IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 18, 2019, accepted October 8, 2019, date of publication October 28, 2019, date of current version November 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2949025

Optimizing the Sensor Movement for Barrier Coverage in a Sink-Based Deployed Mobile Sensor Network

SHUANGJUAN LI[1], HONG SHEN[2,3], QIONG HUANG[1], AND LONGKUN GUO[4,5]

[1]College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China
[2]School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510275, China
[3]School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia
[4]School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, China
[5]Shandong Computer Science Center (National Supercomputer Center in Jinan), Jinan 250101, China

Corresponding author: Hong Shen (hongsh01@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702198, Grant 61402184 and Grant 61872152, in part by the National Key Research and Development Program of China Project under Grant 2017YFB0203201, in part by the Australian Research Council Discovery Project under Grant DP150104871, and in part by Guangdong Program for Special Support of Topnotch Young Professionals under Grant 2015TQ01X796.

ABSTRACT Barrier coverage is an important coverage model for intrusion detection. Clearly energy consumption of sensors is a critical issue to the design of a sensor deployment scheme. In mobile sensor network, it costs the sensors much energy to move. In this paper, we study how to optimize the sensor movement while scheduling the mobile sensors to achieve barrier coverage. Given a line barrier and n sink stations that can supply a required number of mobile sensors, we study how to find the mobile sensors' final positions on the line barrier so that the barrier is covered and the total sensor movement is minimized. We first propose a fast algorithm for determining the nearest sink for the given point on the barrier. We then propose a greedy algorithm and an optimal polynomial-time algorithm for calculating the optimal sensor movement. To obtain an optimal algorithm, we first introduce a notion of the virtual-cluster which represents a subset of sensors covering a specified line segment of the barrier and their sensor movements are minimized. Then we construct a weighted barrier graph with the virtual-clusters modeled as vertexes and the weight of each vertex as the total sensor movements of the virtual-cluster. We also prove that the minimum total sensor movements for achieving barrier coverage is the minimum total weights of the path between the two endpoints of the line barrier in this graph. We also solve this barrier coverage problem for the case when the barrier is a cycle by extending the techniques used for the line barrier. Finally, we demonstrate the effectiveness and efficiency of our algorithms by simulations.

INDEX TERMS Barrier coverage, MinSum, mobile sensors, sink-based deployment.

I. INTRODUCTION

Wireless sensor network has received a great interest in applications such as border surveillance, battlefield surveillance and critical infrastructure security. Barrier coverage is an important coverage model in wireless sensor network, which can provide a sensor-chain barrier for detecting the intruders crossing the boundaries of the surveillance area. Sensors are often randomly dispersed from the airplane. However, it is difficult to achieve barrier coverage only using stationary sensors. Recently, with the development of mobile sensors (e.g., Robomote [1], Packbot [2] and Khepera [3]), it is possible to deploy mobile sensors in practical applications. Mobile sensors can move to the desired positions for constructing a sensor-chain barrier, which can save a lot of stationary sensors. However, the movement of mobile sensors consumes much more energy than sensor's sensing and communication. Most of sensors are equipped with batteries, thus it is a critical problem how to minimize the sensor movement for saving the energy, thus prolonging the network's lifetime.

A widely-used random sensor deployment model assumes that sensors, dispersed from an airplane, are randomly distributed in a deployed area [4]. This deployment model requires that sensors are dispersed by a airplane flying at a low altitude because of sensors' small measurement and light weight; otherwise, sensors may have a very large offset from their target landing points, and some sensors cannot

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenliang Zhang.

communicate with other sensors. In some applications such as the battlefield surveillance and nuclear leakage monitoring, it is impossible that the airplane is flying at a low altitude and thus a new deployment model should be explored. We assume that a sink station and some mobile sensors are packed up as a package. These packages are dispersed by a airplane, which are distributed randomly in a deployed area. After these packages are deployed, the sink stations communicate with their neighbor sink stations, then compute the final positions of mobile sensors and schedule the sensors to move to the desired positions for achieving barrier coverage. Since the mobile sensors may have quite different moving distances, the mobile sensors in a package are equipped with different initial energies. The sensor with more initial energy is sent to a farther final position. This kind of deployment model is called as the sink-based deployment model, which has some advantages. First, sinks can communicate with neighbor sinks easily for their large communication range. Second, sinks can compute the final positions of sensors for their powerful computing ability and storage capacity. Third, fewer mobile sensors are used than the random deployment model.

This model was first studied in the target coverage problem in the work [5]. It assumed that initially all the sensors are located at k sink stations and proposed a polynomial-time approximation scheme to minimize the moving distance of sensors to cover all targets in the surveillance region. However, little work has been studied in the barrier coverage problem based on this deployment model and we are the first to study this problem. Given a line barrier and n sink stations that can supply a required number of sensors, we study the barrier coverage problem which aims to find the mobile sensors' final positions on the line barrier so that the barrier is covered and the total sensor movement is minimized. We find that the barrier coverage problem under the sink-based deployment model can be solved by a polynomial-time algorithm. The challenge of this problem is that the sensors can move to arbitrary point of the line barrier.

The main contributions of this paper are summarized as follows:

1) To the best of our knowledge, we are the first to study the barrier coverage problem under the sink-deployment model. This model is suitable for the scenario that sensors can only be dispersed from a airplane flying at a high altitude.

2) We present a fast algorithm for determining the nearest sink for a given point on the barrier and a greedy algorithm for finding the final positions of sensors to cover the line barrier energy-efficiently.

3) We propose an optimal algorithm for finding the final positions of sensors to cover the line barrier while minimizing the total sensor movements. First, we define a concept of virtual-cluster which represents a subset of sensors covering a specified line segment of the barrier and their sensor movements are minimized. Then we construct a weighted barrier graph with the virtual-clusters modeled as vertexes and the weight of each vertex as the total sensor movements of the virtual-cluster. Two vertexes have an edge if these two corresponding sensor-clusters overlap. We prove that the minimum total sensor movements for achieving barrier coverage is the minimum total weights of the path between the two endpoints of the line barrier in this graph.

4) By extending the techniques used for the line barrier coverage, we also propose a polynomial-time algorithm for the barrier coverage problem when the barrier is a cycle.

5) We conduct simulations to evaluate the performance of our algorithms.

The remainder of this paper is organized as follows. We present the previous work in Section 2. The network model and the problem definition are given in Section 3. We present a fast algorithm for determining the nearest sink for a given point and then propose a greedy algorithm for the line barrier coverage problem in Section 4. An optimal polynomial-time algorithm is also proposed for the line barrier coverage problem in Section 5. We propose an optimal algorithm for the cycle barrier coverage problem in Section 6. Extensive performance evaluations of our algorithms are presented in Section 7. Finally, we conclude this paper in Section 8.

II. PREVIOUS WORK

Barrier coverage in wireless sensor network has been studied for over ten years. Most of the work focuses on stationary sensor network [6]–[11]. Since the mobile sensors can move to the desired positions, fewer sensors are used to achieve barrier coverage. Recently, the researchers turn to study how to deploy the mobile sensor to achieve barrier coverage in mobile sensor network, especially minimizing the sensor movement.

Some work studied the barrier coverage problem in one-dimensional setting. Suppose the barrier is a line segment and all sensors are initially located on the line containing the barrier. Some work studied the minimum maximum sensor movement problem (MinMax). The work [12] first proposed an $O(n^2)$ time algorithm for solving this problem when the sensing range of sensors are uniform. The work [13] improved the time complexity to $O(n \log n)$ for the uniform case, and also proposed an $O(n^2 \log n)$ time algorithm when the sensing ranges of sensors are arbitrary. The work [14] studied the minimum total sensor problem (MinSum). It proposed an $O(n^2)$ time algorithm when the sensing ranges of sensors are uniform and proved the problem is NP-complete by reduction to the 3-partition problem for the general case. The work [15] studied the minimum sensor number problem (MinNum). It proved the problem is NP-hard for the general case and proposed efficient algorithms for the uniform case.

Some work studied the barrier coverage problem in two-dimensional setting. Suppose the barrier is a line segment and all sensors are initially located in a plane containing the barrier. The minimum total sensor problem (MinSum) was studied in [16] and was proved to be NP-hard when the sensing range of sensors are arbitrary and can be solved in $O(n^2)$ time when the sensors can only move vertically to the barrier. The minimum maximum sensor movement problem (MinMax) was studied in [16] and was proved be NP-hard when the sensing range of sensors are arbitrary. This problem can be solved in $O(n \log n)$ time when the sensors can only move vertically to the barrier. The work [17] first studied the MinMax problem when the sensing ranges of sensors are uniform and presented an optimal algorithm. The work [18] studied how to form the maximum number of barriers and also minimize the maximum sensor movement when the number of sensors is given and the positions of the barriers are not known a priori. It proposed a two-phase sensor mobility scheme, which was proved order optimal.

