Received February 12, 2018, accepted March 13, 2018, date of publication April 2, 2018, date of current version May 2, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2822263

Distributed Deployment Algorithm for Barrier Coverage in Mobile Sensor Networks

TRI GIA NGUYEN[1,2], (Member, IEEE), AND CHAKCHAI SO-IN[2], (Senior Member, IEEE)
[1]Faculty of Information Technology, Duy Tan University, Da Nang 550000, Vietnam
[2]Applied Network Technology Laboratory, Department of Computer Science, Faculty of Science, Khon Kaen University, Khon Kaen 40002, Thailand

Corresponding author: Chakchai So-In (chakso@kku.ac.th)

This work was supported by the Research Affairs and Graduate School, Khon Kaen University, Thailand, through the Post-Doctoral Training Program under Grant 59257.

ABSTRACT The deployment of sensor nodes (SNs) to form a network with coverage ability is one of the most important challenges of wireless sensor networks. In this paper, we study an efficient distributed deployment algorithm for barrier coverage improvement with mobile sensors, in which the SNs can be relocated after the initial deployment. To achieve the maximum number of barriers, we propose a distributed algorithm to construct k-barrier coverage by relocation of the SNs. Different from existing approaches, we propose a novel clustering technique based on the network area to reduce the information exchange messages. Then, based on the SNs clusters, we propose a heuristic method to assign the SNs evenly into each cluster with regard to the required number of SNs of each cluster and decide the moving SNs by computing the optimal relocation, considering moving distance minimization. The main goal of this approach is to relocate the SNs to form the maximum number of barriers with a minimum relocation cost, in terms of sensor energy consumption of communication and movement. The simulation results demonstrate the effectiveness of our algorithm when compared with other competing approaches.

INDEX TERMS Barrier coverage, distributed algorithm, mobile sensor networks, sensor deployment, self-deployment, wireless sensor networks.

I. INTRODUCTION

In recent years, coverage has been become one of the most important issues in wireless sensor networks (WSNs) [1], [2]. The coverage problem is usually interpreted as how well a WSN will monitor a field of interest [3]–[11]. There are many coverage applications therein, and barrier coverage [12]–[17] is one of the active research fields on the coverage problem. In this particular problem, a barrier coverage is similar to an electric fence, which detects a mobile object that infiltrates the sensor field.

The concept of barrier coverage was first introduced in the context of robotic systems [18]. The main objective in barrier coverage applications is to detect intruders as they pass over a monitored area. Border surveillance and intrusion detection are the two major applications of the barrier coverage problem [12]–[17]. The requirement of a barrier coverage application is generally to have at least one barrier (1-barrier covered) in the field of interest. Maximizing the number of barriers of the sensor nodes (SNs) deployed within the monitored area is one of the key factors in the design of barrier coverage applications for WSNs. However, the SNs are usually deployed in a large area, and each can only monitor within a limited sensing range. Therefore, to improve the barrier coverage in WSNs, the designing deployment algorithm with mobile sensors [19] is one of the main factors to be considered.

According to the accessibility of the monitored area, the sensor deployment can be classified as either random or deterministic [20], [22]–[25]. For deterministic sensor deployment, the monitored area can be controlled in a human-friendly environment. In contrast, random sensor deployment is usually in an inaccessible area. In many barrier coverage applications, the monitored area is usually inaccessible, and so the SNs can be arranged only by random deployment. In such deployment, the number of redundant SNs that do not belong to any barrier may be large. Hence, how to relocate the SNs to achieve the maximum number of barriers with a minimum relocation cost, in terms of sensor energy consumption of communication and movement, is a challenging problem.

The problem of barrier coverage improvement in WSNs by relocation of the SNs has been extensively studied in the literature [26]–[36]. Most approaches have proposed centralized deployment algorithms, and only a few studies have considered the distributed solutions. Moreover, although these algorithms have been developed to improve barrier coverage deployment, none of the techniques focuses on minimizing the relocation cost in both communication and moving distance with the goal of maximizing the number of barriers.

In this research, our approach focuses on the distributed sensor deployment with mobile SNs for maximizing the number of barriers by minimizing the energy consumption of communication and SN movement. After an initial random placement of SNs, we divide the network area into equal cells based on the length and width of the network area, such that the node sensing range can cover the left and right boundary of a cell. Some cells combine into a cluster.

Then, the cluster head (CH) nodes of each cluster exchange information together and make a decision on the final moving by computing the optimal location for each SN. The clustering approach helps reduce the number of exchange messages (communication cost). A heuristic approach is proposed to compute the new location, which helps minimize the moving distance (movement cost). To the best of our knowledge, this research is the pioneer regarding a distributed clustering protocol for barrier coverage formation with the goal of optimizing energy usage to address communication and movement costs.

The main objectives and contributions of our algorithm are as follows:

- We propose an architecture to redeploy the SNs in the monitored area by an effective distributed protocol for k-barrier coverage formation with mobile sensors.
- We propose a clustering technique in which the cluster size is defined based on the network area and the node sensing range to reduce the number of exchange messages.
- We propose a heuristic method to assign the SNs evenly to clusters based on the required number of SNs of each cluster and each cell, then decide the moving SNs by calculating the optimal relocation, considering moving distance minimization.

The organization of the article is as follows. In section II, we review some of the related work on sensor deployment for barrier coverage improvement. Section III provides the network model, problem formulation, assumptions, and definitions of our algorithm. In section IV, our distributed deployment algorithm is presented in detail. Section V presents and discusses the performance evaluation of our proposed algorithm through simulation results. Finally, the conclusions and future research directions are discussed in section VI.

II. RELATED WORK

Some algorithms that have been proposed to achieve optimized barrier coverage focus on sensor deployments for WSNs. In this section, we briefly summarize related work concerning these two types of barrier coverage deployment algorithms, i.e., centralized and distributed.

A. CENTRALIZED ALGORITHMS

Centralized deployment algorithms for mobile sensors in order to achieve barrier coverage have been recently considered in the literature [26]–[32].

Shen *et al.* [30] studied a deployment approach with mobile sensors in which the SNs can be relocated to achieve barrier coverage. The authors proposed a centralized algorithm to compute the new positions based on the initial positions of all SNs. They assumed that the relocated SNs are aligned into a straight line $y = ax + b$; therefore, their approach does not achieve the maximum number of barriers.

Saipulla *et al.* [31] proposed a deployment algorithm to relocate mobile SNs based on the deployed line to improve the barrier coverage. They considered a line-based sensor deployment strategy where SNs are dropped along a straight line. They further proposed an algorithm that finds barrier gaps and relocates mobile SNs to desirable locations to fill those gaps while minimizing the maximum energy consumption among the SNs. Their algorithm can achieve the optimal moving distance; however, it only considers construction of *1*-barrier coverage. In other words, they do not provide a solution for k-barrier coverage.

