Metaheuristics for Maximization of Obstacles Constrained Area Coverage in Heterogeneous Wireless Sensor Networks

Nguyen Thi Hanh[a,b], Huynh Thi Thanh Binh[a,*], La Van Quan[a], Nguyen Duc Nghia[a], Nilanjan Dey[c]

[a] *Hanoi University of Science and Technology, Vietnam*
[b] *Phuong Dong University, Vietnam*
[c] *Techno India College of Technology, Kolkata, India*

Abstract

Wireless Sensor Networks (WSNs) collect and transfer environmental data from a predefined field to the base station where these data are processed and analyzed. A major problem of designing wireless sensor networks is coverage maximization, whose goal is to deploy a given number of sensor nodes in a way that maximizes area coverage of a given network without violating practical constraints. This problem is known to be NP-hard and thus requires metaheuristic approaches for practical problem sizes. The academic community has extensively explored this problem, but most prior studies dealt with WSNs that are free of obstacles. In this paper, we propose a new problem formulation that captures the pragmatic presence of obstacles and sensor nodes with heterogeneous communication ranges. Two metaheuristics, namely Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), are proposed to tackle this problem. Our new contributions include partial use of heuristic initialization, new fitness function, modified virtual force algorithm (MVFA), addition of a uniform deceleration to the calculation of inertia weight and addition of the influence of sub-populations' head individuals. The proposed algorithms are comprehensively experimented and compared. Experimental results not only suggest which algorithms should be applied to which cases, but also provide insights into parameter settings, effect of heuristic initialization and effect of MVFA in each case. These conclusions are meaningful for our future research on obstacles constrained area coverage problems related to connectivity and lifetime of WSNs.

Keywords:
`Area Coverage Optimization, Genetic Algorithm, Particle Swarm Optimization, Heuristic Initialization, Wireless Sensor Network, Monte Carlo, Virtual Force Algorithm.`

*Corresponding author. Huynh Thi Thanh Binh, Hanoi University of Science and Technology, Vietnam.
Email addresses: `hanhnt@dhpd.edu.vn` (Nguyen Thi Hanh), `binhht@soict.hust.edu.vn` (Huynh Thi Thanh Binh), `lavanquan.kstn.cntt@gmail.com` (La Van Quan), `nghiand@soict.hust.edu.vn` (Nguyen Duc Nghia), `neelanjan.dey@gmail.com` (Nilanjan Dey)

1. Introduction

Networks have so far been playing a crucial role in human society. However, with the surging popularity of embedded systems and wireless communication, the traditional model of networks, wired network, is gradually becoming out of date in terms of convenience and esthetics. This is why wireless sensor networks are coming to spotlight. Wireless Sensor Networks (WSNs) comprise a large number of sensors working as network nodes. Those sensors are able to monitor physical phenomena and produce sensory data. Due to their great number of strengths including easy deployment in various region, compact size, etc., they are applied to a wide range of fields such as surveillance, object tracking, health care, biological detection [**?** **?** **?** **?**] and most noticeably, Internet of Things (IoT) [**?** **?** **?** **?**].

Typically, a WSN comprises two main components which are base station (also called sink node) and a set of sensors nodes. The capacity of collecting and transferring information of a WSN is based on the function of sensor nodes. In fact, each sensor node consists of two parts, namely a data processor and a communicator. While the data processor carries out the task of collecting information, the communicator is responsible for transferring those data directly or through other nodes in the systems to the base station. Information stored and processed in the base station is used for other purposes such as monitoring and surveillance. Moreover, because the mobility and flexibility of WSN make the sensor deployment easier in various kinds of environments [**?** **?**], such processes could even be achieved in complicated terrains or harsh conditions.

Depending on types of WSNs's application, the terrain for deploying sensors may vary, e.g., highlands, mountains, valleys and may contain many kinds of obstacles. Such obstacles appear to be challenging for sensor deployment in a lot of means [**?** **?**]. Those obstacles might not allow sensors to be deployed within its area, which reduces available space for deployment. In addition, obstacles might obstruct the sensing range of each sensor and thus, become a threat to quality of coverage. Obviously, a random deployment is never bound to ensure a good performance of network because sensors are highly put inside obstacles or the overlapping of area coverage of sensors themselves is great.

There are a couple of models for sensor deployment. In 2013, Yourim Yoon et al. [**?**] proposed a new problem which aims to optimize area coverage in a predetermined domain with a given set of different types of sensors. Based on this model, D. Thi Ha Ly et al. [**?**] proposed an improved genetic algorithm with better computational complexity. Sensor deployment in environments containing obstacles also attracted a lot of attention by WSN researchers. Various aspects of this topic are extensively studied in [**?** **?** **?** **?**].

In this paper, we consider an area coverage maximization problem which focuses on finding a node placement that provides the largest coverage in the domain. In this problem, the coverage of a sensor is defined as the area of a disk centered at the sensor with the radius of its sensing range r_s (binary coverage model). There have been a number of works in the literature that solved the coverage maximization problem starting with the assumption that all the sensor nodes are homogeneous [**?**]. Later, different types of sensor nodes with varying ranges were introduced to the problem for increasing the practicality in [**?** **?**]. The problem was first stated and proved to be NP-hard [**?**]. Consequently, heuristics and

meta-heuristics are potential and practical approaches for solving the problem in [**?** **?** **?**]. One of the main limitations of the existing models for solving the coverage maximization problem is that they do not include obstacles in their problem settings. In this paper, we will extend the problem formulation to include obstacles for more practical problem settings. All of the obstacles are rectangles with edges parallel to the axes. The areas taken up by obstacles also block the sensing of sensors. The main contributions of this paper, therefore, are:

- A novel problem formulation considering the presence of obstacles with the connectivity constraints to guarantee a feasible solution. This proposal requires the overlapping concept introduced in [**?**] to be improved so that it performs well in case of intersection between one node's coverage and the obstacles.
- Two algorithms GA and PSO for solving the formulated problem. The proposed methods include improvement of population initialization with the use of a heuristic algorithm, a new fitness function and a modified virtual force algorithm.
- Analysis of experimental results and meaningful conclusions with respect to the problems and the proposed algorithms.

The rest of this paper is organized as follows. Problem formulation and related works are presented in Section 2 and 3 respectively. Section 4 introduces the proposed algorithms. Section 5 explains our experiments and provides an analysis of the results. Finally, section 6 concludes the paper and suggests new directions for future works.

2. Problem Formulation

The problem of maximizing area coverage of a WSN with a given number of sensors was introduced by Y. Yoon in [**?**]. However, realistic sensor fields usually have obstacles (e.g. walls, buildings, trees, etc...), which block communication among the sensors to different extents depending on their shapes, electrical conductivity, as well as the distance to the sensors. Therefore, this paper focuses on the problem of maximizing area coverage of a given number of heterogeneous sensors on a surveillance region with obstacles. We assume that obstacles totally block the wireless communication, thus incurring zero coverage within the area taken up by an obstacle as illustrated in Fig.2.

We define the maximization of obstacles constrained area coverage in WSNs as follows: Given a surveillance region A with n static sensors of k types, let n_i be the number of sensors of type i, with $i = 1, \ldots, k$ such that $n = \sum_{i=1}^{k} n_i$. In this model, each sensor s has a sensing range r_s and a communication range r_c such that $r_c = 2 \times r_s$ [**?**]. The sensing range characterizes the sensing capability of a sensor and the communication range the connectivity of the network. Let $O = \{o_1..o_m\}$ represnt the set of obstacles in the given region, where o_i is a rectangle obstacle specified by its lower-left and upper-right corners (u_{i_1}, v_{i_1}) and (u_{i_2}, v_{i_2}) respectively with $i = 1, \ldots, m$. The objective of the problem is to find locations $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ for sensors $s_1, s_2, \ldots, s_n$ so that the area coverage on

the given surveillance region A is maximized. This paper uses the Boolean disk coverage model from [**?**]. Let $d(s,x)$ denote the Euclidean distance between sensor s and point x and constant r_s the sensing range. The coverage function of the model is given by:

$$f(d(s,x)) = \begin{cases} 1, \text{if } d(s,x) \leq r_s \\ 0, \text{otherwise} \end{cases} \tag{1}$$

Let $sr_i(x_i, y_i)$ denote the region covered by sensing range r_i of sensor s_i centered at (x_i, y_i) and $so_t(u_{t_1}, v_{t_1}, u_{t_2}, v_{t_2})$ the region covered by rectangle obstacle defined by its two corners (u_{t_1}, v_{t_1}) and (u_{t_2}, v_{t_2}). The problem can be formulated as follows:

$$\begin{aligned} \text{maximize } & coA = area\left(\bigcup_{i=1}^{n} sr_i(x_i, y_i) \bigcap \left(A \setminus \bigcup_{t=1}^{m} so_t(u_{t_1}, v_{t_1}, u_{t_2}, v_{t_2})\right)\right) \\ \text{subject to } & (x_i, y_i) \in A; (x_i, y_i) \notin o_t; \\ & t = 1, \ldots, m; i = 1, \ldots, n \end{aligned} \tag{2}$$

where (x_i, y_i) is the center of sensor s_i.

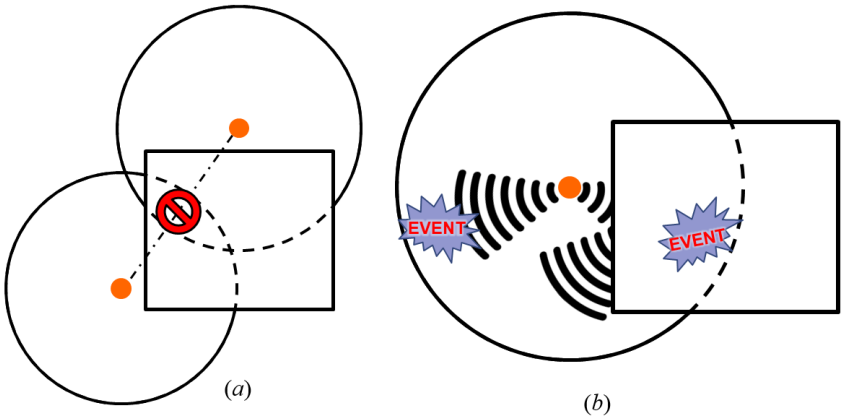


Figure 1: (a) An obstacle blocking wireless communication between two sensors, (b) An obstacle blocking the sensing of a sensor. The disk represents the sensing ranges of sensors and the rectangles represent obstacles blocking the communication.

It should be noted that the same problem but without obstacles has been proven to be NP-hard [**?**]. Therefore, the problem formulated in this paper is also NP-hard. Our approaches to approximately solving the problem are explained in the Proposed Algorithms section.

3. Related Work

Coverage is considered one of the most important factors contributing to the quality of WSN systems. The coverage maximization problem has been extensively investigated in various works [**? ? ?**].

The coverage problem in WSNs [**?**] involved six designing issues, which are coverage type, deployment method, coverage degree, coverage radio, activity scheduling, and network connectivity. Based on three different coverage types, coverage problems are classified into three categories called point coverage (also called as target coverage), area coverage and barrier coverage.

Taking area coverage into consideration, in 2008 You-Chiun and Yu-Chee [**?**] proposed a competition-based scheme and a pattern-based scheme to solve the $k - coverage$ sensor deployment problem. Their solutions considered both the binary and probabilistic sensing models and freed the relationship between the communication range and the sensing range of sensors. Experimental simulations demonstrated the efficiency of proposed schemes towards the area coverage problem in WSNs.

Later, in [**?**] Ozturk C. et al. addressed the area coverage problem by dynamically deploying WSNs that consists of both static and mobile sensors. To obtain more realistic results while computing the effectively covered area, they considered a probabilistic detection model. The artificial bee colony algorithm was applied, and its efficiency was compared with that of another swarm based optimization technique, particle swarm optimization. Simulation results showed that the artificial bee colony algorithm obtained better deployments for WSNs than its opponent does.

Arguing that there is a trade-off between the coverage and network lifetime in WSNs, Changlei and Guohong in 2011 [**?**] scheduled the active period for sensor nodes in order to maximize the spatial-temporal coverage, which was defined as the product of the size of a given area and the length of the period during which that area is covered. A distributed parallel optimization protocol was proposed, whose performance was shown through simulations to be better than other schemes in terms of network lifetime, coverage redundancy, convergence time, and event detection probability. With a different approach, Qin and Qianping 2012 [**?**] suggested a virtual force directed coverage optimization (VFCSO) problem in WSNs. This method was based on node energy, virtual forces and the distances between nodes to elect the cluster head. Theoretical analysis showed that network coverage and node survival time were optimized, and simulation results compared with the adaptive geographical fidelity algorithm (Xu Y [**?**]) strongly supported this conclusion.