Some work studied the case when the barrier is a circle or simple polygon. The work [19] studied the MinMax problem and proposed an $O(n^{3.5} \log n)$ time algorithm for cycle barriers and proposed an $O(mn^{3.5} \log n)$ time for polygon barriers, where m is the number of edges of the polygon. The work [19] studied the MinSum problem and proposed a PTAS. The MinMax result for polygon barriers was improved by [20], which proposed an $O(n^{2.5} \log n)$ algorithm. It also proposed an $O(n^4)$ time algorithm for the MinSum problem when the sensors are moved from the circle perimeter to a regular n-gon. The work [21] studied how to cover the targets on the line by mobile sensors while minimizing the maximum sensor movement and it was proved that this problem is NP-hard when the sensors are initially on this line and the sensing ranges of sensors are non-uniform.

The work [22] studied the hybrid sensor network composed by stationary sensors and mobile sensors. The problem is how to move mobile sensors to fill the gaps while balancing the energy consumption when the stationary sensors are deployed under the line-based deployment. The work [23] studied how many mobile sensors are needed to form k barriers when the stationary sensors are deployed and proposed an optimal algorithm. The work [24] studied the problem when there are not enough sensors to form a barrier and proposed a dynamic sensor patrolling algorithm. It proved that the total sensor movement during one time slot is minimized. The work [25] studied the dynamic barrier coverage problem by combining an inspection robot and stationary sensors and proposed a heuristic algorithm based on game theory.

III. PRELIMINARIES AND PROBLEM STATEMENT

In this section, we present the network model and the problem formulation.

A. NETWORK MODEL

We assume that the deployed area is a two-dimensional rectangular area with the length L and the width W. A barrier is a line segment starting from $[0, 0]$ to $[L, 0]$ in the deployed area. Suppose n sink stations $SK = \{sk_1, sk_2, \ldots, sk_n\}$ are randomly deployed along the barrier, whose positions are $\{(x_i, y_i)|i = 1, 2, \ldots, n\}$. The sink stations can supply a required number of sensors and send the sensors to locate at one point on the barrier for achieving barrier coverage. Sensors can move in any direction. The movement of sensor s_i, denoted by d_i, is the Euclidean distance of the sensor's initial position p_i and its final position p'_i, that is $d_i = dist(p_i, p'_i)$. The sensing range of the sensors are denoted as r.

We also study the case when the barrier is a cycle. The cycle barrier is located in the deployed area with the center at $[x_0, y_0]$ and the radius R. Let p denote the point on the circle barrier, then it can be represented by a 4-tuple $< R, \theta, x_i, y_i >$, where (R, θ) is polar coordinate of p and the (x_i, y_i) is the rectangular coordinate of p. We choose the point with its polar coordinate $[R, 0]$ as the origin of B denoted as p_0. Suppose n sink stations $SK = \{sk_1, sk_2, \ldots, sk_n\}$ are randomly deployed along the cycle. The sink stations can supply a required number of sensors and send the sensors to locate at one point on the cycle for achieving barrier coverage.

B. PROBLEM DEFINITION

Our barrier coverage problem focuses on how to schedule the sensors to cover the barrier so that the total sensor movement is minimized.

Suppose the barrier is a line segment, we define n-Sink Minimum Sum of Movement for Line Barrier Coverage (L-MSBC) problem as follows:

Definition 1 (n-Sink Minimum Sum of Movement for Line Barrier Coverage Problem, Short for L-MSBC Problem): There are n sink stations $SK = \{sk_1, sk_2, \ldots, sk_n\}$ which send mobile sensors to cover the line barrier. The L-MSBC problem is to schedule the sinks to send the mobile sensors $S = \{s_1, s_2, \ldots, s_m\}$ to locate on the line barrier so that this line barrier is covered by the sensing ranges of the mobile sensors and the total sensor movement $\sum_{i=1}^{m}\{dist(p_i, p'_i)\}$ is minimized.

Suppose the barrier is a cycle, we define n-Sink Minimum Sum of Movement for Cycle Barrier Coverage (C-MSBC) problem as follows:

Definition 2 (n-Sink Minimum Sum of Movement for Cycle Barrier Coverage Problem, Short for C-MSBC Problem): There are n sink stations $SK = \{sk_1, sk_2, \ldots, sk_n\}$ which send mobile sensors to cover the cycle barrier. The C-MSBC problem is to schedule the sinks to send the mobile sensors $S = \{s_1, s_2, \ldots, s_m\}$ to locate on the cycle barrier so that this barrier is covered by the sensing ranges of the mobile sensors and the total sensor movement $\sum_{i=1}^{m}\{dist(p_i, p'_i)\}$ is minimized.

IV. THE FAST ALGORITHM FOR THE L-MSBC PROBLEM

In this section, we first present a fast algorithm for determining the nearest sink for a given point on the barrier and then propose a greedy and fast algorithm for the L-MSBC problem.

A. THE ALGORITHM FOR DETERMINING THE NEAREST SINK

Given a point on the barrier, how to find the nearest sink to send a sensor to locate at this point?

A trivial way is to compute all the distances between this target point and each sink, and then find the nearest sink for

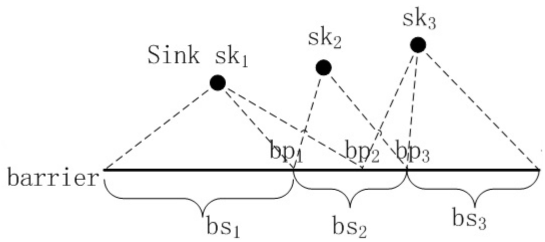

FIGURE 1. Illustration of b_segments.

this point. Obviously, at least $\lceil L/2r \rceil$ sensors are needed to cover the barrier. Thus, it costs $O(nL/2r)$ time to find the nearest sinks for all the sensors covering the barrier. If L is a big number, this method is time consuming. We'll show a fast method to find the nearest sinks.

The main idea of our algorithm is to compute the set of equaldistant points on the line barrier for every pair of sinks, then find a subset of equaldistant points, called boundary-points, such that any point between two consecutive boundary-points in this subset has a same nearest sink, and compute the nearest sink for the subsegment between such two consecutive points. Given a target point, we first identify the subsegment which include it and then return the nearest sink for this subsegment as the nearest sink for this target point.

Before showing the detail of this algorithm, we first introduce some definitions.

Definition 3 (balance-Point): A point on the line barrier is called balance-point, denoted as bp_k, if the distance from sink s_i to it is equal to that from sink s_j to it.

A balance-point is an equaldistant point on the line barrier for a pair of sinks. Let BP denote the set of balance-points. For a pair of sinks s_i and s_j, we compute the balance-point bp_k by drawing the vertical bisector of the line segment $\overline{s_i s_j}$, and bp_k is the intersection point between this vertical bisector and the line barrier. For every pair of sinks, we can compute all the balance-points and sort them increasingly. As shown in Fig.1, bp_1, bp_2, bp_3 are balance-points.

Definition 4 (boundary-Point): A balance-point is called boundary-point, denoted as dp_i, if the two points dp_i-ε and $dp_i + \varepsilon$ have a different nearest sink, where ε is a very small positive number.

Let DP denote the set of boundary-points. We can compute all the boundary-points by binary searching all the balance-points and computing their nearest sink.

Lemma 5: Any point on the subsegment bounded by two consecutive boundary-points has the same nearest sink.

Proof: Suppose there exist two points p_{i_1}, p_{i_2} on the subsegment bounded by two consecutive boundary-points dp_{j_1}, dp_{j_2} has different nearest sink sk_{t_1}, sk_{t_2}.

It is easy to know that there is a balance-point bp_k between p_{i_1} and p_{i_2} which has the same distance to sk_{t_1} and sk_{t_2}. It implies that bp_k is also a boundary-point. It is a contradiction, since dp_{j_1}, dp_{j_2} are two consecutive boundary-points.

Therefore, the lemma is proved. ■

Definition 6 (b_Segment): A line segment is called a b_segment if its endpoints are two consecutive boundary-points.

Let bs_i denote the ith b_segment and $BS = \{bs_1, bs_2, \ldots, bs_\tau\}$ denote the set of b_segments. Let $BE = \{be_i | 1 \leq i \leq \tau\} = \{b_0, b_1\} \cup DP$ denote the set of the endpoints of b_segments, which consists of the endpoints of the line barrier and the set DP. That is $bs_i = \overline{dp_{i-1} dp_i}$.

Definition 7 (Assigned Sink): Sink s_k is said to be assigned sink of b_segment bs_i if s_k is the nearest sink of one point on bs_i.

Let as_i denote the assigned sink of b_segment bs_i and $AS = \{as_1, as_2, \ldots, as_\tau\}$ be the set of these assigned sinks. As shown in Fig.1, $\{bp_1, bp_2, bp_3\}$ is a set of balance-points. Since bp_2 is not a boundary-point, $\{bp_1, bp_3\}$ is the set of boundary-points. Thus, $\{bs_1, bs_2, bs_3\}$ is the set of b_segments and $BE = \{b_0, bp_1, bp_3, b_1\}$, where b_0 and b_1 are the endpoints of the barrier and $\{sk_1, sk_2, sk_3\}$ is the set of assigned sinks.

These b_segments have the following property:

Lemma 8: Any two b_segments have different assigned sinks.

Proof: We'll prove it by contradiction. Suppose there are two b_segments which have the same assigned sink.

