Computer Science and Information Systems 17(3):737–758 https://doi.org/10.2298/CSIS200205018M

Energy-Efficient Non-linear K-barrier Coverage in Mobile Sensor Network ⋆

Zijing Ma[1], Shuangjuan Li[1] , Longkun Guo[2], and Guohua Wang[1]

[1] College of Mathematics and Informatics, South China Agricultural University, China
mazijingscau@hotmail.com,lishj2013@hotmail.com,w.guohuascut@gmail.com
[2] School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan, China
forkun@mail.ustc.edu.cn

Abstract. K-barrier coverage is an important coverage model for achieving robust barrier coverage in wireless sensor networks. After initial random sensor deployment, k-barrier coverage can be achieved by moving mobile sensors to form k barriers consisting of k sensor chains crossing the region. In mobile sensor network, it is challenging to reduce the moving distances of mobile sensors to prolong the network lifetime. Existing work mostly focused on forming linear barriers, that is the final positions of sensors are on a straight line, which resulted in large redundant movements. However, the moving cost of sensors can be further reduced if non-linear barriers are allowed, which means that sensors' final positions need not be on a straight line. In this paper, we propose two algorithms of forming non-linear k barriers energy-efficiently. The algorithms use a novel model, called horizontal virtual force model, which considers both the euclidean distance and horizontal angle between two sensors. Then we propose two barrier forming algorithms. To construct a barrier, one algorithm always chooses the mobile sensor chain with the largest horizontal virtual force and then flattens it, called sensor chain algorithm. The other chooses the mobile sensor with the largest horizontal virtual force to construct the barrier, other than the mobile sensor chain, called single sensor algorithm. Simulation results show that the algorithms significantly reduce the movements of mobile sensors compared to a linear k-barrier coverage algorithm. Besides, the sensor chain algorithm outperforms the single sensor algorithm when the sensor density becomes higher.

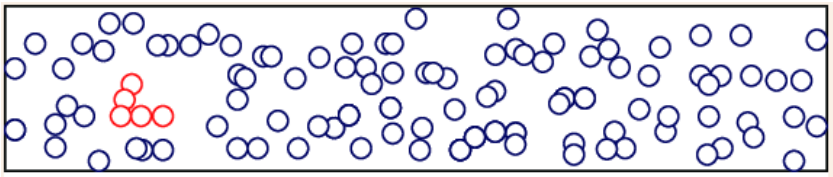
Keywords: wireless sensor networks, k-barrier coverage, virtual force, non-linear barrier.

1. Introduction

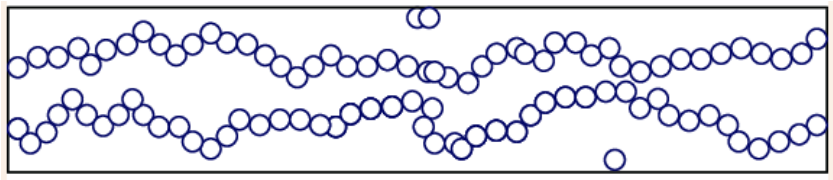
Wireless Sensor Networks(WSNs) have been widely applied in many fields such as intrusion detection, border protection, and environment monitoring. Nowadays, many problems in WSNs have been widely studied such as topology control, localization technology, data aggregation, and coverage problem. Among them, coverage problem is a significant problem in WSNs, which can be classified into many different coverages, including area coverage, target coverage, barrier coverage, and sweep coverage. Barrier coverage is an

⋆ It is an extended version of the ($\mathbf{10}_{th}$ International Symposium on Parallel Architectures, Algorithms and Programming (PAAP 2019))

important problem in these coverage problems. It was first proposed in the work[9] and often used to detect intruders by forming a sensor chain in a belt region of interest(ROI) so that any intruder will be detected when passing through the ROI vertically along any paths. K-barrier coverage is a kind of robust barrier coverage, which guarantees that any intruder crossing the region will be detected by at least k sensors. Initially, sensors are often deployed randomly in the ROI. However, it is not likely to form k-barrier coverage after initial random sensor deployment, as shown in Figure 1(a). With the development of mobile sensor technology, k-barrier coverage can be achieved by moving mobile sensors to the desired positions, as shown in Figure 1(b).



(a) Initial sensor deployment



(b) Final sensor deployment

Fig. 1. Initial sensor deployment and final sensor deployment

However, mobile sensor is equipped with batteries, and it costs much more energy during the sensor movement than the sensing. Thus, it is important to minimize the sensor movements for prolonging the network lifetime while achieving k-barrier coverage. Some algorithms [13, 10] are proposed to form linear barriers using mobile sensors, which means that the sensors move to locate on a line segment spanning the region. Obviously, it results in large redundant sensor movements by mobile sensors to form linear barriers. To further reduce the sensor movements, the work in [1, 16, 12] formed a non-linear barrier energy-efficiently, which means the sensors' final positions are on a curve, other than a straight line, as shown in Figure 1(b). It was showed that the sensor movements can be reduced for forming a non-linear barrier than a linear barrier. However, very few works studied how to form k non-linear barriers energy-efficiently, which is a challenging problem. This paper tries to propose solutions to solve this problem.

The work in [1] proposed an algorithm to form a non-linear barrier coverage energy-efficiently, which outperformed other existing algorithms. However, this algorithm cannot be extended to the case of k-barrier coverage directly. Inspired by the work[1], we propose two energy-efficient algorithms of forming non-linear k barriers based on the initial deployment of mobile sensors. In the work[1], virtual force model has been proposed to pull one sensor chain to touch another sensor chain. This traditional virtual force only con-

siders the euclidean distance of two sensors, which might cause part of barriers formed vertically. Instead, we define the notion of horizontal virtual force by considering the euclidean distance and also horizontal angle. In this paper, we propose two solutions for achieving non-linear k-barrier coverage based on the horizontal virtual force, called sensor chain algorithm and single sensor algorithm respectively. The main idea of the two solutions are to first divide the region into several subregions, and then construct k sensor chains, called sub-barriers, crossing from the left boundary of each subregion to the right boundary respectively, and finally connect the sub-barriers in neighbor subregions for forming k barriers in the whole region. We use two different sub-barrier forming algorithms in these solutions. One algorithm always chooses the mobile sensor chain with the largest horizontal virtual force and then flattens it, called mobile sensor chain movement algorithm. The other algorithm always chooses the mobile sensor with the largest horizontal virtual force to construct the sub-barrier, other than the mobile sensor chain, called single sensor movement algorithm. Simulation results show that our proposed algorithms efficiently decrease the movements of mobile sensors compared to a linear k-barrier coverage algorithm. The simulations also show that the sensor chain algorithm outperforms the single sensor algorithm when the sensor density becomes higher.

