(\text{moves})$ represents the time complexity related to the movement operators for both female and males spiders, as given by Equation (17).

4. REAL-CODED SSO-BASED SENSOR DEPLOYMENT SCHEME FOR WSNs

As exposed in Section 2, the main objective behind the deployment of a WSNs is to maximize the sensor network's coverage area by choosing optimal locations within a given ROI for a set of available sensor nodes. For simplicity it is often considered that the service area has a finite amount of candidate locations where sensor nodes can be placed, commonly represented by a 2-D grid. Under this approach, the measure of how good a given sensor arrangement is may be given by its coverage area rate, which is related to the amount of locations of the modeled lattice that are covered by the sensor network. On the literature, the BSM is commonly applied in order to determine whether or not a given location (x, y) of the modeled 2-D grid is covered or not by any of the available sensor nodes, and then, the total amount of covered positions is compared against the total number available points within the grid to determine the coverage rate [19,32,33]. In a summary, this approach for optimal WSNs deployment represents a combinatorial optimization problem in which the aim is to provide an appropriate combination of discrete positions $\mathbf{s} = \left[x_1, y_1, x_2, y_2, \ldots, x_{N_{\text{nodes}}}, y_{N_{\text{nodes}}}\right]$ so that the rate of covered area $R_{\text{cov}}(\mathbf{s})$ achieved by the sensor nodes at these positions is maximized; this is:

$$\begin{aligned} \text{Maximize:} \quad & R_{\text{cov}}(\mathbf{s}) = \frac{A_{\text{cov}}(\mathbf{s})}{A_{\text{total}}} \\ \text{Subject to:} \quad & x_j \in \{0, 1, \ldots, M\} \\ & y_j \in \{0, 1, \ldots, N\}, \end{aligned} \tag{21}$$

where $A_{\text{cov}}(\mathbf{s})$ denotes the amount of the service area that is covered by applying the sensor deployment configuration modeled by solution $\mathbf{s}$, while $A_{\text{total}} = M \times N$ stand for the total area (number of locations) that are required to be covered by the deployed sensor nodes.

In [19], the authors proposed a sensor deployment scheme based on the SSO algorithm, in which optimal locations for a set of sensor nodes $S = \{s_1, s_2, \ldots s_n\}$ are chosen from among a set of candidate positions within a 2-D grid $G(x, y)$ of size $M \times N$. In their proposed SSO-based sensor deployment scheme, the SSO algorithm starts by generating a set of N_{pop} random solutions (spiders) $\mathbf{S} = \left\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{N_{\text{pop}}}\right\}$ within the feasible solution space, represented in this case by the discrete experimental environment. In the context of OSD for WSNs, each spider $\mathbf{s}_i$ represents a possible configuration of 2-D positions for the available sensor nodes, such that:

$$\mathbf{s}_i = \left[x_1^i, y_1^i, x_2^i, y_2^i, \ldots, x_{N_{\text{nodes}}}^i, y_{N_{\text{nodes}}}^i\right], \tag{22}$$

where the elements x_j^i and y_j^i represent a pair of (x, y) discrete locations corresponding to sensing device "j," while N_{nodes} stand for the number of sensor nodes that are required to be deployed within a given service area.

Guided by the SSO's cooperative behavior operators, each spider $\mathbf{s}_i$ moves around the available solution space while looking for the OSD configuration. In this context, the quality (fitness) of the solutions represented by each spider $\mathbf{s}_i$ is evaluated concerning the sensing coverage rate which is provided by the sensor deployment configuration modeled by each of said solutions.

While the SSO algorithm has demonstrated competent results when applied to solve such a challenging combinatorial optimization problem, it is worth noting that the discrete nature of these sensor deployment models somewhat restricts the ability of the SSO to find potentially better placement configurations for the available sensing devices. Motivated by this fact, in this paper, a real-coded implementation based on the SSO algorithm is proposed to solve the problem of OSD in WSNs. Different to previous works, where the sensor's deployment area is modeled by a discrete grid $G(x, y)$, in this paper we consider such service area to be modeled by a continuous flat surface $f(x, y)$ of size $M \times N$. Also, it is considered that each sensor node $s_j \in S$ can be placed on any real location (x, y). With the previous being said, the optimization problem that is required to solve may now be expressed as follows:

$$\begin{aligned} \text{Maximize:} \quad & R_{\text{cov}}(\mathbf{s}) = \frac{A_{\text{cov}}(\mathbf{s})}{A_{\text{total}}} \\ \text{Subject to:} \quad & 0 \le x_j \le M; \quad 0 \le y_j \le N. \end{aligned} \tag{23}$$

As previously mentioned, the key difference between our proposed approach and the one presented in [19] is the fact that in the former each available sensor node s_j is able to be placed in any real positions within the modeled ROI, whereas in the later, only a limited set of locations are candidates to place sensing devices. This gives our proposed approach the flexibility to explore a wider set of candidate solutions, while also giving it the ability to find potentially better solutions in comparison to a discrete-coded sensor deployment scheme.

Similar to the approach presented in [19], a BSM has been considered for the calculation of the sensor network's coverage area. The main difference, however, is that instead of calculating the covered area by counting the number of discrete locations that are covered by the sensor network, our proposed approach calculates the actual area that is covered by a given sensor deployment configuration. For this purpose, it is considered that the covered area $A_{\text{cov}}(\mathbf{s})$ (with $\mathbf{s} = \left[x_1, y_1, x_2, y_2, \ldots, x_{N_{\text{nodes}}}, y_{N_{\text{nodes}}}\right]$ representing the set of sensor node locations) may be given by the following expression:

$$A_{\text{cov}}(\mathbf{s}) = \iint_A \left(A_{\text{sensing}}(\mathbf{s}) \cap A_{\text{deploy}}\right) dA, \tag{24}$$

where A_{deploy} describes the deployment area that is required to be covered, whereas $A_{\text{sensing}}(\mathbf{s})$ denotes the total area covered by all of the deployed sensor nodes, as given as follows:

$$A_{\text{sensing}}(\mathbf{s}) = \bigcup_{j=1}^{N} A_j, \tag{25}$$

where A_j denotes the circular region within the deployment area that is covered by the j-th sensor node. It should be noted that, for all purposes, the sensing region A_j described by a sensor node s_j is represented by a circular area around the sensor's location (x_j, y_j), which size is delimited by the sensor's coverage radius r_s, as described in Section 2 (see Figure 4).

5. EXPERIMENTAL SETUP AND RESULTS

In this paper, a real-coded SSO (R-SSO)-based sensor deployment scheme is proposed for solving the problem of optimal placement

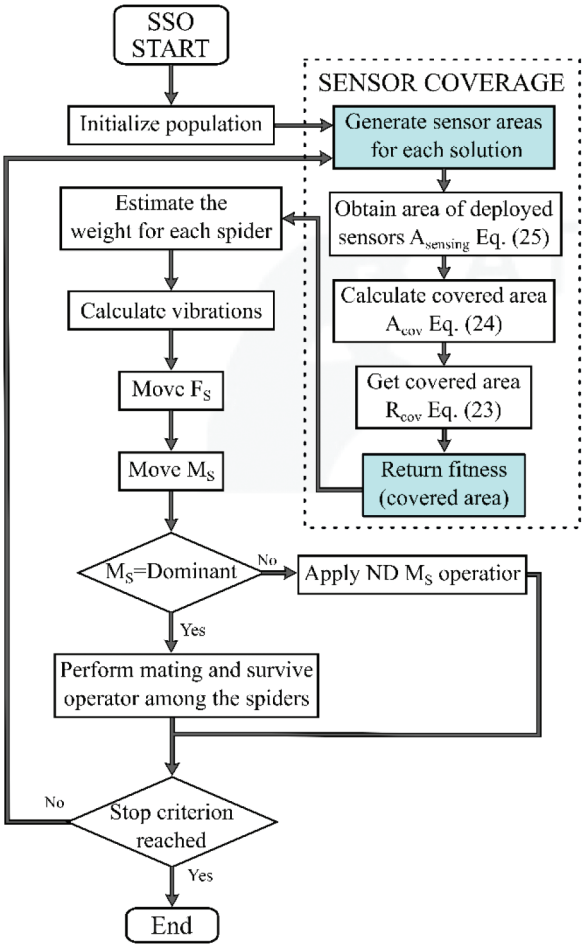

Figure 4 | Flowchart illustrating the computational procedure for the proposed real-coded Social Spider Optimization (R-SSO) optimal sensor deployment (OSD) approach.

of sensor nodes in WSNs. Different to other similar approaches currently reported on the literature, where the arrangement of sensor nodes is performed by considering a set of discrete positions modeled by a 2-D mesh grid, our proposed method considers a continuous experimental environment, where such sensing devices can adopt virtually any real position. As a result of this, our proposed OSD approach is able to explore a much wider set of candidate solutions, allowing it to find potentially better solutions in comparison to those provided by discrete-coded methods. In order to evaluate the performance of our proposed OSD approach, a series of comparative experiments against other similar techniques were performed. Our experimental results are divided into two parts: in Section 5.1, we present a comparative analysis between our proposed R-SSO-based approach and the discrete-coded SSO (D-SSO)-based implementation proposed in [19]; furthermore, in Section 5.2, we compare the performance of our proposed method with that of other similar techniques modified to work with real-coded positions, such as the MFO algorithm [20], FA [21], PSO [22], ABC [23], WOA [24], CSA [25], and GWO [26]; finally, in Section 5.3., additional comparative experiments, which consider the task of covering several disjointed regions of interest (ROIs) within a given service area are presented.

5.1. R-SSO-Based OSD VS D-SSO-Based OSD

Our first set of experiments involves a comparison between our proposed R-SSO-based OSD approach and the D-SSO-based OSD implementation proposed in [19]. As previously stated, the key difference between both of the compared approaches is that in the former the locations (x, y) given to each of the available sensor nodes are coded as a pair of real positions, whereas in the later these positions are conditioned to be any pair of discrete positions modeled within a 2-D grid; intuitively, this suggests that our proposed approach is able to explore a much wider set of solutions and, as a result, it could potentially be able to find much better solutions in comparison to the discrete-coded case. Motivated by this premise, we have compared the performance for both the R-SSO and the D-SSO with regard to the optimization problem illustrated by Equation (23). Both of the compared methods were tested by considering a population size of $N_{pop} = 50$ individuals (search agents), and a maximum number of iterations of $k_{max} = 100$ as stop criterion. Also, for both algorithms, the female attraction probability parameter is equally set as $PF = 0.7$. All experiments were performed on MATLAB® R2016a, running on a computer with an Intel® Core™ i7 - 3.40 GHz processor, and Windows 8 (64-bit, 8GB of memory) as its operating system.