In [32], Nguyen *et al.* proposed two centralized deployment algorithms to optimize the number of barriers by minimizing the moving distance and the number of moving nodes. After an initial random placement of the SNs, the first approach applies a heuristic method to move all SNs close to the optimal location to maximize the total number of barriers. The second approach is used to determine the barriers based on a coverage graph and then to fill up the barrier gaps by moving the other SNs that do not belong to any available barrier path. The first approach is used to achieve the maximum number of barriers, while the second approach can achieve the optimal number of moving SNs and total moving distance.

Most centralized deployment algorithms can achieve the maximum number of barriers; however, these approaches suffer from drawbacks typical to centralized solutions, such as single high message overhead and lack of scalability.

B. DISTRIBUTED ALGORITHMS

There have been several studies with a special focus on distributed deployments to improve barrier coverage in WSNs [30], [33]–[37].

In [30], the authors designed a distributed barrier algorithm (DBarrier) based on a virtual force model. To form a barrier, their algorithm let the SNs gather into a certain position nearby a line crossing from the left boundary to the right boundary of the monitored area. The SNs adjust their positions into a barrier crossing the given area according to the two types of forces, i.e., the total repulsive and attractive forces. To compute the virtual force on a single SN itself in a distributed way, these authors only considered the forces exerted by neighboring nodes. The SN collects the information of two-hop neighbors, from which the one-hop neighbors are picked based on new positions after their virtual moving.

Note that this algorithm can form only *1*-barrier coverage and does not provide an effective solution for k-barrier coverage; and this is a main drawback.

Ban *et al.* [35] focused on the problem of how to relocate SNs to construct a k-barrier with minimum energy consumption. The authors proposed an Approximate to Horizontal and Vertical Grid Barrier (AHVGB) algorithm to construct k-barrier coverage in large scale sensor networks. First, their algorithm divides the network area into equal-sized sub-regions. Then, the authors construct a horizontal grid barrier and a vertical grid barrier in each sub-region. Each sub-region can independently form barriers by local information in its own area without communication with other sub-regions. Although AHVGB can minimize the energy of communication and movement, it cannot maximize the number of barriers.

Wang *et al.* [36] proposed an algorithm for movement of the SNs to satisfy the line k-barrier coverage while minimizing the total SN movements. First, the SNs are placed into several evenly distributed fixed positions. They form a group of SNs and a group of fixed points as the two groups of a bipartite graph. Then, the algorithm incorporates the two groups to satisfy the line k-barrier coverage requirement. This algorithm can minimize the total moving distance; however, it cannot achieve a good number of barriers. In addition, the authors do consider the energy for communication.

Silvestri and Goss [37] proposed an autonomous algorithm for k-barrier coverage formation (MobiBar). The objective of MobiBar is to relocate the SNs to achieve a strong k-barrier coverage over the network area. Their algorithm aims at constructing k barriers of SNs such that every crossing path from the entrance boundary to the exit boundary of the monitored area intersects the sensing ranges of at least k different SNs. This algorithm defines a line parallel to the long edge of the area, called a baseline. Then, the model requires the SNs to construct a barrier on the baseline and the adjacent barriers to be located at a given distance. In general, the algorithm relocates the SNs to construct k-barrier in a normal way, and it does not achieve the optimizing of the moving distance.

III. PRELIMINARIES

In this section, we describe the system overview of the proposed algorithm, including some assumptions and definitions for the algorithm.

A. NETWORK MODEL

We assume that a network N consists of a large number of SNs S_i such that $\{i | 1, \dots, n\}$. Each SN has a unique identification – *id*; each i^{th} SN is located at a coordinate (x_i, y_i); and all of them are randomly deployed over a rectangular two-dimensional region with a length (l) and width (w). An example of sensor network deployment for barrier coverage is illustrated in Fig. 1.

We assume that all SNs have the same sensing radius and communication radius. The communication range is much larger than the sensing range [38]; this assumption is realistic based on the actual motes, such as MICAz, whose communication range is much higher than the sensing range of several typical sensors [39]. The communication links between SNs are bi-directional.

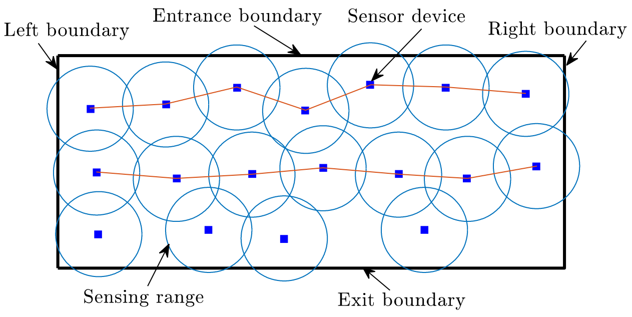

FIGURE 1. A sensor network deployment that provides 2-barrier coverage.

We also assume that each SN knows its position (with low cost GPS) [22], [40], but we consider possible localization and positioning inaccuracies, bounded by a maximum error ε [37]. Therefore, we assume that the actual sensing radius (r) of SNs is equal to the minimum sensing radius $r_{\min}$ minus the maximum error (ε), i.e., $r = r_{\min} - \varepsilon$.

Table 1 introduces some notations used in our network models and our algorithm.

TABLE 1. Notations.

Symbols	Descriptions
n	Number of SNs in the network
l	Length of the network area
w	Width of the network area
l_c	Length of cell
w_c	Width of cell
h	The sequence number of the cell from left to right
v	The sequence number of the cell from bottom to top
e_{hv}	The required number of nodes of cell hv
$e_{c_{S_i}}$	The required number of nodes of the cluster that node S_i belongs to
$id_{c_{S_i}}$	The *id* of the cluster that node S_i belongs to
x_{S_i}	The coordinate x of node S_i
y_{S_i}	The coordinate y of node S_i
$\Theta_{c_{S_i}}$	The coordinates of the center point of the cluster that node S_i belongs to
$u_{c_{S_i}}$	The current number of nodes of the cluster that node S_i belongs to
Θ_{k_z}	The coordinates of the new location z in cell k

B. DEFINITIONS

In this subsection, we introduce some definitions to support our algorithm.

Definition 1 (Barrier): A barrier is a group of SNs that together form a node-disjoint path between the left boundary and right boundary of the network area (see Fig. 1).

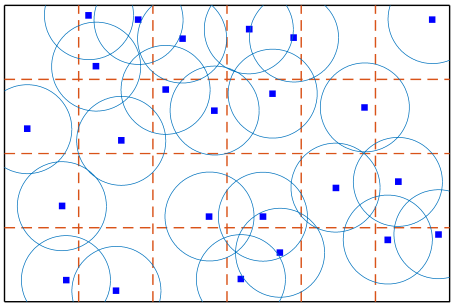

FIGURE 2. An example of the network area with cell division.