Suppose b_segment bs_i and $bs_{(i+1)}$ have a common endpoint be_i. Sink sk_i is the assigned sink of bs_i while sink $sk_{(i+1)}$ is the assigned sink of $bs_{(i+1)}$. We claim that the line segment $sk_{i+1}be_i$ is on the right of $sk_i be_i$. If not, the distance from sink sk_{i+1} to be_{i+1} is larger than that from sink sk_i to be_{i+1}, which is contradiction.

Suppose there is one b_segment bs_{i+1} between these two b_segments bs_i and bs_{i+2}. Suppose bs_i and bs_{i+2} have the same assigned sink sk_i while sk_{i+1} is the assigned sink of bs_{i+1}. As shown in Fig.2(a), we draw a circle c_i with be_i as its center and $be_i sk_i$ as its radius. We also draw another circle c_{i+1} with be_{i+1} as its centre and $be_{i+1}sk_i$ as its radius. By the claim, $sk_{i+1}be_i$ is on the right of $sk_i be_i$. Thus, sk_{i+1} is in the circle c_{i+2}, which implies that $be_{i+1}sk_{i+1}$ is shorter than $be_{i+1}sk_i$. However, $be_{i+1}sk_{i+1}$ is equal to $be_{i+1}sk_i$ since be_{i+1} is the common point of bs_{i+1} and bs_{i+2}, which is a contradiction. Thus, if there is one b_segment between two b_segment, these three b_segments have different assigned sinks.

Suppose there are two b_segments bs_{i+1} and bs_{i+2} between these two b_segments bs_i and bs_{i+3}. As shown in Fig.2(b), bs_i and bs_{i+3} have the same assigned sink sk_i, while sk_{i+1} and sk_{i+2} are the assigned sinks of bs_{i+1} and bs_{i+2} respectively. We draw a circle c_i with be_i as its centre and $be_i sk_i$ as its radius. We also draw another circle c_{i+1} with be_{i+2} as its centre and $be_{i+2}sk_i$ as its radius. By the claim, $sk_i be_{i+2}$ is on the right of $sk_{i+2}be_{i+2}$. Thus, sk_{i+2} is in the circle c_i, which implies $be_i sk_{i+2}$ is shorter than $be_i sk_i$. Since sk_i is the assigned sink of bs_i, $be_i sk_{i+2}$ is not shorter than $be_i sk_i$, which is a contradiction. Thus, if there are two b_segments between two b_segments, these four b_segments have different assigned sinks.

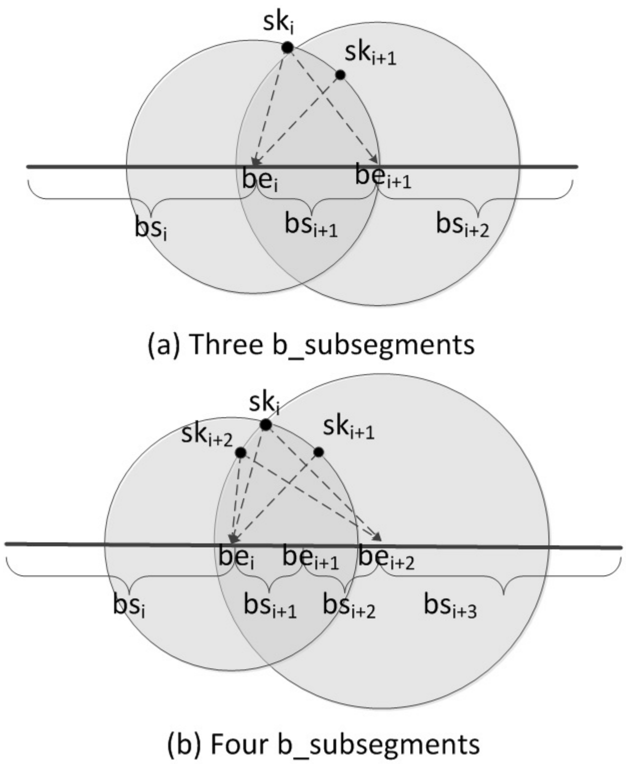

FIGURE 2. Illustration of lemma 8.

Suppose there are $m(m > 2)$ b_segments between these two b_segments. Similarly, we can prove that these two b_segments have different assigned sinks using the above method.

Therefore, the lemma is proved. ■

Next, we propose an algorithm to find all the b_segments and their assigned sinks by computing the boundary-points.

First, compute all the balance-points and sort them increasingly, denoted as $bp_1, bp_2 \ldots bp_m$; Let bp_0, bp_{m+1} denote the endpoints of the line barrier; Second, let $i = 0, j = 0$ and $be_i = bp_j$, compute the nearest sink of the point $bp_j + \varepsilon$ as sk, where ε is a small positive number; Third, use binary search technique to find the largest balance-point bp_l whose nearest sink is also sk. Let $i = i+1$ and $be_i = bp_l$. Next, if l is $m+1$, the algorithm stops; Otherwise, compute the nearest sink of the point $be_i + \varepsilon$ as sk. Go to the third step. The pseudocode of the algorithm is presented in Algorithm 1. It costs $O(n^2 \log n)$ time.

Given one point of the line barrier $(x_p, 0)$, we propose a method to find its nearest sink by binary searching the set BE and finding the index i such that $be_{i-1} < x \leq be_i$. The nearest sink of this point is as_i. The detail of this algorithm is in Algorithm 2. It costs $O(\log n)$ time.

B. GREEDY ALGORITHM FOR THE L-MSBC PROBLEM

Given the locations of sinks and a line barrier, the L-MSBC problem is how to calculate the final locations of sensors sent by sinks such that the line barrier is covered and the total sensor movements are minimized. We propose a greedy and fast algorithm for the L-MSBC problem. In the greedy algorithm, we send sensors to cover the grid points on the line barrier $G = \{r, 3r, 5r, \ldots, \}$. We always choose the closest sink to send the sensor to locate at each grid points until all the grid points are covered. This algorithm runs in $max\{O(L\log n/2r), O(n^2 \log n)\}$ time.

Algorithm 1 Algorithm for Computing the Set of b_Segments

INPUT: $SK = \{sk_1, sk_2, \ldots, sk_n\}, L$
OUTPUT: $BE = \{be_0, be_1, \ldots, be_\tau\}$, $AS = \{as_1, as_2, \ldots, as_\tau\}$
$bp_0 \leftarrow 0$, $k \leftarrow 1$, $AS \leftarrow \phi$;
for each sk_i in SK
  for each sk_j $(i < j)$ in SK
    calculate bp_k;
    if $bp_k < L$
      $k++$;
    endif
  endfor
endfor
$bp_k \leftarrow L$;
sort $bp_0, bp_1, \ldots, bp_k$ increasingly;
$i \leftarrow 0$, $be_0 \leftarrow bp_0$, $mid \leftarrow 0$, $BE \leftarrow BE \cup \{be_0\}$;
while $be_i != L$
  $l \leftarrow mid + 1$; $r \leftarrow k$;
  find the nearest sink sk_1 for the point $be_i + 0.001$;
  while $l \leq r$
    $mid \leftarrow (l + r)/2$;
    find the nearest sink sk_2 for the point $bp_{mid} + 0.001$;
    if $sk_1 \neq sk_2$
      $r \leftarrow mid - 1$;
    else
      $l \leftarrow mid + 1$;
    endif
  endwhile
  $be_{i+1} \leftarrow bp_{mid}$;
  $BE \leftarrow BE \cup \{be_{i+1}\}$;
  $AS \leftarrow AS \cup \{sk\}$;
  $i \leftarrow i + 1$;
endwhile
return BE, AS;

V. THE OPTIMAL ALGORITHM FOR THE L-MSBC PROBLEM

In this section, we propose an optimal algorithm for the L-MSBC problem by introducing a weighted barrier graph model.

A. WEIGHTED BARRIER GRAPH

We first give some definitions as follows.

Definition 9 (Projective-Point): A point on the line barrier, denoted as pp_i, is called the projective-point of sink s_i if sink s_i's x-coordinate equals to pp_i's x-coordinate.

Algorithm 2 Algorithm for Finding the Nearest Sink When the Barrier Is a Line Segment

```
INPUT: BE, AS, x_p
OUTPUT: sk
l ← 1; r ← |BE|;
while l ≤ r
  mid ← (l + r) /2;
  if BE_mid > x_p
    r ← mid − 1;
  else
    l ← mid + 1;
  endif
endwhile
return sk_mid;
```

Algorithm 3 Greedy Algorithm for the L-MSBC Problem

```
INPUT: SK = {sk_1, sk_2, ..., sk_n}, L, r
OUTPUT: P, d
compute BE, AS using Algorithm 1;
p ← r, d ← 0;
while p < L + r
  find the nearest sink sk_i to p using Algorithm 2;
  d+ = dist(p, sk_i);
  put p into P;
  p = p + 2r;
endwhile
return P, d;
```

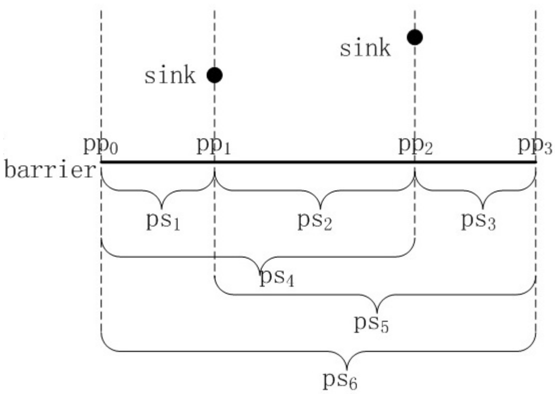

FIGURE 3. Illustration of projective-subsegments.