Yourim and Yong-Huyk 2013 [**?**] proposed another model for the area coverage problem which aims at finding the optimal placement locations for a predefined number of sensor nodes having different sensing ranges, such that the largest area is covered. In this model, the term coverage represents the sensing coverage and is defined by a perfect circle disk centered at the sensor node whose radius equals the sensing range. Assuming the sensing ranges of sensors are heterogeneous, the authors proved that the problem is NP-hard and proposed a genetic algorithm for finding a proper deployment topology. Specifically, an individual representing a solution is denoted by an array of sensor nodes' coordinates, each of which shows the actual position of the corresponding sensor node in reality. The sensor nodes are ordered descendingly by sensing range. An individual X will be evaluated by its coverage area (CoA) corresponding to its real coordinates, which is computed by Monte Carlo technique. The idea of Monte Carlo method is to randomly create a very large number of points (L points) and check if those points are in the sensor node X's coverage area.

The complexity of this method for computing the fitness of each individual in the genetic algorithm is $O(nL)$ with $L >> n$, where n is the number of sensors [**?**]. The experimental results showed a great improvement in the quality of area coverage, yet it is still far from reaching the optimal solution, and the fitness computational time is rather expensive (since $L >> n$).

To avoid the expensive fitness computation time in [**?**], a new approach for fitness computation that has the complexity of $O(n^2)$ called the Improved Genetic Algorithm (IGA) was proposed in [**?**]. In order to reduce the complexity, the authors introduced a new concept of the overlapping called *Olap*, which highly reduces the computational complexity required in the evaluation stage. Moreover, a modified version of the original mutation was investigated in the algorithm, which utilizes the dynamic Gaussian function compared with the static one in [**?**]. Thanks to these modifications, the new genetic algorithm in [**?**] not only obtained a better set of experimental results but also achieved the better fitness computation complexity reducing from $O(nL)$ with $L >> n$ in [**?**] to $O(n^2)$.

In 2015, Anju and Rishi [**?**] proposed a new model for this problem, whose goal is to find the smallest density of sensor nodes such that they entirely cover a given field. This problem was investigated with two placement strategies, namely, deterministic node placement and random node deployment, and they solved the problem using mathematical analysis.

The above models were somehow far from fully reflecting some pragmatic situations, especially when there exist obstacles (or banned regions) that sensor nodes cannot be located inside. This scenario was introduced in [**?**] by Andrew Howard et al., in which the potential-field-based approach is applied to maximize the total covered area of sensor nodes. The fields were deployed such that each node is repelled by both obstacles and other nodes, thereby forcing a fair distribution of sensors throughout the environment. In 2009 C. Y. Chang et. al. [**?**] presented the efficient obstacle-resistant robot deployment (ORRD) algorithm, which involves the design of a node placement policy, a serpentine movement policy, obstacle-handling rules, and boundary rules. By applying the proposed ORRD, the robot rapidly deploys a near-minimal number of sensor nodes to achieve full sensing coverage, even though there exist unpredicted obstacles with regular or irregular shapes.

Thus, we propose a new model as introduced in the previous section, which overcomes the mentioned weaknesses by investigating obstacles in rectangular shape and maintaining the connectivity between nodes using distance constraints. The sensors will be developed by adding a certain property to environment, which is the presence of obstacles. Given a number of sensors of different types, the question is how those sensors should be deployed to gain the largest area coverage in a predetermined domain.

Since our formulated problem admits the formulation in [**?**] as a special case (when the number of obstacles is 0), it is also NP-hard. We propose an improved version of fitness function and virtual force algorithm to adapt well to the existence of obstacles while still keeping their complexity of *Olap* as good as those in [**?**]. Moreover, we propose two metaheuristics to solve this problem. They will be described in the next section.

4. Proposed Algorithm

We propose two metaheuristics to deal with this model, namely Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Both algorithms use the same fitness function. The details of each algorithm are explained below:

4.1. Genetic Algorithm

In the following subsections, we will introduce the five steps (individual representation, population initialization, calculation of the fitness function, application of genetic operators and survival selection) of the proposed genetic algorithm (GA) in details.

4.1.1. Individual Representation

Each individual in the genetic population represents a solution to the problem. Due to the difference in radii of the sensors, genes representing the sensors are divided into k types, and it is complicated to represent individuals of different types. The representation is thus fragmented into k sections. To be more specific, each individual is encoded by an array of n pairs (x_i, y_i) where (x_i, y_i) are the center coordinates of the i^{th} sensor in a two-dimensional space satisfying:

$$\begin{cases} r_{s_i} \leq x_i \leq W - r_{s_i} \\ r_{s_i} \leq y_i \leq H - r_{s_i} \\ (x_i, y_i) \notin o_t; t = 1, \ldots, m; i = 1, \ldots, n \end{cases} \tag{3}$$

where r_{s_i} is sensing range of sensor i^{th}.

For example, with $n = 9$ sensors, the number of sensor types $k = 3$ including $n_1 = 2$ sensors of the type I, $n_2 = 4$ sensors of the type II, and $n_3 = 3$ sensors of the type III, an individual is encoded as in Fig.2, in which r_1, r_2, and r_3 correspond to the sensors of type I, II and III respectively.

4.1.2. Population Initialization

In GA, a population includes N individuals, each of which will be generated in one of three ways: randomly, with the new heuristic, or with the heuristic proposed in [**?**]. The initialization heuristics should allow the deployment of sensors on surveillance region A to be such that the sensors overlap each other, the boundaries and the obstacles as little as possible. The initialization processes are described as follows:

Random Initialization: The coordinates of sensors deployed randomly on surveillance region A have to satisfy conditions (3). Therefore, overlapping sensors are many as illustrated in Fig.3(a). *Heuristic Initialization proposed in* [**?**]: sensors are deployed end to end from boundary 0 to boundary (W, H). Therefore, the overlap of the sensors is greatly reduced, but the area near the upper boundary (W, H) has hardly any sensors deployed. It is hard to reach these positions even after many evolutionary generations, because the populations themselves do not contain the genetic materials that allow for deployment in these areas (Fig.3(b)). To overcome this difficulty, this paper proposes a *New Heuristic Initialization*

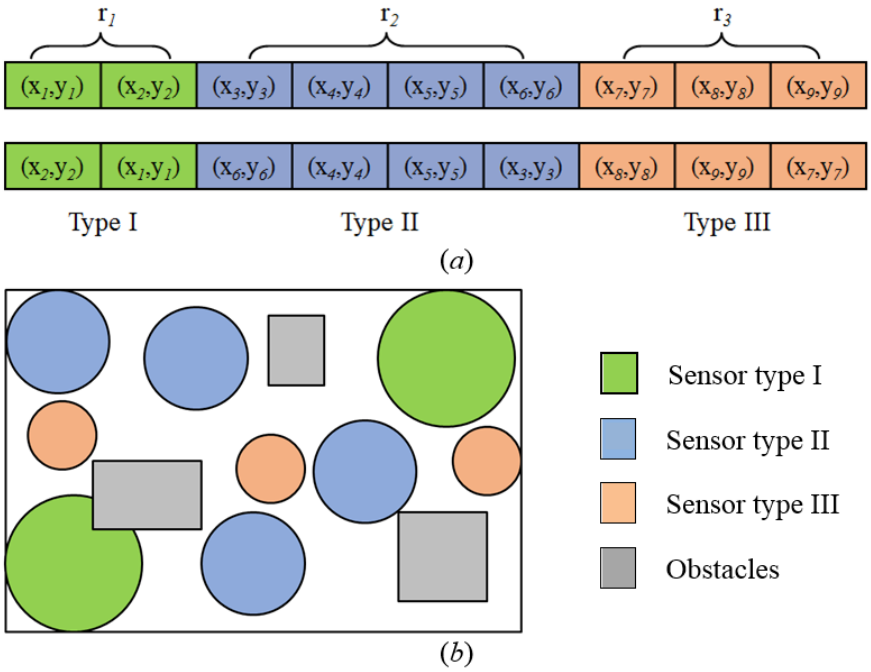


Figure 2: Individual representation: (a) Two the geneotype representations of one solution, (b) The phenotype of the solution.

as follows: sensors will be deployed end to end starting from the (W, H) boundary. Fig.3(c) depicts this initialization. After all three types of initializations are used, the population has genomic diversity and phenotypic diversity. The pseudo code for the new heuristic initialization algorithm is shown in Algorithms.1.

4.1.3. Fitness Function

The GA uses an improved version of the fitness function introduced in [**?**]. Originally, the fitness function [**?**] is calculated as follows: considering an individual $S = (s_1, s_2, .., s_n)$ with $s_i = (x_i, y_i)$ as the center coordinates of the i^{th} sensor, the overlap ($Olap$) of S is defined in [**?**] as follows:

$$Olap\,(S) = \sum_{i=1}^{n} \left(\sum_{j=i+1}^{n} overlap\,(s_i, s_j) + \sum_{m=1}^{4} overlap\,(s_i, b_p) \right) \tag{4}$$

where $overlap\,(s_i, s_j)$ is the coverage overlap between two sensors s_i and s_j; and $overlap\,(s_i, b_p)$ is the part of the coverage of sensor s_i that is out of the surveillance region A.

Using this method, the $Olap$ of the sensor set represents the overlapping areas among sensors and the parts of their coverage that are out of surveillance region A. The fitness function in [**?**] is the value of $Olap$, i.e. the higher the $Olap$ value, the worse the solution. Unlike the model presented in [**?** **?**], the proposed model in this paper has obstacles. Therefore, a modified fitness function that also takes into account the overlaps between

Algorithm 1: Proposed Heuristic Initialization

Input : n: number of sensors
k: number of sensor types
n_i: number of sensors of type $i, i = 1..k$
r_i: radii of sensor type $i, i = 1..k$
$O = \{o_1, o_2, \ldots, o_m\}$: list of obstacles
W, H: width and height of surveillance region $A = W \times H$

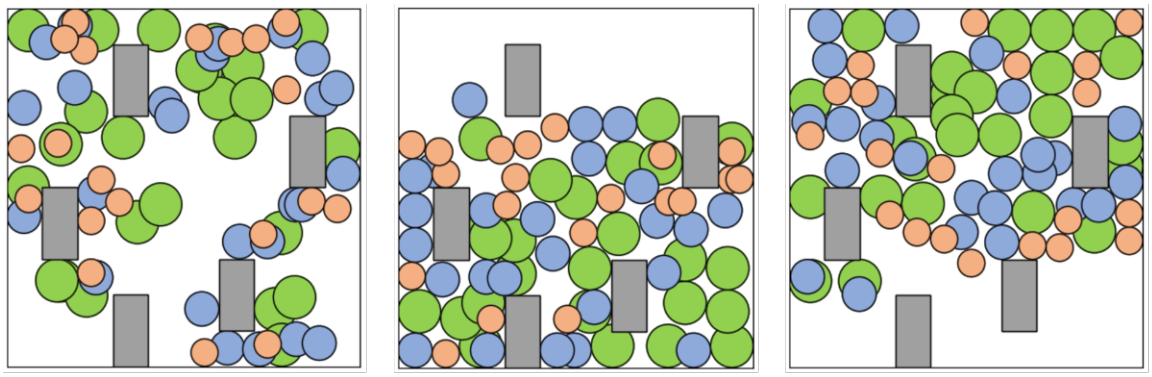
Output : The heuristic individual $S = [(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)]$.

```
Generate a random set of radii setR with n_1 radius r_1,...,n_k radius r_k
 Initialize S_1 = (W − setR_1, H − setR_1)
 count1 = 1
 count2 = n
 while i < n − 1 do
 |  S_i = S_{i−1}.x − setR_{i−1} + setR_i
 |   if S_i.x < 0 then
 |  |  count2 = count1
 |  |   count1 = 1
 |  |   S_i.x = r_i
 |  |   S_i.y = S_{i−count2}.y − setR_{i−count2} − setR_i
 |  end
 |  if S_i.x ≥ 0 then
 |  |  count1 = count1 + 1
 |  |   if i − count2 ≤ 1 then
 |  |  |  S_i.y = setR_i
 |  |  end
 |  |  if i − count2 > 1 then
 |  |   |  S_i.y = S_{i−count2}.y − setR_{i−count2} − setR_i
 |  |  end
 |  end
end
Adjust coordinates of the sensors using the MVFA
 Sort elements in S by radius
 return(S)
```

sensors and obstacles is needed. In this paper, we proposed a new fitness function called $OSlap$ that meets this need. For an individual $S = (s_1, s_2, .., s_n)$ with $s_i = (x_i, y_i)$, the overlap ($OSlap$) of S is defined as follows:

$$OSlap\,(S) = \sum_{i=1}^{n} \left(\sum_{j=i+1}^{n} overlap\,(s_i, s_j) + \sum_{p=1}^{4} overlap\,(s_i, b_p) + \sum_{q=i}^{m} overlap\,(s_i, o_q) \right) \quad (5)$$