For each experimental run, it is considered that there is an experimental environment with an area size of $A = M \times N$ (being $M = 800$ and $N = 700$) in which it is required to deploy a set of distributed sensor nodes equal to $N_{sensors} = 20$, each with a fixed coverage radius $r_s = 90$. For the simulation of both, the R-SSO-based OSD and the D-SSO-based OSD case, the applied objective functions are given by Equations. (21) and (23), respectively. In Table 1, the comparative results corresponding to 30 individual runs for each of the compared methods are shown with the best outcomes being indicated in boldface. The results reported in the table consider the following performance indexes: the mean, median, and standard deviation of the best fitness values (f_{mean}, f_{median}, and f_{std}, respectively) found on each of set of experimental runs, as well as the worst and best fitness values (f_{worst} and f_{best}, respectively) from among each individual run. As shown by the data provided in this table, the R-SSO-based approach manifests a slightly better performance in comparison to its discrete-coded counterpart. While the search algorithms implemented in both cases are essentially the same, the flexibility to choose any real position within the experimental environment gives our proposed approach the edge when applied for the task of OSD.

Table 1 | Optimization results for R-SSO and D-SSO applied for the OSD of $N_{sensors} = 20$ sensor nodes (each with a coverage radius $r_s = 90$) within an experimental environment with an area size of $A = 800 \times 700$. The statistical results correspond to 30 individual runs per method, each by considering a population size of $N_{pop} = 50$ individuals, and a maximum number of iterations of $k_{max} = 100$ as stop criterion.

	R-SSO	D-SSO
f_{mean}	0.8436	0.8419
f_{median}	0.8459	0.8418
f_{std}	0.0087	0.0057
f_{min}	0.8243	0.8321
f_{max}	0.8553	0.8553

OSD, optimal sensor deployment; D-SSO, discrete-coded Social Spider Optimization; R-SSO, real-coded Social Spider Optimization.

Furthermore, Figure 5 presents the convergence curves corresponding to the averaged performance over the set of experimental runs of each of the compared techniques. As illustrated by the curves, the R-SSO algorithm demonstrates to have better convergence rate than his counterpart, with D-SSO manifesting a comparable performance (though still inferior when compared to R-SSO). R-SSO achieves a better fitness it each stage of the iterations.

Finally, in Figures 6 through 8, we show an example of an initial arrangement of sensor positions as well as the best sensor deployment configurations achieved by each of the compared methods from among their respective sets of experimental runs.

5.2. Comparison with Other Real-Coded OSD Approaches Based on Metaheuristics

To further demonstrate the performance of the proposed R-SSO-based OSD, a set of comparative experiments against some other similar techniques, modified to work with real-coded positions, is presented. Specifically, methods based on metaheuristic optimization algorithms such as MFO algorithm [20], FA [21], PSO [22], ABC [23], WOA [24], CSA [25], and GWO [26] were considered to perform our comparative experiments. The parameter setup considered for each of these algorithms is as follows:

1. R-SSO: The female attraction probability parameter is set as $PF = 0.7$ [27].
2. MFO: The number of flames is set as $N_{\text{flames}} = \text{round}\left(\left(N_{\text{pop}} - k\right) * \left(N_{\text{pop}} - 1\right)/k_{\max}\right)$, where N_{pop} denotes the population size, k the current iteration and $k_{\max}$ the maximum number of iterations [20].
3. FA: The parameters setup for the randomness factor and the light absorption coefficient are set to $\alpha = 0.2$ and $\gamma = 1.0$, respectively [21].
4. PSO: The cognitive and social coefficients are set to $c_1 = 2.0$ and $c_2 = 2.0$, respectively. Also, the inertia weight factor ω is set to decreases linearly from 0.9 to 0.2 as the search process evolves [22].
5. ABC: The algorithm was implemented by setting the parameter *limit* = num Of Food Sources * dims, where num Of *Food Sources* = N (population size) and dims = d (dimensionality of the solution space) [23].

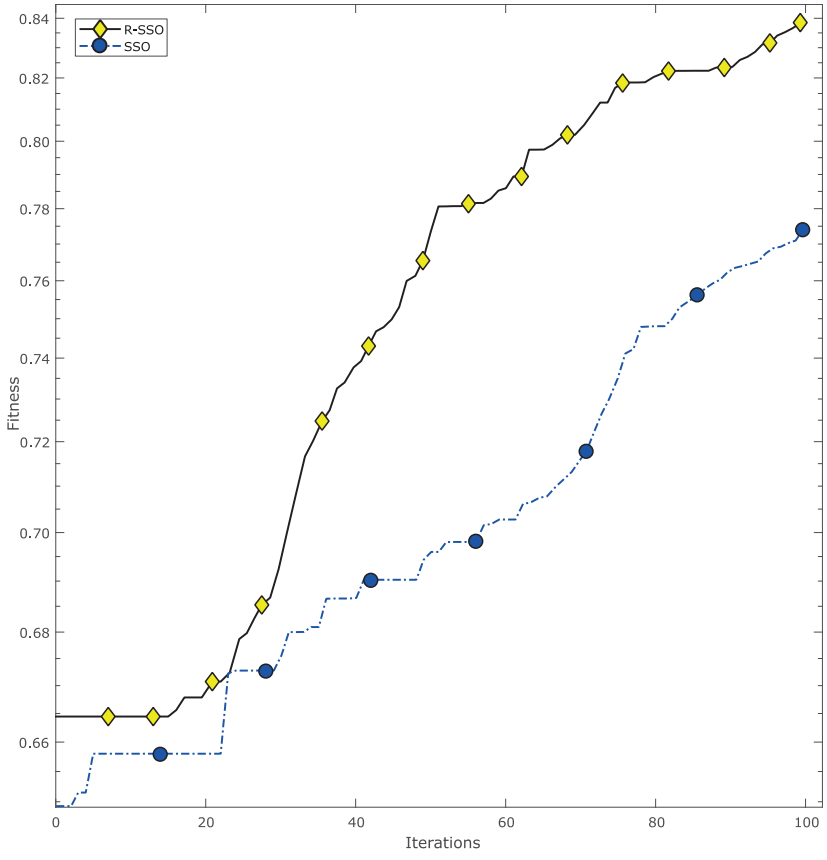

Figure 5 | Convergence curves obtained by real-coded Social Spider Optimization (R-SSO) and discrete-coded SSO (D-SSO) when applied to solve the proposed optimal sensor deployment (OSD) problem (with $A = 800 \times 700$, $N_{\text{sensors}} = 20$ and $r_s = 90$). The shown curves correspond to the averaged performance over each set of experimental runs. For all cases, a population size of $N_{\text{pop}} = 50$ individuals and a maximum number of iterations of $k_{\max} = 100$ are considered.

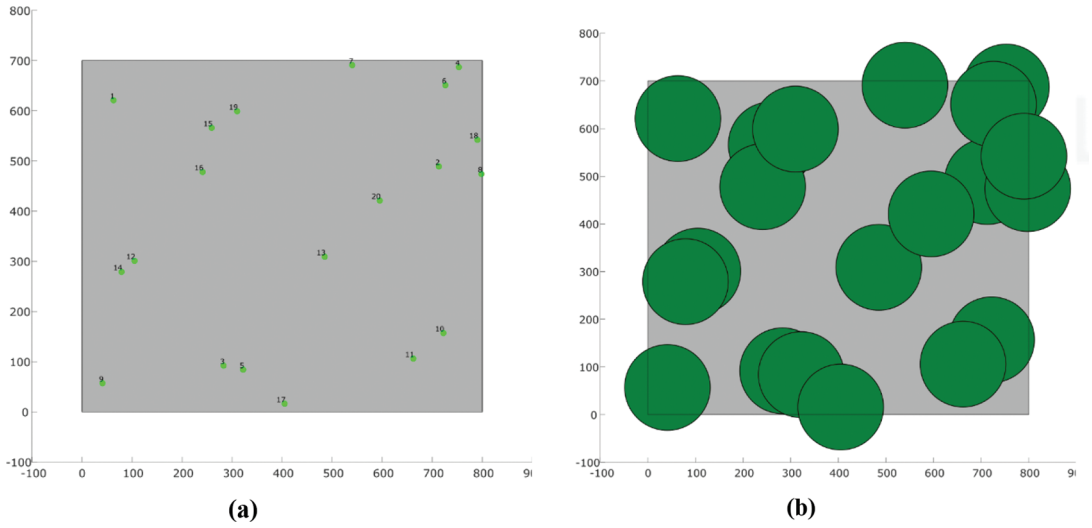

Figure 6 | Initial sensor positions for a set of 20 sensor nodes within an experimental environment of area size A = 800 × 700: (a) sensor nodes locations and (b) area covered by the sensing devices on their initial locations.

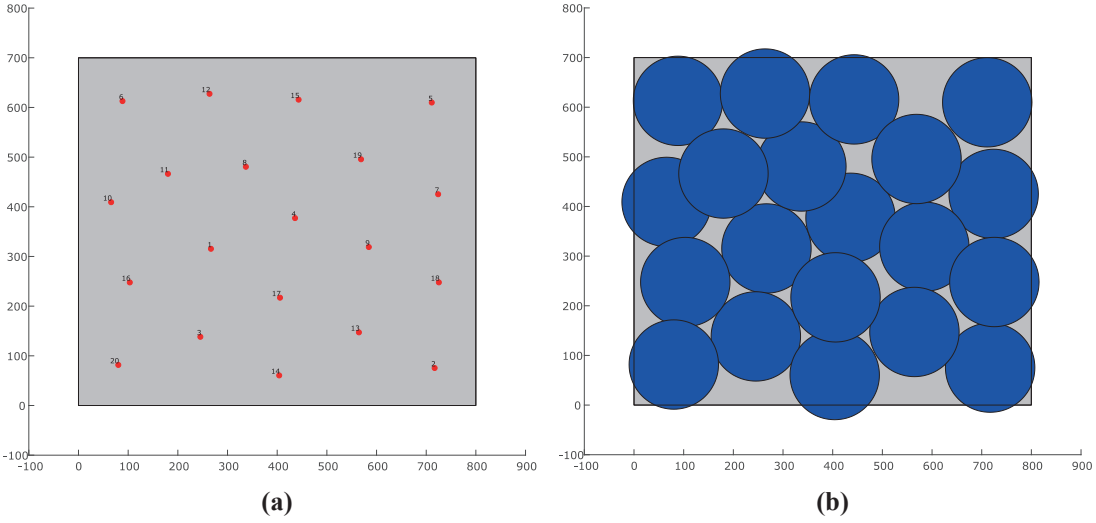

Figure 7 | Sensor deployment configuration for 20 sensor nodes, obtained by applying the real-coded Social Spider Optimization (R-SSO) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

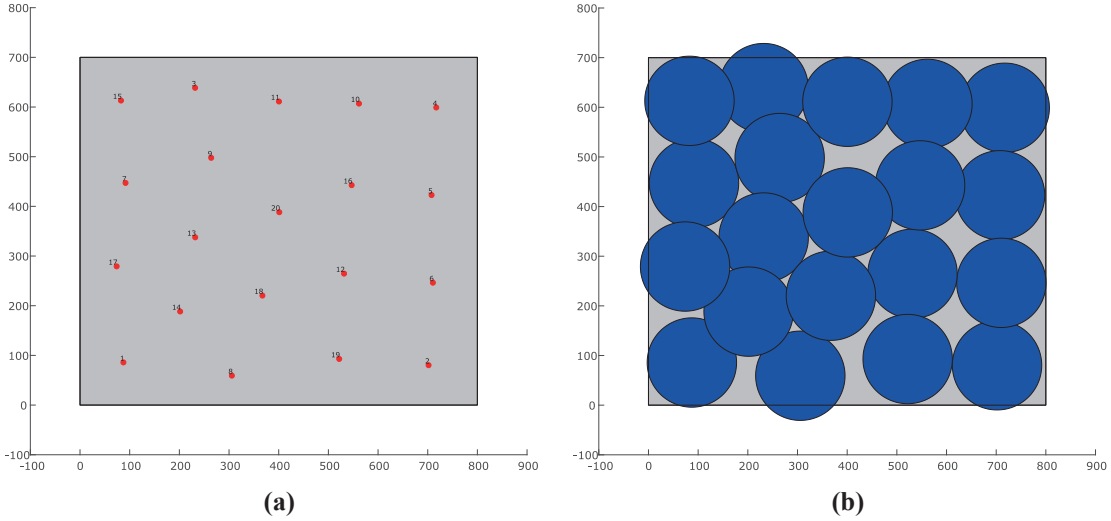

Figure 8 | Sensor deployment configuration for 20 sensor nodes, obtained by applying the discrete-coded Social Spider Optimization (D-SSO) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

6. WOA: The internal parameters A and C are set to decreases linearly from 2 to 0, and -1 to -2, respectively [24].
7. CSA: The awareness probability is set to $AP = 0.1$, while the flight length is given as $fl = 2$ [25].
8. GWO: The algorithm's parameter a is set to decrease linearly from 2 to 0 [26].