Definition 2 (Cell): The monitored area is divided into equal cells (see Fig. 2). At the end of the algorithm, the SNs must move so that each cell contains a certain number of SNs. The length and width of the cell are defined as follows:

$$l_c = \frac{l}{\left\lceil \frac{l}{2r} \right\rceil}, \tag{1}$$

$$w_c = \frac{w}{\left\lceil \frac{w}{2r} \right\rceil}. \tag{2}$$

The required number of SNs of each cell is calculated as follows:

$$e_{hv} = \frac{n}{\frac{l}{l_c} \cdot \frac{w}{w_c}} + \frac{n \, mod \left(\frac{l}{l_c} \cdot \frac{w}{w_c} \right)}{v \cdot \frac{l}{l_c} + h}, \tag{3}$$

where l_c and w_c are the length and width of the cells; l and w are the length and width of the network; e_{hv} is the required number of SNs of cell hv (h is the sequence number of the cell from left to right – horizontal; v is the sequence number from bottom to top – vertical).

Definition 3 (Cluster): A cluster is a group of λ^2 cells located next to each other ($\lambda = \{2, 3, \ldots\}$). In this research, we consider $\lambda = 2$, based on the relationship between sensing range and communication range. This means that each cluster contains 4 cells. Note that, in case the number of rows is odd, there are some clusters located at the top that contain only 2 cells.

Here, each cluster has its own *id*; the cluster *id* is numbered in order, as shown in Fig. 3. Each SN can know the *id* of the cluster where it is located by calculating as follows:

$$\begin{aligned} id_{cs_i} = {} & \left(\left\lceil \frac{xs_i}{2l_c} \right\rceil - 1 \right) \cdot \left\lceil \frac{w}{2w_c} \right\rceil + \left\lceil \frac{ys_i}{2w_c} \right\rceil^{(x \, mod \, 2)} \\ & \cdot \left(\left\lceil \frac{w}{2w_c} \right\rceil + 1 - \left\lceil \frac{ys_i}{2w_c} \right\rceil \right)^{\left(\left(\left\lceil \frac{xs_i}{2l_c} \right\rceil + 1 \right) mod \, 2 \right)}. \end{aligned} \tag{4}$$

The required number of SNs of each cluster is calculated as follows:

$$e_{cs_i} = \frac{n}{\left\lceil \frac{l}{2l_c} \right\rceil \cdot \left\lceil \frac{w}{2w_c} \right\rceil} + \frac{n \, mod \left(\left\lceil \frac{l}{2l_c} \right\rceil \cdot \left\lceil \frac{w}{2w_c} \right\rceil \right)}{\left\lceil \frac{ys_i}{2w_c} \right\rceil \cdot \left\lceil \frac{l}{2l_c} \right\rceil + \left\lceil \frac{xs_i}{2l_c} \right\rceil}. \tag{5}$$

Definition 4 (Neighbor Cluster): Cluster B is called a neighbor cluster of cluster A if cluster B has cluster *id* greater than the *id* of cluster A, and cluster B is located adjacent to cluster A (horizontal, vertical, or diagonal).

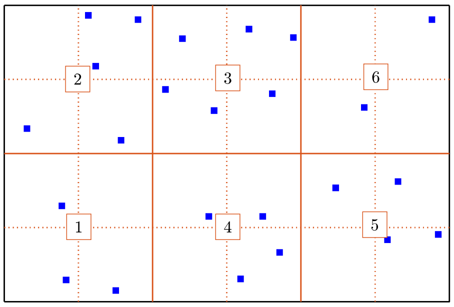

FIGURE 3. Clusters and the numbering of cluster *id*.

For example, in Fig. 3, cluster 1 has three neighbor clusters, including clusters 2, 3, and 4; and cluster 4 has two neighbor clusters, including clusters 5 and 6.

In our algorithms, we define some different messages as follows:

- *Info-Msg* (node *id*, cluster *id*, node location).
- *Cluster-Msg* (node *id*, cluster *id*, number of nodes in cluster).
- *No-Req* (node *id*, destination *id*, *P*). *P* is set of cluster *id*; it will be described in detail later in section IV.
- *Pull-Req* (node *id*, destination *id*, *P*).
- *Push-Req* (node *id*, destination *id*, *P*).
- *Change-Req* (node *id*, destination *id*, new CH *id*).
- *Join-Msg* (node *id*, destination *id*, node location).
- *Move-Req* (node *id*, destination *id*, new node location).

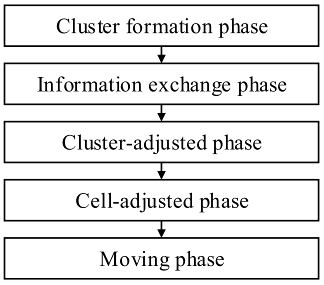

FIGURE 4. Five phases of DDABC.

IV. DISTRIBUTED DEPLOYMENT ALGORITHM FOR BARRIER COVERAGE (DDABC)

In this section, we discuss our distributed deployment algorithm for barrier coverage in detail. Fig. 4 shows an overall view of DDABC. DDABC consists of a cluster formation phase, an information exchange phase, a cluster-adjusted phase, a cell-adjusted phase, and a moving phase.

A. CLUSTER FORMATION PHASE

The first phase of DDABC is cluster formation, which is used to get the location of SNs and the number of SNs in the same cluster. In this phase, each SN broadcasts *Info-Msg*, which contains the information about the cluster *id* (see Definition 3) and its location.

After receiving the *Info-Msg* from all SNs in the same cluster, an SN S_i will compare the distance between node location and the center of the cluster $d\left(S_i, \Theta_{cs_i}\right)$ of each node within the cluster. The location of the cluster center that node S_i belongs to is calculated as follows:

$$\Theta_{cs_i} = \left(\left\lceil \frac{x_{S_i}}{2l_c} \right\rceil \cdot 2l_c - l_c, \left\lceil \frac{y_{S_i}}{2w_c} \right\rceil \cdot 2w_c - w_c\right). \qquad (6)$$

If the distance between the location of an SN and the center of the cluster is shortest, it will become the CH node of that cluster; otherwise, it will become a cluster member (CM) node. Algorithm 1 provides the details of the first phase.

Algorithm 1 The Cluster Formation Phase

```
$S_i$ broadcasts *Info-Msg*
receive and process *Info-Msg* from all nodes in the
  same cluster
**if** $d(S_i, \Theta_{cs_i}) = \min\{d(S_j, \Theta_{cs_i}) | S_j \in c_{S_i}\}$ **then**
    $S_i$ become a CH node
**else**
    $S_i$ become a CM node
**end if**
```

B. INFORMATION EXCHANGE PHASE

In this phase, each CH node exchanges *Cluster-Msg* with other CH nodes; the *Cluster-Msg* contains the information about the cluster *id* and the number of SNs in the cluster. If a CH node receives *Cluster-Msg* from a CH node of its neighbor cluster, it must save that information. Each CH node maintains a table containing the information of all its neighbor clusters from which it has received *Cluster-Msg* thus far.

Algorithm 2 shows the pseudo-code for this phase.