The GA uses an improved version of the fitness function which had been introduced in formula (4). The improved fitness function, which is presented in formula (5), is based on the same concept of that in formula (4) but turns out to be more accurate as it calculates the area of the overlap ($OSlap$) instead of the width of the overlapping area. In formula (5), we consider three following cases:



(a) Random Initialization (b) Heuristic Initialization (Ly et al.) (c) New Heuristic Initialization

Figure 3: Three cases of Population Initialization

a) Two sensors overlap: $overlap\,(s_i, s_j)$, which is the overlapping area between the coverage areas of two sensors s_i and s_j, is calculated by formula (6) and illustrated in Fig.4.

$$Overlap\,(s_i, s_j) = \begin{cases} 0, & \text{if } d\,(s_i, s_j) \geq r_{s_i} + r_{s_j} \\ area_{ij}, & \text{if } \left|r_{s_i} - r_{s_j}\right| \leq d\,(s_i, s_j) < r_{s_i} + r_{s_j} \\ \pi.\left(\min\left(r_i, r_j\right)\right)^2, & \text{if } d\,(s_i, s_j) < \left|r_{s_i} - r_{s_j}\right| \end{cases} \tag{6}$$

where $d\,(s_i, s_j)$ is Euclidean distance between s_i and s_j; and $area_{ij}$ is calculated as the common area of the two sensors (illustrated by the green area in Fig.4).

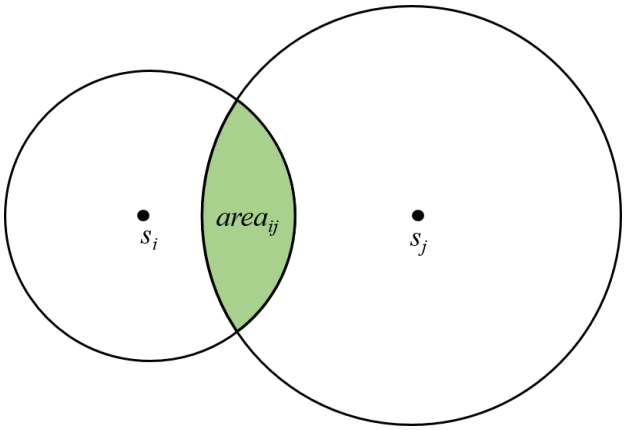


Figure 4: Two intersecting circle.

b) Part of the area coverage of a sensor is out of surveillance region A***:*** $overlap\,(s_i, b_p)$ is the part of the area coverage of sensor s_i that is out of surveillance region A. Possible sensor positions that lead to this case are illustrated in Fig.5 (a). We divide this case into three subcases, which are:

Subcase 1) The area coverage of sensor s_i intersects one of the four boundaries of the surveillance region A.

Fig.5 (b) depicts this subcase. $overlap\,(s_i, b_p)$ is equal to the area of circular segment BC (outside surveillance region A) of circle (A, AB) and is calculated using the following formula:

$$overlap\,(s_i, b_p) = \frac{1}{2} \cdot (\theta - sin\,(\theta)) \cdot r_{s_i}^2 \tag{7}$$

where $\theta = \angle BAC = 2 \times \arccos \frac{AD}{AB}$, AD is the distance from s_i to the boundary.

Subcase 2) The area coverage of sensor s_i intersects two of the four boundaries of domain A, and the intersection point of these two boundaries lies within the area coverage.
Fig.5 (c) depicts this subcase. $overlap\,(s_i, b_p)$ is equal to the area of circular segment BC (outside surveillance region A) and segment ED (outside surveillance region A) of circle (A, AB). $overlap\,(s_i, b_p)$ is calculated using the following formula:

Subcase 3) The area coverage of sensor s_i intersects two of the four boundaries of surveillance region A, and the intersection point of these two boundaries lies outside the area coverage.
Fig.5 (d) depicts this subcase. $overlap\,(s_i, b_p)$ is equal to the area of the circular segment BGC that lies outside surveillance region A and can be calculated by the following formula:

$$overlap\,(s_i, b_p) = \frac{1}{2}\left\{(\alpha + \beta) \cdot r_{s_i}^2 - (AE \cdot BE + AF \cdot FC)\right\} + \frac{\pi \cdot r_{s_i}^2}{4} - AF \cdot AE \tag{8}$$

where $\alpha = \angle BAE = \arccos \frac{AE}{AB}$, $\beta = \angle FAC = \arccos \frac{AF}{AC}$

c) Overlap between the coverage area of the sensor and that of obstacle: $overlap\,(s_i, o_q)$ is the overlap between the coverage area of sensor s_i and that of obstacle k^{th}. Because of the diversity of obstacles in terms of size, it is challenging to calculate the exact overlapping area. Therefore, $overlap\,(s_i, o_q)$ is calculated by an approximate method.

We divide the area around each obstacle into nine parts, each of which is a possible location to deploy a sensor. We label the locations as in Fig.6.

The similarity among case II, case IV, case VI and case VIII as well as that among case III, case V, case VII and case IX reduce the work. This paper will only discuss case I, case II and case III, which can represent other cases.

Case I: The sensor is deployed in area I (illustrated in Fig. 7(a)). The overlapping area is calculated as follows:

$$overlap\,(s_i, o_k) = \{min\,(r_{s_i}, AD) + min\,(r_{s_i}, AE)\}\,.\,\{min\,(r_{s_i}, AG) + min\,(r_{s_i}, AH)\} \tag{9}$$

Case II: The sensor is deployed in area II (illustrated in Fig. 7 (b)). The overlapping area is calculated as follows:

$$overlap\,(s_i, o_k) = \gamma \cdot (r_{s_i} - AD)\,.\left\{min\left(\sqrt{r_{s_i}^2 - AD^2}, DB\right) + min\left(\sqrt{r_{s_i}^2 - AD^2}, DC\right)\right\} \tag{10}$$

where DE is perpendicular to two opposite boundaries of surveillance region A and BC is perpendicular to the other two boundaries.

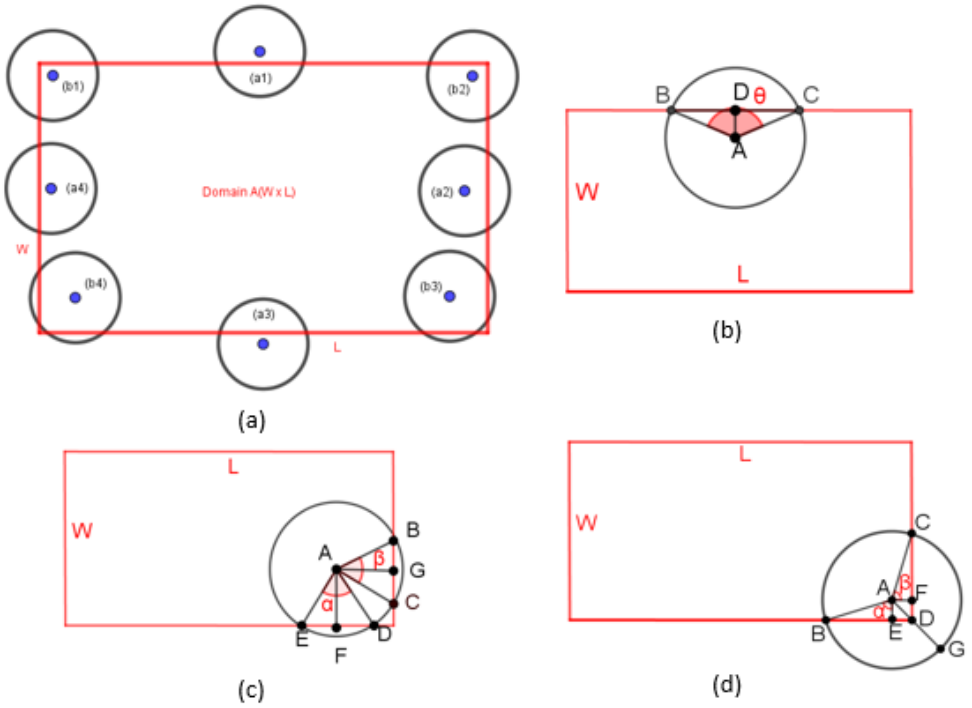


Figure 5: Cases in which part of the area coverage of a sensor is out of surveillance region A.

A constant value, γ ($\gamma < 1$) is added to increase the accuracy of formula (10), because $overlap\,(s_i, o_k)$ is approximated as the area of a rectangle whose two sides are BC and DE (Fig. 7 (b)).

Case III: The sensor is deployed in area III (illustrated in Fig. 7 (c)). The overlapping area is calculated as follows:

$$overlap\,(s_i, o_k) = \frac{1}{2\delta} \cdot \left\{\sqrt{r_{s_i}^2 - AF^2} - AE\right\} . \left\{\sqrt{r_{s_i}^2 - AE^2} - AF\right\} \tag{11}$$

where E and F are the projections of A on DC and BD. A constant value, $\delta < 1$ is added to increase the accuracy of formula (11), because $overlap\,(s_i, o_k)$ is approximated as the area of the right triangle BDC (Fig. 7 (c)).

Thus, the fitness of the individual S is:

$$Fitness\,(S) = OSlap\,(S) \tag{12}$$

$OSlap\,(S)$ calculated by (5) requires the time $O\,(n^2 + n.m)$ where n is the number of sensors and m is the numbers of obstacles. Therefore, the computing of the new fitness function value has the complexity of $O\,(n^2 + n.m)$.

4.1.4. Genetic Operators

a) Crossover Operator:

The proposed GA applies the crossover operator BLX_α, which has the property that the location of each offspring depends on the distance between its parents. The larger the

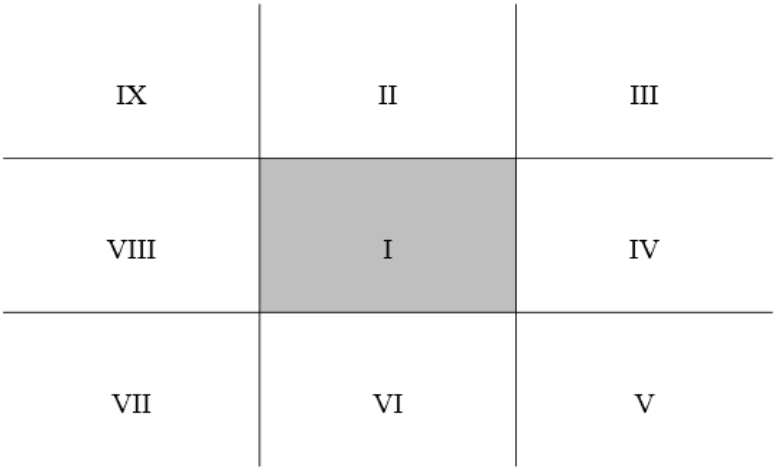


Figure 6: Nine parts divided by edges of obstacles.