In summary, the main contributions of this paper are listed as follows:

- We study the problem of forming non-linear k-barrier coverage using mobile sensors and propose two energy-efficient solutions.
- We introduce the horizontal virtual force model. It considers both euclidean distance and horizontal angle between two sensors, which can avoid forming part of the barriers vertically.
- We first divide the region into several subregions, and then construct k sensor chains, called sub-barriers, crossing from the left boundary of each subregion to the right boundary respectively, and finally connect the sub-barriers in neighbor subregions for forming k barriers in the whole region.
- We propose two algorithms to efficiently form k sub-barriers in each subregion.
- Simulation results demonstrate the efficiency of our proposed algorithms.

The rest of paper is organized as follows. Section 2 reviews some related work about barrier coverage using mobile sensors. Section 3 establishes the networks model and gives some terms about our algorithms. Section 4 describes an algorithm of forming one sub-barrier in a subregion called mobile sensor chain movement algorithm. Section 5 proposes an algorithm of forming one sub-barrier in a subregion called single sensor movement algorithm. In Section 6 we propose the solutions of forming k barriers in the whole region. In Section 7 we evaluate the performance of the algorithms. Section 8 concludes our paper.

2. Related Work

Barrier coverage has been widely studied in wireless sensor networks. The notion of k-barrier coverage was first proposed by the work[9]. In the work [9], two kinds of barrier coverage were proposed: weak barrier coverage and strong barrier coverage. Weak barrier coverage aims at detecting those intruders which cross the ROI along the vertical paths, while the strong barrier coverage aims to detect any intruder crossing the ROI along any

paths. In this paper, we aim to form strong barrier coverage. Existing barrier coverage algorithms can be divided into centralized algorithms and distributed algorithms.

The centralized algorithms are under the assumption that all the information of region and the locations of mobile sensors are known beforehand. A central device can compute the final location of each mobile sensor and then sensors can move directly to their final location. The work in [2] constructed linear k barriers by dividing the region into subregions and forming a baseline grid barrier and an isolation grid barrier in each subregion. However, a large number of redundant sensors are needed for constructing isolation grid barriers. Especially when there are not enough sensors remained in one subregion to fill the isolation grid, the algorithm has to move mobile sensors from other subregions to form the grids, which might cause lots of redundant movements. The work in [7] defined a novel barrier coverage model, called hetebar, and gave a centralized algorithm to find out the maximum lifetime of hetebar after sensors' initial deployment. The work in [13] tried to move the sensors to the grid points while minimizing the maximum sensor movement. The work in [10] proposed a polynomial-time algorithm to move the sensors to cover the barrier line while minimizing the maximum sensor movement, which can reduce the sensor movement. All the above algorithms formed linear barriers, which means the sensors in the barriers must locate on a straight line. The work in [1] proposed an energy-efficient algorithm based on virtual force to form non-linear barrier coverage, which outperformed the linear barrier algorithms in the work[2]. The work in [16] proposed a centralized algorithm to form barriers with mobile sensors under the influence of both sunny and rainy days. The work in [14] studied the hybrid network consisting of stationary sensors and mobile sensors and proposed an algorithm of relocating mobile sensors to improve barrier coverage by filling the gap resulted by stationary sensors. In [19], similar to [14], the authors tried to use mobile sensors and stationary sensors to form barrier coverage. They proposed a two-phase deployment algorithm, where the stationary sensors are first deployed, and then the mobile sensors are deployed to fill the gaps between stationary sensors to form a barrier. Moreover, they proposed a scheme based on probabilistic model to minimize the total sensor cost. The work in [17] achieved barrier coverage in heterogeneous WSNs by leveraging various types of mobile sensors and proposing a greedy movement algorithm to fill the gaps between stationary sensors deployed.

In fact, the performance of centralized algorithms might be limited by the central device if there are a large number of sensors. Hence, some researchers studied the distributed algorithms. Mobile sensor adjusts its location according to its environment and it does not need to know all the information of other sensors. However, these algorithms might cause large redundant movements of mobile sensors in practical applications. The work in [4] proposed an algorithm, which was inspired by the animal aggregations, to solve the problem of establishing barrier coverage between two landmarks. The work in [8] proposed a fully distributed algorithm based on virtual force to relocate the sensors from the original positions to uniformly distribute on the convex hull of the region. The work in [15] presented a distributed algorithm called MobiBar to form linear k-barrier coverage. Furthermore, the authors proved the algorithm terminated in a finite time. The work in [12] proposed two distributed algorithms for forming barriers based on virtual force. However, it was not energy-efficienct. The work in [3] studied the sensors' movement in barrier coverage with a game theory approach. In [18], the authors formed barrier coverage using directional sensors in a line-based model. The algorithm indicated that

after deployed along a predetermined line, the directional sensor can rotate itself based on the information of its adjacent sensors to form barrier coverage.

3. Network Model

Assuming that there is a rectangular belt region of length *L* and width *H*, which is $L \gg H$. A set of mobile sensors is deployed randomly in this region. The sensing range of these sensors is R_s and the number of the sensors is *N*. The initial position of mobile sensor s_i is (x_i, y_i). We assume that each mobile sensor knows its coordinate using GPS system. Each sensor can move in all direction, whose moving distance is the euclidean distance of its initial position and final position.

In this paper, we focus on how to form k-barrier coverage energy-efficiently using mobile sensors. K barriers are formed by k chains of sensors whose sensing range overlap with each other crossing from the left boundary of the region to the right boundary. We study how to find the sensors' final positions so that the sensors can move to form k barriers crossing the region while minimizing the average sensor moving distances.

Before showing the algorithm, we will define some terms below.

Definition 1. ***Mobile Sensor Chain****: A mobile sensor chain is a set of mobile sensors in which the sensing range of each sensor should intersect with that of adjacent sensor in this set, which means the distance between these two mobile sensors is less than or equal to* $2R_s$.

A mobile sensor chain, denoted in red, can be seen in Figure 1(a). Note that a single sensor is the minimum mobile sensor chain. A barrier is formed when there is a mobile sensor chain in which there are two mobile sensors whose sensing ranges intersect the left boundary and the right boundary respectively.

Definition 2. ***Main Mobile Sensor Chain****: A main mobile sensor chain is a kind of mobile sensor chain in which the sensing range of one sensor intersects with the left boundary.*

We form a barrier by constructing a main mobile sensor chain from the left boundary to the right boundary of the region.

Now we'll define the notion of horizontal virtual force by considering the euclidean distance and also horizontal angle.