The line segment between two projective-points is called a projective-subsegment.

Definition 10 (Projective-Subsegment): A line segment on the line barrier, denoted as $ps_i = \overline{pp_ipp_j}$, is called a projective-subsegment if both endpoints are also the projective-points.

There are $O(n^2)$ projective-subsegments. As shown in Fig.3, pp_0, pp_1, pp_2, pp_3 are the projective-points and $ps_1, ps_2, \ldots ps_6$ are the projective-subsegments.

Lemma 11: For a projective-subsegments $ps_k = \overline{pp_ipp_j}$, the minimum number of sensors needed for covering it, denoted as n_k^1, is $\lceil((pp_j - pp_i) - 2r)/2r\rceil$, while the maximum number of sensors needed, denoted as n_k^2, is $\lceil((pp_j - pp_i) + 2r)/2r\rceil$.

Proof: The line segment starting from pp_i to $pp_i + r$, denoted as $\overline{pp_i\,(pp_i + r)}$, can be covered by the sensor located within its left neighbor projective-subsegment. Similarly, $\overline{(pp_j - r)\,pp_j}$ can be covered by the sensor located within the right neighbor projective-subsegment. Thus, the minimum number of sensors needed for covering ps_k is the maximum number of sensors needed to cover the line segment $\overline{(pp_i + r)\,(pp_j - r)}$. Since these sensors are in attaching positions, $\lceil((pp_j - pp_i) - 2r)/2r\rceil$ sensors are needed. Thus, the minimum number of sensors needed is $\lceil((pp_j - pp_i) - 2r)/2r\rceil$.

Suppose the line segment $\overline{(pp_i - r)\,pp_i}$ is covered by a sensor located at ps_k and $\overline{pp_j\,(pp_j + r)}$ is covered by a sensor located at ps_k. Thus, the maximum number of sensors needed is $\lceil((pp_j - pp_i) + 2r)/2r\rceil$. ■

Definition 12: Sensors $s_{i_1}, s_{i_2}, \ldots, s_{i_q}$ are said to be in attaching positions if any pair of neighbor sensors overlap at only one point, that is $\forall 1 \le j < q, x_{i_{j+1}} = x_{i_j} + 2r$ holds.

Lemma 13: In a sensor deployment of minimizing the total sensor movements, the sensors are in attaching positions if their positions are within, not including, two consecutive projective-points.

Proof: Suppose there is a sensor in these sensors which overlaps with its left neighbor sensor at more than one point. If the x-coordinate of this sensor's final position is smaller than that of this sensor's initial position, the sensor's final position can be shifted to the right a bit, which can reduce its movement; otherwise, the final position of its left neighbor sensor can be shifted to the left a bit, which can reduce its movement. It is a contradiction of the optimality.

Suppose there is a sensor in these sensors which overlaps with its right neighbor sensor at more than one point. We can also prove that it is a contradiction of the optimality.

Thus, the lemma is proved. ■

Now we define a notion of sensor-cluster.

Definition 14 (Sensor-Cluster): A set of sensors $\{s_{i_1}, \ldots, s_{i_\tau}\}$ with final positions $\left\{p'_{i_1}, \ldots, p'_{i_\tau}\right\}$ is called a sensor-cluster within $\overline{pp_ipp_j}$ if $pp_i \le p'_{i_1} < \ldots < p'_{i_\tau} \le pp_j$ holds and for any $k\,(1 \le k < \tau)$ which satisfies that $p'_{i_{k+1}} - p'_{i_k} = 2r$ holds, where $n_k^1 \le \tau \le n_k^2$.

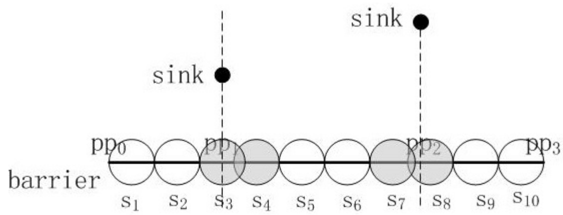

FIGURE 4. Illustration of sensor-clusters.

As shown in Fig.4, there are three sensor-clusters, which are the set of sensors s_1, s_2, s_3, the set of sensors s_4, s_5, s_6, s_7 and the set of sensor s_8, s_9, s_{10}.

Suppose the sensors' initial positions are fixed and their final positions should be within $\overline{pp_ipp_j}$, we can calculate a sensor-cluster such that the sensors' total movements are minimized.

Definition 15 (Virtual-Cluster): A sensor-cluster, consisting of sensors $\{s_{i_1}, \ldots, s_{i_\tau}\}$ with final positions $\{p'_{i_1}, \ldots, p'_{i_\tau}\}$, is called a virtual-cluster, denoted as vc_{ijl}, within $\overline{pp_ipp_j}$ if initial positions of sensors $\{p_{i_1}, \ldots, p_{i_\tau}\}$ are fixed and $\sum_{j=1}^{\tau}\left(p'_{i_j} - p_{i_j}\right)$ is minimized, where $1 \leq l \leq \gamma$.

We'll define the notion of a weighted barrier graph as follows.

Definition 16 (A Weighted Barrier Graph): $G = (V, E, W)$ of a mobile sensor network is constructed as follows: The set V consists of vertices corresponding to the left endpoint of line barrier (s), the virtual-clusters $\left(\{vc_{ijl}|1 \leq i \leq j \leq n, 1 \leq l \leq n\}\right)$ and the right endpoint of line barrier (t). That is, $V = \{s \cup t \cup \{vc_{ijl}|1 \leq i \leq j \leq n, 1 \leq l \leq n\}\}$. The set $E = \{e(v_i, v_j)\}$ consists of edges which are between two vertices if their corresponding sensor-clusters overlap. A vertex has an edge with s or t if the corresponding sensor-cluster covers the left or right endpoint of line barrier. $W : V \rightarrow \mathbb{R}$ is the set of the weights of each vertices, where the weight $w(v_i)$ of vertex v_i is the total movement of sensors in the corresponding sensor-cluster.

Then we have the following lemma.

Lemma 17: Any path from s to t on the weighted barrier graph G means that the barrier can be covered by the corresponding sensor-clusters of the vertices on the path.

Proof: By the definition of a weighted barrier graph, two vertices have an edge if their corresponding sensor-clusters overlap. A vertex has an edge with s or t if the corresponding sensor-cluster covers the left or right endpoint of line barrier. Thus, if there is a path from s to t on the weighted barrier graph G, there are sensor-clusters which cover the barrier from left to right. ■

Theorem 18: The minimum total sensor movement needed to form a barrier under the sink-deployment model is exactly the minimum total weight of $s-t$ path on the weighted barrier graph G.

Proof: Suppose the minimum total weight of $s - t$ path on the weighted barrier graph G is larger than the minimum total sensor movement needed to form a barrier. Recall that the path with the minimum total weight is composed of virtual-clusters which cover the whole line barrier.

By lemma 13 and the definition of the sensor-cluster, it implies that the sensor deployment of minimizing the total sensor movements, called the optimal sensor deployment, is composed of sensor-clusters.

Now we can produce a sensor deployment whose total sensor movement is smaller than this optimal sensor deployment. We check the sensor-clusters in the optimal sensor deployment from the left to the right.

If the ith sensor-cluster is not a virtual-cluster, we can replace this sensor-cluster by the corresponding virtual-cluster. If there is a gap between it and its left neighbor virtual-cluster, then these two virtual-clusters can be replaced by the corresponding virtual-cluster; If there is still a gap between this new virtual-cluster and its left neighbor, continue replacing these two virtual-clusters by the corresponding virtual-cluster until there is no gap between it and its left neighbor. Since the virtual-cluster is a sensor-cluster with the minimum total sensor movement, thus the total sensor movement can be decreased by the replacement of virtual-clusters. Thus, a sensor deployment consisting of virtual-clusters is produced, whose total sensor movement is smaller than the optimal sensor deployment, which is a contradiction.

Thus, the Theroem is proved. ■

B. ALGORITHM DESCRIPTION

In this subsection, we'll present an optimal algorithm of the L-MSBC problem.

The main idea of this algorithm is to compute all possible projective-subsegments, then compute the virtual-clusters for each projective-subsegment, and construct a weighted barrier graph with the virtual-clusters as the vertices. The optimal sensor deployment is the vertices on the path of the minimum total weights.

1) COMPUTING THE VIRTUAL-CLUSTERS

In this subsection, we'll show how to compute the virtual-clusters after the projective-subsegments are calculated.

Recall that the virtual-clusters are the optimal deployments of sensors located within the projective-subsegment. The difficulty to compute the virtual-clusters is that the initial positions of sensors deployed are unfixed. We solve this problem by dividing the range of the sensors' final positions into some subranges such that the initial-positions of sensors are fixed. In each subrange, we can compute the optimal sensor deployment by using optimization technique as a virtual-cluster.