The previously illustrated sets of parameters were determined through exhaustive experimentation; thus, these sets of parameters represent the best possible configurations for each of the compared methods. All of the compared methods were tested by considering a population size of $N_{pop} = 50$ individuals (search agents), and a maximum number of iterations of $k_{max} = 300$ as stop criterion. All experiments were performed on MATLAB® R2016a, running on a computer with an Intel® Core™ i7 - 3.40 GHz processor, and Windows 8 (64-bit, 8GB of memory) as its operating system.

Our experiments aim to compare the performance of R-SSO against those of MFO, FA, PSO, ABC, CSA, WOA, and GWO when applied for the task of OSD in WSNs. Similarly to the experiments reported in Section 5.1, we consider an experimental environment with an area size of $A = M \times N$ (being $M = 800$ and $N = 700$) in which it is required to deploy a WSN comprised by $N_{sensors} = 30$ distributed sensor nodes, each with a fixed coverage radius $r_s = 90$. For all cases, the implemented objective function is that presented in Equation (23). The experimental results, corresponding to 30 individual runs for each of the considered method are reported in Table 2, where the best outcomes are boldfaced. Similar to the results reported in the previous section, we consider as performance indexes the mean, median, and standard deviation of the best fitness values (f_{mean}, f_{median}, and f_{std}, respectively) found on each of the sets of experimental runs, as well as the worst and best fitness values (f_{worst} and f_{best}, respectively) found on each set of experiments. As shown by experimental results, the R-SSO-based scheme seems to have a better performance when compared to all other methods. This result could be attributed to the task distribution scheme proposed by the SSO algorithm, as well as the specialized operators applied by both male and female individuals, which lead to a better tradeoff between the exploration and exploitation of solutions.

Table 2 Optimization results for R-SSO, MFO, FA, PSO, CSA, ABC, WOA, and GWO for the OSD of $N_{sensors} = 30$ sensor nodes (each with a coverage radius $r_s = 90$) within an experimental environment of size $A = 800 \times 700$. The statistical results correspond to 30 individual runs per method, each by considering a population size of $N_{pop} = 50$ individuals, and a maximum number of iterations of $k_{max} = 300$ as stop criterion.

	R-SSO	MFO	FA	PSO	CSA	ABC	WOA	GWO
f_{mean}	**0.9804**	0.9656	0.9504	0.9417	0.8016	0.7867	0.9090	0.9593
f_{median}	**0.9831**	0.9673	0.9514	0.9459	0.8029	0.7815	0.9078	0.9607
f_{std}	**0.0081**	0.0104	0.0119	0.0176	0.0115	0.0177	0.0113	0.0082
f_{worst}	**0.9586**	0.9388	0.9102	0.9028	0.7864	0.7582	0.8873	0.9404
f_{best}	**0.9908**	0.9773	0.9647	0.9637	0.8217	0.8266	0.9308	0.9769

ABC, Artificial Bee Colony; CSA, Crow Search Algorithm; FA, Firefly Algorithm; GWO, Grey Wolf Optimizer; MFO, Moth-flame Optimization; OSD, optimal sensor deployment; PSO, Particle Swarm Optimization; ROI, regions of interest; R-SSO, real-coded Social Spider Optimization; WOA, Whale Optimization Algorithm.

Furthermore, Figure 9 presents the convergence curves corresponding to the averaged performance over the set of experimental runs of each of the compared techniques. As illustrated by these curves, the R-SSO algorithm demonstrates to have better convergence rate than all methods, with MFO manifesting a comparable performance (though still inferior when compared to R-SSO). Interestingly, while R-SSO, MFO, FA, PSO, and GWO seem to exhibit continuous convergence toward the global best solution, methods such as ABC, CSA, and WOA seem to have notorious difficulties to handle the proposed OSD problem, with each of them suffering from stagnation even at the earliest stages of their search process.

Finally, in Figures 10 through 18, we show an example of an initial arrangement of sensor positions as well as the best sensor deployment configurations achieved by applying R-SSO, MFO, FA, PSO, CSA, ABC, WOA, and GWO to said initial set of positions. As evidenced by these graphical results, R-SSO can achieve a much uniform distribution of sensor nodes, while also allowing it to cover the most area when compared to the other applied methods.

5.3. R-SSO for OSD within a Set of Disjointed ROIs

In order to further demonstrate the capabilities of the proposed R-SSO-based OSD approach, we have formulated an additional set of

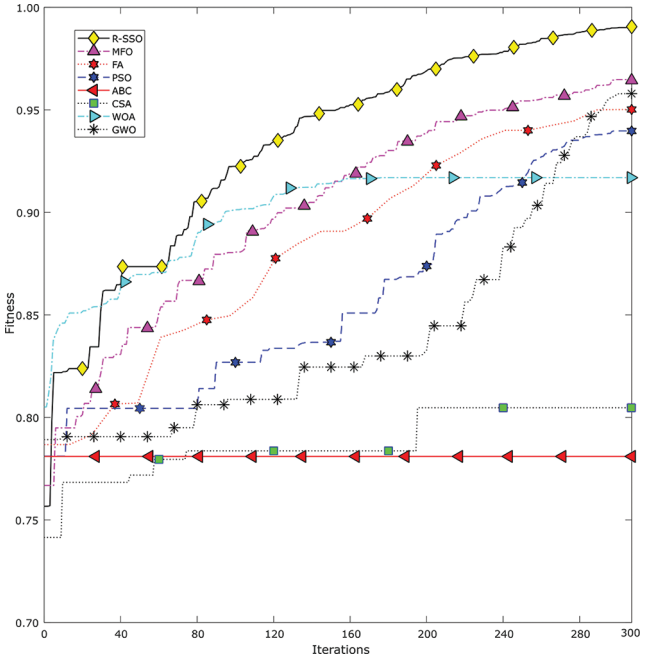

Figure 9 Evolution curves obtained by real-coded Social Spider Optimization (R-SSO), Moth-flame Optimization (MFO), Firefly Algorithm (FA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Crow Search Algorithm (CSA), Whale Optimization Algorithm (WOA), and Grey Wolf Optimizer (GWO) when applied to solve the proposed optimal sensor deployment (OSD) problem (with $A = 800 \times 700$, $N_{sensors} = 30$ and $r_s = 90$). The shown curves correspond to the averaged performance over each set of experimental runs. For all cases, a population size of $N_{pop} = 50$ individuals and a maximum number of iterations of $k_{max} = 300$ are considered.

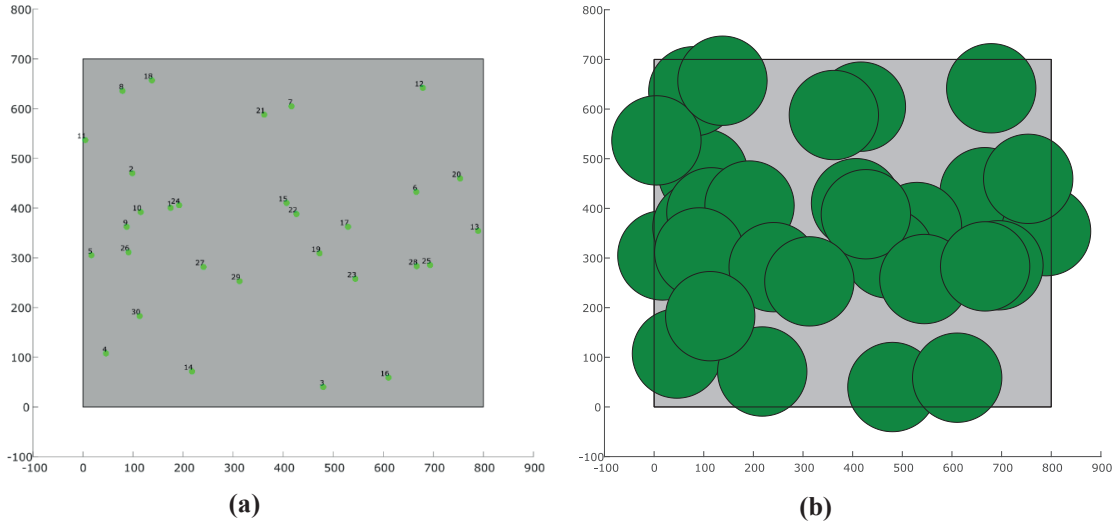

Figure 10 | Initial configuration of positions for a set of 30 sensor nodes within an experimental environment of area size $A = 800 \times 700$: (a) sensor nodes locations and (b) area covered by the sensing devices on their initial positions.