Algorithm 2 The Information Exchange Phase

```
**if** ($S_i$ is the CH node) **then**
    $S_i$ sends *Cluster-Msg* to the CH nodes of its
    neighbor clusters
    **if** ($S_i$ receives *Cluster-Msg* from the CH nodes of
    its neighbor clusters) **then**
        save information of these clusters
    **end if**
**end if**
```

C. CLUSTER-ADJUSTED PHASE

After the information exchange phase, the CH nodes begin the operation of the cluster-adjusted phase, which is illustrated in Algorithm 3. In this phase, each cluster adjusts the number of SNs such that they have exactly the required number of SNs in the end of this phase. The operation of this phase is carried out sequentially in the order of the cluster *id*;

Algorithm 3 The Cluster-Adjusted Phase

```
**if** ($S_i$ is a CH node of cluster $p$) **then**
    **if** ($S_i$ receives *Pull-Req* from a CH node) **then**
        choose the nodes closest to the requested
        cluster by sending *Change-Req* to these nodes,
        update the current number of nodes
        update the information of its neighbor clusters
    **end if**
    **if** ($S_i$ receives *Push-Req* from a CH node) **then**
        update the current number of nodes (if it was
        assigned more nodes)
        update the information of its neighbor clusters
    **end if**
    **if** ($id_{cs_i} = 1$ or $S_i$ receives a request from a CH
      node of the previous cluster) **then**
        **if** $\left(u_{cs_i} = e_{cs_i}\right)$ **then**
            send *No-Req* to the CH node of the next
            cluster (*id* is greater than 1 unit)
        **else**
            **if** $\left(u_{cs_i} < e_{cs_i}\right)$ **then**
                $x \leftarrow e_{cs_i} - u_{cs_i}$
                **while** $x > 0$ **do**
                    $\min \leftarrow n$
                    **for** $q \leftarrow 1$ to $|V_p|$ **do**
                        $x_q \leftarrow 1; o_q \leftarrow d_q + o_{q1} + o_{q2}$
                        **if** $(o_q < \min)$ **then**
                            $\min \leftarrow o_q; s \leftarrow q$
                        **end if**
                    **end for**
                    $P \leftarrow P + [id_q]; x \leftarrow x - 1$
                **end while**
                send *Pull-Req* to the CH nodes of the
                neighbor clusters
            **else**
                $x \leftarrow u_{cs_i} - e_{cs_i}$
                **while** $x > 0$ **do**
                    $\min \leftarrow n$
                    **for** $q \leftarrow 1$ to $|V_p|$ **do**
                        $x_q \leftarrow 1; o_q \leftarrow d_q + o_{q3} + o_{q4}$
                        **if** $(o_q < \min)$ **then**
                            $\min \leftarrow o_q; s \leftarrow q$
                        **end if**
                    **end for**
                    $P \leftarrow P + [id_q]; x \leftarrow x - 1$
                **end while**
                send *Push-Req* to the CH nodes of the
                neighbor clusters
                choose the nodes for assigning by sending
                *Change-Req* to these nodes, update the
                current number of nodes
            **end if**
        **end if**
    **end if**
    **if** ($S_i$ receives *Join-Req* from a node) **then**
        update the list of current number of nodes
    **end if**
**else**($S_i$ is a CM node of cluster $p$)
    **if** ($S_i$ receives *Change-Req* from a CH node) **then**
        send *Join-Req* message to the new CH node
    **end if**
**end if**
```

this will ensure the clusters will get exactly the required number of SNs.

At the beginning of this phase, the CH node of the cluster with $id = 1$ will perform the operation first; it starts by checking the current number of SNs and the required number of SNs. There are three cases to consider as follows:

Case 1: $u_{cs_i} = e_{cs_i}$, the current number of SNs is equal to the required number of SNs. For this case, the CH node sends *No-Req* to the CH node of the next cluster (*id* is greater than 1 unit).

Case 2: $u_{cs_i} < e_{cs_i}$, the current number of SNs is less than the required number of SNs. In this situation, this cluster must pull the lacking number of nodes from its neighbor clusters, but first, the CH node must calculate in order to find the optimal solution by considering the following three objectives.

$$\sum_{q\in V_p}^{|V_p|} x_q d_q \rightarrow \min,$$

$$o_{q1} = \frac{e_q}{u_q - x_q}, \tag{7}$$

$$\sum_{q\in V_p}^{|V_p|} x_q o_{q1} \rightarrow \min,$$

$$o_{q2} = \frac{e_q}{\frac{\sum_{k\in V_q}^{|V_q|} u_k + (u_q - x_q)}{|V_q|+1}}, \tag{8}$$

$$\sum_{q\in V_p}^{|V_p|} x_q o_{q2} \rightarrow \min, \tag{9}$$

where V_p is a set of neighbor clusters of cluster p; x_q is the number of nodes that cluster p wants to get from cluster q; u_q is the current number of nodes of cluster q; e_q is the required number of nodes of cluster q; and d_q is the distance between cluster p and cluster q. Assume that $d_q = 1$ if the two neighbor clusters p and q are on the same row or column, and $d_q = 1.3$ if the two neighbor clusters p and q are on the same diagonal.

The first objective is to minimize the moving distance of the SNs (Equation 7); the second objective is to pull the SNs from the cluster with highest number of SNs (Equation 8); and the third objective is to pull the SNs from the cluster with highest number of SNs of its neighbor clusters (Equation 9). Note that the common objective is expressed as follows:

$$\sum_{q\in V_p}^{|V_p|} x_q d_q + \sum_{q\in V_p}^{|V_p|} x_q o_{q1} + \sum_{q\in V_p}^{|V_p|} x_q o_{q2} \rightarrow \min. \tag{10}$$

It should be noted that the above objective must satisfy:

$$\sum_{q\in V_p}^{|V_p|} x_q = |e_p - u_p|. \tag{11}$$

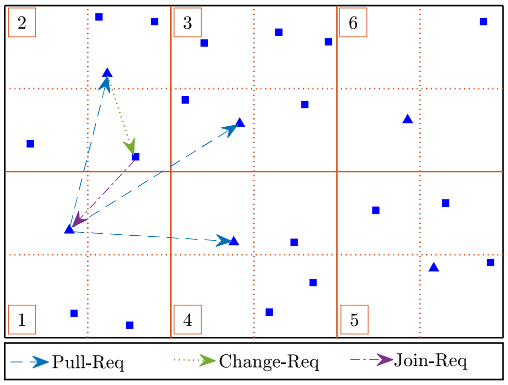

FIGURE 5. An example of requesting for cluster-adjusted.

After calculating to get the optimal solution, the CH node has to send *Pull-Req* to the other CH nodes of its neighbor clusters. Note that if a CH node receives *Pull-Req*, it must choose the SNs (in quantity as required) closest to the center of the requested cluster and sends *Change-Req* to these nodes. Once an SN receives *Change-Req*, it has to send *Join-Msg* to the new CH node to confirm that it will join the new cluster. Fig. 5 illustrates an example of the process of requesting.