First, we calculate the range of sensors' final positions. For a projective-subsegments $ps_k = \overline{pp_ipp_j}$, the final positions of the sensors within it should satisfy the two conditions:

1) Their final positions should be within the projective-subsegment;

2) These sensors should cover the line segment starting from $pp_i + r$ to $pp_j - r$;

By lemma 11, the minimum number of sensors needed for covering the projective-subsegment $ps_k = \overline{pp_ipp_j}$ is n_k^1. We calculate the range of the sensors' final position as follows:

1) Suppose these sensors are in attaching positions. We consider the final position of first sensor, which is called archor-sensor. We can calculate the range of the archor-sensor's final position, called as sensor-range, as follows:

Case 1: Suppose the number of sensors in the virtual-cluster is n_k^1. If $\left(pp_j - pp_i\right)\%2r = 0$, then the sensor-range is $[2r + pp_i, 2r + pp_i]$; Otherwise, the sensor-range is $\left[\left(pp_j - pp_i\right)\%2r + pp_i, 2r + pp_i\right]$.

Case 2: Suppose the number of sensors in the virtual-cluster is $n_k^1 + 1$. If $\left(pp_j - pp_i\right)\%2r = 0$, then the sensor-range is $[pp_i, 2r + pp_i]$; Otherwise, the sensor-range is $\left[pp_i, \left(pp_j - pp_i\right)\%2r + pp_i\right]$.

2) Suppose the first sensor is located the projective-point and the other sensors are in attaching positions. Then we choose the second sensor as the archor-sensor. This case can be calculated as Case 1.

3) Suppose the last sensor is located the projective-point and the other sensors are in attaching positions. Then we choose the first sensor as the archor-sensor. This case can be calculated as Case 1.

Second, we divide the sensor-range into some subranges, called as sensor-subrange, such that the initial-positions of sensors are fixed when the archor-sensor is located in this subrange.

The sensor-subranges are calculated as follows:

1) check out all the boundary-points within this subsegment;

2) calculate the anchor-sensor's final position, called change-point, such that there is a sensor in this sensor-cluster exactly located at one boundary-point;

3) these change-points divide the sensor-range into sensor-subranges.

Suppose sensor-range is $[a, b]$ and the boundary-point is bp_i. Then the change-point cp_j can be calculated as $(bp_i - a)\%2r + a$. If cp_j is larger than b, this change-point is not within the sensor-range and ignored.

Third, we compute the optimal sensor deployment when the archor-sensor is located with a sensor-subrange.

Suppose the sensors for covering $ps_k = \overline{pp_i pp_j}$ are $s_1, s_2, \ldots s_m$. The archor-sensor is located in one sensor-subrange $[a_1, b_1]$ with initial position (x_1, y_1) and final position $(x, 0)$. Let y_{p_1}, y_{p_2} be the sensor movement when a sensor is located at the left and right endpoints of for ps_k respectively. Depending on four cases, we calculate the minimum total sensor movement as follows:

(1) When the number of sensors is $m = n_k^1$, the minimum total sensor movement M_1 is calculated as follows:

If $\left(pp_j - pp_i\right)\%2r = 0$,

$M_1 \leftarrow \sum_{i=1}^{m} \sqrt{(x_1 - (pp_i + 2r) - (i-1)*2r)^2 + y_1^2}$

else

$M_1 \leftarrow Minimize \sum_{i=1}^{m} \sqrt{(x_1 - x - (i-1)*2r)^2 + y_1^2}$

s.t. $x \in [a_1, b_1] \subseteq \left[\left(pp_j - pp_i\right)\%2r + pp_i, 2r + pp_i\right]$

Then optimization method, such as interpolation method, can be adopted to find the minimum total sensor movement value.

(2) When the number of sensors is $m = n_k^1 + 1$ and a sensor is located at the left endpoint of the projective-subsegment, the minimum total sensor movement $M_2 = M_1 + y_{p_1}$.

(3) When the number of sensors is $m = n_k^1 + 1$ and there is no projective-sensor, the minimum total sensor movement M_3 is calculated as follows:

If $\left(pp_j - pp_i\right)\%2r = 0$,

$M_3 \leftarrow \sum_{i=1}^{m} \sqrt{(x_1 - (pp_i + r) - (i-1)*2r)^2 + y_1^2}$

else

$M_3 \leftarrow minimize \sum_{i=1}^{m} \sqrt{(x_1 - x - (i-1)*2r)^2 + y_1^2}$

s.t. $x \in [a_1, b_1] \subseteq \left[pp_i, \left(pp_j - pp_i\right)\%2r + pp_i\right]$

Then optimization method, such as interpolation method, can be adopted to find the minimum total sensor movement value.

(4) When the number of sensors is $m = n_k^1 + 1$ and a sensor is located at the right endpoint of the projective-subsegment, the minimum total sensor movement $M_4 = M_1 + y_{p_2}$.

Finally, we'll compute all the virtual-clusters.

The virtual-cluster is denoted by $< M, l, r, indx >$. Let M denote the minimum total sensor movement of this cluster, while l(resp. r) is represented as the leftmost(resp. rightmost) covering point of this cluster. Let $indx$ denote the index of the projective-subsegment. The left endpoint of the line barrier is also regarded as a virtual-cluster denoted by $< 0, 0, 0, 0 >$, while the right endpoint of the line barrier is also regarded as a virtual-cluster denoted by $< 0, L, L, size(PS) + 1 >$. Note that $size(PS)$ is the number of the projective-subsegments.

There are $O(n^2)$ projective-subsegments. Besides, there are $O(n)$ sensor-subranges for each projective-subsegment. Thus, there are $O(n^3)$ virtual-clusters and it costs $O(n^3 C)$ time to calculate all virtual-clusters, where C is the time consumed by the optimization method adopted to find the minimum total sensor movement value.

2) CONSTRUCTING WEIGHTED BARRIER GRAPH

Now we'll show how to construct the weighted barrier graph.

We construct a weighted barrier graph $G = (V, E)$. Each node in V represents a virtual-cluster and its weight is the virtual-cluster's M value. Let s and t be the virtual-clusters of the left and right endpoint of the line barrier respectively. The weight of each edge is set as follows:

Initially, the weight of each edge is set to be infinity.

For two nodes $vc_i = \{M_i, l_i, r_i, indx_i\}$ and $vc_j = \left\{M_j, l_j, r_j, indx_j\right\}$, the weights of the edges $e(i,j)$ and $e(j,i)$ are calculated as follows:

1) If $indx_i \neq indx_j$, $r_j \geq r_i$, $r_i \geq l_j$ and $l_j > l_i$ hold, $e(i,j)$ is set to be M_j;

2) if $indx_i \neq indx_j$, $r_j < r_i$, $r_i \leq l_j$ and $l_j < l_i$ hold, $e(j,i)$ is set to be M_i;

Our optimization problem is converted into finding a path with the minimum total weights from s to t in G. We use Djikstra algorithm to find the minimum total weight path, which represents the minimum total sensor movement. The Djikstra algorithm costs $O(E + V\lg V)$ time.

Theorem 19: The L-MSBC problem can be solved in $O(n^3 lgn)$ time.

Proof: By Theorem 18, Algorithm 4 can find the minimum total sensor movements for the L-MSBC problem.

There are $O(n^3)$ virtual-clusters, thus there are $O(n^3)$ vertices. There are $O(n^3)$ edges, since only two vertices belonging to the neighbor projective-subsegments have an edge.

Thus, Algorithm 4 runs in $O(n^3 \lg n)$ time.

Therefore, the Theorem is proved. ■

Algorithm 4 The Optimal Algorithm for the L-MSBC Problem

```
Input: SK = {sk_1, ... sk_n}, L
Output: M // the total sensor movement
Generate all possible projective-subsegments;
Generate all virtual-clusters;
Generate the weighted barrier graph G = (V, E).
Run djikstra algorithm on this graph and find a path with the minimum total weights from s to t in G.
return the minimum total weights of the path as M;
```

VI. THE OPTIMAL ALGORITHM FOR THE C-MSBC PROBLEM

In this section, we study the C-MSBC problem for achieving barrier coverage when the barrier is a cycle by extending the technique used for line barrier coverage.

A. OVERVIEW

Although the cycle barrier coverage problem is as similar as the line barrier coverage problem, the solution for line barrier coverage cannot be directly applied to solve the cycle barrier coverage case. There are some challenging issues. First, how to determine the starting point and ending point of the cycle barrier as the line barrier? Second, even though these two points are determined, there exists a sensor in the sensor deployment which may cover both the starting point and ending point. How to solve this special case? Third, we cannot sort the points on the circle barrier by their x-coordinates like the line barrier.

We'll study these issues and try to extend the technique used for line barrier coverage to solve the circle barrier coverage problem. The basic idea is to introduce weighted barrier graph and find the minimum weight cycle on the weighted barrier graph to obtain the minimum total sensor movement needed to form a cycle barrier.

B. DETERMINING THE NEAREST SINK

In this subsection, we propose a method to determine the nearest sink for one point of the cycle barrier.