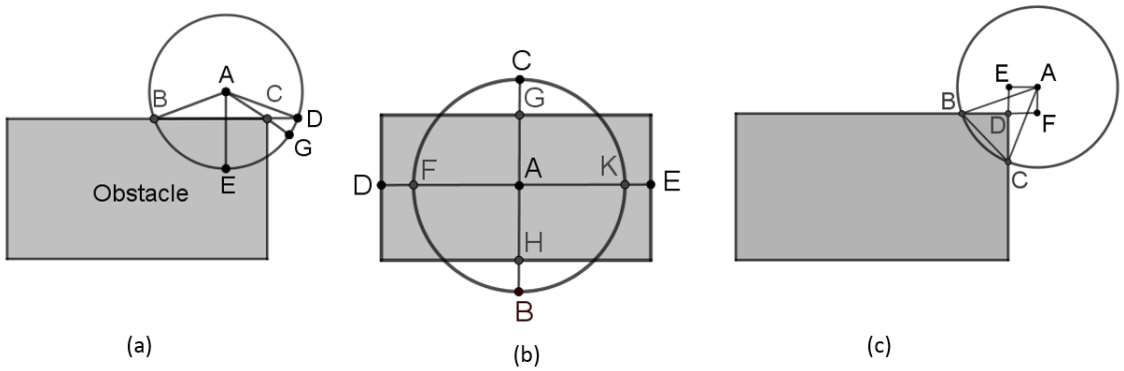


Figure 7: Overlapping areas between the coverage of a sensor and an obstacle.

distance between the parents, the larger the distance between the offspring and the parents. BLX_α allows for slightly more exploration thanks to a random factor in the reproduction process [**?**].

Consider two parents $S_1 = \left\{(x_i, y_i), i = \overline{1,n}\right\}$ and $S_2 = \left\{(x_i{}', y_i'), i = \overline{1,n}\right\}$. Offspring $Z = \left\{(u_i, v_i), i = \overline{1,n}\right\}$ is generated as follows.

$$\begin{cases} u_i \in \left[min\left(x_i, x_i^{'}\right) - \alpha I_x, max\left(x_i, x_i^{'}\right) + \alpha I_x\right] \\ v_i \in \left[min\left(y_i, y_i^{'}\right) - \alpha I_y, max\left(y_i, y_i^{'}\right) + \alpha I_y\right] \end{cases} \tag{13}$$

where $I_x = \left|x_i - x_i^{'}\right|$, $I_y = \left|y_i - y_i^{'}\right|$, and α is expansion coefficient of BLX_α. The value of α is 0.5 [**?**]. The execution of the crossover operator is illustrated in Fig. 8.

b) Mutation Operator:

Offspring $Z = \{(u_1, v_1), (u_2, v_2), ..., (u_n, v_n)\}$ produced by crossing over the parents $S_1 =$

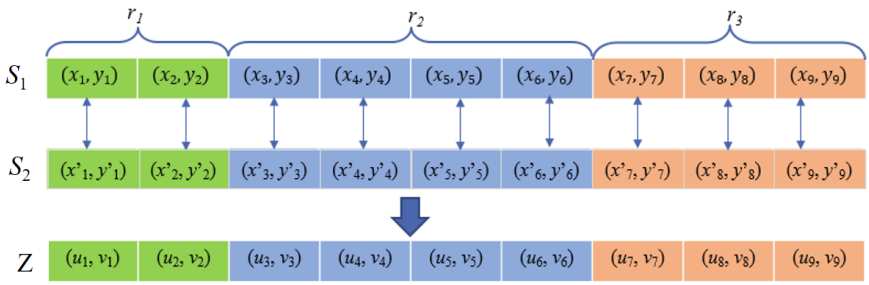


Figure 8: The process of generating offspring Z by performing crossover between S_1 and S_2.

$\{(x_i, y_i), i = \overline{1,n}\}$ and $S_2 = \{(x_i', y_i'), i = \overline{1,n}\}$ will be mutated by dynamic Gauss [**?**] as in equation (14).

$$u_i^{'} = u_i + \mathcal{N}\left(0, \left|x_i - x_i^{'}\right|^2\right) \tag{14}$$

$$v_i^{'} = v_i + \mathcal{N}\left(0, \left|y_i - y_i^{'}\right|^2\right)$$

where $\mathcal{N}\left(0, \left|x_i - x_i^{'}\right|^2\right)$ and $\mathcal{N}\left(0, \left|y_i - y_i^{'}\right|^2\right)$ are Gaussian distribution. Fig. 9 illustrates the implementation of the mutation operator.

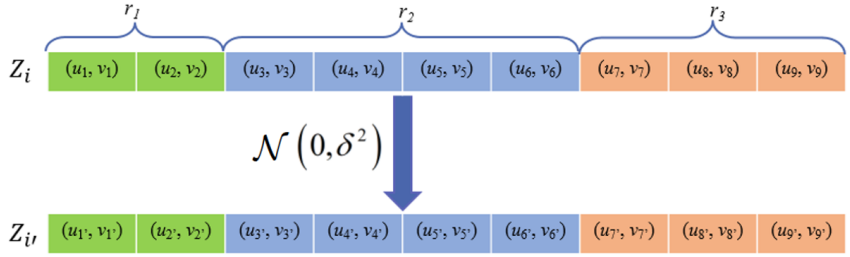


Figure 9: The implementation of mutation on an individual using the dynamic Gauss mutation.

Some of the individuals added to population after crossover and mutation are "bad" individuals, which produce solution deployments such that the sensors may overlap one another, the boundaries or obstacles. To overcome this difficulty, we applied the virtual force algorithm (VFA) which was utilized in [**?**], with some improvement to make it work well in the new model. In our modified version of VFA named Modified Virtual Force algorithm ($MVFA$) , there are three types of forces used on sensors:

- Mutual forces between sensors.
- Mutual forces between sensors and boundaries of A.
- Mutual forces between sensors and obstacles.

The first and second type of forces are introduced in [**?**]. This paper introduces the third type of forces, which is the interaction between sensors and obstacles. The pseudocode for the MVFA is given in Algorithm 2.

Algorithm 2: Modified Virtual Force Algorithm

while $s_i = (x_i, y_i) \in S$ **do**
  Resistive force $\vec{F_r} = \vec{0}$, Attractive force $\vec{F_a} = \vec{0}$
  Number of resistance $n_r = 0$, number of attraction $n_a = 0$
  while $s_j \in S$ & $s_j \neq s_i$ **do**
    if $d(s_i, s_j) < r_{s_i} + r_{s_j}$ **then**
      $\vec{F_r} = \vec{F_r} + (1 - \frac{r_{s_i}+r_{s_j}}{d(s_i,s_j)}).(s_j - s_i)$
      $n_r = n_r + 1$
    end
    if $d(s_i, s_j) > r_{s_i} + r_{s_j}$ **then**
      $\vec{F_a} = \vec{F_a} + (1 - \frac{r_{s_i}+r_{s_j}}{d(s_i,s_j)}).(s_j - s_i)$
      $n_a = n_a + 1$
    end
  end
  while $b \in \{(x_i, 0), (x_i, H), (0, y_i), (W, y_i)\}$ **do**
    if $d(s_i, b) < r_{s_i}$ **then**
      $\vec{F_r} = \vec{F_r} + (1 - \frac{r_{s_i}}{d(s_i,b)}).(b - s_i)$
      $n_r = n_r + 1$
    end
    if $d(s_i, b) > r_{s_i}$ **then**
      $\vec{F_a} = \vec{F_a} + (1 - \frac{r_{s_i}}{d(s_i,b)}).(b - s_i)$
      $n_a = n_a + 1$
    end
  end
  while $o_j \in O$ **do**
    if $d(s_i, o_j) < r_{s_i}$ **then**
      $\vec{F_r} = \vec{F_r} + (1 - \frac{r_{s_i}}{d(s_i,o_j)}).(o_j - s_i)$
      $n_r = n_r + 1$
    end
    if $d(s_i, o_j) > r_{s_i}$ **then**
      $\vec{F_a} = \vec{F_a} + (1 - \frac{r_{s_i}}{d(s_i,o_j)}).(o_j - s_i)$
      $n_a = n_a + 1$
    end
  end
  $s_i = s_i + \alpha_r.\frac{\vec{F_r}}{n_r} + \alpha_a.\frac{\vec{F_a}}{n_a}$
end
Adjust coordinate of sensors

4.2. Particle Swarm Optimization

The traditional Particle Swarm Optimization (PSO) is inspired by the habits of a flock of animals. The main difference between the GA and the PSO is that individuals in a PSO population interact with one another [**?**]. In the PSO, the experience and acquired knowledge of individuals can be used for surviving and finding food effectively. The PSO

algorithm includes the following steps: individual representation, population initialization, fitness calculation and individual update. Similar to the fitness function in GA, the fitness function in PSO is the *Olap* value calculated by formula (5). In the following subsections, we will introduce the tacks in details.

4.2.1. Individual Representation

In the problem of Area Coverage Optimization in WSNs which has n sensors of k different types, a population S has N individuals $S = \{s_1, s_2, \ldots, s_N\}$, each of which is specialized by its velocity and position on surveillance region A. The position of an individual, which is a possible solution to this problem, is presented by a n-dimensional vector U. PSO uses the same individual representation as that of GA in which U is constituted by n elements $\{u_1, \ldots, u_n\}$ and the i^{th} element $u_i = (x_i, y_i)$ is the coordinates of the sensor i on surveillance region A. Meanwhile, the velocity of individual s_i, which is a n-dimensional vector $V = [(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)]$, specifies the displacement of an individual after a generation. The best position of the i^{th} individual will be assigned to $pbest_i$.

4.2.2. Population Initialization

Each individual is initialized randomly and $\overrightarrow{0}$ is assigned to its velocity V. Despite bad quality of randomly initialized individuals, the random initialization guarantees the diversity of individuals. Consequently, it will increase the possibility of finding optimal results. Furthermore, the best position among those of the individuals in a population will be stored in a variable, namely *gbest*, which is the output of PSO.

$$gbest = \arg\min_{s_i \in S} fitness\,(S_i) \tag{15}$$

Even though PSO is widely practiced, it still has a disadvantage which is that it suffers from a premature convergence [**?**] due to the tendency of the whole population going in the direction of the best individual. In order to solve this problem, we propose an improved population initialization based on the clustering of individuals. The PSO makes use of not only the experience of the current best individual of each generation but also the experience of every other individual. In other words, PSO has a better searching ability. The modifications are as follows.

A population is divided into k sub-populations, denoted by C_i with $i = 1, ..., k$. Each sub-population C_i has a head individual C_{best_i}. The best individual of the whole population is stored in G_{best}.

Step 1: Initialize k sub-populations randomly.

Step 2: Find the best individual C_{best} of every sub-population. We propose three searching strategies:

- *Strategy 1 (S1-PSO):* Choose a fixed C_{best} for each sub-population. Inspired by an observation of the natural world that if a place has a plenty of food for a species, the whole area around it is also likely to have a good food resource, we choose a fixed C_{best} based on the value of the fitness function right after the sub-population initialization process.

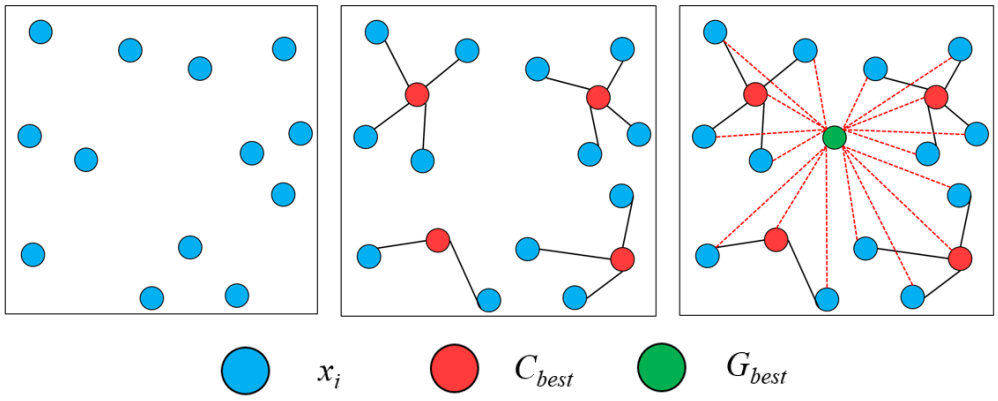


Figure 10: Population Initialization in PSO.

- *Strategy 2 (S2-PSO):* Choose a C_{best} for each sub-population based on the fitness function. Every time the individuals are updated, the individual with the best fitness value is chosen as the head individual of the sub-population, as in formula (16).

$$C_{best_i} = \left\{ x \mid fitness\,(x) = \min_{y \in sub_i} fitness\,(y) \right\} \tag{16}$$

  where sub_i is the i^{th} sub-population with i from 1 to k.