Case 3: $u_{cs_i} > e_{cs_i}$, the current number of SNs is larger than the required number of SNs. In this situation, this cluster has to push the excess number of SNs to its neighbor clusters. Similar to the second case, the CH node must calculate in order to find the optimal solution by considering the three objectives including Equations 7, 12, and 13.

$$o_{q3} = \frac{u_q + x_q}{e_q},$$

$$\sum_{q\in V_p}^{|V_p|} x_q o_{q3} \rightarrow \min, \tag{12}$$

$$o_{q4} = \frac{\frac{\sum_{k\in V_q}^{|V_q|} u_k + (u_q + x_q)}{|V_q|+1}}{e_q},$$

$$\sum_{q\in V_p}^{|V_p|} x_q o_{q4} \rightarrow \min. \tag{13}$$

The objective of Equation 12 is to push the SNs to the cluster with the lowest number of SNs and the objective of Equation 13 is to push the SNs to the cluster with lowest number of SNs of its neighbor clusters. Note that the common objective of this case is expressed as follows:

$$\sum_{q\in V_p}^{|V_p|} x_q d_q + \sum_{q\in V_p}^{|V_p|} x_q o_{q3} + \sum_{q\in V_p}^{|V_p|} x_q o_{q4} \rightarrow \min. \tag{14}$$

Similar to the second case, the above objective must satisfy Equation 11. After a CH node has the optimal solution, it has to send *Push-Req* to the other CH nodes of its neighbor clusters.

Note that *Pull-Req* and *Push-Req* contain a set P. For example, the CH node of cluster 1 sends *Pull-Req* to the CH nodes of clusters 2, 3, and 4, which contains $P = \{2, 3, 3\}$. It means that the CH node of cluster 1 wants to pull 1 node from cluster 2 and 2 nodes from cluster 3, accordingly.

It also should be noted that, after a CH node receives a request message (*Pull-Req*, *Push-Req*, or *No-Req*) from a CH node of the previous cluster (*id* is less than 1 unit), it will perform the same process by following the above three cases. This process will continue until the last CH node. The following pseudo-code provides the details of the cluster-adjusted phase.

D. CELL-ADJUSTED PHASE

After a cluster-adjusted phase, all clusters already have the required number of nodes. However, cells in each cluster may not have exactly (lack or excess) the required number of nodes. In this phase, the CH node must adjust the number of nodes of each cell. In other words, the CH node has to assign each SN to the appropriate cell such that each cell in the cluster has exactly the required number of SNs (see also Equation 3).

The pseudo-code shown in algorithm 4 provides the details of the cell-adjusted phase.

E. MOVING PHASE

Once each cell has exactly the required number of nodes, the network will begin the operation of the moving phase; it is the final phase of DDABC. In this phase, each CH node computes and decides the new location of each SN in its cluster. The new locations of the SNs in each cell are distributed evenly on the vertical line that passes through the center of the cell. This will ensure that the sensing range of the SN overlaps the left boundary and the right boundary of the cell.

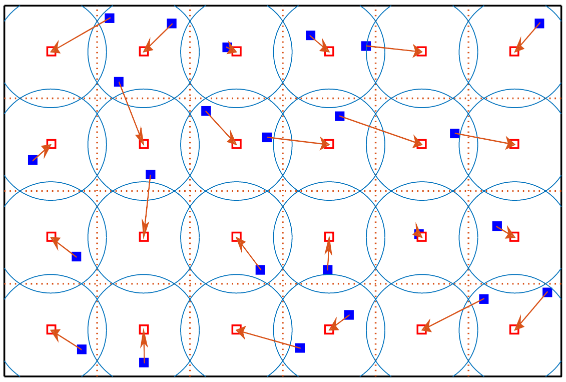
FIGURE 6. An example of the relocation.

After determining the optimal location for each SN, the CH node sends *Move-Req* to its CM nodes. As with the CM nodes, the CH node also moves to a new location to be put together with other SNs to form the barriers. Algorithm 5 provides the details of the moving phase of DDABC (see also Fig. 6 as an example of node relocation).

Algorithm 4 The Cell-Adjusted Phase

```
1  if ($S_i$ is a CH node of cluster $p$) then
2    $x = |C_p|$ //($C_p$ is set of nodes that have joined
      cluster $p$ in the previous phase, but are located
      outside this cluster)
3    if ($x > 0$) then
4      for $j \leftarrow 1$ to $x$ do
5        if $d(S_j, \Theta_{cell_k}) = \min\left\{ d(S_j, \Theta_{cell_k}) \mid k \in \{p_1, p_2, p_3, p_4\} \right\}$
           then
6          assign $S_j$ to $L_k$ //$L_k$ is a set of nodes of
           cell $k$
7        end if
8      end for
9    end if
10   for $k \leftarrow 1$ to 4 do
11     if ($u_k < e_k$) then
12       $a \leftarrow e_k - u_k$
13       for $j \leftarrow k + 1$ to 4 do
14         if ($u_j > e_j$) then
15           $b \leftarrow u_j - e_j$
16           if ($a \leq b$) then
17             choose $a$ nodes in cell $j$ closest to
              cell $k$ for cell assigning, update the
              information of each cell
18             break
19           else
20             choose $b$ nodes in cell $j$ closest to
              cell $k$ for cell assigning, update the
              information of each cell
21             $a \leftarrow a - b$
22           end if
23         end if
24       end for
25     end if
26     if ($u_k > e_k$) then
27       $a \leftarrow u_k - e_k$
28       for $j \leftarrow k + 1$to 4 do
29         if ($u_j < e_j$) then
30           $b \leftarrow e_j - u_j$
31           if ($a \leq b$) then
32             choose $a$ nodes in cell $k$ closest to
              cell $j$ for cell assigning, update the
              information of each cell
33             break
34           else
35             choose $b$ nodes in cell $k$ closest to
              cell $j$ for cell assigning, update the
              information of each cell
36             $a \leftarrow a - b$
37           end if
38         end if
39       end for
40     end if
41   end for
42 end if
```

Algorithm 5 The Moving Phase

```
1  if ($S_i$ is a CH node of cluster $p$) then
2    for $k \leftarrow 1$ to 4 do
3      $x = |L_k|$ //$L_k$ is a set of SNs in cell $k$
4      for $j \leftarrow 1$ to $x$ do
5        if $d(S_j, \Theta_{k_z}) = \min\{d(S_j, \Theta_{k_z}) | z \in W_k\}$
         then // $W_k$ is a set of points of new location
         $(\Theta_{k_z})$ in cell $k$
6          send Move-Req to $S_j$
7          $W_k \leftarrow W_k \backslash \Theta_{k_z}$
8        end if
9      end for
10   end for
11 else
12   if ($S_i$ receives Move-Req from its CH node) then
13       move to the new location
14   end if
15 end if
```

V. PERFORMANCE EVALUATION

In this section, we investigate the performance of our algorithm DDABC compared to that of other distributed deployment algorithms for barrier coverage, such as DBarrier [30], AHVGB [35], and MobiBar [37].

A. SIMULATION SETUPS

A MATLAB simulator has been used for the simulation. All SNs were randomly distributed across a plain area of 100×400 m^2. We simulated the algorithms with various numbers of SNs, i.e., $n = \{100, 200, 300, 400, 500\}$. We set the actual sensing radius r to be 10 meters and the communication radius to be 80 meters [39].