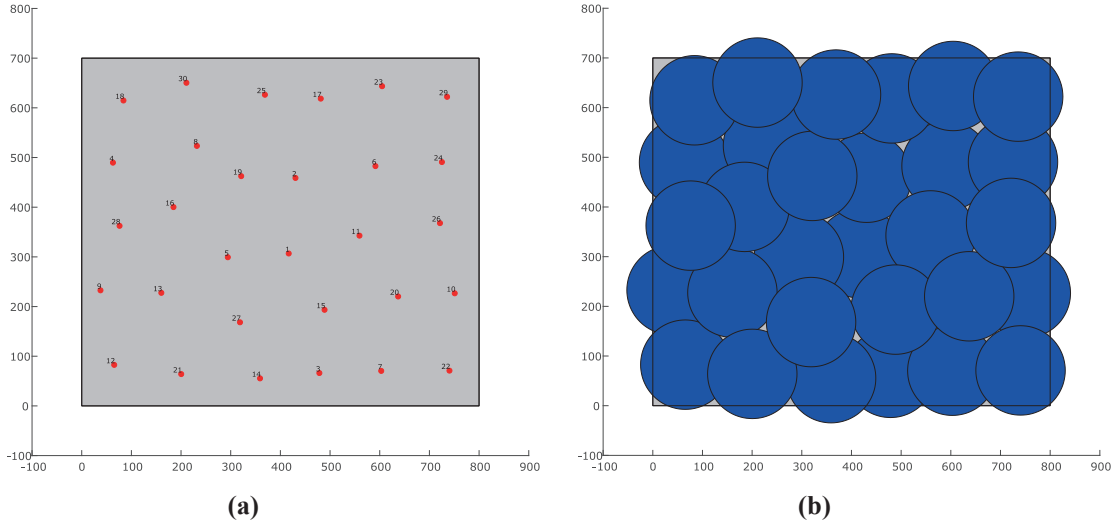

Figure 11 | Sensor deployment configuration for 20 sensor nodes, obtained by applying the real-coded Social Spider Optimization (R-SSO) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

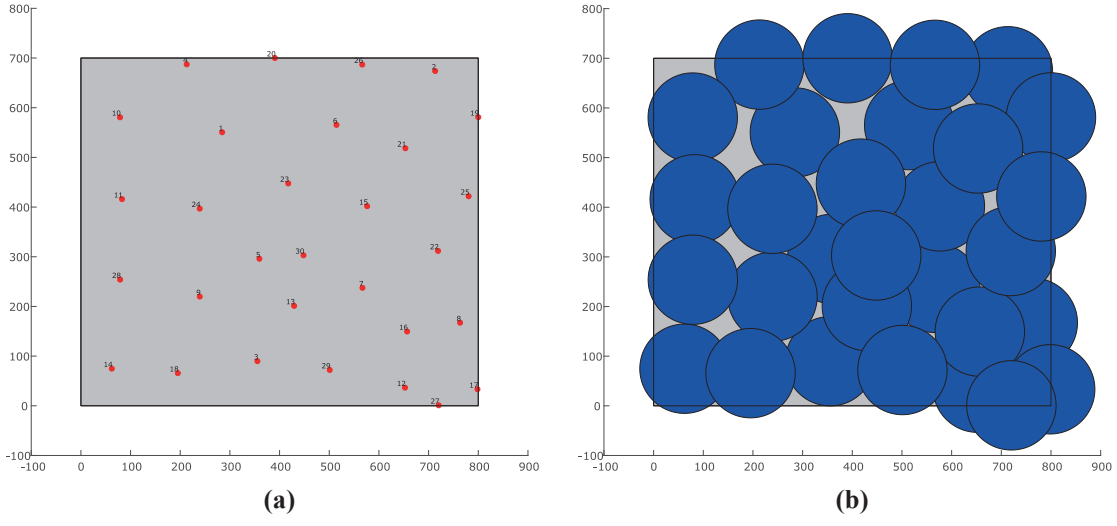

Figure 12 | Sensor deployment configuration for 20 sensor nodes, obtained by applying the Moth-flame Optimization (MFO) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

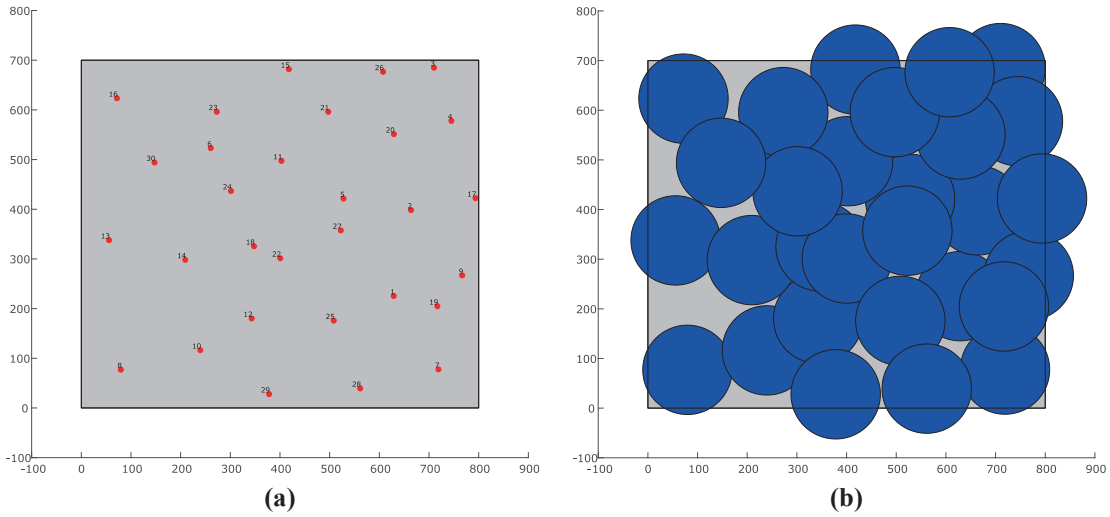

Figure 13 | Sensor deployment configuration for 20 sensor nodes, obtained by applying the Firefly Algorithm (FA) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

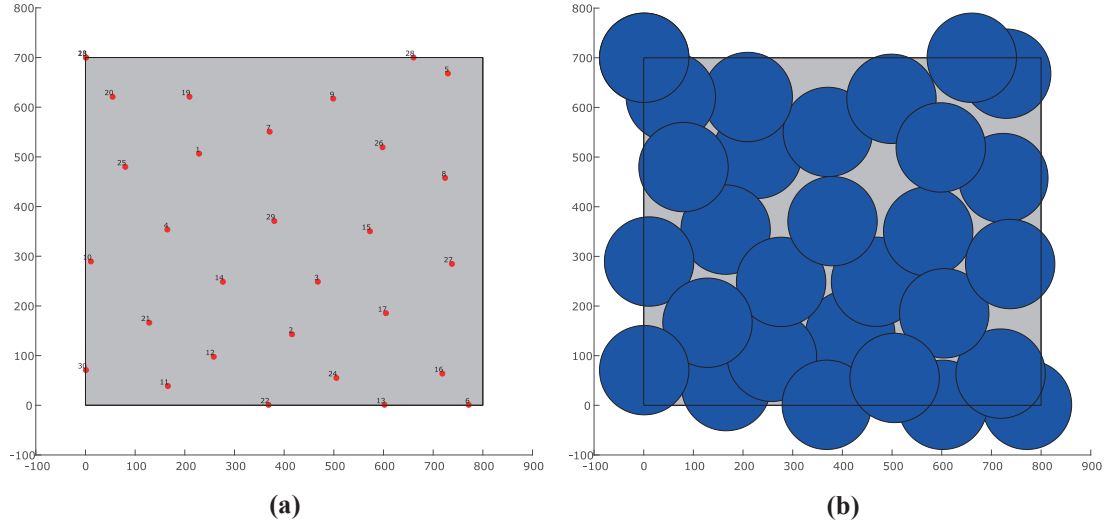

Figure 14 | Sensor deployment configuration for 20 sensor nodes, obtained by applying the Particle Swarm Optimization (PSO) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

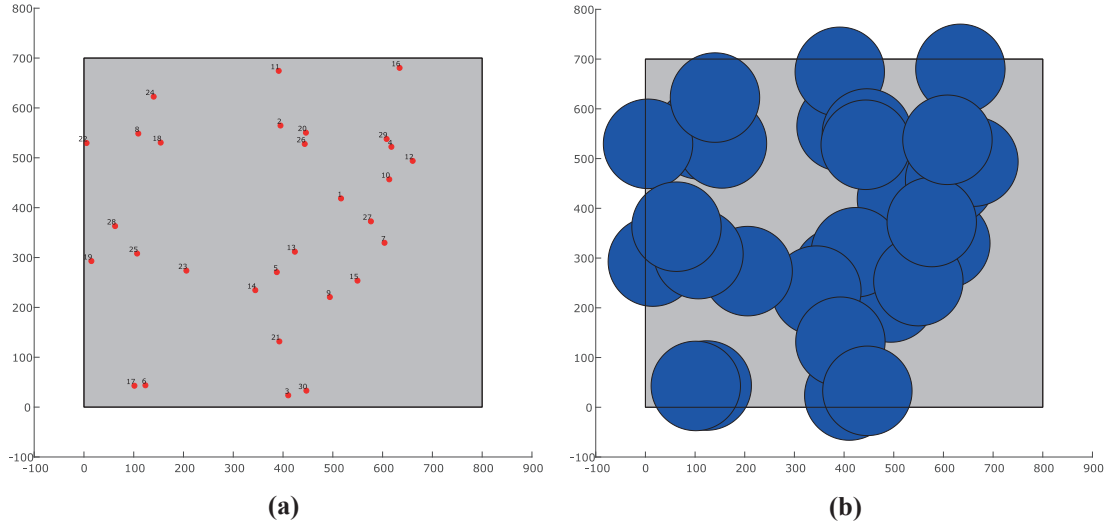

Figure 15 | Sensor deployment configuration for 20 sensor nodes, obtained by applying the Crow Search Algorithm (CSA) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

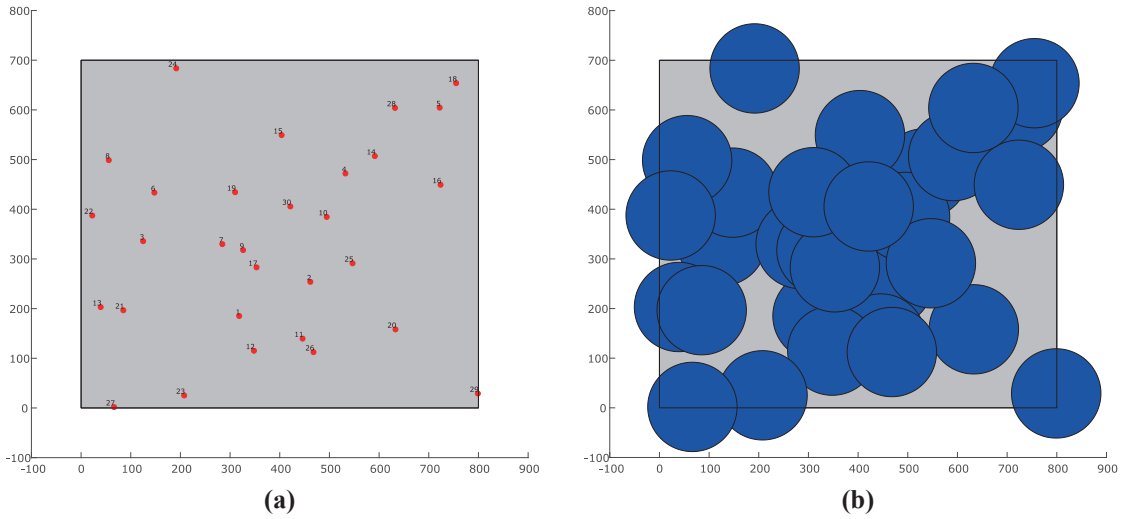

Figure 16 | Sensor deployment configuration for 20 sensor nodes, obtained by applying the Artificial Bee Colony (ABC) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