- *Strategy 3 (S3-PSO):* Choose a random C_{best} for each sub-population. To avoid falling into local optimum, this strategy chooses the head individual in sub-population randomly. The process of updating C_{best} in every generation is based on the probability below:
  Assume that the i^{th} sub-population has m individuals y_j with $j = 1, .., m$. Each individual y_j has a probability $P\,(y_j)$ of being chosen as the C_{best} as follows.

$$P\,(y_j) = \frac{\frac{1}{fitness(y_j)}}{\sum_{t=1}^{m} \frac{1}{fitness(y_t)}}, j = \overline{1, m} \tag{17}$$

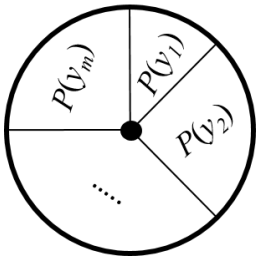


Figure 11: Probabilities of individuals to be chosen as C_{best}.

Step 3: Find the best individual in terms of fitness of the population and assign it to G_{best}.

4.2.3. Individual updating

In the individual updating process of the PSO introduced in [**?**], the velocity and position of individual s_i at the t^{th} generation are updated by formula (18) and formula (19).

$$V_i^{t+1} = \omega V_i^t + c_1 r_1 \left(pbest_i - U_i^t\right) + c_2 r_2 \left(gbest - U_i^t\right) \tag{18}$$

$$U_i^{t+1} = U_i^t + V_i^{t+1} \tag{19}$$

where ω is inertia weight, V_i^t is current velocity, U_i^t is current position, and c_1, c_2, r_1 and r_2 are parameters that characterize the individual's experience and the population's experience.

However, our proposed PSO cannot use this formula, as the velocity of each individual was not only influenced by the experience of the individual itself and of the best individual, but also by the experience of the head individual of its sub-population, as a result of the proposed population initialization. Therefore, in this paper, we propose modified calculations of the updated velocity and position for the individual updating process, which are described as follows:

During the t^{th} generation, individual s_i of each sub-population j at the position U_{ij}^t will move at its updated velocity V_{ij}^{t+1} to the new position U_{ij}^{t+1}. The velocity and the new position is updated by formula (20) and formula (21).

$$V_{ij}^{t+1} = \omega V_{ij}^t + c_1 r_1 \left(P_{best_{ij}} - U_{ij}^t\right) + c_2 r_2 \left(G_{best} - U_{ij}^t\right) + c_3 r_3 \left(C_{best_j} - U_{ij}^t\right) \tag{20}$$

$$U_{ij}^{t+1} = U_{ij}^t + V_{ij}^{t+1} \tag{21}$$

where ω is inertia weight, V_{ij}^t is current velocity, U_{ij}^t is current position, and c_1, c_2, c_3, r_1, r_2, and r_3 are parameters that characterize the individual's experience and the population's experience.

The inertia weight ω represents the movement of an individual. The greater the inertia weight ω, the faster the individual moves and vice versa. It can be seen that if the individual moves too fast, there will be a state of chaos and it will be difficult for the algorithm to converge. If the individual moves too slowly, the algorithm will take very long to finish. Therefore, our proposed PSO involves a uniform deceleration, i.e. the inertia weight decreases over generations and is calculated by:

$$\omega = \omega_{\max} - \frac{t}{T}\left(\omega_{\max} - \omega_{\min}\right) \tag{22}$$

where t is the current number of generations and T is the maximum number of generations.

In short, the updated information of an individual includes its current velocity, position and experience. Fig. 12 illustrates the individual update in details.

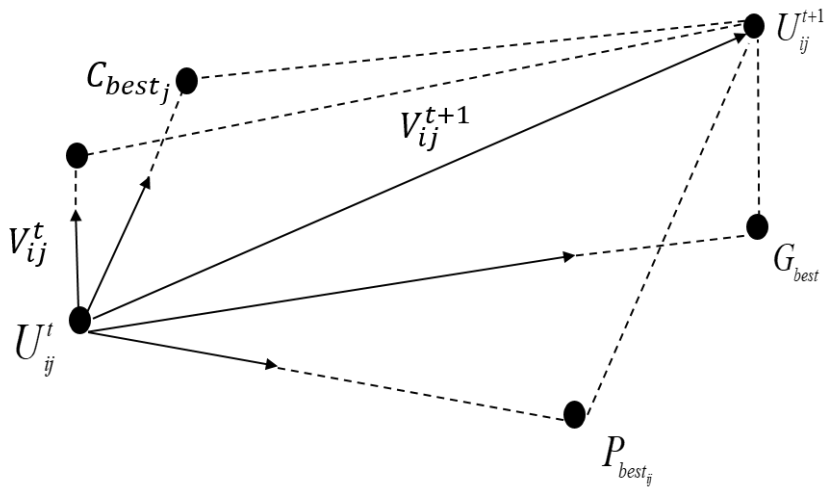


Figure 12: Update of an individual on its position and velocity based on personal experience and the experience of the best individual in PSO.

5. Experiments and Results

5.1. Experimental Scenarios

We devise five scenarios with 23 instances to evaluate the performances of the proposed algorithms when applied to networks with different properties. The scenarios are as follows:

Scenario 1: This scenario uses three datasets as given in Table 1. The impact of the obstacles' locations on the quality of the solutions is assessed. We examine three subscenarios of obstacles' locations:

- *Subscenario 1*: The obstacles concentrate near the four boundaries of surveillance region A (details of this subscenario is shown in Table 1 (s1-1)).
- *Subscenario 2*: The obstacles concentrate near the center of surveillance region A (details of this subscenario is shown in Table 1 (s1-2)).
- *Subscenario 3*: The obstacles are randomly distributed on surveillance region A (details of this subscenario is shown in Table 1 (s1-3)).

Scenario 2: This scenario uses five datasets as given in Table 2. The impact of the ratio of the total area of all obstacles to the area of surveillance region A on the quality of the solutions is assessed. For this purpose, we developed five different subscenarios by keeping the number of obstacles, radii of sensors and the area of surveillance region A but changing the ratio of the total area of all obstacles to the area of surveillance region A successively to: 5%, 10%, 15%, 20%, and 30% (details of these subscenarios are shown in the following Table 2).

Scenario 3: This scenario uses five datasets as given in Table 3. The impact of the number of obstacles on surveillance region A on the quality of the solutions is assessed.

Table 1: Scenario 1

Dataset	n	n_1	n_2	n_3	*No. Obstacles*	*Obstacle/A*	r_1	r_2	r_3	A
s1-1	104	34	35	35	5	20%	6	4.8	3.84	100 x 100
s1-2	104	34	35	35	5	20%	6	4.8	3.84	100 x 100
s1-3	104	34	35	35	5	20%	6	4.8	3.84	100 x 100

Table 2: Scenario 2

Dataset	n	n_1	n_2	n_3	*No. Obstacles*	*Obstacle/A*	r_1	r_2	r_3	A
s2-1	122	42	40	40	5	5%	6	4.8	3.84	100 x 100
s2-2	116	40	37	39	5	10%	6	4.8	3.84	100 x 100
s2-3	111	36	37	38	5	15%	6	4.8	3.84	100 x 100
s2-4	104	34	35	35	5	20%	6	4.8	3.84	100 x 100
s2-5	94	28	30	36	5	30%	6	4.8	3.84	100 x 100

For this purpose, we have developed five different subscenarios by keeping the ratio of the total area of all obstacles to the area of surveillance region A, the number of sensors of each type, the sensor radius of each type, and the area of surveillance region A but changing the number of obstacles successively to: 5, 10, 15, 20, and 30 (details of these subscenarios are shown in the following Table 3).

Scenario 4: This scenario uses five datasets as given in Table 4. The impact of the ratio of the precalculated maximum possible coverage by the sensors to the area of surveillance region A on the quality of the solutions is evaluated. For this purpose, we have developed five subscenarios by keeping the ratio of the total area of all obstacles to the area of surveillance region A, the number of obstacles, the sensor radius of each type, and the area of domain A and changing the ratio of the sensors' total coverage to the area of surveillance region A successively to 70%, 75%, 80%, 85%, and 90% (details of these subscenarios are shown in the following Table 4).

Scenario 5: This scenario uses five datasets as given in Table 5. The impact of the sensing radius on the quality of the solutions is evaluated. For this purpose, we developed five subscenarios by keeping the ratio of the total area of all obstacles to the area of surveillance region A, the number of obstacles, and the area of surveillance region A but changing the sensors' radii of each type (details of these subscenarios are shown in the following Table 5).

Table 3: Scenario 3

Dataset	n	n_1	n_2	n_3	*No. Obstacles*	*Obstacle/A*	r_1	r_2	r_3	A
s3-1	94	28	30	36	5	30%	6	4.8	3.84	100 x 100
s3-2	94	28	30	36	10	30%	6	4.8	3.84	100 x 100
s3-3	94	28	30	36	15	30%	6	4.8	3.84	100 x 100
s3-4	94	28	30	36	20	30%	6	4.8	3.84	100 x 100
s3-5	94	28	30	36	30	30%	6	4.8	3.84	100 x 100

Table 4: Scenario 4

Dataset	n	n_1	n_2	n_3	*No. Obstacles*	*Obstacle/A*	r_1	r_2	r_3	A
s4-1	91	30	30	31	5	10%	6	4.8	3.84	100 x 100
s4-2	100	30	33	37	5	10%	6	4.8	3.84	100 x 100
s4-3	104	34	35	35	5	10%	6	4.8	3.84	100 x 100
s4-4	111	36	37	38	5	10%	6	4.8	3.84	100 x 100
s4-5	117	38	40	39	5	10%	6	4.8	3.84	100 x 100

Table 5: Scenario 5

Dataset	n	n_1	n_2	n_3	*No. Obstacles*	*Obstacle/A*	r_1	r_2	r_3	A
s5-1	23	6	7	10	5	10%	14	11.2	8.96	100 x 100
s5-2	32	7	11	14	5	10%	12	9.6	7.68	100 x 100
s5-3	46	11	14	21	5	10%	10	8	6.4	100 x 100
s5-4	73	16	23	34	5	10%	8	6.4	5.12	100 x 100
s5-5	130	28	41	61	5	10%	6	4.8	3.84	100 x 100

5.2. Parameter Settings

To examine the performance of the proposed GA and PSO, we implement the PSO in [**?**] (denoted by PSO*) and compare the results. The experimental parameters are provided in Table 6,7, 8.

All the systems were implemented in Java 8 and run on a machine with Intel Core i7-3.60GHz, RAM 16GB computer using Windows 8 64-bit.

Table 6: Experimental parameters of GA

Parameter	**Value**
Number of runs	30
Population size (N)	50
Maximum number of generations	2000
Maximum number of loops without producing better results	400
Crossover rate	80%
Mutation rate	$1/n$
Number of points randomly generated by Monte Carlo method (L)	10000000

5.3. Computational Results and Discussions

We conducted five experiments to evaluate the results of the proposed algorithms. These are implemented on 23 datasets explain in Experimental Scenarios.

- *Experiment 1:* Find parameter values for each algorithm to achieve the best performance.

Table 7: Experimental parameters of PSO*

Parameter	**Value**
Number of runs	30
Population size (N)	50
Maximum number of generations	2000
Random variable r_1 and r_2	Uniform distribution [0,1]
Maximum number of loops without producing better results	400
Maximum weight ($\omega_{\max}$)	0.9
Minimum weight ($\omega_{\min}$)	0.1
Number of points randomly generated by Monte Carlo method (L)	10000000

Table 8: Experimental parameters of PSO

Parameter	**Value**
Number of runs	30
Population size (N)	50
Sub-population size	10
Number of sub-populations	5
Maximum number of generations	2000
Random variable r_1, r_2 and r_3	Uniform distribution [0,1]
Maximum number of loops without producing better results	400
Maximum weight ($\omega_{\max}$)	0.9
Minimum weight ($\omega_{\min}$)	0.1
Number of points randomly generated by Monte Carlo method (L)	10000000

- *Experiment 2:* Evaluate the effect of the percentage of heuristic initialization on performance of the proposed algorithms.
- *Experiment 3:* Evaluate the effectiveness of the proposed strategies for finding the head individual C_{best} in IPSO algorithm.
- *Experiment 4:* Evaluate the effectiveness of *MVFA* to achieve the best performance.
- *Experiment 5:* Evaluate the performances of the proposed algorithms when applied to networks with different properties. The scenarios explained in Experimental Scenarios are used for this experiment.