Note that our energy model follows the models defined in [22], [37], and [40]. In particular, the energy consumption of sending and receiving a message is 1.125 units and 1 unit, respectively. For energy consumption of movement, 1 meter of movement costs 300 times that of sending a message, and starting or stopping a movement costs the equivalent of a 1 meter movement [37].

We evaluated the performance of DDABC by making a comparison with three other algorithms; DBarrier [30], AHVGB [35], and MobiBar [37].

The DBarrier is a distributed algorithm for *1*-barrier coverage. DBarrier adjusts the SNs' positions according to the total repulsive and attractive forces. In our simulation, we modified DBarrier for k-barrier coverage by horizontally splitting the network area equally.

The AHVGB is a distributed algorithm for k-barrier coverage. AHVGB divides the network area into rectangular sub-regions. In each sub-region, an SN is selected as a CH node. The CH node calculates the new positions of the SNs in its sub-region; a subset of SNs fills the right side of the rectangle, while another set forms a line that horizontally crosses the sub-region. In our simulation, to get a fair result, we increased the number of sub-regions proportionally to the number of SNs.

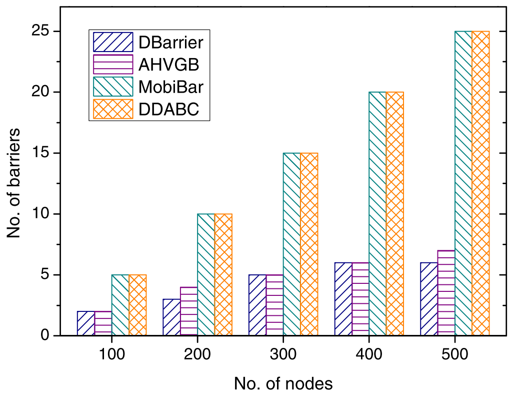

FIGURE 7. Number of barriers of the final deployment.

MobiBar is also a distributed algorithm for k-barrier coverage. MobiBar forms the barriers by moving the SNs toward the horizontal lines.

There are five key metrics of these evaluations, i.e., the number of barriers, the average moving distance of the SNs, the ratio of the average moving distance to the number of barriers, the average energy consumption of movement and communication, and the ratio of the average energy consumption to the number of barriers. The results of all the simulations are averaged over twenty simulation runs in each configuration.

B. SIMULATION RESULTS

Fig. 7 shows the barrier coverage level after performing the movement of the SNs formulated by DBarrier, AHVGB, MobiBar, and DDABC. It is clear that the number of barriers of MobiBar and DDABC are the largest and reach the maximum value. In contrast, the number of barriers of DBarrier and AHVGB are very low due to their lack of adaptability. Upon increasing the number of SNs, the number of barriers of all algorithms increased. This is justifiable because the larger the number of SNs is, the larger the number of available barriers is. However, when the number of SNs is increased, DBarrier and AHVGB cannot exploit the extra SNs to form many barriers because the structure of the area or sub-region has limited the formation of more barriers.

Fig. 8 shows the average moving distance per SN under different numbers of SNs. The average moving distance of AHVGB is the lowest; this is justifiable because in this algorithm, the SNs move locally within each sub-region. DDABC achieves the average moving distance, similar to DBarrier and much better than MobiBar.

We provide additional insights on the ratio of the average moving distance to the number of barriers in Fig. 9. DDABC is superior; we observe a similar trend of the average moving distance to the number of barriers for the four algorithms, in which our DDABC algorithm results in the lowest total distance of only approximately 40% of the values for DBarrier and AHVGB, and only approximately 60% to 80% of the value for MobiBar.

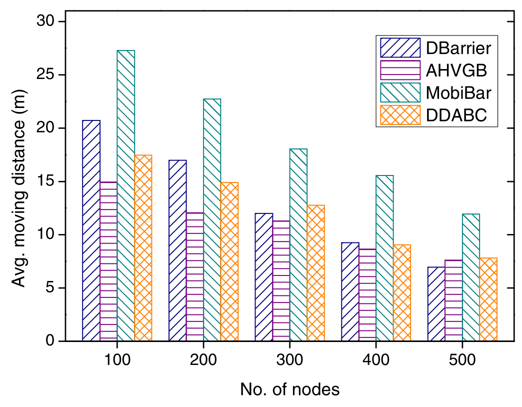

FIGURE 8. Average moving distance.

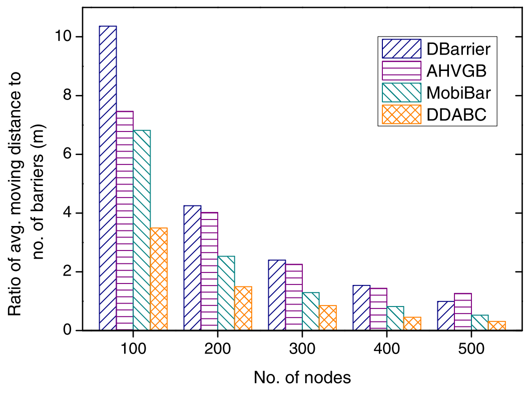

FIGURE 9. Ratio of average moving distance to number of barriers.

Fig. 10 compares the average energy consumption of communication spending on the four algorithms. It is clear from the charts that DDABC and AHVGB show very good performance. This is justifiable because DDABC divides the network area into many clusters similar to the sub-regions in AHVGB. This helps these algorithms to limit the number of sending and receiving messages. In contrast, there is no range limit for sending messages in DBarrier and MobiBar. This leads to the SNs having to receive many messages.

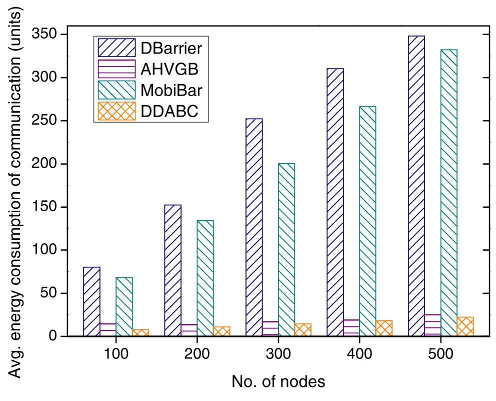

FIGURE 10. Average energy consumption of communication.

Fig. 11 shows the average energy consumption of movement. For this metric, AHVGB incurs a low energy consumption of movement because this algorithm provides a low moving distance. However, this comes at the expense of a lower number of barriers. The average energy consumption of movement of DDABC is similar to DBarrier and is only approximately 65% to 75% of the value for MobiBar.

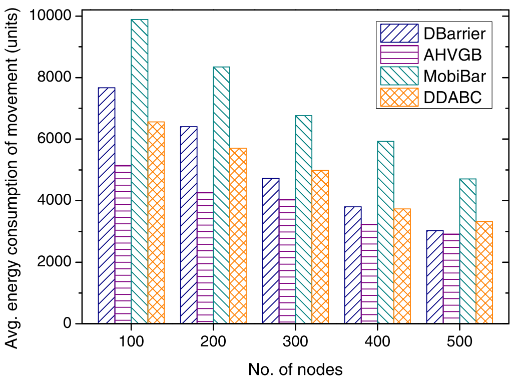

FIGURE 11. Average energy consumption of movement.