Let N represent the set of all sensors deployed in the region and N_c represent all the mobile sensors in the main mobile sensor chain. For each mobile sensor $c \in N_c$ and $v \in N - N_c$, we define horizontal virtual force $h(c, v)$ from v to c as follows:

$$h(c,v) = \frac{\alpha}{distance(c,v)} \times cos\theta \tag{1}$$

$$c \in N_c, v \in N - N_c, \theta \in (0, \frac{\pi}{2})$$

In the formula, $distance(c, v)$ is the Euclidean distance between mobile sensor c and mobile sensor v. The direction of $h(c, v)$ starts from v and points to c. θ is the included

angle of the horizontal line and the line of mobile sensor c and v . Note that $\theta \in (0, \frac{\pi}{2})$ and α is a scaling parameter.

For each sensor $c \in N_c$, we can calculate the maximum horizontal virtual force $H\ (c)$ as follows:

$$H(c) = \underset{v}{max} h(c, v) \tag{2}$$

$$c \in N_c, v \in N - N_c, \theta \in (0, \frac{\pi}{2})$$

The action point is defined as the sensor $c_m \in N_c$ which satisfies that $H\ (c_m)$=$\underset{c}{max} H(c)$, $\forall c \in N_c$. The reaction point is defined as the sensor $v_m \in N - N_c$ whose horizontal virtual force to sensor c_m is $H\ (c_m)$.

4. Mobile Sensor Chain Movement Algorithm

In this paper, the main idea of forming k barriers is to first divide the region into several subregions, then construct k sub-barriers from the left boundary of each subregion to the right boundary and finally connect the sub-barriers in adjacent subregions for forming k barriers in the whole region.

In this section, we will show an energy-efficient algorithm of forming a sub-barrier called mobile sensor chain movement algorithm. The whole solution of forming k barriers will be discussed in section 6.

The main idea of mobile sensor chain movement algorithm is to always choose the mobile sensor chain with the largest horizontal virtual force to construct the sub-barrier and then flatten it. This algorithm can be described in three phases.

In the first phase, called the left-fix phase, we will find out the leftmost mobile sensor and then pull it to the left boundary as well as the chain where the mobile sensor belongs to. This chain is regarded as the main mobile sensor chain.

In the second phase, called the extending phase, we will select the rightmost sensor of this main mobile sensor chain and compute the sensor chain with the largest horizontal virtual force towards the rightmost sensor, and pull the mobile sensor chain towards the main mobile sensor chain until the mobile sensor chain touches the rightmost sensor.

In the third phase, called the right-fix phase, when the main mobile sensor chain touches the right boundary of the subregion, we will select the corresponding main mobile sensor chain in the next subregion and move sensors to fill the gap between the chains. If it is the last subregion, then this phase will be ignored. Once the main mobile sensor chain touches its corresponding chain or the right boundary of subregion, it implies that one sub-barrier is formed.

4.1. Flattening Algorithm

Before showing the detail of these three phases, we first present the flattening algorithm.

The flattening algorithm is used when a mobile sensor chain is selected to move to the target which may be the left boundary of the region or a main mobile sensor chain. The main idea of this flattening algorithm is that we first compute the horizontal path of this

mobile sensor chain and then pull the sensors in this chain toward the target one by one by extending the horizontal path.

The detail of this algorithm is described as in Algorithm 1.

Algorithm 1 Flattening Algorithm

Input: main mobile sensor chain *M*, mobile sensor chain *G*
Output: updated main mobile sensor chain *M*

Find out the rightmost sensor of *M*, regard it as an **action point**
Calculate the sensor with the largest horizontal force towards the action point, regard it as a **reaction point**
if the reaction point is on the rightside of action point
  Compute the horizontal path *H* starting from the reaction point.
  for each mobile sensor $c \in H$
    moving sensor = *c*
    The moving sensor *c* moves towards action point along the straight line and touches the action point
    action point = moving mobile sensor
    if the action point intersects with the mobile sensor chain **then**
      return *M*
    end if
    if moving sensor's degree ≥ 2 **then**
      Compute the redundant sensor chain *R* which touches the moving sensor
      for each mobile sensor $d \in R$
        moving sensor = *d*
        The moving sensor *d* moves towards action point along the straight line and touches the action point.
        action point = moving sensor
        if the action point intersects with the mobile sensor chain *G* **then**
          return *M*
        end if
      end for
    end if
  end for
else
  reaction point moves to action point's position
  action point moves towards the right boundary of subregion until it is tangent with the reaction point
return *M*

Let N denote all mobile sensors and N_a denote those mobile sensors in the main mobile sensor chain.

First, we enumerate all the mobile sensors in the main mobile sensor chain and find out the sensor with rightmost X coordinate. We regard this sensor as the action point a. For each mobile sensor $r \in N - N_a$, compute all the horizontal virtual force from a to r and we need to find out the largest horizontal virtual force among them. The sensor with the largest horizontal virtual force is the reaction point. Let P denote the mobile sensor

chain where the reaction point is. We'll extend the main mobile sensor chain by pulling P towards it.

Second, compute the horizontal path of P. we enumerate all possible paths starting from this reaction point by doing a depth-first search and calculate the variance of Y coordinate of these paths and select the path with the lowest variance.

Third, if the reaction point is on the rightside of a, for each sensor c in the horizontal path, move c towards the main mobile sensor chain by extending the horizontal path P. We first compute the degree of the reaction point and then move the reaction point to touch the action point of the main mobile sensor chain. Then the reaction point becomes the new action point. If the reaction point's degree is greater than or equal to 2, it means there is at least one redundant mobile sensor chain R_G connecting the reaction point before it moves. Therefore, moving R_G is preferential and we will move the sensors in R_G towards the action point along the straight line one by one until the new action point intersects with the mobile sensor chain or all the mobile sensors in R_G have been moved. If the new action point intersects with P, the procedure stops; If all mobile sensors in R_G have been moved, then we will continue moving those mobile sensors in the horizontal path P.

However, if the reaction point is on the left of a, we need to insert the reaction point of the mobile sensor chain into the main mobile sensor chain. First we move the reaction point to the position of a. Then, to extend the main mobile sensor chain, we pull a towards the right boundary of the subregion until a is tangent with the reaction point.

To illustrate the algorithm more clearly, we will show two simple examples. Figure 2 shows the example when the reaction point is on the rightside of the action point and figure 3 shows the example when the reaction point is on the left.

Figure 2(a) shows the initial deployment of the main mobile sensor chain and another mobile sensor chain D_G. As is shown in the figure, the rightmost mobile sensor in main mobile sensor chain is selected as the action point and the reaction point in D_G is selected as the moving mobile sensor. By traversing the horizontal path, move the moving mobile sensor towards action point along the straight line until keeping their sensing ranges just intersect at one point, which is tangent. Then this moving sensor will become the new action point and the next mobile sensor in horizontal path will be the new reaction point, which is becoming the moving sensor.

In Figure 2(b), we can find that the degree of moving sensor is 2, which means there is a redundant mobile sensor chain touching the moving sensor, so we need to record this redundant mobile sensor chain, and then move the moving sensor.