To evaluate the goodness of each algorithm in Experiment 1 and Experiment 2, we calculate the value of the function $f_j(i)$ in formula 23 for each dataset j and each $i \in P$ (P will be specifically defined later for each experiment).

$$f_j(i) = \frac{U_j - area_j}{\underset{i \in P}{max}\left(U_j - area_j(i)\right)} \tag{23}$$

where U_j is the maximum rate of coverage area on surveillance region A of the sensors in the j^{th} dataset and $area_i(j)$ is the coverage rate obtained by the algorithm for each value of i on dataset j.

From formula (23), it can be noticed that the smaller the value of $f_j(i)$, the greater the coverage, which results in higher solution quality. Each experiment is implemented and summarized as follows.

5.3.1. Experiment 1:

Experiment 1 finds the values of the parameters c_1, c_2 in the PSO* and c_3 in the PSO for best performance of each algorithm.

In the PSO, the c_1 characterizes the effect of individual experience, while c_2 characterizes the effect of the overall population's experience on each individual. In PSO*, we assume that $c_1 = c_2$. The values of c_1 and c_2 range from 0.1 to 1.0.

To find the appropriate values of the parameters, we try different values of c_1 and c_2 in the range between 0.1 and 1.0. Note that as we assume $c_1 = c_2$, the corresponding value of i in formula (23) is equal to c_1 and c_2 and set P is defined as $P = \{0.1, 0.2, .., 1.0\}$. The value of $f_i(j)$ is then calculated according to formula (23) for each dataset j.

Experimental results for parameter selection of PSO* are illustrated in Fig.13. It can be noted from these results that the PSO*1 algorithm corresponding to $i = 0.1$ returns the best results (largest coverage area in surveillance region A). Therefore, we concluded that the most suitable values for parameters in PSO* are $c_1 = c_2 = 0.1$.

The PSO, which is an improved version of the PSO*, reuses the parameters c_1 and c_2 of the PSO*. However, the difference between the two algorithms is that the PSO has parameter c_3 which characterizes the degree of influence the head individual exerts on each of the other individuals in the same cluster. The PSO includes three strategies to build the cluster, namely S1-PSO, S2-PSO and S3-PSO, explained in the Proposed Algorithms

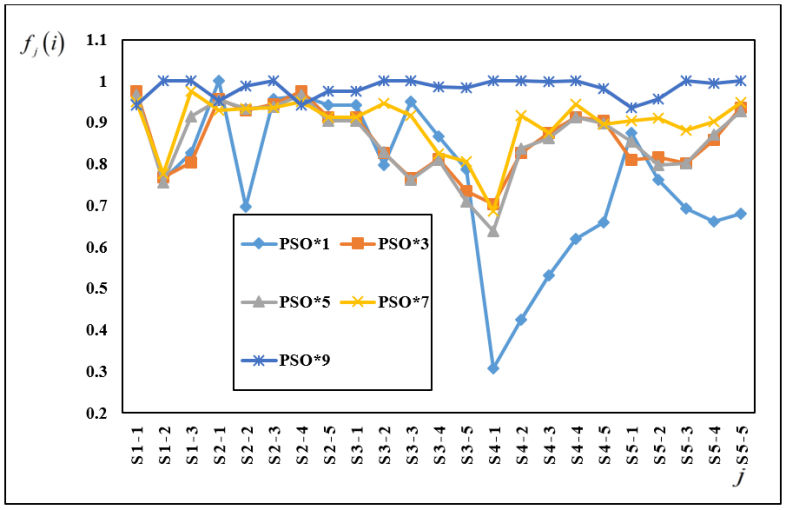


Figure 13: Select parameters for PSO*

section. Depending on the characteristics of each proposed strategy, we try different values of c_3 ranging from 0.1 to 1.0 while keeping the values of c_1 and c_2 equal to 0.1, which is the most appropriate value found for the parameters of PSO*. We calculate the value of $f_i(j)$ by formula 23 for each value of c_3.

Note: Sm-PSOn is the version of PSO using Strategy m for finding the head individual of the m^{th} sub-population with $n = \{1, 2, .., 10\}$ (corresponding to $c_3 = \{0.1, 0.2, .., 1.0\}$) and $P = \{1, 2, 3\}$. Experimental results show that:

- *Strategy 1 (S1-PSO):* The S1-PSO returns the best coverage when $c_3 = 0.1$ as illustrated in Fig.14.
- *Strategy 2 (S2-PSO):* The S2-PSO returns the best coverage when $c_3 = 0.9$ as illustrated in Fig.15.
- *Strategy 3 (S3-PSO):* The S3-PSO returns the best coverage when $c_3 = 0.1$ as illustrated in Fig.16.

It can be noted that while the head individual of each sub-population is fixed over generations in Strategy 1, in Strategy 3 each head individual is randomly selected from the sub-population, which leading to the existence of some bad head individuals. Therefore, the impact coefficient of the head individual is small ($c_3 = 0.1$) as demonstrated by the results of Strategy 1 and Strategy 3 in Experiment 1. In contrast, the individual chosen as the head individual by Strategy 2 is always the best individual in the sub-population. Therefore, the impact coefficient of the head individual in the sub-population is great ($c_3 = 0.9$) as demonstrated by the results of Strategy 2 in Experiment 1.

5.3.2. Experiment 2:

This experiment evaluates the effect of heuristic initialization on the performance of each algorithm. To find the best heuristic initialization rate, we try different heuristic initializa-

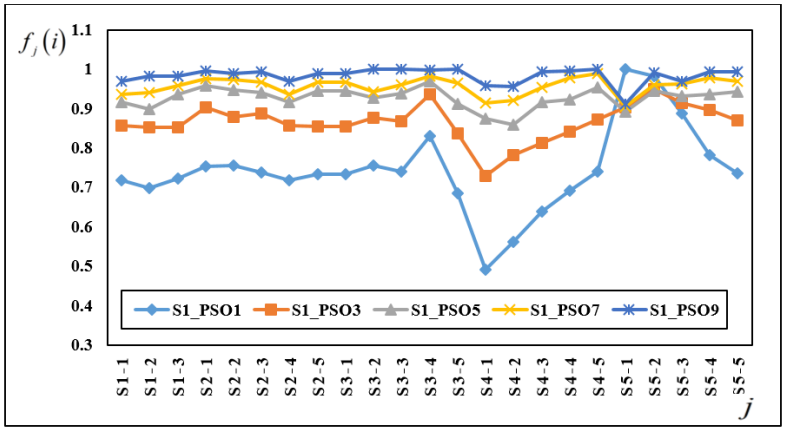


Figure 14: Select parameters for PSO with Strategy 1.

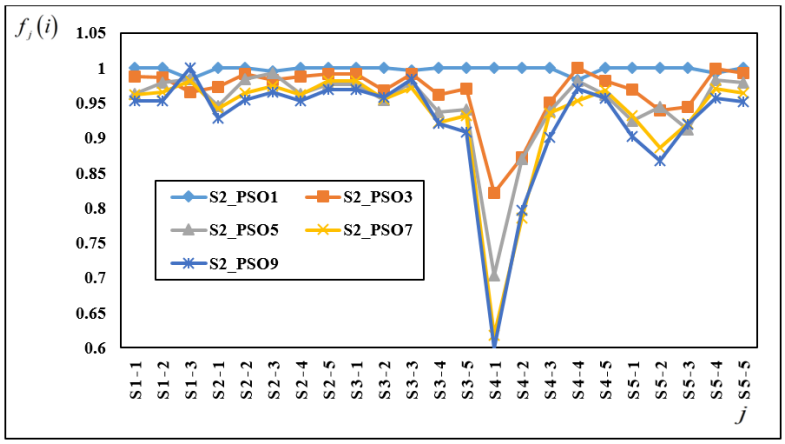


Figure 15: Select parameters for PSO with Strategy 2.

tion rates of 0%, 50% and 100% and calculate the value of $f_i(j)$ with $P = \{0\%, 50\%, 100\%\}$.

PSO*: In this experiment, we evaluate the goodness in terms of coverage of PSO* with heuristic initialization rates (HR) of 0%, 50% and 100%. The experimented variants of PSO* are named PSO*-HR0 (HR: 0%), PSO*-HR50 (HR: 50%) and PSO*-HR100 (HR: 100%). Fig.17 shows that HR of 50% returns the best solution quality on most datasets.

In the case of non-heuristic initialization (HR: 0%), the initial individuals are randomly generated. Therefore, the random head individual is not necessarily good in the sub-population, which causes the other individuals follow a wrong direction. However, the advantage of non-heuristic initialization is that the population is diverse as illustrated in Fig.17.

In the case of 100% heuristic initialization (HR: 100%), all initial individuals are initialized by heuristic, so they lie close to one another. If these initialized positions fall in the domain that has a local optimum, all individuals will move towards that local optimum. Consequently, the PSO* with 100% heuristic initialization converges very fast as illustrated in Fig.17.

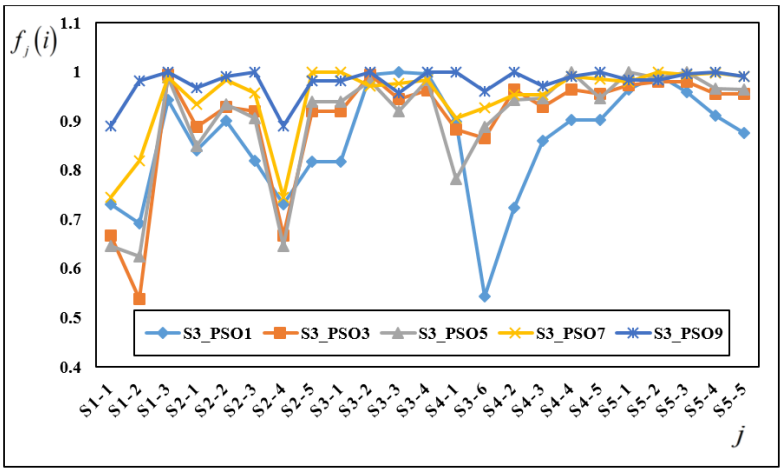


Figure 16: Select parameters for PSO with Strategy 3.

In the case of 50% heuristic initialization (HR: 50%), the population is both diverse as a result of partial random initialization and guaranteed to include good individuals. Therefore, we choose to heuristic initialization rate of 50% as the most appropriate rate. The results are illustrated in Fig.17.

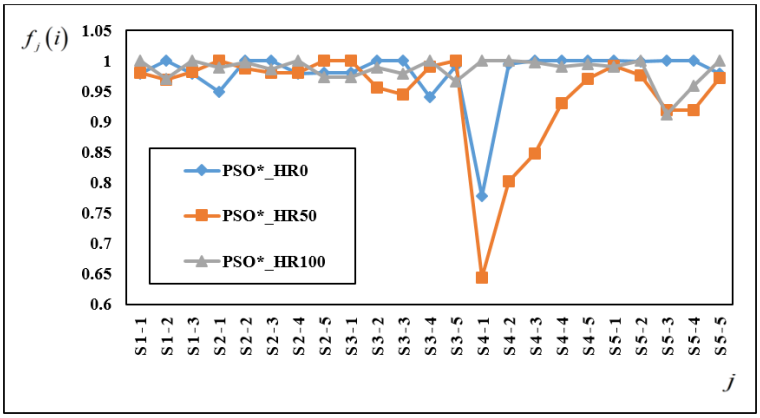


Figure 17: Select the heuristic initialization rates for the PSO*.

GA: In this experiment, GA applies heuristic initialization rates (HR) of 0%, 50% and 100% as in PSO*. The experimented variants of GA are named GA-HR0 (HR: 0%), GA-HR50 (HR: 50%) and GA-HR100 (HR: 100%). HR of 100% returns the best solution quality on most datasets as illustrated in Fig.18.

In the case of non-heuristic initialization (HR: 0%), all of the initial individuals are generated such that their sensors are randomly deployed. As a result, the sensors are likely to overlap, leading to reduction of fitness of the individuals but at the same time bringing more genetic diversity to the population.

In the case of 50% heuristic initialization (HR: 50%), the population is not only genetically diverse, but it also contains good genetic materials thanks to heuristic initialization.

Therefore, the solution quality obtained by 50% heuristic initialization is better than the quality obtained by non-heuristic initialization.