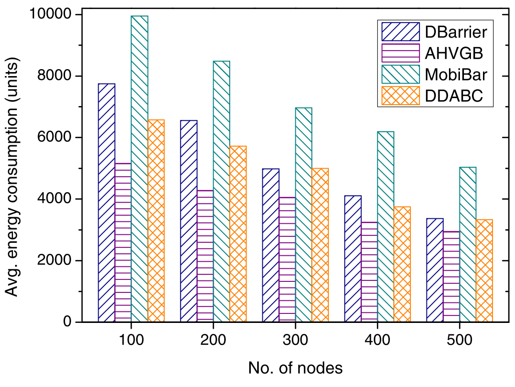

FIGURE 12. Average energy consumption.

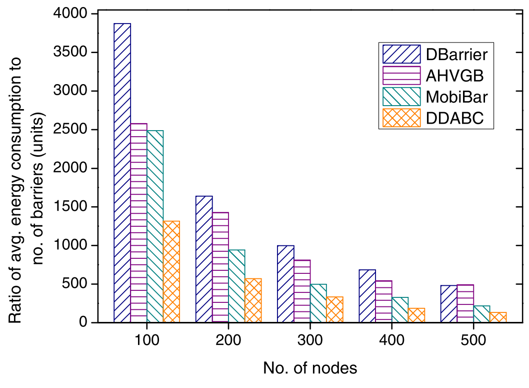

FIGURE 13. Ratio of average energy consumption to number of barriers.

Fig. 12 compares the overall average energy consumption (both communication and movement) of the four algorithms. The energy consumption of the movement is significantly higher than the energy consumption of the communication. Therefore, this bar chart is similar to the bar chart of the average energy consumption of movement. This is true of the values of DDABC and AHVGB due to the lower energy consumption of communication. Different from the two above algorithms, DBarrier and MobiBar consume more energy for communication; and this has led to a significant increase in their total energy consumption.

Fig. 13 shows the ratio of the average energy consumption to the number of barriers. Again, DDABC is superior. The ratio of the average energy consumption to the number of barriers of DDABC is only approximate at 20% to 40%, 30% to 55%, and 60% to 75% of the values of DBarrier, AHVGB, and MobiBar, respectively. The results have shown the effectiveness of the energy consumption optimization of DDABC in the process of formulating the maximum number of barriers.

VI. CONCLUSION

In this research, we proposed a distributed deployment algorithm for maximizing the number of barriers with mobile sensors, called DDABC. DDABC optimizes the relocation cost that includes communication and movement by considering a cluster-based processing.

First, DDABC considers reducing the number of exchange messages by dividing the monitored area into clusters. The cluster size is determined based on the monitored area and the node sensing range. The clustering approach helps optimizing energy consumption of communication. Moreover, it helps minimizing the maximum moving distance of SNs. Second, DDABC considers finding the optimal solution to adjust the required number of SNs in each cluster and each cell such that the moving distance is lowest.

To evaluate the performance of our proposed scheme, we compared the performance of our algorithm – DDABC – to the performance of the DBarrier, AHVGB, and MobiBar algorithms in terms of the number of barriers, the average moving distance, and the average energy consumption of communication and movement after the final deployment. Through simulation experiments, the results show that our algorithm has achieved the maximum number of barriers by moving the SNs to the optimal locations. In particular, our algorithm outperforms the compared algorithms in almost all performance parameters.

Although our algorithm can achieve some improvements for barrier coverage, there are still some modifications that can be made for further enhancements. Our future work will solve that problem by applying soft-computing algorithms to achieve global optimization.

REFERENCES

[1] I. F. Akyildiz and M. C. Vuran, *Wireless Sensor Networks*. Hoboken, NJ, USA: Wiley, 2010.

[2] C. M. Cordeiro and D. P. Agrawal, *Ad Hoc and Sensor Networks: Theory and Applications*. Singapore: World Scientific, 2006.

[3] B. Wang, *Coverage Control in Sensor Networks* (Computer Communications and Networks). London, U.K.: Springer, 2010.

[4] J. N. Al-Karaki and A. Gawanmeh, "The optimal deployment, coverage, and connectivity problems in wireless sensor networks: Revisited," *IEEE Access*, vol. 5, pp. 18051–18065, 2017.

[5] Y. Wang, S. Wu, Z. Chen, X. Gao, and G. Chen, "Coverage problem with uncertain properties in wireless sensor networks: A survey," *Comput. Netw.*, vol. 123, pp. 200–232, Aug. 2017.

[6] S. Mini, S. K. Udgata, and S. L. Sabat, "Sensor deployment and scheduling for target coverage problem in wireless sensor networks," *IEEE Sensors J.*, vol. 14, no. 3, pp. 636–644, Mar. 2014.

[7] Z. Liao, J. Wang, S. Zhang, J. Cao, and G. Min, "Minimizing movement for target coverage and network connectivity in mobile sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 7, pp. 1971–1983, Jul. 2015.

[8] T. G. Nguyen and C. So-In, "An energy-efficient coverage aware clustering mechanism for wireless sensor networks," *Int. J. Ad Hoc Ubiquitous Compu.*, to be published. [Online]. Available: http://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=IJAHUC, doi: 10.1504/IJAHUC.2016.10001915.

[9] Y. Yoon and Y.-H. Kim, "An efficient genetic algorithm for maximum coverage deployment in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1473–1483, Oct. 2013.

[10] H. Mahboubi, K. Moezzi, A. G. Aghdam, and K. Sayrafian-Pour, "Distributed deployment algorithms for efficient coverage in a network of mobile sensors with nonidentical sensing capabilities," *IEEE Trans. Veh. Technol.*, vol. 63, no. 8, pp. 3998–4016, Oct. 2014.

[11] T. G. Nguyen, C. So-In, N. G. Nguyen, and S. Phoemphon, "A novel energy-efficient clustering protocol with area coverage awareness for wireless sensor networks," *Peer-Peer Netw. Appl.*, vol. 10, no. 3, pp. 519–536, 2017.

[12] S. Kumar, T. H. Lai, and A. Arora, "Barrier coverage with wireless sensors," in *Proc. 11th Annu. Int. Conf. Mobile Comput. Netw.*, 2006, pp. 284–298.

[13] S. Kumar *et al.*, "Maximizing the lifetime of a barrier of wireless sensors," *IEEE Trans. Mobile Comput.*, vol. 9, no. 8, pp. 1161–1172, Aug. 2010.

[14] A. Chen, S. Kumar, and T. H. Lai, "Local barrier coverage in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 4, pp. 491–504, Apr. 2010.

[15] T. G. Nguyen, C. So-In, and N. G. Nguyen, "Maximum barrier coverage deployment algorithms in wireless sensor networks," in *Proc. Int. Joint Conf. Comput. Sci. Softw. Eng.*, Khon Kaen, Thailand, Jul. 2016, pp. 1–5.