We adopt the algorithm from the line barrier coverage, which is to first compute the boundary-points and then find all the b_segments and the assigned sinks. However, there are two differences between the line barrier coverage and cycle barrier coverage. First, there may be two boundary-points for each pair of sinks, thus the algorithm for the line barrier can not be directedly applied to the cycle barrier coverage. It implies that two b_segments may have the same assigned sinks. Second, there may exist a b_segment which crosses the starting point and ending point of the cycle barrier.

To handle these two challenges, we propose an algorithm.

We choose the point with its polar coordinate $[R, 0]$ as the origin of B denoted as bp_0 and $bp_{\mu+1}$. Then calculate all the boundary-points, and then sort these boundary-points by their polar angles, denoted by $bp_1, \ldots bp_\mu$. Our algorithm considers the set of boundary-points $BP = \{bp_0, bp_1, \ldots bp_\mu, bp_{\mu+1}\}$. For simplicity, $bp_i = \theta_i$, where θ_i is the polar angle of the point bp_i. The basic idea is to compute all b_segments satisfying that the neighbor b_segments share different assigned sinks. Let bs_i denote the ith b_segment and $BS = \{bs_1, bs_2, \ldots, bs_\tau\}$ denote the set of b_segments. Let $BE = \{be_0, be_1, \ldots be_\tau\}$ denote the set of the b_segments' endpoints. That is $bs_i = \widehat{be_{i-1}be_i}$. The procedures of the algorithm are described as follows:

1) Initialize BE as an empty set. Let $i = 0, j = 0$ and $be_0 = bp_0$.

2) Calculate the assigned sink sk which is the nearest sink of the point $be_i + 0.001$.

3) Let $j = j + 1$. Check the boundary-point $bp_j + 0.001$ whether its nearest sink is also sk. If yes, go to 3); Otherwise, let $i = i + 1$ and $be_i = bp_j$.

4) If j is $\eta + 1$, stop; otherwise, go to 2).

The pseudocode of the algorithm is presented in Algorithm 5. It costs $O(n^3)$ time.

Algorithm 5 Algorithm of Finding the Nearest Sink When the Barrier Is a Cycle

```
INPUT: SK = {sk_1, sk_2, ..., sk_n}
OUTPUT: BE = {be_0, be_1, ..., be_τ}, AS = {as_1, as_2, ..., as_τ}
bp_0 ← 0, k ← 1, BE ← φ, AS ← φ;
for each sink sk_i
  for each sink sk_j (i < j)
    calculate bp_k;
    k++;
  endfor
endfor
bp_k ← 2π;
sort bp_0, bp_1, ..., bp_k increasingly by their angles;
i ← 0, j ← 0, be_0 ← bp_0; BE ← BE ∪ {be_0};
calculate the nearest sink sk_1 of the point be_0 + 0.001.
while bp_j != bp_k
  calculate the nearest sink sk_2 of the point bp_j + 0.001.
  if sk_1 ≠ sk_2
    BE ← BE ∪ {bp_j};
    AS ← AS ∪ {sk_1};
    i++;
    sk_1 ← sk_2;
  endif
  j ← j + 1;
endwhile
return BE, AS;
```

C. CALCULATING PROJECTIVE-SUBSEGMENTS

In this subsection, we present a method to divide the cycle barrier into segments called projective-subsegments.

We first calculate the projective-points. Note that for each sink, there are two projective-points and we only choose

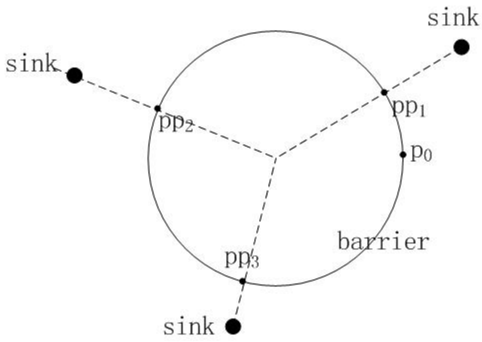

FIGURE 5. Illustration of projective-points on the cycle barrier.

the nearer projective-point. Let $PP = \{pp_1, \ldots pp_\tau\}$ denote the set of projective-points. The p_segments are the arcs satisfying that the endpoints of the arc are two projective-points. Notes that for any pair of two projective-points, there may exist two p_segments and one p_segment may cross the starting point and ending point of the cycle barrier, which is named crossed p_segment. As seen in Fig 5, the projective-subsegment $\widehat{pp_3pp_1}$ crosses the point p_0. To handle the crossed p_segments, we add a duplication of the projective-points to the set of projective-points, with the polar angle added by 2π. Sort the projective-points by their polar angles and the set of projective-points is denoted by $PP = \{pp_1, pp_2, \ldots, pp_{2(n-1)}, pp_{2n}\}$ with their polar angles $\{\theta_1, \theta_2, \ldots 2\pi + \theta_{n-1}, 2\pi + \theta_n\}$. A projective-subsegment is denoted by $ps_i = \widehat{pp_jpp_k}$ $(j < k)$ and Let $PS = \{ps_1, ps_2, \ldots\}$ denote the set of p_segments.

Then we enumerate all possible projective-subsegments and there are four types of projective-subsegments. Let pp_i denote the starting endpoint of one projective-subsegment and pp_j be the ending endpoint.

For the first type of projective-subsegment, $pp_i < 2\pi$ and $pp_j < 2\pi$ hold.

For the second type of projective-subsegment, $pp_i \geq 2\pi$ and $pp_j \geq 2\pi$ hold.

For the third type of projective-subsegment, $pp_i < 2\pi$, $pp_j > 2\pi$ and $pp_j - pp_i \leq 2\pi$ hold.

For the fourth type of projective-subsegment, $pp_i < 2\pi$, $pp_j > 2\pi$ and $pp_j - pp_i > 2\pi$ hold.

It is easy to know that the second type and the fourth of projective-subsegment can be transformed into the first type of projective-subsegment. Thus, we only choose the first type and the third type of projective-subsegments and put them into the set PS, which has $O(n^2)$ projective-subsegments.

The procedures of the algorithm are described as follows:

1) Initialize PS and PP as an empty set.

2) For each sink, compute its projective-points. Choose the nearer projective-point and add it to PP. We also add a duplication of this projective-point to PP, with the polar angle added by 2π.

3) Sort these projective-points by their polar angles.

4) For each pair of pp_i and pp_j $(i < j)$, if $pp_i < 2\pi$ and $pp_j - pp_i \leq 2\pi$ hold, then add the p_segment $\overline{pp_jpp_k}$ into PS.

D. DEFINING VIRTUAL-CLUSTER

Now we compute all the virtual-clusters after calculating the projective-subsegments. First, we compute the minimum number of sensors within each projective-subsegment. Similar as the line barrier, we divide the range of sensors' final positions into subranges such that the sensors' initial positions are fixed. We use the polar angle of the archor-sensor' final position instead of its x-coordinate and use $arcsin(r/R)$ instead of r. The sensor-range and the sensor_subrange is calculated as the line barrier.

Suppose the archor-sensor is located in one sensor-subrange $[a_1, b_1]$ with final position (R, θ_{t_1}). Let y_{p_1}, y_{p_2} be the polar radius of the initial position of the sensors which will be located at the left and right endpoints of the projective-subsegment. Let $rr = arcsin(r/R)$. We calculate the ith sensor's initial position and final position. The ith sensor's final position is calculated as $(R, \theta_{t_1} + (i-1) * 2rr)$. That is, the x-coordinate of the ith sensor is $x'_i = Rcos(\theta_{t_1} + (i-1) * 2rr)$ and the y-coordinate is $y'_i = Rsin(\theta_{t_1} + (i-1) * 2rr)$. The ith sensor's initial position is the position of its assigned sink denoted by (x_{t_i}, y_{t_i}). The virtual-cluster is calculated as the line barrier. There are $O(n^3)$ virtual-clusters and it costs $O(n^3C)$ time to calculate all virtual-clusters, where C is the time consumed by the optimization method adopted to find the minimum total sensor movement.

E. CONSTRUCTING WEIGHTED BARRIER GRAPH

We construct a directed weighted barrier graph $G = (V, E)$ and the C-MSBC problem can be converted into finding a cycle with the minimum total weights in G. Now we'll show how to construct the directed weighted barrier graph $G = (V, E)$. Each node in V represents a virtual-cluster and its weight is the virtual-cluster's M value. The weight of the edge of two nodes is not infinity if the corresponding virtual-clusters belongs to different projective-subsegments and they are connected. The weight of the edges are set as follows:

Initially, the weight of all the edges are set to be infinity.