In the case of 100% heuristic initialization(HR: 100%), all individuals in the initial population are good individuals, i.e. populations contains only good genetic materials, but this method of initialization does not create genetic diversity. However, GA applies the mutation operator to overcome this disadvantage. Fig. 18 shows that GA with 100% heuristic initialization returns the best solutions among the three proposed variants of GA. Therefore, we choose heuristic initialization rate of 100% as the most appropriate rate.

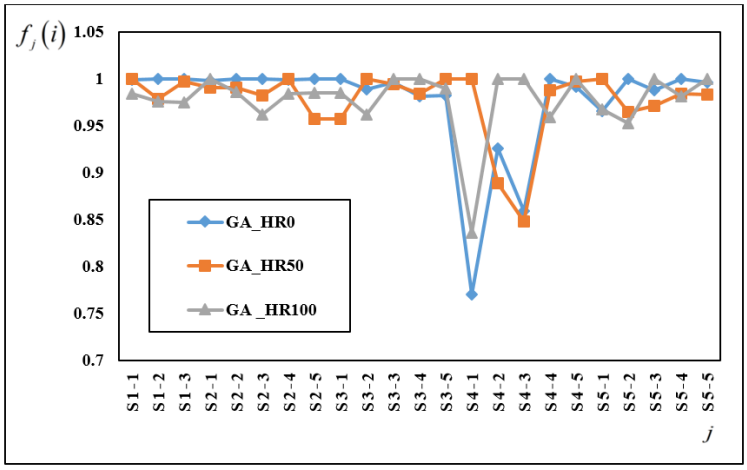


Figure 18: Select the heuristic initialization rates for the GA.

PSO: In this experiment, PSO applies three strategies introduced in the Proposed Algorithms section with heuristic initialization rates (HR) of 0%, 50% and 100%.

Note: Sm_PSO_HRn represents the variant of PSO using Strategy m with heuristic initial rate n ($m \in \{1, 2, 3\}$ and $n \in \{0\%, 50\%, 100\%\}$).

Fig.19, Fig.20 and Fig.21 indicate that using 50% heuristic initialization (HR: 50%) returns the best solution quality on most datasets for all of the three strategies. This conclusion is explicable, as PSO*, which is the primitive version of PSO, also returns the best results when the HR is 50%.

5.3.3. Experiment 3:

This experiment selects the best strategy for finding the head individual in a subpopulation from the three strategies of PSO.

The optimal parameters found by Experiment 1 and the optimal HR of 50% found by Experiment 2 are used in this experiment. The experimented variants of PSO are named S1_PSO_H50 (Strategy 1 and HR: 50%), S2_PSO_H50 (Strategy 2 and HR: 50%) and S3_PSO_H50 (Strategy 3 and HR: 50%). It can be concluded from Fig. 22 that S2_PSO_H50 obtains the best results. Therefore, we choose S2_PSO_H50 to be the best version for the proposed PSO. Hereinafter, PSO refers to the variant of PSO that uses Strategy 2 and HR of 50%.

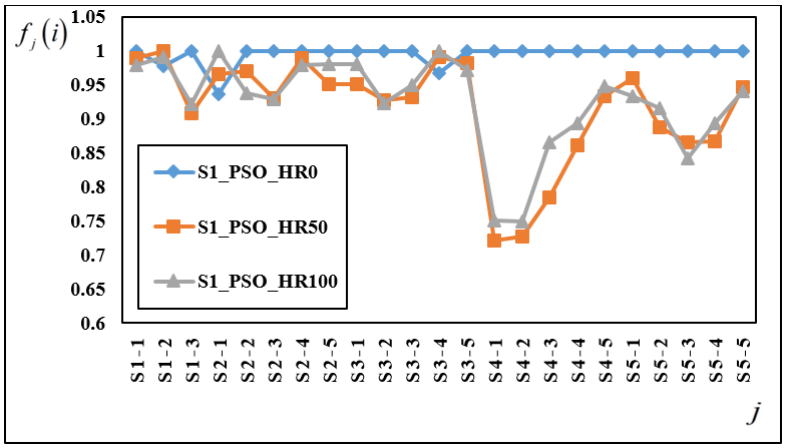


Figure 19: Select the heuristic initialization rates for the S1-PSO.

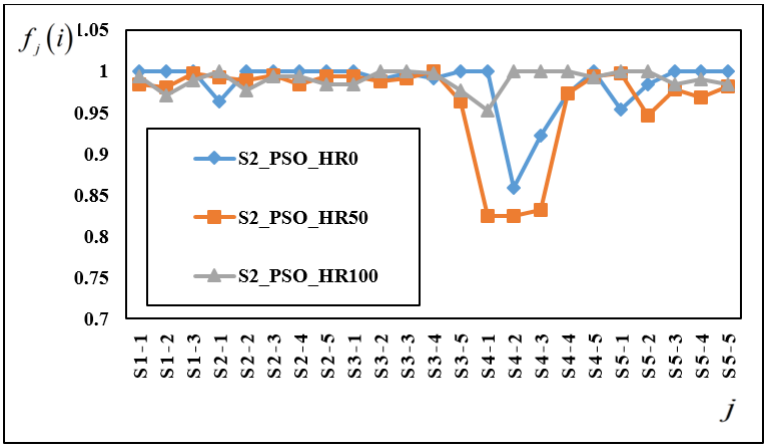


Figure 20: Select the heuristic initialization rates for the S2-PSO.

5.3.4. Experiment 4:

This experiment evaluates the effect of MVFA on each proposed algorithm. To examine the effect of MVFA, we experimented two versions of each algorithm: with MVFA (GA-MVFA, PSO*-MVFA and PSO-MVFA) and without MVFA (GA-non-MVFA, PSO*-non-MVFA and PSO-non-MVFA). It can be inferred from Fig.23 that the versions with MVFA of the proposed algorithms returns the better solution quality than those without MVFA on most datasets. Therefore, MVFA is an essential contributing factor to the improvement of solution quality of the proposed algorithms.

5.3.5. Experiment 5:

This experiment evaluates and compares the performances of the proposed algorithms (PSO, PSO* and GA) in five network scenarios set up in section Experimental Scenarios.

Scenario 1: This scenario, which consists of three subscenarios, evaluates the impact of obstacles' locations on solution quality. Details of the datasets used in this scenario are

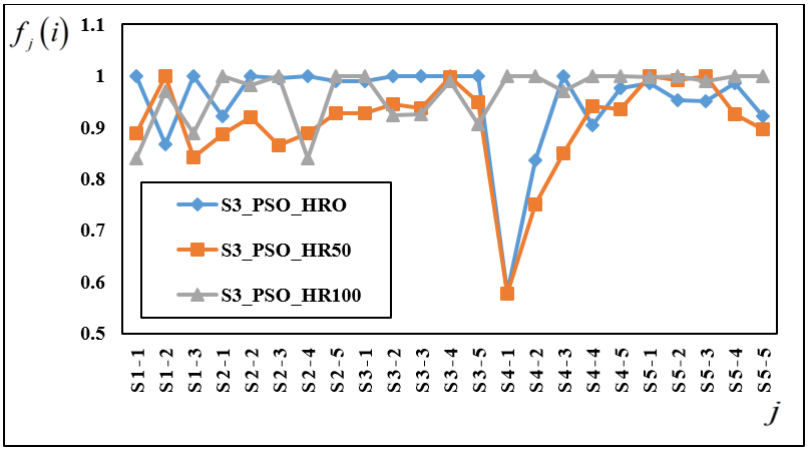


Figure 21: Select the heuristic initialization rates for the S3-PSO.

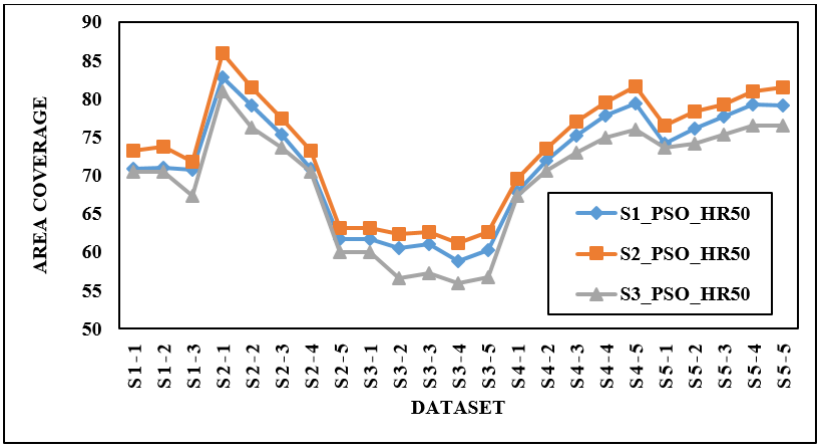


Figure 22: Select strategy with heuristic initialization rate of 50% for PSO.

provided in Table.1. From the results illustrated in Fig.24, it can be seen that PSO is better than GA for two out of three subscenarios (corresponding to datasets s1-1 and s1-2) and GA is better than PSO for one sub-scenario (corresponding to dataset s1-3). The reason is that for Subscenario 1 (dataset s1-1) and Subscenario 2 (dataset s1-2) the obstacles either lie close to the boundaries or are concentrated near the center of the surveillance region A, so the areas to deploy sensors are less divided by the obstacles. As a result, good solutions are more easily to be found by PSO. More specifically, the average area coverages obtained by PSO in Subscenario 1 (dataset s1-1) and in Subscenario 2 (dataset s1-2) are 73.2% and 73.8%, respectively.

In Subscenario 3 (dataset s1-3), the obstacles are scattered throughout the surveillance region A. Because individuals in both PSO* and PSO are not able to avoid obstacles, PSO* and PSO turn out less effective in this case, with the average coverage of only 71.8%. If there are obstacles lying on the path that the leading individuals move, they are more likely to get blocked by the obstacles. Therefore, it is harder in this case to achieve a good solution.

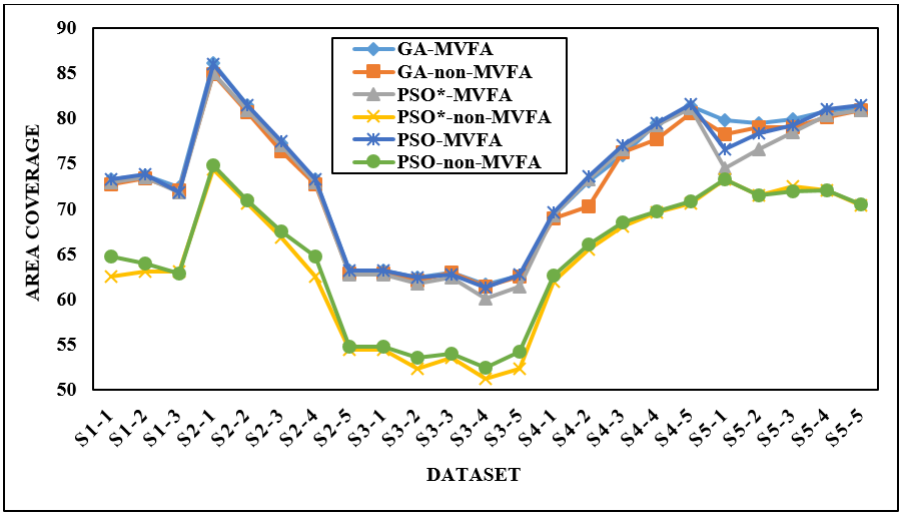


Figure 23: Select VFA for PSO*, PSO and GA.

However, as GA involves mutation, its population becomes more diverse and thus includes individuals that are able to get rid of obstacles. The average area coverage of GA is 72.3% in Subscenario 3 (dataset s1-3). From the above analysis, it can be concluded the proposed algorithms achieve the best results when obstacles are concentrated in the center or lie close to the boundaries.

It is evident that the positions of the obstacles greatly influence the performances of the proposed algorithms. To achieve objectivity in our experiments, from Scenario 2 to Scenario 5, we will construct subscenarios in which the obstacles are randomly distributed in surveillance region A.