In Figure 2(c), since we have recorded this redundant mobile sensor chain, we have to move the redundant mobile sensor chain instead of mobile sensors in the horizontal path. For each mobile sensor in the redundant mobile sensor chain, we will move it towards the action point along the straight line and then it become the new action point until all mobile sensors are added to the main mobile sensor chain.

In Figure 2(d), the redundant mobile sensor chain has been moved to touch the main mobile sensor chain. Note that adding the redundant mobile sensor chain increases the horizontal length of the main sensor chain. Next, we keep moving the mobile sensors in horizontal path until there is another redundant mobile sensor chain or no mobile sensors in the horizontal path.

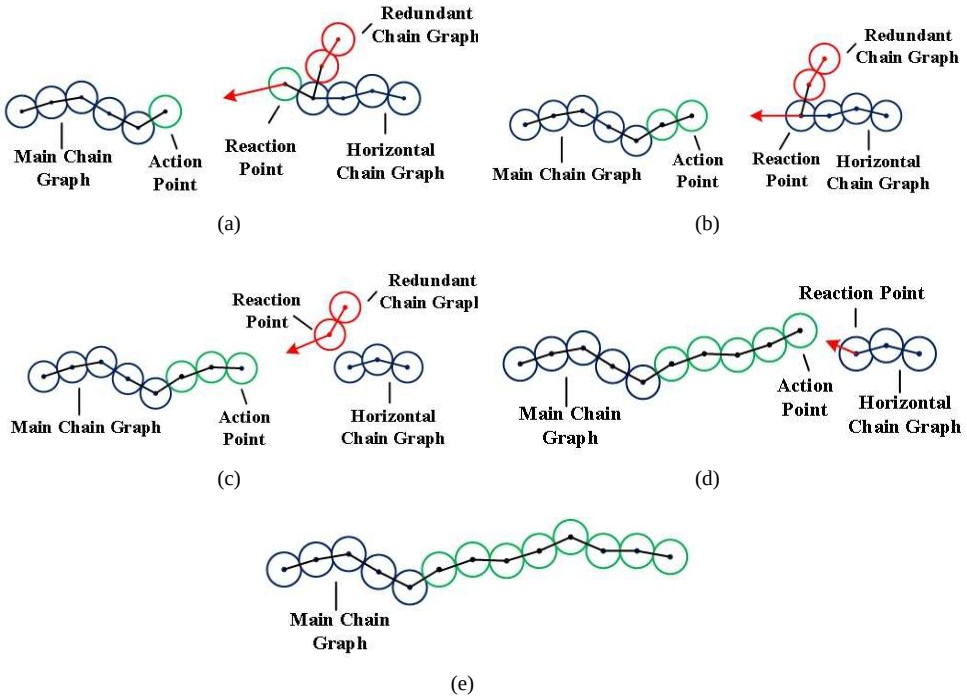


Fig. 2. An example of flattening algorithm when *reaction point* is on the right

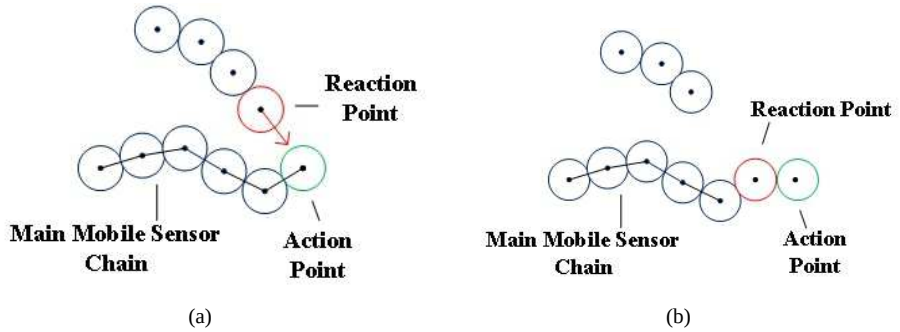


Fig. 3. An example of flattening algorithm when *reaction point* is on the left

Figure 2(e) shows the result of the flattening algorithm. We can see the main mobile sensor chain merges the other mobile sensor chain and extends in horizontal direction.

Figure 3 shows an example when the reaction point is on the left of the action point. As we can see, the reaction point moves to the action point's position, then the action point moves towards the right boundary of subregion and finally, it is tangent with the reaction point. Note that when the reaction point is on the left of the action point, we only move the reaction point to its corresponding position, not the mobile sensor chain where the reaction point is.

4.2. Forming One Sub-barrier

Now we'll show the mobile sensor chain movement algorithm of forming one sub-barrier. The algorithm is divided into three phases: Left-Fix Phase, Extending Phase and Right-Fix Phase.

In the Left-Fix Phase, we will identify the main mobile sensor chain in the subregion. We find out the leftmost mobile sensor chain, which is the closest mobile sensor chain L_G to the left boundary of the subregion. Next, we will move L_G to the left boundary, making the leftmost mobile sensor in L_G tangent with the left boundary. Note that we move L_G using the flattening algorithm, and the action point of the algorithm is a virtual sensor, and its coordinate is $(-R_s, Y_{leftmost})$. The $Y_{leftmost}$ is the Y coordinate of the leftmost mobile sensor in L_G. In the end, we regard L_G as the main mobile sensor chain.

In the Extending Phase, we try to extend the main mobile sensor chain by moving other mobile sensor chains towards it. We will first calculate the reaction point and the action point, where the reaction point is the sensor with the largest horizontal virtual force. Move the mobile sensor chain containing reaction point to the main mobile sensor chain using the flattening algorithm. We continue iterating this phase until the main mobile sensor chain touches the right boundary of subregion or no redundant sensor can be selected. When the phase stops, we come to the right-fix phase.

In the Right-Fix phase, since the main mobile sensor chain has touched the right boundary of subregion, we have to connect it to another main mobile sensor chain in the next subregion to form a barrier.

5. Single Sensor Movement Algorithm

In section 4, we proposed a mobile sensor chain movement algorithm and the flattening algorithm. However, when the length of ROI is not long, flattening algorithm works not well since there is no space to move the mobile sensor chain. Motivated by this, we will introduce a single sensor movement algorithm.

The main idea of this algorithm is that we first divide the ROI into several subregions and in each subregion we select k mobile sensors as the sub-barriers, also called main mobile sensor chain. Then we extend these sub-barriers by moving the sensors with the largest horizontal virtual force towards the sensors in the main mobile sensor chains until the main mobile sensor chain touches another main mobile sensor chain in the next subregion or the right boundary of the ROI.

5.1. Extending Chain Algorithm

In this subsection, we propose an algorithm of forming one sub-barrier. This algorithm can be divided into three phases: Left-Fix Phase, Extending Phase, and Right-Fix Phase.