For two nodes $vc_i = \{M_i, l_i, r_i, indx_i\}$ and $vc_j = \{M_j, l_j, r_j, indx_j\}$, the weight of the edges $e(i, j)$ and $e(j, i)$ are calculated as follows:

1) If $indx_i \neq indx_j$, $r_j \geq r_i$, $r_i \geq l_j$ and $l_j > l_i$ hold, $e(i, j)$ is set to be M_i;

2) if $indx_i \neq indx_j$, $r_j < r_i$, $r_i \leq l_j$ and $l_j < l_i$ hold, $e(j, i)$ is set to be M_j;

3) if $indx_i \neq indx_j$, $r_i \geq 2\pi$, $r_j \geq r_i - 2\pi$ and $r_i - 2\pi > l_j$ hold, $e(i, j)$ is set to be M_i;

4) if $indx_i \neq indx_j$, $r_j \geq 2\pi$, $r_i \geq r_j - 2\pi$ and $r_j - 2\pi > l_i$ hold, $e(j, i)$ is set to be M_j;

5) if $r_i - l_i \geq 2\pi$, $e(i, i)$ is set to be M_i;

Our optimization problem is converted into finding a cycle with the minimum total weights in G. We can use the algorithm in [26] to find the minimum weight cycle, which runs in $O(VE)$ time.

Theorem 20: The minimum total sensor movement needed to form a cycle barrier is exactly the total weights of the minimum weight cycle on the weighted barrier graph G.

Proof: Suppose there is an optimal sensor deployment of minimum total sensor movement forming a cycle barrier, whose sensor movement is smaller than that of the minimum weight cycle on G.

It is easy to know that there is a sensor-cluster in the optimal sensor deployment, which is not a virtual-cluster. By using the similar method in Theorem 19, we can also produce a sensor deployment which has a smaller sensor movement than the optimal sensor deployment, which is a contradiction.

Thus, the Theorem is proved. ■

Theorem 21: The C-MSBC problem can be solved in $O(n^6)$ time.

Proof: By Theorem 20, the C-MSBC problem can be transformed to finding the minimum weight cycle on the weighted barrier graph G. We can use the algorithm in [26] to find the minimum weight cycle, which runs in $O(VE)$ time. There are $O(n^3)$ vertices and $O(n^3)$ edges in G. Thus, the C-MSBC problem can be solved by $O(n^6)$ time. ■

VII. EVALUATION

In this section we evaluate the performance of the proposed algorithms by simulation in two different cases.

A. THE LINE BARRIER CASE

We first evaluate the solution for the L-MSBC problem called SMC solution. The sinks are deployed randomly in a belt region with length L and width W. The barrier is a line segment in the belt region starting from [0, 0] to $[L, 0]$. L is set to be 1057 and W is 30. The number of sinks is 5. Each sink can send sensors to cover the barrier. The sensing range of sensors is 22. We mainly focus on the minimum total sensor movement for barrier coverage. This metric is evaluated on the length of the line barrier, the width of the deployed area, the number of sinks and the sensor's sensing range. We compare the SMC algorithm with the greedy algorithm, which run 100 times. The data points are a average of 100 experiments.

Fig 6 shows the effect of the length of line barrier on the minimum total sensor movement. The length of the line barrier varies from 177 to 1200 with step 176. As the length of the line barrier increases, the minimum total sensor movement by two algorithms both increases. The reason is that more sensors are needed to cover the line barrier . We can see that the minimum total sensor movement by SMC is always smaller than the result by the Greedy algorithm, which implies that the SMC algorithm outperforms the Greedy algorithm.

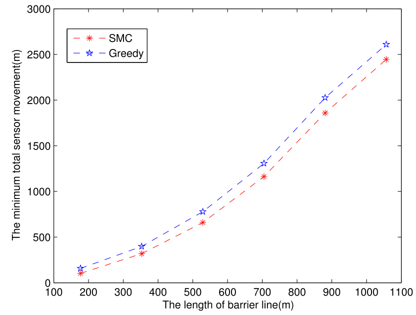

FIGURE 6. The length of the line barrier changes.

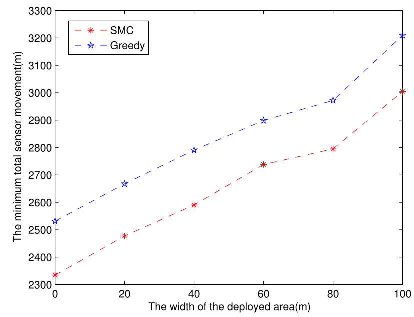

FIGURE 7. The width of the deployed area changes.

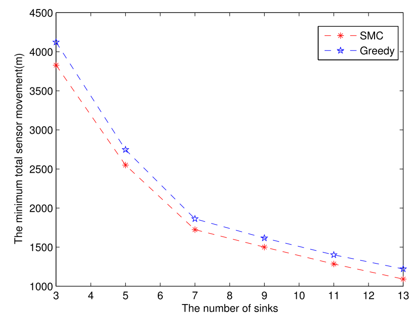

FIGURE 8. The number of sinks changes.

Fig 7 shows the effect of the width of the deployed area on the minimum total sensor movement. The width of the deployed area varies from 0 to 100 with the step 20. With the increasing of the width of the deployed area, the minimum total sensor movement by two algorithms both increases nearly linearly. We can see that the result obtained by SMC algorithm is about 90 percent of that by Greedy algorithm.

Fig 8 shows the effect of the number of sinks on the minimum total sensor movement. The number of sinks varies from 3 to 13 with the step 2. With the increasing of the number of sinks, the minimum total sensor movement by two algorithms both decreases sharply until the number of sinks approaches 7. After passing this value, the result decreases slowly. This demonstrates a tradeoff between the number of sinks and the minimum total sensor movement required to achieve barrier coverage. The result by SMC is always smaller than the result by the Greedy algorithm.

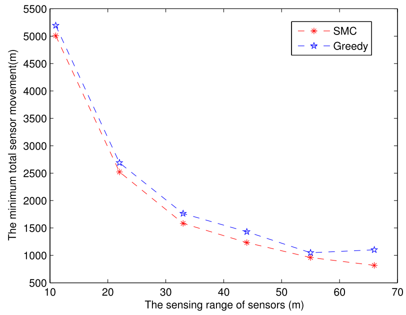

FIGURE 9. The sensing range of sensors changes.

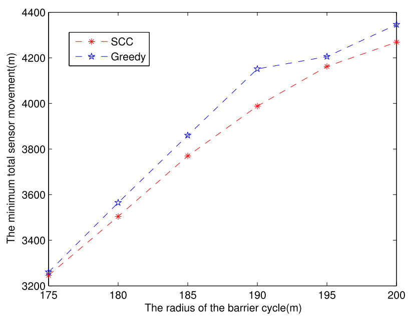

FIGURE 10. The radius of the barrier cycle changes.

Fig 9 shows the effect of the sensing range of sensors on the minimum total sensor movement. The sensing range of sensors varies from 11 to 66 with 11 as the step size. As the sensing range of sensors increases, the minimum total sensor movement decreases. That's because more sensors are needed to achieve barrier coverage when the sensing range increases. The result first decreases sharply, and then decreases slowly. When the sensing range is larger than 55, the effect on the result is not significant.

B. THE CYCLE BARRIER CASE

We evaluate the solution for the C-MSBC problem called SCC solution. The barrier is a cycle with the radius R. The sinks are deployed randomly along the cycle barrier with the maximum offset W. Each sink can send sensors to cover the cycle barrier. R is set to be 190 and W is set to be 22. The sensing range of sensors, denoted as r, is set to be 22. The number of sinks is 3. We mainly focus on the minimum total sensor movement for barrier coverage. This metric is evaluated on the radius of the barrier cycle, the maximum offset of the sinks from the cycle barrier, the number of sinks and the sensor's sensing range. As similar as the line barrier case, we also choose the greedy algorithm for comparison. The greedy algorithm and the SCC algorithm are run 100 times. The data points are a average of 100 experiments.

Fig 10 shows the effect of the radius of barrier cycle on the minimum total sensor movement. The radius of barrier cycle varies from 175 to 200 with step 5. The result by the SCC

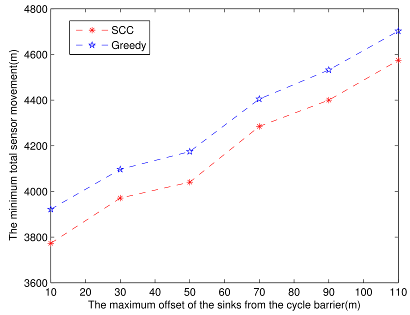

FIGURE 11. The maximum offset of the sinks from the cycle barrier changes.

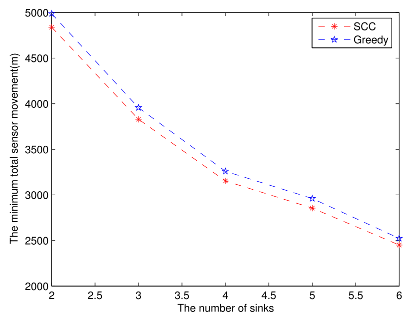

FIGURE 12. The number of sinks changes.

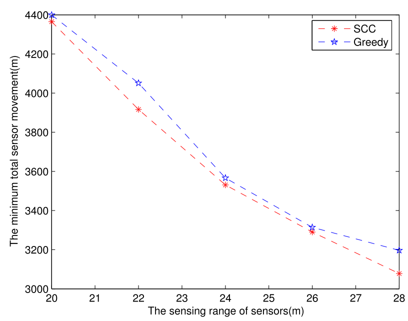

FIGURE 13. The sensor's sensing range changes.

algorithm is smaller than that by the Greedy algorithm, thus SCC algorithm always outperforms the Greedy algorithm. As the radius of barrier cycle increases, the minimum total sensor movement also increases. We find that when the radius of the barrier cycle approaches 190, the gap of the two curves is the largest. Fig 11 shows the effect of the maximum offset of the sinks from the cycle barrier on the minimum total sensor movement. The maximum offset of the sinks from the cycle barrier varies from 10 to 110 with the step 20. The result by the two algorithms both increase as the maximum offset of the sinks from the cycle barrier increases. That's because the sensors need to move a larger distance to cover the barrier cycle.