[16] J. He and H. Shi, "Constructing sensor barriers with minimum cost in wireless sensor networks," *J. Parallel Distrib. Comput.*, vol. 71, no. 12, pp. 1654–1663, 2012.

[17] C.-F. Cheng, T.-Y. Wu, and H.-C. Liao, "A density-barrier construction algorithm with minimum total movement in mobile WSNs," *Comput. Netw.*, vol. 62, pp. 208–220, Apr. 2014.

[18] D. W. Gage, "Command control for many-robot systems," *Unmanned Syst. Mag.*, vol. 10, no. 4, pp. 28–34, 1992.

[19] G. T. Sibley, M. H. Rahimi, and G. S. Sukhatme, "Robomote: A tiny mobile robot platform for large-scale ad-hoc sensor networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, Washington, DC, USA, May 2002, pp. 1143–1148.

[20] M. R. Senouci and A. Mellouk, *Deploying Wireless Sensor Networks: Theory and Practice*. Amsterdam, The Netherlands: Elsevier, 2016.

[21] S. Abdollahzadeh and N. Navimipour, "Deployment strategies in the wireless sensor network: A comprehensive review," *Comput. Commun.*, vols. 91–92, pp. 1–16, Oct. 2016.

[22] N. Bartolini, T. Calamoneri, T. F. La Porta, and S. Silvestri, "Autonomous deployment of heterogeneous mobile sensors," *IEEE Trans. Mobile Comput.*, vol. 10, no. 6, pp. 753–766, Jun. 2011.

[23] D. S. Deif and Y. Gadallah, "An ant colony optimization approach for the deployment of reliable wireless sensor networks," *IEEE Access*, vol. 5, pp. 10744–10756, 2017.

[24] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization in distributed sensor networks," *ACM Trans. Embedded Comput. Syst.*, vol. 3, no. 1, pp. 61–91, 2004.

[25] M. R. Senouci, A. Mellouk, and K. Assnoune, "Localized movement-assisted sensordeployment algorithm for holedetection and healing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 5, pp. 1267–1277, May 2014.

[26] A. Saipulla, B. Liu, G. Xing, X. Fu, and J. Wang, "Barrier coverage with sensors of limited mobility," in *Proc. 11th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Chicago, IL, USA, 2010, pp. 201–210 .

[27] J. Du, K. Wang, H. Liu, and D. Guo, "Maximizing the lifetime K-discrete barrier coverage using mobile sensors," *IEEE Sensors J.*, vol. 13, no. 12, pp. 4690–4701, Dec. 2013.

[28] D. Z. Chen, Y. Gu, J. Li, and H. Wang, "Algorithms on minimizing the maximum sensor movement for barrier coverage of a linear domain," *Discrete Comput. Geometry*, vol. 50, no. 2, pp. 374–408, 2013.

[29] S. Li and H. Shen, "Minimizing the maximum sensor movement for barrier coverage in the plane," in *Proc. IEEE Conf. Comput. Commun.*, Hong Kong, Apr./May 2015, pp. 244–252.
[30] C. Shen, W. Cheng, X. Liao, and S. Peng, "Barrier coverage with mobile sensors," in *Proc. Int. Symp. Parallel Architectures, Algorithms, Netw.*, Sydney, NSW, Australia, May 2008, pp. 99–104.
[31] A. Saipulla, C. Westphal, B. Liu, and J. Wanga, "Barrier coverage with line-based deployed mobile sensors," *Ad Hoc Netw.*, vol. 11, no. 4, pp. 1381–1391, 2013.
[32] T. G. Nguyen, C. So-In, and N. G. Nguyen, "Barrier coverage deployment algorithms for mobile sensor networks," *J. Internet Technol.*, vol. 8, no. 7, pp. 1689–1699, 2017.
[33] L. Kong, X. Liu, Z. Li, and M.-Y. Wu, "Automatic barrier coverage formation with mobile sensor networks," in *Proc. IEEE Int. Conf. Commun.*, Cape Town, South Africa, May 2010, pp. 1–5.
[34] J. Jia, X. Wu, J. Chen, and X. Wang, "An autonomous redeployment algorithm for line barrier coverage of mobile sensor networks," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 16, no. 1, pp. 58–69, 2014.
[35] D. Ban, W. Yang, J. Jiang, J. Wen, and W. Dou, "Energy-efficient algorithms for k-barrier coverage in mobile sensor networks," *Int. J. Comput., Commun. Control*, vol. 5, no. 5, pp. 616–624, 2010.
[36] Y. Wang, S. Wu, X. Gao, F. Wu, and G. Chen, "Minimizing mobile sensor movements to form a line K-coverage," *Peer-Peer Netw. Appl.*, vol. 10, no. 4, pp. 1063–1078, 2017.
[37] S. Silvestri and K. Goss, "MobiBar: An autonomous deployment algorithm for barrier coverage with mobile sensors," *Ad Hoc Netw.*, vol. 54, pp. 111–129, Jan. 2017.
[38] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc Sensor Wireless Netw.*, vol. 1, nos. 1–2, pp. 89–124, 2005.
[39] *MICAz*. Accessed: Jan. 2, 2018. Online]. Available: http://www.memsic.com/userles/les/Datasheets/WSN/micaz_datasheet-t.pdf
[40] G. Wang, G. Cao, and T. F. La Porta, "Movement-assisted sensor deployment," *IEEE Trans. Mobile Comput.*, vol. 5, no. 6, pp. 640–652, Jun. 2006.

TRI GIA NGUYEN (M'17) received the B.Ed. degree from the Hue University of Education, Vietnam, in 2011, the M.Sc. degree from Duy Tan University, Vietnam, in 2013, and the Ph.D. degree from Khon Kaen University, Thailand, in 2017, all in computer science. He is currently a Post-Doctoral Researcher with the Department of Computer Science, Faculty of Science, Khon Kaen University. His research interests include the Internet of Things, computer networks, wireless communications, sensor networks, energy harvesting, mobile computing, parallel and distributed systems, network security, and modeling and analysis.

CHAKCHAI SO-IN (SM'14) received the Ph.D. degree in computer engineering from Washington University in St. Louis, MO, USA, in 2010. He was an Intern at CNAP-NTU (SG), Cisco Systems, WiMAX Forums, and Nokia Bell Labs, USA. He is currently a Professor with the Department of Computer Science, Khon Kaen University. He has authored over 80 publications and 10 books, including some in the IEEE Journal on Selected Areas in Communications, IEEE Magazines, and Computer Network/Network Security Labs. His research interests include mobile computing, wireless/sensor networks, Internet of Things, parallel and distributed systems, and computer networking and security. He is a Senior Member of the ACM. He has served as an Editor at SpringerPlus, PeerJ, and ECTI-CIT and as a Committee Member and a Reviewer for many conferences/journals, including GLOBECOM, ICC, VTC, WCNC, ICNP, ICNC, and PIMRC; and IEEE Transactions, IEEE journals and magazines, and *Computer Networks/Communications*.