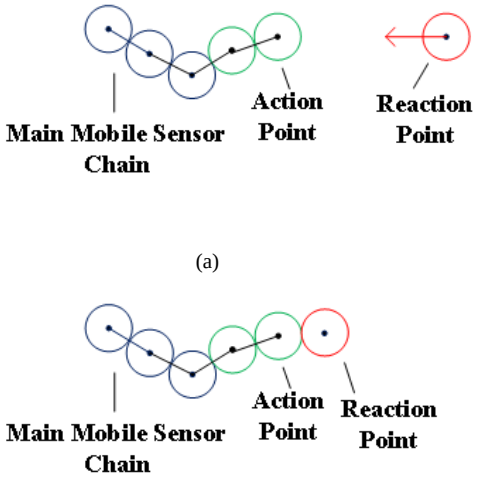


Fig. 4. An example of extending chain algorithm when *reaction point* is on the right

In the Left-Fixed phase, in each subregion, at first, we find out k sensors which are the closest to the left boundary of subregion by calculating the distances between sensors in the subregion and the left boundary of the subregion. Then move these sensors to touch the left boundary of subregion. The final positions of these k sensors should be $(i_{subregion} * L_{subregion} + R_s, Y_{sensor})$, where $i_{subregion}$ is the number of region, starting from 0. $L_{subregion}$ is the length of subregion, Y_{sensor} is the sensors' Y coordinate. Note that we only move these k sensors, which are regarded as the main mobile sensor chain.

In the Extending Phase, we extend the main mobile sensor chain. First, we calculate all the horizontal virtual forces between the sensors in the main mobile sensor chain and other sensors in this subregion, and choose the sensor with the largest horizontal virtual force. Then find out the action point and reaction point. If the reaction point is on the right side of the action point, we just move the reaction point to touch the action point along the line segment between them. But if the reaction point is on the leftside of the action point, we need to move the reaction point to the action point's position, and then, for each sensor on the right side of the action point on the chain, except the rightmost sensor(the last sensor of the chain), it moves to touch its right neighbor sensor. Note that the rightmost sensor does not have the right neighbor sensor, so move it towards the right boundary of subregion.

In the Right-Fixed phase, it is the same as mobile sensor chain movement algorithm.

(a) (b)

Fig. 5. An example of extending chain algorithm when *reaction point* is on the left

Algorithm 2 Extending Chain Algorithm

Input: mobile sensor set *S*, main mobile sensor chain *M*
Output: updated main mobile sensor chain *M*

for each sensor $c \in M$
  calculate the horizontal virtual force between *c* and other sensors which are not in all the main mobile sensor chain
  select the sensor with the largest horizontal force, as the reaction point of *c*
end for
find out the action point whose horizontal virutal force between action point and reaction point is the largest
if the reaction point is on the rightside of action point
  moving sensor = reaction point
  The moving sensor moves towards action point along the straight line and touches the action point
else
  insertPosition = *M*.indexOf(action point)
  reaction point moves to action point's position
  for each sensor $c_{insertPosition} \in M$
    if insertPosition >= *M*.size-1
      $c_{insertPosition}$ moves towards the right boundary of subregion while touching its left neighbor sensor
    else
      $c_{insertPosition}$ moves to $c_{insertPosition+1}$ point's position
    insertPosition = insertPosition + 1
  end for
return *M*

The detail of this algorithm is described in Algorithm 2.

Also, to illustrate the algorithm clearly, we present two examples of extending chain algorithm in the extending phase. Figure 4 shows an example of extending chain algorithm when the reaction point is on the right side of the action point. The reaction point moves towards the action point along the line segment between reaction point and the action point until the reaction point touches the action point. Figure 5 shows how the algorithm works when the reaction point is on the leftside of the action point. The reaction point moves to action point's position. Then the action point moves to the position of its right neighbor sensor. Also, this sensors on the rightside of the action point moves to the position of its right neighbor sensor except the rightmost sensor. The rightmost sensor moves towards the right boundary of the subregion while touching its left neighor sensor. Note that the extending chain algorithm is used to extend the length of main mobile sensor chain.

6. Forming K-barrier

In this section, we'll show how to form k barriers. Actually, mobile sensors chain movement algorithm and single sensor movement algorithm in the former two section aim to forming one sub-barrier in a subregion. We will propose two solutions of forming k barriers based on these two algorithms, called sensor chain algorithm and single sensor algorithm.

The main idea is to first divide the region into several subregions, and then use the two algorithms k times in each subregion to form k sub-barriers respectively. After forming k sub-barriers, we will connect the k sub-barriers with the other sub-barriers in the right subregions so that the k barriers is formed.

The detail of the solutions are described as follows:

First, the region is divided into equal-sized subregions whose length are L_r and the width are W, where $L_r = L/n$ and *n* is the number of subregions. Empirical, *n* should not be too large or too small. On one hand, if *n* is too large, there might be not enough sensors to form barriers in the subregion. On the other hand, if *n* is too small, the length of subregion will be longer, which might result in a larger moving distance of mobile sensors.

Second, form k sub-barriers in each region independently by running the two algorithms of forming one sub-barrier k times. The number of sensors constructing one barrier is limited to be N_i/k, avoiding that there is not enough sensor to construct the k_{th} sub-barrier, where N_i is the number of sensors deployed in i_{th} subregion. After completing k sub-barriers in each subregion, it can be observed that these sub-barriers in two adjacent subregions may be not connected, as shown in Figure 6(a).

Third, for simplicity, k sub-barriers in one subregion is called the left k sub-barriers while k sub-barriers in its right neighbor subregion is called the right k sub-barriers. The left or right k sub-barriers are numbered increasingly by their locations on the right or left boundary of their subregion respectively. Each left sub-barrier connects with the same number of the right sub-barrier by pulling the sensors one by one to fill the gap between them. For example, sensors denoted in red are moved to connect sub-barriers in adjacent subregions, as shown in Figure 6.

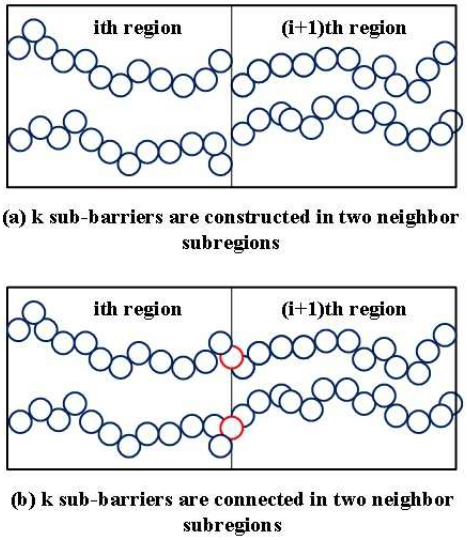


Fig. 6. Connecting two sub-barriers in two adjacent subregion

Finally, k barriers will be formed in the whole region. Note that the solutions will form non-linear barriers in the region, so the shapes of barriers are curves. The time complexity of the solutions are $O(n^2)$.