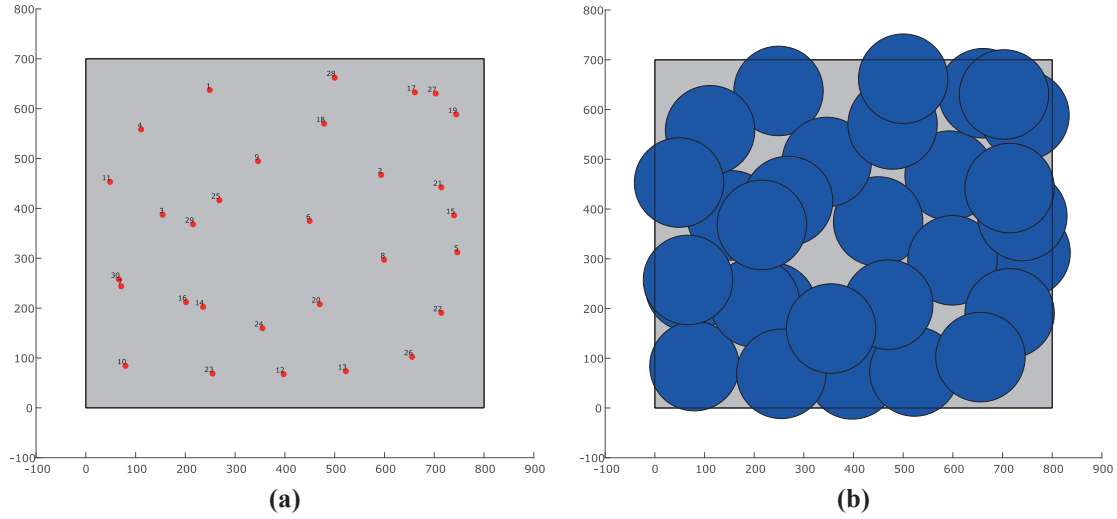

Figure 17 | Sensor deployment configuration for 20 sensor nodes, obtained by applying the Whale Optimization Algorithm (WOA) approach: (a) sensor nodes locations, and (b) area covered by the deployed sensing devices.

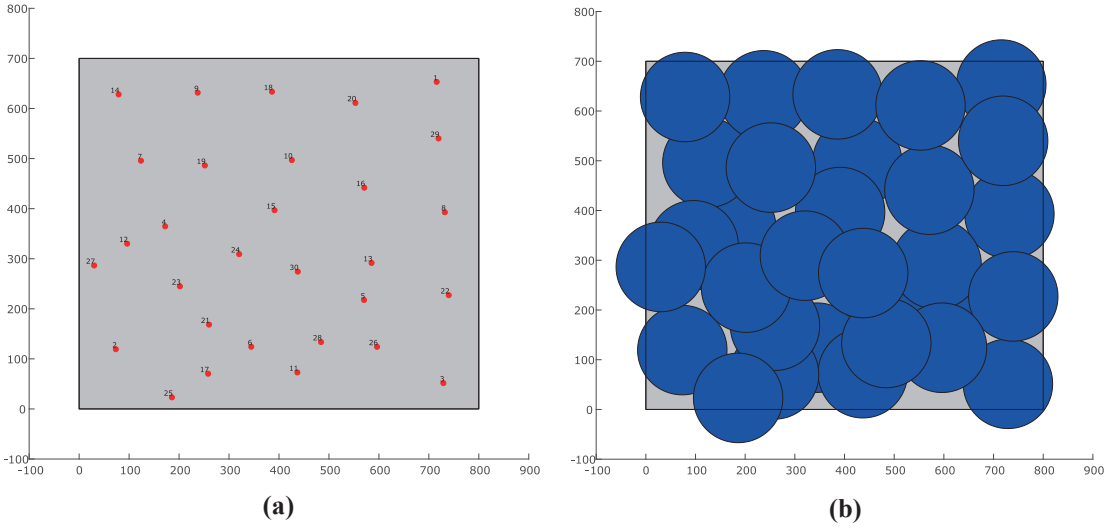

Figure 18 | Sensor deployment configuration for 20 sensor nodes, obtained by applying the Grey Wolf Optimizer (GWO) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

experiments which involve the optimal placement of sensor nodes within several disjointed ROIs within a given experimental environment. Specifically, it is considered that an experimental environment of size of $A = M \times N$ there exists a number of ROIs (each with a different shape and size) that are required to be covered by a set of deployed sensor nodes (see Figure 19). For these final set of experiments, we have considered $A = 800 \times 700$, $N_{\text{sensors}} = 15$, and $r_s = 90$ as the fixed parameters for the deployment of the required WSN. While the optimization problem is essentially the same as in the previous two cases (i.e., maximizing the sensor network's coverage), it is worth noting that the disjointed nature of the modeled ROIs adds a notorious level of complexity to the OSD problem, and as such it is interesting to explore the capabilities of the studied OSD approaches under this conditions.

Similar to the experiments reported in the previous section, the objective is to compare the performance of R-SSO against those of MFO, FA, PSO, ABC, CSA, WOA, and GWO when applied for this specific OSD case; with that being said, and for the sake of consistency, the parameter setup applied to each algorithm is kept as reported in Section 5.2. All of the compared methods were tested by considering a population size of $N_{\text{pop}} = 50$ individuals (search agents), and a maximum number of iterations of $k_{\max} = 200$ as stop criterion. Also, as in the experiments reported in the previous section, the fitness function considered for this set of experiments is that given in Equation (23). It should be noted that the number of deployable sensor nodes N_{sensors} and the maximum number of iterations applied for this set of experiments are both lower than those considered on the experiments reported in Section 5.2; this is done due to the fact that the total area that is required to be covered (represented by the disjointed ROIs) is significantly smaller than that of the total experimental environment, thus, this problem may be solved by considering fewer sensor nodes while also requiring less iteration cycles for each method to converge toward the global best solutions. All experiments were performed on MATLAB® R2016a, running on a computer with an Intel® Core™ i7 - 3.40 GHz processor, and Windows 8 (64-bit, 8GB of memory) as its operating system.

The experimental results, corresponding to 30 individual runs for each of the considered methods are shown in Table 3, where the best outcomes appear in boldface. As in the results reported in Section 5.2, the analyzed performance indexes correspond to the mean, median, and standard deviation of the best fitness values (f_{mean}, $f_{\text{median}}f_{\text{mean}}$, and f_{std}, respectively) found on each of the sets of experimental runs, and the worst and best fitness values (f_{worst} and f_{best}, respectively) found on each of such set of experiments. As illustrated by the data presented in this table, R-SSO outperforms all other methods in terms of covered area rate. Like in the experiments reported in the previous section, R-SSO's high performance is mainly attributed to the task distribution and specialized operators applied in by the SSO algorithm.

In Figure 20, we show the evolution curves corresponding to the averaged performance over the set of experimental runs of each of the compared techniques. Like in the experiments reported in Section 5.2, the R-SSO approach once again demonstrates to have a better convergence rate than all methods, followed by the MFO algorithm. It is also worth noting that once again algorithms such as ABC, CSA, and WOA seem to struggle when applied to this

Table 3 Optimization results for R-SSO, MFO, FA, PSO, CSA, ABC, WOA, and GWO for the OSD of $N_{\text{sensors}} = 15$ sensor nodes (each with a coverage radius $r_s = 90$) within an experimental environment of size $A = 800 \times 700$ with four disjointed ROIs of different shape and area size. The statistical results correspond to 30 individual runs per method, each by considering a population size of $N_{\text{pop}} = 50$ individuals, and a maximum number of iterations of $k_{\max} = 200$ as stop criterion.

	SSO	MFO	FA	PSO	CSA	ABC	WOA	GWO
f_{mean}	**0.9923**	0.9803	0.9753	0.9629	0.8003	0.7369	0.8965	0.9771
f_{median}	**0.9948**	0.9861	0.9749	0.9728	0.8012	0.7274	0.9043	0.9763
f_{std}	**0.0083**	0.0184	0.0086	0.0273	0.0239	0.0300	0.0427	0.0144
f_{min}	**0.9757**	0.9286	0.9581	0.8938	0.7601	0.6909	0.9323	0.7862
f_{max}	**1.0000**	0.9995	0.9865	0.9975	0.8453	0.7990	0.9516	0.9985

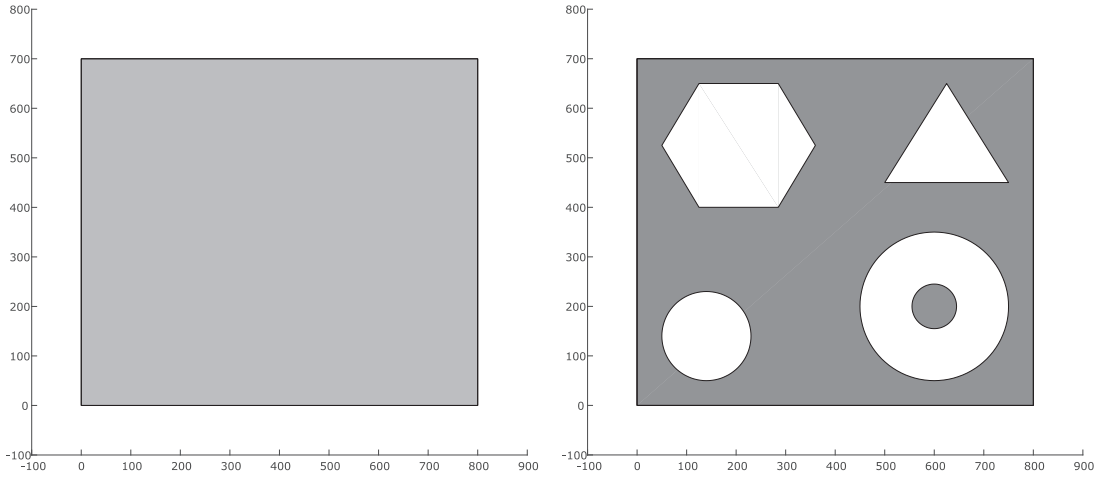

Figure 19 Comparison of experimental environments of optimal sensor deployment (OSD): (a) experimental environment of area size , where here the objective is to maximize the area covered by a set of sensor nodes deployed within said service area; and (b) experimental environment with four disjointed regions of interest (ROIs) (shapes colored white), were the area that is required to be covered by the set of sensor nodes is that which is delimited by each ROIs.