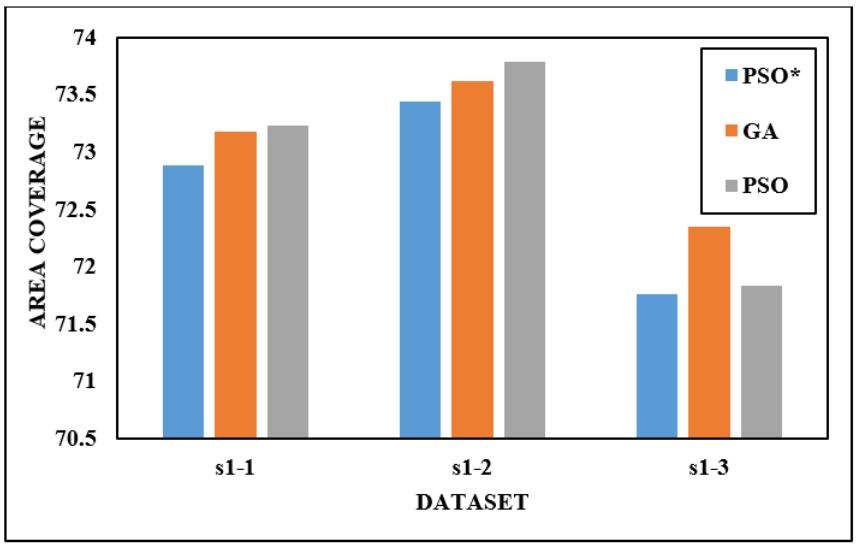


Figure 24: Percentage area coverage of Scenario 1.

Scenario 2: This scenario, which includes five subscenarios, evaluates the impact of the ratio of the total area of all obstacles to the area of surveillance region A on solution quality. Details of the datasets used in this scenario are provided in Table.2. From the results illustrated in Fig.25, it can be seen that PSO and GA perform equally well and slightly better than PSO* in all of the five subscenarios. This can be explained with the fact that the search for competent individuals become difficult when sensors are densely deployed in surveillance region A. As a result, the performances of PSO*, PSO and GA do not vary much.

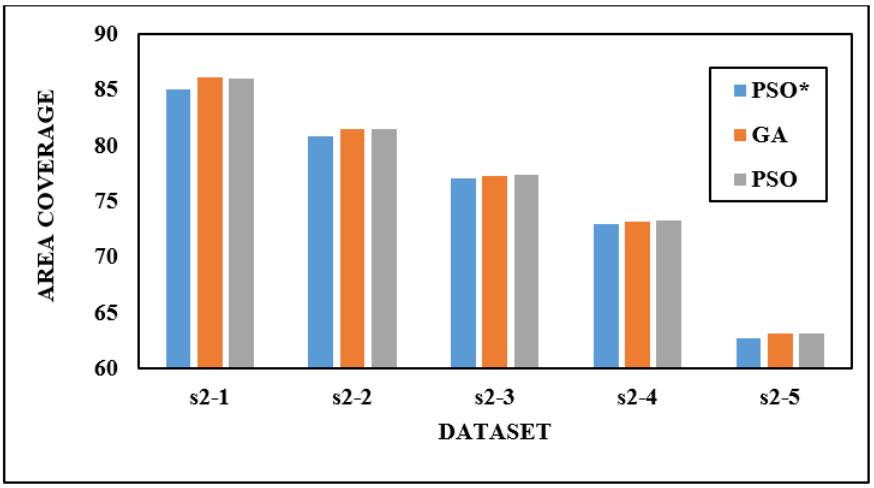


Figure 25: Percentage area coverage of Scenario 2.

Scenario 3: This scenario, which consists of five subscenarios, evaluates the impact of the number of obstacles on solution quality. Details of the datasets used in this scenario are provided in Table.3. From the results illustrated in Fig.26, it can seen that GA performs better than PSO* and PSO in two out of five subscenarios (datasets s3-3 and s3-4) and PSO performs better than PSO* and GA in three out of five subscenarios (datasets s3-1, s3-2 and s3-5). This can be explained by the fact that the increase in the number of obstacles leads to the increase in constraints-violating areas in the surveillance region.

PSO is the most effective among the three algorithms when applied to datasets with a small number of obstacles (5 and 10). The reason for this is that as the head individual in each sub-population in PSO acts as a local optimum, it can lead the other individuals in the sub-population out of the constraints-violating areas, improving the solution quality.

However, when applied to datasets with greater numbers of obstacles (15 and 20), GA turns out to be more effective than PSO and PSO*. Because GA involves mutation, it is possible to escape the areas with obstacles, which results in increased solution quality. Nonetheless, when the number of obstacles increases even more (30), PSO becomes more efficient than GA. This is because with the same ratio between the total areas of obstacles and the area of the surveillance region, the great number of obstacles means each obstacle is very small. Therefore, PSO can easily avoid constraints-violating zones.

Scenario 4: This scenario, which consists of five subscenarios, evaluates the impact of

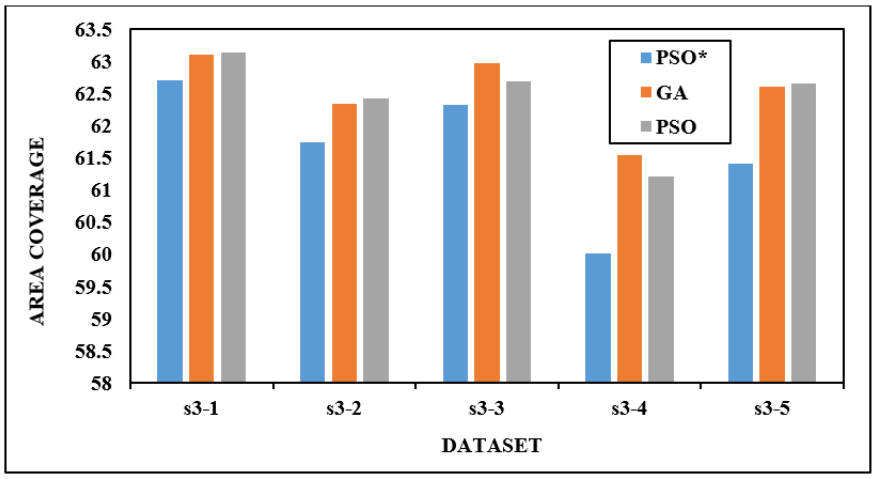


Figure 26: Percentage area coverage of Scenario 3.

the sensor density (the ratio between the precalculated maximum coverage by the sensors and the area of surveillance region A) on solution quality. Details of the datasets used in this scenario are provided in Table.4. From the results illustrated in Fig.27, it can be inferred that PSO is better than PSO* and GA in all subscenarios. PSO* performs better than GA in two sub-scenarios (datasets s4-2 and s4-3).

When the ratio is 70% (corresponding to s4-1), all three proposed algorithms return results, near the precalculated upper bound. This can be explained by the fact that when the number of sensors is small and the sensor density is low, sensors hardly overlap one another ($Olap = 0$).

As the sensor density increases, it becomes more difficult to reach the upper bound. For example, PSO returns results that cover 69.5%, 77% and 81.6% of the surveillance region when the upper bound is 70% (s4-1), 80% (s4-3) and 90% (s4-5) respectively.

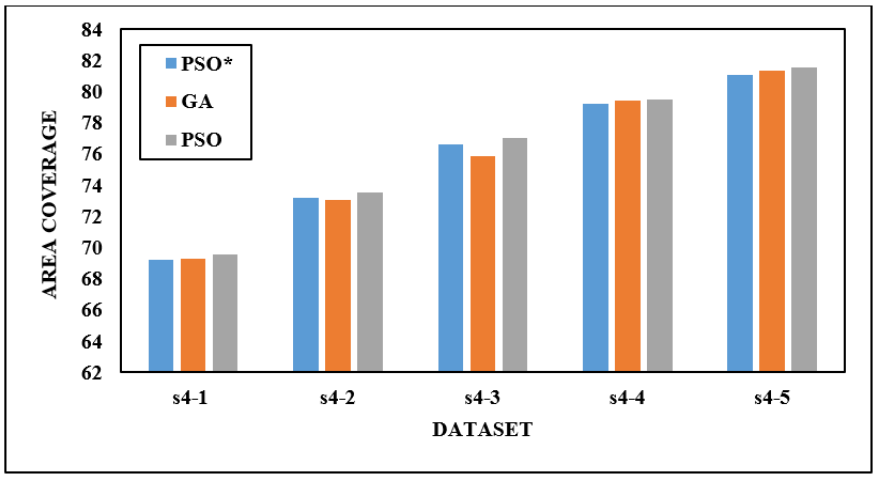


Figure 27: Percentage of covered area in Scenario 4.

Scenario 5: This scenario, which consists of five subscenarios, evaluates the impact of sensing radius on solution quality. Details of the datasets used in this scenario are provided in Table.5. From the results illustrated in Fig.28, it can be inferred that PSO outperforms the other algorithms in subscenarios with small sensing radii (datasets s5-4 and s5-5). This is because each time an individual moves to a new location, the displacement is small and thus less likely to lead to the individual's falling into constrainst-violating areas. Even when the individual falls into a constraint-violating area, they can easily get out of that area.

In contrast, in subscenarios with large sensor radii (datasets s5-1, s5-2 and s5-3), GA performs better than PSO and PSO* as a result of the greatness of the individuals' displacements in PSO and PSO*. Thus, the possibility of individuals' falling into constraint-violating area is high and they cannot easily get out of these areas.

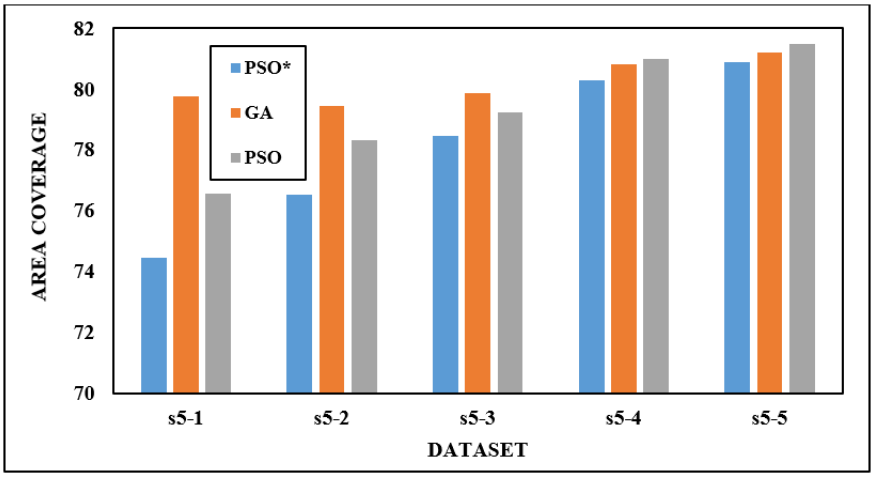


Figure 28: Percentage covered area in Scenario 5.

6. Conclusion

In this paper, we proposed a new model for maximizing area coverage in WSNs with heterogeneous sensing ranges and obstacles. We also proposed two metaheuristic algorithms for solving this problem: a genetic algorithm (GA) and a particle swarm optimization (PSO).

Our contributions include a new heuristic initialization, a new fitness function and an improved version of the virtual force algorithm. A uniform deceleration was added to the calculation of the inertia weight to improve the convergence rate in PSO. In addition, three strategies for finding head individuals in PSO were introduced. The proposed methods were comprehensively experimented on 23 datasets with various scenarios devised for different purposes. Through these experiments, we have drawn several meaningful conclusions related to parameter settings, effect of heuristic initialization, strategies to select head individuals, effect of MVFA and performances of the proposed algorithms (PSO and GA) when applied to networks with different properties. We also implemented the PSO* in [**?**] to compare the results with those of our proposed methods.

Experimental results proved that properties such as obstacles' locations, total areas taken up by obstacles, number of obstacles, maximum coverage of sensors and sensors' radii greatly influenced the performances of the proposed algorithms. In most experimental scenarios, PSO outperformed PSO* and GA, especially when dealing with datasets whose numbers of sensors are great. Among other factors, the addition of sub-population head individuals improved solution quality by making use of experience of every individual instead of taking only the best individual into account, thus avoiding premature convergence. On the other hand, GA turned out to perform the most effectively of all algorithms on datasets with great numbers of obstacles, as its involvement with mutation help to escape the areas with obstacles.

These conclusions are meaningful for future work on the problem and can pave the way for advancements in the field of deployment in WSNs and the field of evolutionary computation. In the future, we are aiming to study the maximization of obstacles constrained area coverage problems related to connectivity and lifetime of WSNs.

7. Acknowledgment

This research is supported by Vietnam National Foundation for Science and Technology Development (NAFOSTED) Grant Number DFG 102.01-2016.03

References