Fig 12 shows the effect of the number of sinks on the minimum total sensor movement. The number of sinks varies from 2 to 6 with the step 1. The minimum total sensor movement by two algorithms both decreases as the number of sinks increases. The minimum total sensor movement decreases by 50 percent when the number of sinks increases from 2 to 6. Fig 13 shows the effect of the sensing range of sensors on the minimum total sensor movement. The sensing range of sensors varies from 20 to 28 with 2 as the step size. The minimum sensor movements by the two algorithms both decrease as the sensing range of sensors increases. The result decreases by 30 percent when the sensing range of sensors increases from 20 to 28. The gap between these two curves is the largest when the sensing range of sensors is 22.

VIII. CONCLUSION

In this paper, we study the minimum total sensor movement problem for barrier coverage under sink-based deployment and propose some algorithms both for the line barrier and the cycle barrier. To solve the line barrier case, we enumerate all the projective-subsegments and compute all possible optimal sensor deployments for each projective-subsegment as virtual-clusters. Then we construct a weighted barrier graph and find the path with the minimum weight, which is the minimum total sensor deployment for achieving barrier coverage. We also solve the cycle barrier case by extending the techniques used in the line barrier case. In the future, we will study an approximation algorithm for solving the minimum total sensor movement problem for barrier coverage under the assumption that sensors are randomly distributed in the surveillance area, which has been proved to be NP-hard.

REFERENCES

[1] K. Lembke, L. Kietliński, M. Golański, and R. Schoeneich, "RoboMote: Mobile autonomous hardware platform for wireless Ad-hoc sensor networks," in *Proc. IEEE Int. Symp. Ind. Electron.*, Jun. 2011, pp. 940–944.

[2] K. Michael and M. Michael, "The packbots are coming: Boosting security at the 2014 FIFA world cup," *IEEE Consum. Electron. Mag.*, vol. 3, no. 3, pp. 59–61, 2014.

[3] J. M. Soares, I. Navarro, and A. Martinoli, *The Khepera IV Mobile Robot: Performance Evaluation, Sensory Data and Software Toolbox*. Springer, 2016.

[4] B. Liu, O. Dousse, J. Wang, and A. Saipulla, "Strong barrier coverage of wireless sensor networks," in *Proc. 9th Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2008, pp. 411–420.

[5] Z. Chen, X. Gao, F. Wu, and G. Chen, "A PTAS to minimize mobile sensor movement for target coverage problem," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[6] G. Yang and D. Qiao, "Multi-round sensor deployment for guaranteed barrier coverage," in *Proc. INFOCOM*, Mar. 2010, pp. 1–9.

[7] Y. Wang and G. Cao, "Barrier coverage in camera sensor networks," in *Proc. 12th Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2011, p. 12.

[8] H. Ma, M. Yang, D. Li, Y. Hong, and W. Chen, "Minimum camera barrier coverage in wireless camera sensor networks," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 217–225.

[9] J. Li, J. Chen, and T. H. Lai, "Energy-efficient intrusion detection with a barrier of probabilistic sensors," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 118–126.

[10] 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.

[11] A. Saipulla, C. Westphal, B. Liu, and J. Wang, "Barrier coverage of line-based deployed wireless sensor networks," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 127–135.

[12] J. Czyzowicz, E. Kranakis, D. Krizanc, I. Lambadaris, L. Narayanan, J. Opatrny, L. Stacho, J. Urrutia, and M. Yazdani, "On minimizing the maximum sensor movement for barrier coverage of a line segment," in *Ad-Hoc, Mobile Wireless Networks*. Springer, 2009, pp. 194–212.

[13] 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, Jul. 2013.

[14] J. Czyzowicz, E. Kranakis, D. Krizanc, I. Lambadaris, L. Narayanan, J. Opatrny, L. Stacho, J. Urrutia, and M. Yazdani, "On minimizing the sum of sensor movements for barrier coverage of a line segment," in *Ad-Hoc, Mobile Wireless Networks*. Springer, 2010, pp. 29–42.

[15] M. Mehrandish, L. Narayanan, and J. Opatrny, "Minimizing the number of sensors moved on line barriers," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Mar. 2011, pp. 653–658.

[16] S. Dobrev, S. Durocher, M. Eftekhari, K. Georgiou, E. Kranakis, D. Krizanc, L. Narayanan, J. Opatrny, S. Shende, and J. Urrutia, "Complexity of barrier coverage with relocatable sensors in the plane," in *Algorithms Complexity*. Springer, 2013, pp. 170–182.

[17] S. Li and H. Shen, "Minimizing the maximum sensor movement for barrier coverage in the plane," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2015, pp. 244–252.

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

[19] B. Bhattacharya, M. Burmester, Y. Hu, E. Kranakis, Q. Shi, and A. Wiese, "Optimal movement of mobile sensors for barrier coverage of a planar region," *Theor. Comput. Sci.*, vol. 410, no. 52, pp. 5515–5528, 2009.

[20] X. Tan and G. Wu, "New algorithms for barrier coverage with mobile sensors," in *Frontiers in Algorithmics*. Springer, 2010, pp. 327–338.

[21] P. Huang, W. Zhu, and L. Guo, "Optimizing movement for maximizing lifetime of mobile sensors for covering targets on a line," *Sensors*, vol. 19, no. 2, p. 273, Jan. 2019. doi: 10.3390/s19020273.

[22] A. Saipulla, C. Westphal, B. Liu, and J. Wang, "Barrier coverage with line-based deployed mobile sensors," *Ad Hoc Netw.*, vol. 11, no. 4, pp. 1381–1391, 2013.

[23] Z. Wang, J. Liao, Q. Cao, H. Qi, and Z. Wang, "Achieving k-barrier coverage in hybrid directional sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 7, pp. 1443–1455, 2013.

[24] S. He, J. Chen, X. Li, X. Shen, and Y. Sun, "Mobility and intruder prior information improving the barrier coverage of sparse sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 6, pp. 1268–1282, Jun. 2015.

[25] F. Fan, Q. Ji, G. Wu, M. Wang, X. Ye, and Q. Mei, "Dynamic barrier coverage in a wireless sensor network for smart grids," *Sensors*, vol. 19, no. 1, p. 41, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/1/41

[26] J. B. Orlin and A. Sedeno-Noda, "An O(nm) time algorithm for finding the min length directed cycle in a graph," in *Proc. 28th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2017, pp. 1866–1879.

SHUANGJUAN LI received the B.S. and M.S. degrees from Wuhan University, in 2005 and 2007, respectively, and the Ph.D. degree with Sun Yat-sen University, in 2016. She was an Engineer with the Guangzhou Communication Research Institute, from 2007 to 2013. She is currently a Lecturer with South China Agricultural University. Her research interests include wireless sensor networks, ad hoc networks, and optimization algorithm.

HONG SHEN received the B.Eng. degree from the Beijing University of Science and Technology, the M.Eng. degree from the University of Science and Technology of China, and the Ph.Lic. and Ph.D. degrees from Abo Akademi University, Finland, all in computer science. He was a Professor and the Chair of the Computer Networks Laboratory, Japan Advanced Institute of Science and Technology, from 2001 to 2006; he has been a Professor (Chair) of computer science with Griffith University, Australia, where he taught nine years, since 1992. He is currently a specially-appointed Professor with Sun Yat-sen University, China, and a tenured Professor (Chair) of computer science with The University of Adelaide, Australia. He has published more than 300 articles, including more than 100 articles in international journals, such as a variety of the IEEE and ACM Transactions. His main research interests include parallel and distributed computing, algorithms, data mining, privacy preserving computing, high performance networks, and multimedia systems. He was a recipient of many honours and awards.

QIONG HUANG received the B.S. and M.S. degrees from Fudan University, in 2003 and 2006, respectively, and the Ph.D. degree from the City University of Hong Kong, in 2010. He is currently a Professor with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China. He has published more than 100 research articles in international conferences and journals in the area of cryptography and information security, and served as a program committee member in many international conferences. His research interests include cryptography and information security, in particular, and cryptographic protocols design and analysis.

LONGKUN GUO received the B.S. and Ph.D. degrees in computer science from the University of Science and Technology of China, in 2005 and 2011, respectively. He is currently a Full Professor with the School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences). In 2011, he joined Fuzhou University. From 2015 to 2016, he was a Research Associate with The University of Adelaide. After that, he served as an Associate Professor and Head of the Department of Computer Science, College of Mathematics and Computer Science, Fuzhou University. He has authored over 40 academic articles in reputable journals or conferences, such as Algorithmica, the IEEE TMC, IJCAI, IEEE TPDS, and SPAA. His major research interests include efficient algorithm design, and computational complexity analysis for optimization problems in high-performance computing systems and networks.