7. Simulation Results

In this section, we evaluate the performance of our proposed algorithms using Java, called sensor chain algorithm and single sensor algorithm. These two algorithms are compared with the CBIGB algorithm in the work [2]. The CBIGB algorithm constructs k barriers by dividing the region into equal-sized subregions, selecting a baseline based on the distance between sensors and baselines, and forming a baseline grid barrier and an isolation grid barrier using hungarian algorithm in each subregion. The results obtained are the average of running the experiments 100 times.

Figure 7,8,9 and 10 show how the average moving distance of sensors changes as the number of sensors increases. Sensors are deployed in the regions with length 30m, 50m, 100m, and 150m respectively and width 8m. Sensors' sensing radius is 0.5m. The number of sensors is different according to the length of regions. We divide the region into 3 subregions. It can be observed that our algorithms result in less average moving distances of sensors than CBIGB algorithm in all the figures.

Figure 7 shows the performance of algorithms when sensors are deployed in a 30m$\times$8m region. As the number of sensors increases, our proposed algorithms always obtain a smaller average moving distance than CBIGB algorithm. Meanwhile, the single sensor algorithm performs better than sensor chain algorithm.

Figure 8 shows the performance of our algorithms when the length of the region is 50m. It can be seen that the result obtained by our algorithms is almost half of that by

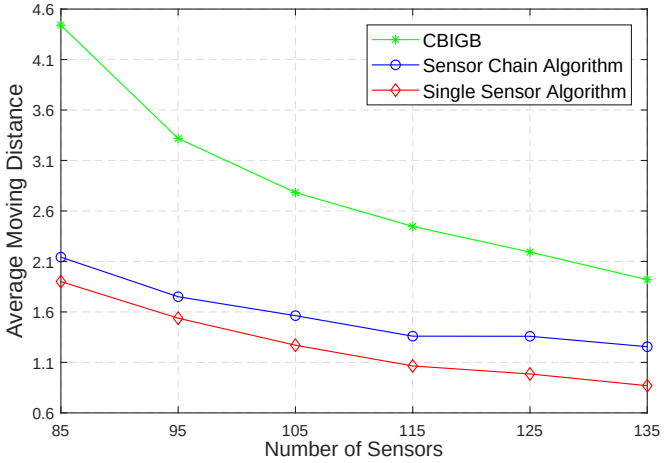


Fig. 7. Average moving distance vs number of sensors in 30m×8m ROI

CBIGB algorithm. At first, the sensor chain algorithm performs worse than single sensor algorithm. However, as the number of sensors increases, the results by the sensor chain algorithm improve sharply. When the number of sensors is 170, they achieve almost the same result.

Figure 9 shows the performance of our algorithms when the length of the region is 100m. At first, the sensor chain algorithm results in a larger moving distance than single sensor algorithm. As the number of sensors increases, the moving distances by these two algorithms both decrease. But the result of the sensor chain algorithm decreases more sharply. When the number of sensors is 275, the results of these two algorithms become the same. As the number of sensors increases, the sensor chain algorithm has a smaller result than the second algorithm. It implies in the middle region case, the sensor chain algorithm is more suitable when the number of sensors is large.

In figure 10, sensors are deployed in the large region with length 150m and width 8m. It can be seen that the sensor chain algorithm results in a larger moving distance than single sensor algorithm. However, as the number of sensors increases, the result of the first algorithm decreases sharply and that by the second algorithm is almost the same. Thus, when the number of sensors is small, single sensor algorithm is more suitable.

Figure 7,8,9 and 10 imply that a larger number of sensors leads to less average moving distance. That is because a larger number of sensors means a higher node density, making sensors move less distance. Additionally, our algorithms are more energy-efficient than CBIGB algorithm in different size of regions since our algorithms result less average moving distance. When the number of sensors is small, single sensor algorithm outperforms the sensor chain algorithm. When the number of sensors becomes larger, the sensor chain algorithm is more suitable.

Next, we study the performance of our proposed algorithms and CBIGB algorithm when the sensors' sensing radius varies. Sensors are randomly deployed in a region with length 50m and width 8m. The number of sensors is 220. Figure 11 shows that the average

moving distance decreases as the sensors' sensing radius becomes larger, since sensors with larger sensing radius can cover larger areas and thus other sensors can move less to touch it. It is observed that the result of sensor chain algorithm and single sensor algorithm is almost the same. When the sensing radius is 0.3m, the average moving distance is almost 2m. When the sensing radius is 0.55m, the moving distance is almost 1m. Hence, a larger sensing radius is preferable for reducing the moving distance.

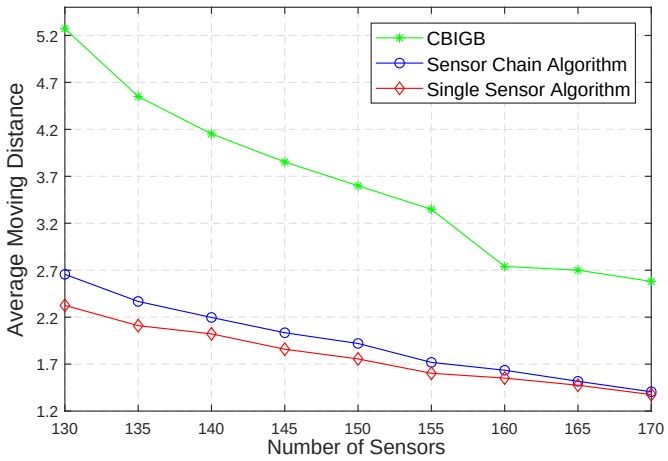


Fig. 8. Average moving distance vs number of sensors in 50m$\times$8m ROI

At last, we will evaluate the performance of three algorithms when the length of region varies. Assuming that the node density is a constant value and then we increase the length of region to simulate large region. Note that the node density can be calculated by N/S, where N is the number of sensors and S is the area of ROI. Therefore, as the length of region increases, the number of sensors also increases. Sensor's sensing radius is set to be 0.5m and the node density varies from 0.375, 0.4, and 0.425.

In figure 12, the length of region varies from 30m to 100m with step 10m and the number of sensors ranges from 90 to 300 with step 30. As the length of region increases, the result by sensor chain algorithm decreases, while that by single sensor algorithm increases. It implies that the sensor chain algorithm outperforms the single algorithm when the length of the region is long. When the length of the region is 60, they share the same result. The curves in figure 13 and 14 are similar to that of figure 12. The sensors' density is 0.4 and 0.425 and the number of sensors varies from 96 to 320 and 102 to 340 respectively. When the length of region is 60m, these two algorithms obtain the same result.