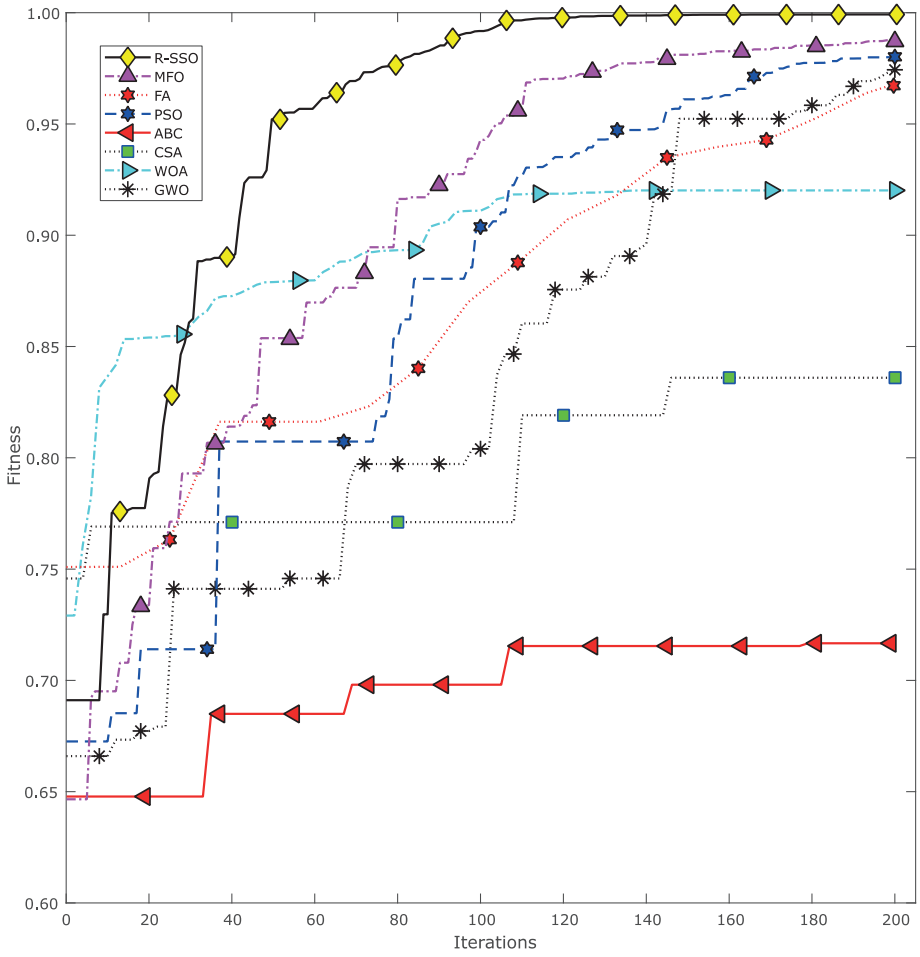

Figure 20 Evolution curves obtained by real-coded Social Spider Optimization (R-SSO), Moth-flame Optimization (MFO), Firefly Algorithm (FA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Crow Search Algorithm (CSA), Whale Optimization Algorithm (WOA), and Grey Wolf Optimizer (GWO) when applied to solve the optimal sensor deployment (OSD) problem for disjointed regions of interest (ROIs) within an experimental environment (with $A = 800 \times 700$, $N_{\text{sensors}} = 15$, and $r_s = 90$). The shown curves correspond to the averaged performance over each set of experimental runs. For all cases, a population size of $N_{\text{pop}} = 50$ individuals and a maximum number of iterations of $k_{\max} = 200$ are considered.

specific OSD task, with each of them suffering from stagnation during a noticeable number of iterations of their search process.

Finally, in Figures 21 through 29, we show an example of an initial arrangement of sensor positions as well as the best sensor deployment configurations achieved by applying R-SSO, MFO, FA, PSO, CSA, ABC, WOA, and GWO to said initial set of positions. From this set of graphical results, it is clear that R-SSO achieves the best sensor nodes configuration from among the compared methods, with it covering practically the whole area of each of the modeled disjointed ROIs.

6. CONCLUSIONS

In this paper, a real-coded sensor deployment approach based on the SSO algorithm has been proposed to solve the problem of OSD in WSNs. Different to most of the methods currently reported on the literature, which only consider a set of discrete-coded locations as candidate positions to deploy sensor nodes, our proposed method is able to explore any spatial location within a given ROI. Under our proposed approach, two technical challenges commonly seen within the problem of OSD are handled: on first instance, by considering real candidate locations instead of a set of finite discrete locations our proposed method has the flexibility to explore potentially better solutions to the sensor deployment problem, while also increasing its performance when placement restrictions are introduced; on second place, since the actual covered area is considered for calculating the rate coverage of a given sensor arrangement configuration, the uncertainty of the model is reduced, as this approach provides a better insight on the actual amount of the ROI that is being monitored by the deployed sensor network.

In order to evaluate the performance of our proposed method, a collection of comparative experiments which evaluate the performance of our proposed method against other similar approaches have been conducted. Our experimental results are divided three

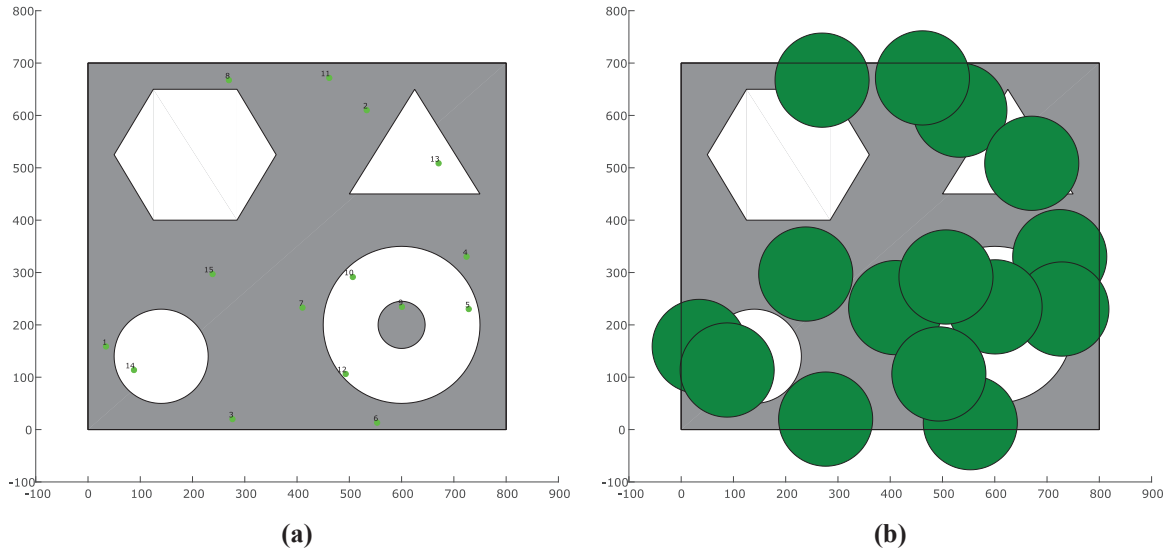

Figure 21 Initial configuration of positions for a set of 15 sensor nodes deployed to cover a set of disjointed regions of interest (ROIs) within an experimental environment of area size $A = 800 \times 700$: (a) sensor nodes locations and (b) area covered by the sensing devices on their initial locations.

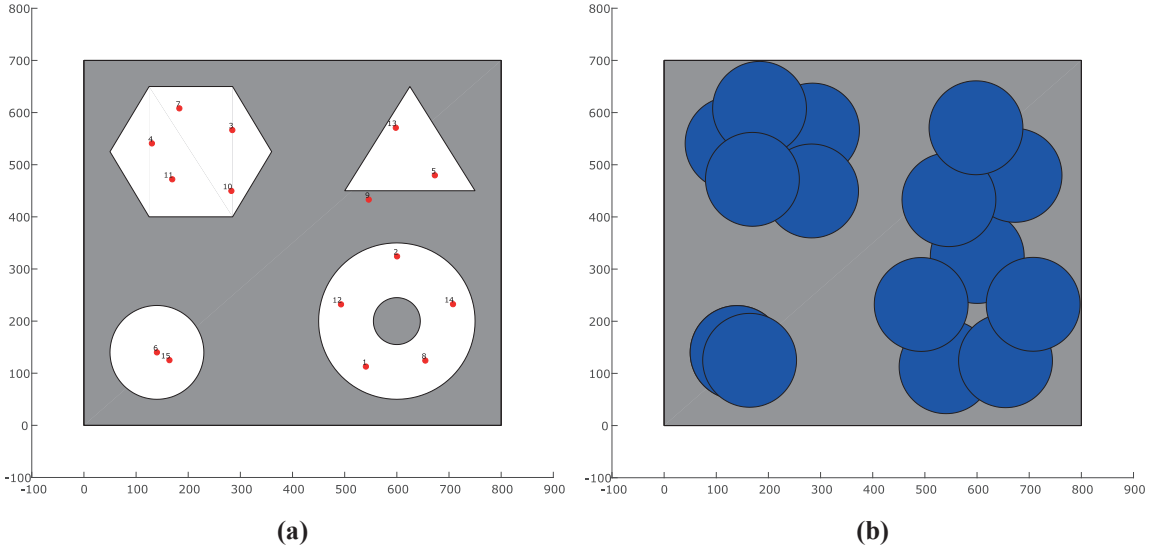

Figure 22 Positions configuration for a set of 15 sensor nodes deployed to cover a set of disjointed regions of interest (ROIs) within an experimental environment of area size $A = 800 \times 700$, obtained by applying the real-coded Social Spider Optimization (R-SSO) approach: (a) sensor nodes locations, and (b) area covered by the deployed sensing devices.

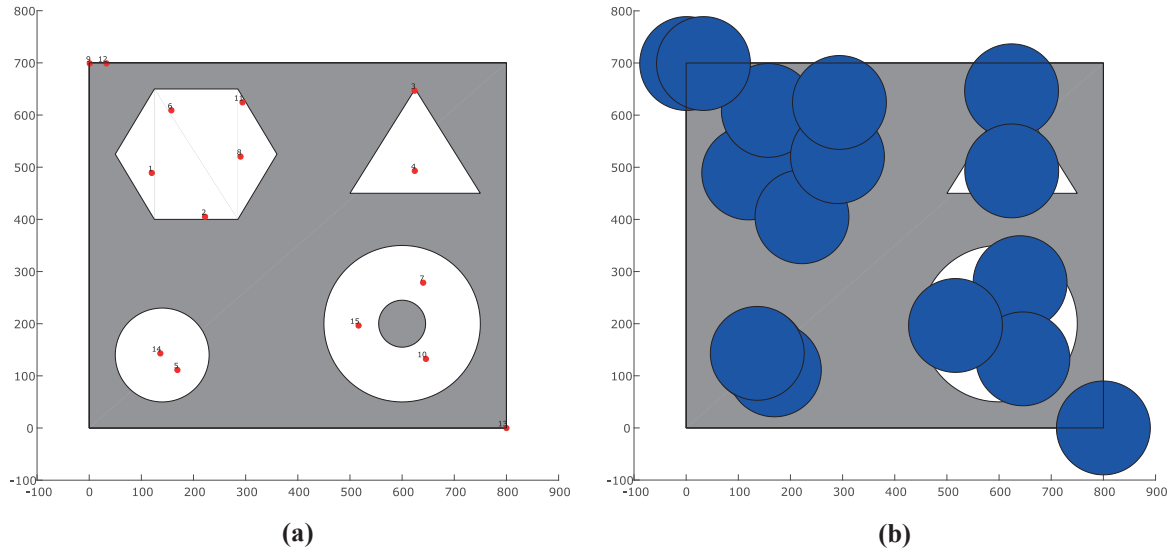

Figure 23 | Positions configuration for a set of 15 sensor nodes deployed to cover a set of disjointed regions of interest (ROIs) within an experimental environment of area size $A = 800 \times 700$, obtained by applying the Moth-flame Optimization (MFO) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

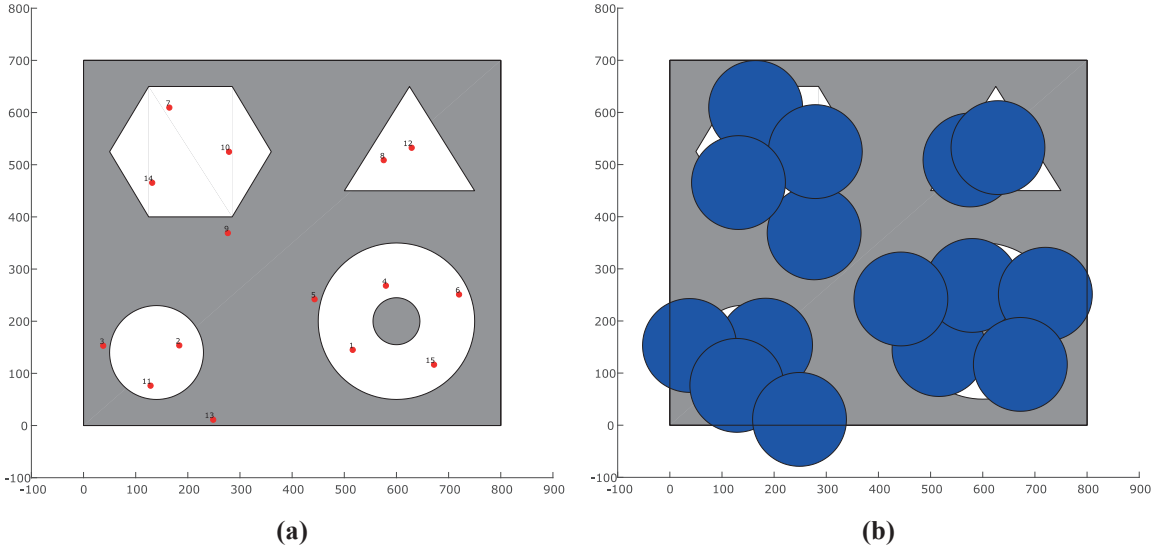

Figure 24 | Positions configuration for a set of 15 sensor nodes deployed to cover a set of disjointed regions of interest (ROIs) within an experimental environment of area size $A = 800 \times 700$, obtained by applying the Firefly Algorithm (FA) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

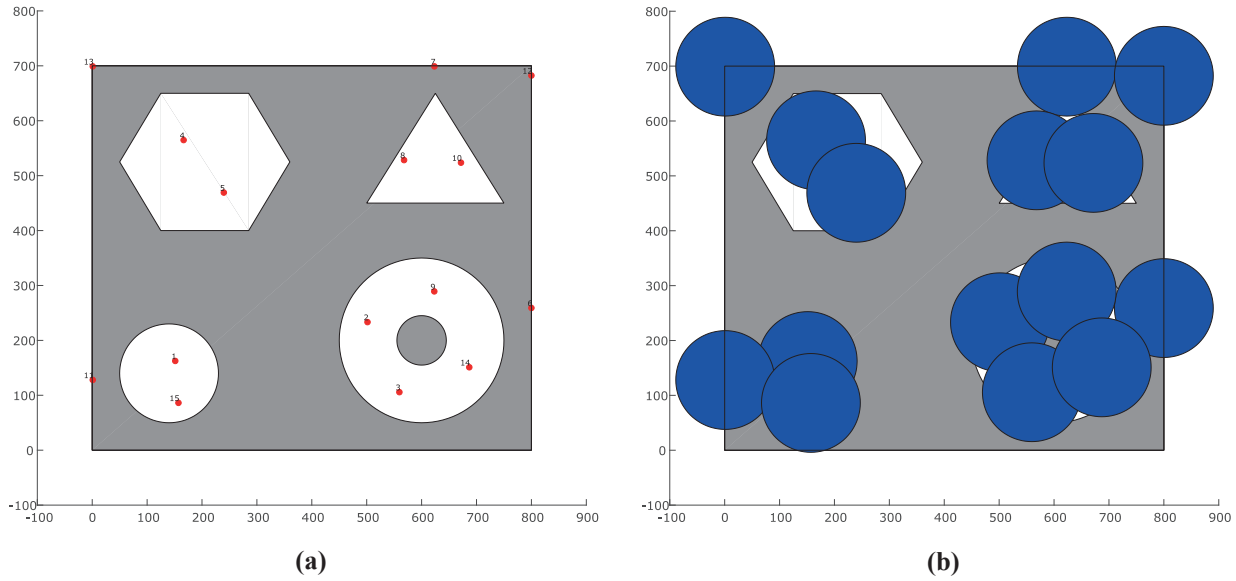

Figure 25 Positions configuration for a set of 15 sensor nodes deployed to cover a set of disjointed regions of interest (ROIs) within an experimental environment of area size $A = 800 \times 700$, obtained by applying the Particle Swarm Optimization (PSO) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

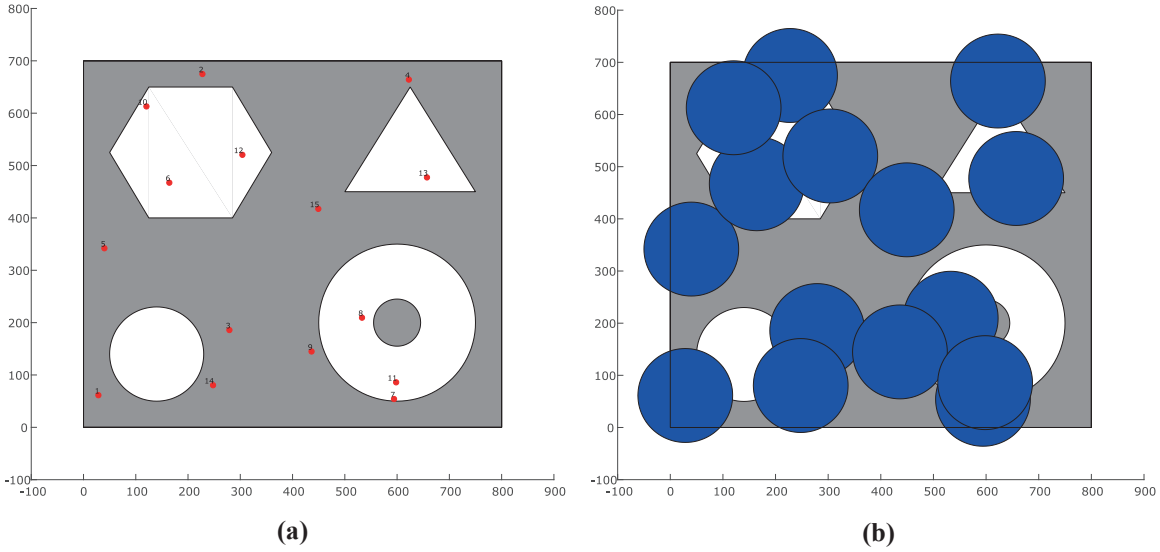

Figure 26 Positions configuration for a set of 15 sensor nodes deployed to cover a set of disjointed regions of interest (ROIs) within an experimental environment of area size $A = 800 \times 700$, obtained by applying the Crow Search Algorithm (CSA) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

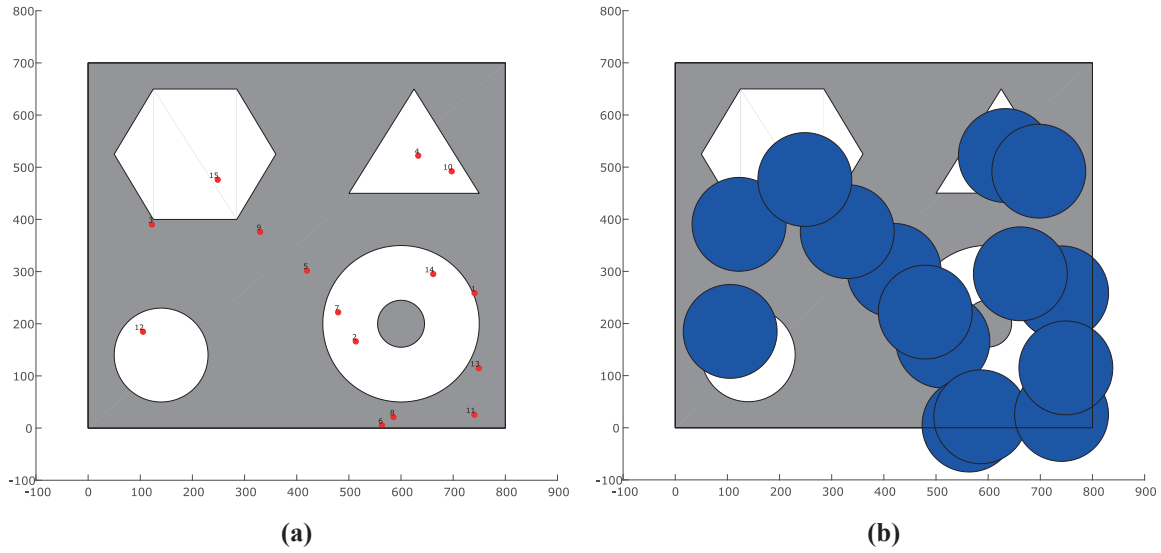

Figure 27 Positions configuration for a set of 15 sensor nodes deployed to cover a set of disjointed regions of interest (ROIs) within an experimental environment of area size $A = 800 \times 700$, obtained by applying the Artificial Bee Colony (ABC) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

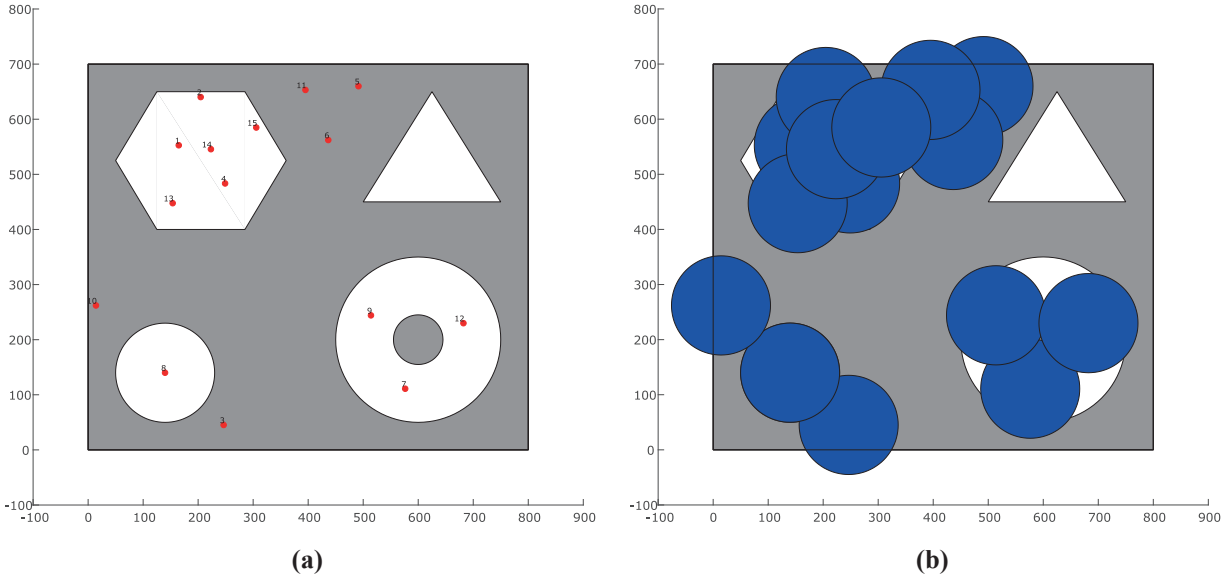

Figure 28 Positions configuration for a set of 15 sensor nodes deployed to cover a set of disjointed regions of interest (ROIs) within an experimental environment of area size $A = 800 \times 700$, obtained by applying the Whale Optimization Algorithm (WOA) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