In summary, figure 12,13, and 14 show that the average moving distance obtained by our algorithm is always less than CBIGB algorithm when the length of region increases, which implies that our algorithms are scalable and can be applied to large scale sensor networks. Moreover, single sensor algorithm outperforms the sensor chain algorithm when

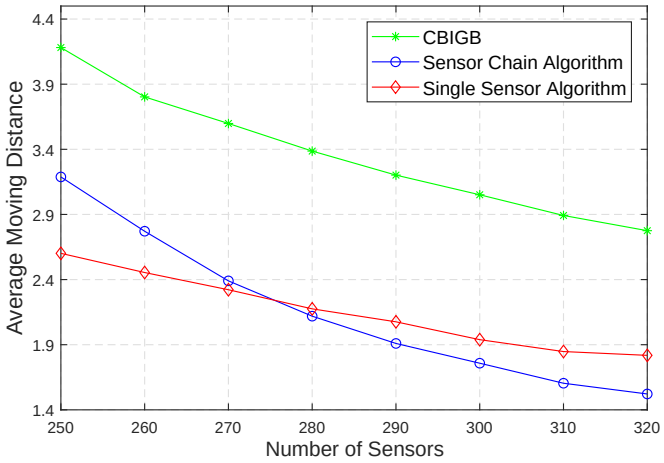


Fig. 9. Average moving distance vs number of sensors in 100m×8m ROI

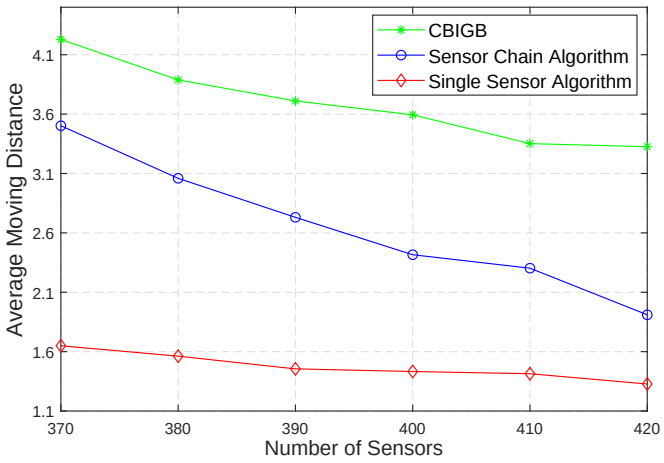


Fig. 10. Average moving distance vs number of sensors in 150m×8m ROI

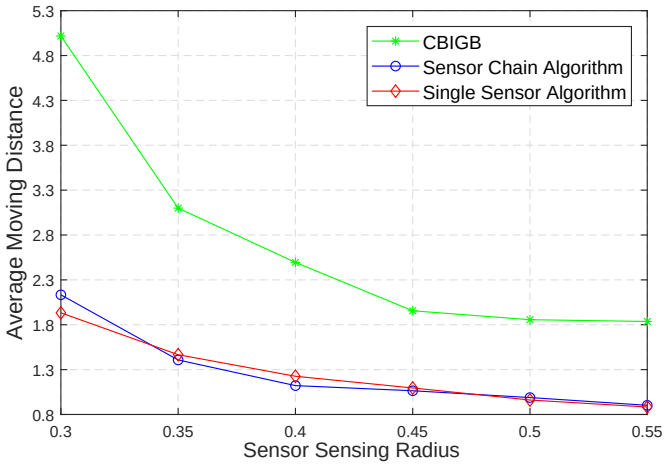


Fig. 11. Average moving distance vs sensor sensing radius in 50m×8m ROI

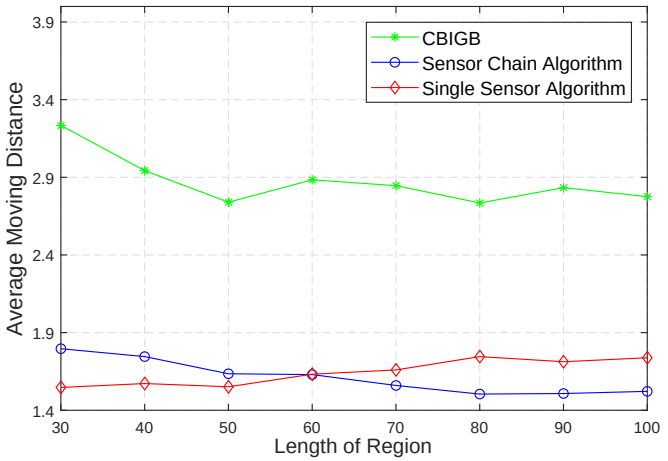


Fig. 12. Average moving distance vs length of region with width 8m and sensors' density 0.375

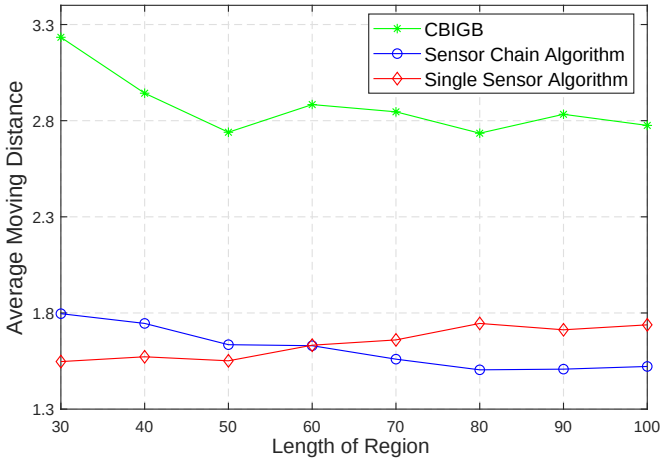


Fig. 13. Average moving distance vs length of region with width 8m and sensors' density 0.4

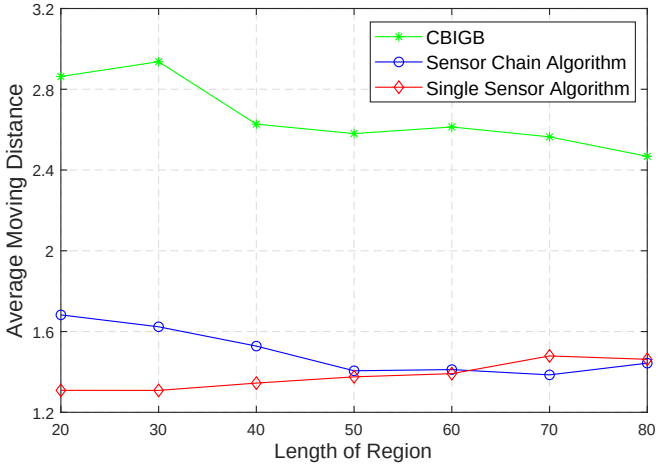


Fig. 14. Average moving distance vs length of region with width 8m and sensors' density 0.425

the length of region is short. However, when the length of region becomes longer, the sensor chain algorithm is more suitable.

Our proposed algorithms outperform CBIGB algorithm in average moving distance due to two reasons. First, CBIGB algorithm forms linear k-barrier coverage by moving sensors to some predetermined baselines, which causes large redundant movements. Second, besides forming sub-barriers in subregions, CBIGB algorithm has to form isolation grid barrier vertically to combine sub-barriers between adjacent subregions. Obviously, it also increases the movements of sensors.

8. Conclusion

In this paper, we propose two algorithms based on horizontal virtual force model, called sensor chain algorithm and single sensor algorithm, to form non-linear k-barrier coverage in mobile sensor networks. Simulation results show that the two proposed algorithms can efficiently reduce the movements of mobile sensors compared to a linear barrier algorithm and can be applied to large scale sensor networks. In the future, we will design a distributed algorithm for achieving non-linear k-barrier coverage using the horizontal virtual force.

Acknowledgments. This work is supported by National Natural Science Foundation of China (Grant No. 61702198). The corresponding author is Shuangjuan Li.

References

1. Baheti, A., Gupta, A.: Non-linear barrier coverage using mobile wireless sensors. In: 2017 IEEE Symposium on Computers and Communications (ISCC). pp. 804–809. IEEE (2017)
2. Ban, D., Jiang, J., Yang, W., Dou, W., Yi, H.: Strong k-barrier coverage with mobile sensors. In: Proceedings of the 6th International Wireless Communications and Mobile Computing Conference. pp. 68–72. ACM (2010)
3. Cheng, C.F., Wu, T.Y., Liao, H.C.: A density-barrier construction algorithm with minimum total movement in mobile wsns. Computer Networks 62, 208–220 (2014)
4. Cheng, T.M., Savkin, A.V.: A problem of decentralized self-deployment for mobile sensor networks: Barrier coverage between landmarks. In: 2009 IEEE International Conference on Control and Automation. pp. 1438–1442. IEEE (2009)
5. Du, J., Wang, K., Liu, H., Guo, D.: Maximizing the lifetime of k-discrete barrier coverage using mobile sensors. IEEE Sensors Journal 13(12), 4690–4701 (2013)
6. He, S., Chen, J., Li, X., Shen, X.S., Sun, Y.: Mobility and intruder prior information improving the barrier coverage of sparse sensor networks. IEEE transactions on mobile computing 13(6), 1268–1282 (2013)
7. Kim, H., Ben-Othman, J.: Heterbar: Construction of heterogeneous reinforced barrier in wireless sensor networks. IEEE Communications Letters 21(8), 1859–1862 (2017)
8. Kong, L., Liu, X., Li, Z., Wu, M.Y.: Automatic barrier coverage formation with mobile sensor networks. In: 2010 IEEE International Conference on Communications. pp. 1–5. IEEE (2010)
9. Kumar, S., Lai, T.H., Arora, A.: Barrier coverage with wireless sensors. In: Proceedings of the 11th annual international conference on Mobile computing and networking. pp. 284–298. ACM (2005)

10. Li, S., Shen, H.: Minimizing the maximum sensor movement for barrier coverage in the plane. In: 2015 IEEE Conference on Computer Communications (INFOCOM). pp. 244–252. IEEE (2015)
11. Qiu, C., Shen, H., Chen, K.: An energy-efficient and distributed cooperation mechanism for¡ inline-formula¿¡ tex-math notation. IEEE Transactions on Mobile Computing (6), 1247–1259 (2018)
12. Rout, M., Roy, R.: Self-deployment of randomly scattered mobile sensors to achieve barrier coverage. IEEE Sensors Journal 16(18), 6819–6820 (2016)
13. Saipulla, A., Liu, B., Xing, G., Fu, X., Wang, J.: Barrier coverage with sensors of limited mobility. In: Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing. pp. 201–210. ACM (2010)
14. Saipulla, A., Westphal, C., Liu, B., Wang, J.: Barrier coverage with line-based deployed mobile sensors. Ad Hoc Networks 11(4), 1381–1391 (2013)
15. Silvestri, S., Goss, K.: Mobibar: An autonomous deployment algorithm for barrier coverage with mobile sensors. Ad Hoc Networks 54, 111–129 (2017)
16. Tian, J., Liang, X., Wang, G.: Deployment and reallocation in mobile survivability-heterogeneous wireless sensor networks for barrier coverage. ad hoc networks 36, 321–331 (2016)
17. Wang, Z., Cao, Q., Qi, H., Chen, H., Wang, Q.: Cost-effective barrier coverage formation in heterogeneous wireless sensor networks. Ad Hoc Networks 64, 65–79 (2017)
18. Wu, Y., Cardei, M.: Distributed algorithms for barrier coverage via sensor rotation in wireless sensor networks. Journal of Combinatorial Optimization 36(1), 230–251 (2018)
19. Zhang, X., Wymore, M.L., Qiao, D.: Cost-efficient barrier coverage with a hybrid sensor network under practical constraints. In: 2017 IEEE International Conference on Communications (ICC). pp. 1–7. IEEE (2017)

Zijing Ma received a Bachelor's degree from South China Agricultural University. He will be a student at Central South University and study for a Master's degree in 2020. His main research interest is wireless sensor networks.

Shuangjuan Li is currently an teacher at South China Agricultural University, China. She received her BS and MS degrees in Wuhan University and her PhD degree in Sun Yat-sen University, China, in 2005, 2007, and 2016, respectively. Her current research interests include efficient algorithm design for wireless sensor network.

Longkun Guo received the B.S. and Ph.D. degrees in Computer Science from University of Science and Technology of China (USTC) in 2005 and 2011, respectively. He was a research associate of the University of Adelaide, and is currently a professor in School of Computer Science, Qilu University of Technology. His major research interest includes efficient algorithm design and computational complexity analysis for optimization problems in high performance computing systems and networks, VLSI etc. He has published more than 60 academic papers in reputable journals/conferences such as Algorithmica, IEEE Transactions on Mobile Computing, IJCAI, SPAA, etc.

Guohua Wang is currently an teaching fellow at South China Agricultural University, China. He received his BS degree in software engineering from South China Normal University and his MS and PhD degrees in computer software and theory from South

China University of Technology, China, in 2011, 2012, and 2016, respectively. His current research interests include computer vision, and pattern recognition.

Received: February 5, 2020; Accepted: July 22, 2020.