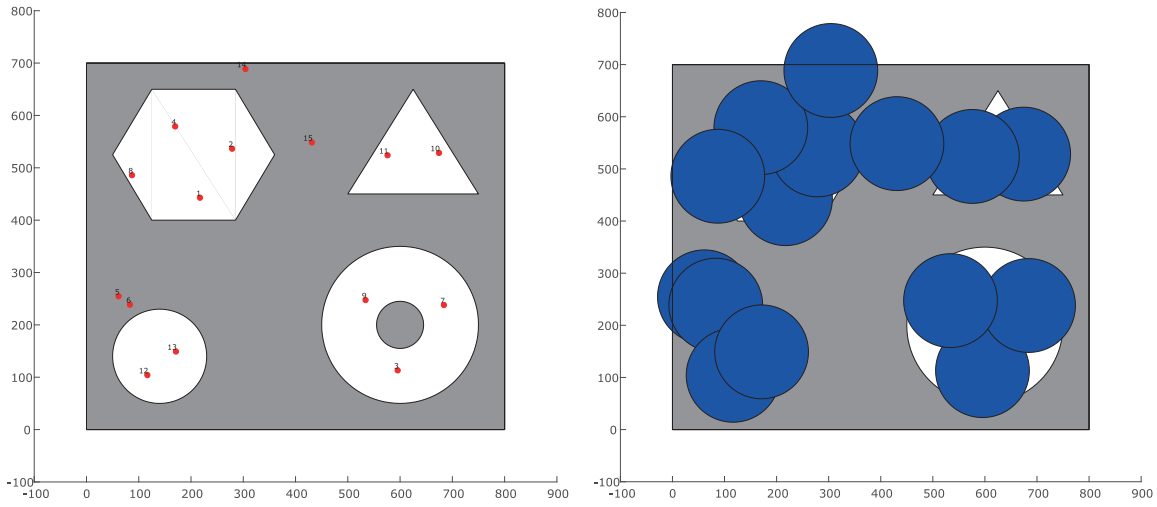

Figure 29 Positions configuration for a set of 15 sensor nodes deployed to cover a set of disjointed regions of interest (ROIs) within an experimental environment of area size $A = 800 \times 700$, obtained by applying the Grey Wolf Optimizer (GWO) approach: (a) sensor nodes locations and (b) area covered by the deployed sensing devices.

separated stages: for our first set of experiments, we have compared the performance of our R-SSO-based OSD approach against that of the D-SSO method previously proposed in [19]; for the second stage, we have compared the performance of our R-SSO approach against other metaheuristics-based methods adapted to work in terms of real-coded OSD, which include optimization schemes such as the MFO algorithm [20], FA [21], PSO [22], ABC [23], WOA [24], CSA [25], and GWO [26]. Finally, for our third and final set of experiments, we have compared the performance of R-SSO, MFO, FA, PSO, CSA, BA, WOA, and GWO with regard to a special case of OSD. As in the previous two sets of experiments, this study involved the OSD of a WSN, aimed cover several specific disjointed regions of different shape and size within the proposed experimental environment, an experiment not commonly seen on the literature.

In all of the performed experiments, the proposed R-SSO OSD approach has demonstrated to have the best performance when compared to all other of the compared methods. The remarkable performance produced by our proposed scheme is related to two important characteristics of the R-SSO algorithm: 1. its task division scheme, which allows the SSO population to be divided in two subpopulations with different functions and 2. the specialized operators applied by male and female spiders, which allows them to experiment different behaviors as the search process goes on. In general, these properties allow the SSO algorithm to manifest a better tradeoff between the exploration and exploitation of solutions, thus enhancing the probability to find more competent solutions.

CONFLICT OF INTEREST

The authors declare that there are no conflict of interest implicted in this work.

AUTHORS' CONTRIBUTIONS

- All authors are credited for their contribution on the writing and revisions of the submitted manuscript.
- All of the experiments reported in this paper were designed and performed by Oscar Maciel-Castillo and Bernardo Morales-Castañeda
- The data and results presented on this work were analyzed and validated Fernando Fausto and Erik Cuevas.

ACKNOWLEDGMENTS

The completion of this work wouldn't have been possible without the colaboration of Dr. Oscar Maciel-Castillo and Dr. Bernardo Morales-Castañeda, who designed and performed all of the experiments reported in this manuscript. We also thanks Dr. Erik Cuevas for their valuable assistance and observations during the development of this work.

REFERENCES

[1] M. Iqbal, M. Naeem, A. Anpalagan, A. Ahmed, M. Azam, Wireless sensor network optimization: multi-objective paradigm, Sensors. 15 (2015), 17572–17620.
[2] H. Chizari, M. Hosseini, T. Poston, S.A. Razak, A.H. Abdullah, Delaunay triangulation as a new coverage measurement method in wireless sensor network, Sensors. 11 (2011), 3163–3176.
[3] H. Fan, M. Li, X. Sun, P.-J. Wan, Y. Zhao, Barrier coverage by sensors with adjustable ranges, ACM Trans. Sens. Netw. 11 (2014), 1–20.
[4] T.P. Lambrou, Optimized cooperative dynamic coverage in mixed sensor networks, ACM Trans. Sens. Netw. 11 (2015), 46.
[5] Y. Wang, G. Cao, Achieving full-view coverage in camera sensor networks, ACM Trans. Sens. Netw. 10 (2013), 1–31.
[6] N. Bartolini, T. Calamoneri, T. La Porta, C. Petrioli, S. Silvestri, Sensor Activation and Radius Adaptation (SARA) in heterogeneous sensor networks, ACM Trans. Sens. Netw. 8 (2012), 1–34.
[7] X. Li, G. Fletcher, A. Nayak, I. Stojmenovic, Placing sensors for area coverage in a complex environment by a team of robots, ACM Trans. Sens. Netw. 11 (2014), 1–22.

[8] P.K. Sahoo, J.P. Sheu, W.S. Lin, Dynamic coverage and connectivity maintenance algorithms for wireless sensor networks, in Proceeding of 2007 2nd International Conference on Communication Systems Software and Middleware, Bangalore, India 2007.
[9] H. Wang, Y. Chen, S. Dong, Research on efficient-efficient routing protocol for WSNs based on improved artificial bee colony algorithm, IET Wirel. Sens. Syst. 7 (2017), 15–20.
[10] O.M. Alia, A. Al-Ajouri, Maximizing wireless sensor network coverage with minimum cost using harmony search algorithm, IEEE Sens. J. 17 (2017), 882–896.
[11] Y. Zeng, C.J. Sreenan, N. Xiong, L.T. Yang, J.H. Park, Connectivity and coverage maintenance in wireless sensor networks, J. Supercomput. 52 (2010), 23–46.
[12] D.S. Deif, S. Member, Y. Gadallah, S. Member, An ant colony optimization approach for the deployment of reliable wireless sensor networks, IEEE Access. 5 (2017), 10744–10756.
[13] M. Argany, F. Karimipour, F. Mafi, A. Afghantoloee, Optimization of wireless sensor networks deployment based on probabilistic sensing models in a complex environment, J. Sens. Actuator Netw. 7 (2018), 20.
[14] H. Wang, Y. Li, T. Chang, S. Chang, An effective scheduling algorithm for coverage control in underwater acoustic sensor network, Sensors. 18 (2018), 2512.
[15] M. Sangeetha, A. Sabari, Genetic optimization of hybrid clustering algorithm in mobile wireless sensor networks, Sens. Rev. 38 (2018), 526–533.
[16] M. Argany, M. Mostafavi, C. Gagné, Context-aware local optimization of sensor network deployment, J. Sens. Actuator Netw. 4 (2015), 160–188.
[17] X. Wu, S. Lei, W. Jin, J. Cho, S. Lee, Energy-efficient deployment of mobile sensor networks by PSO, in Conference of Advanced Web and Network Technologies, and Applications, Harbin, China 2006, vol. 3842, pp. 373–382.
[18] D.B. Jourdan, O.L. de Weck, Multi-objective genetic algorithm for the automated planning of a wireless sensor network to monitor a critical facility, in Proceedings of SPIE 5403, Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense III, Orlando, Florida 2004, pp. 565–575.
[19] Y. Zhou, R. Zhao, Q. Luo, C. Wen, Sensor deployment scheme based on social spider optimization algorithm for wireless sensor networks, Neural Process. Lett. 1 (2018), 71–94.
[20] S. Mirjalili, Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm, Knowledge-Based Syst. 89 (2015), 228–249.
[21] X. Yang, Firefly algorithm, lévy flights and global optimization, in: M. Bramer, R. Ellis, M. Petridis (Eds.), Research and Development in Intelligent Systems XXVI, Springer, London, 2010.
[22] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intell. 1 (2007), 33–57.
[23] D. Karaboga, B. Basturk, On the performance of Artificial Bee Colony (ABC) algorithm, Appl. Soft Comput. J. 8 (2008), 687–697.
[24] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016), 51–67.
[25] A. Askarzadeh, A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm, Comput. Struct. 169 (2016), 1–12.
[26] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014), 46–61.
[27] E. Cuevas, M. Cienfuegos, D. Zaldívar, M. Pérez-cisneros, A swarm optimization algorithm inspired in the behavior of the social-spider, Expert Syst. Appl. 40 (2013), 6374–6384.
[28] M.-A. Díaz-Cortés, E. Cuevas, R. Rojas, Engineering Applications of Soft Computing, Springer International Publishing, Cham, Switzerland, 2017.
[29] E. Cuevas, M.A. Díaz Cortés, D.A. Oliva Navarro, Advances of Evolutionary Computation: Methods and Operators, first ed., Springer International Publishing, Cham, Switzerland, 2016.
[30] E. Cuevas, V. Osuna, D. Oliva, Evolutionary Computation Techniques: A Comparative Perspective, vol. 686, Springer International Publishing, Cham, Switzerland, 2017.
[31] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 2nd ed., MIT Press and McGraw Hill, Cambridge, Massachusetts, 2001. https://mitpress.mit.edu/books/introduction-algorithms-second-edition.
[32] K. Chakrabarty, S.S. Iyengar, H. Qi, E. Cho, Grid coverage for surveillance and target location in distributed sensor networks, IEEE Trans. Comput. 51 (2002), 1448–1453.
[33] Y. Zou, K. Chakrabarty, Sensor deployment and target localization based on virtual forces, Infocom. 2 (2003), 1293–1303.