PAPER

Boundary Node Identification in Three Dimensional Wireless Sensor Networks for Surface Coverage

Linna WEI^{†a)}, Xiaoxiao SONG^{††b)}, Xiao ZHENG^{†c)}, Xuangou WU^{†d)}, *Nonmembers*, and Guan GUI^{†††e)}, *Member*

SUMMARY With the existing of coverage holes, the Quality of Service (such as event response, package delay, and the life time et al.) of a Wireless Sensor Network (WSN) may become weaker. In order to recover the holes, one can locate them by identifying the boundary nodes on their edges. Little effort has been made to distinguish the boundary nodes in a model where wireless sensors are randomly deployed on a three-dimensional surface. In this paper, we propose a distributed method which contains three steps in succession. It first projects the 1-hop neighborhood of a sensor to the plane. Then, it sorts the projected nodes according to their angles and finds out if there exists any ring formed by them. At last, the algorithm validates a circle to confirm that it is a ring surrounding the node. Our solution simulates the behavior of rotating a semicircle plate around a sensor under the guidance of its neighbors. Different from the existing results, our method transforms a three-dimensional problem into a two-dimensional one and maintaining its original topology, and it does not rely on any complex Hamiltonian Cycle finding to test the existence of a circle in the neighborhood of a sensor. Simulation results show our method outperforms others at the correctness and effectiveness in identifying the nodes on the edges of a three-dimensional WSN.

key words: boundary node, distributed, surface coverage, three-dimensional network, coverage problem

1. Introduction

Wireless sensor networks have been widely applied in a great number of areas in human life, such as environment monitoring, industry manufacturing, agricultural irrigation, battlefield surveillance, et al.. They are responsible for collecting information from several Regions of Interest (RoI) and transmitting data to end users for analysis [1], [2]. The accuracy of the data not only depends on the hardware of sensors but also on the topology of networks. A coverage hole, which is a blank area not been covered by any node in the RoI, of a network could vitiate the accumulated data and further misdirect the end users [3]. It may also help to increase the traffic load on the sensors around the holes.

Therefore, it is crucial to eliminate coverage holes to ensure the QoS of networks [4]. As the holes are always surrounded by nodes, we can locate them by finding out the boundary nodes on the edges of them. Thus, it is a key step to firstly identify the boundary nodes before locating the coverage holes.

Generally, there are three types of boundary nodes in a WSN: the nodes on the border of the entire network, the sensors on the edges of the coverage holes, and the points on the boundary of an event. For convenience, we mention the former two as network-boundary nodes and the third one as event-boundary sensors. The later one can be distinguished out by reporting different statistical data through various measurements according to events [5]. However, these methods are almost incapable of identifying network-boundary sensors as there could be no active event in a RoI. In an environment where sensors are deterministically deployed, boundary nodes are the ones surrounding the disabled sensors (which are usually the main reason for the existence of coverage holes) and the ones on the RoI's frontier. However, the shape of the frontier, the position and the number of the coverage holes are often unpredictable in a non-deterministic network where sensors are randomly arranged (such as been dropped out by aircrafts) [6]. Thus, there is a need for an algorithm to identify the network-boundary nodes.

Researchers have proposed numerous solutions to solve the problem in two-dimensional WSNs. They are roughly categorized into three kinds: the algorithms based on geometry [7]–[9], the statistical methods [10], [11], and the techniques using topology [12]–[14]. The first one usually provides accurate results but needs every sensor's coordinate. The second makes decisions according to the statistical results of a sensor's local environment and it needs an even distribution of the nodes. The third kind does not need sensors' positions and is adaptive to various deployments, but it requires to be operated in a centralized way or demands a comprehensive overview of the local area of a sensor.

Several works investigated the problem in a three-dimensional volume WSN. In this model, sensors are arranged both in the body and on the surface of a three dimensional entity [15] where the boundary nodes are the ones on the surface, and there is no coverage hole in it. But sometimes people are interest in transmitting data between nodes on the surface of an entity, eg. monitoring the exterior of a volcano. Such work requires a three-dimensional surface

Manuscript received October 20, 2018.

Manuscript revised January 11, 2019.

Manuscript publicized March 4, 2019.

[†]The authors are with the School of Computer Science and Technology, Anhui University of Technology, China.

^{††}The author is with the School of Information Engineering, Chaohu College, China.

^{†††}The author is with the College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications, China.

a) E-mail: linnawahut@gmail.com (Corresponding author)

b) E-mail: 054060@chu.edu.cn

c) E-mail: Xzheng@ahut.edu.cn (Corresponding author)

d) E-mail: xuanguowu@gmail.com

e) E-mail: guiguan@njupt.edu.cn

DOI: 10.1587/transinf.2018EDP7355

WSN [16] and little effort has been put to solve the problem in it. Imagine a fire alarm WSN deployed on a mountain with curved surface and severe environments. Due to the outdoor bad conditions, sensors in the network are dropped out by airplanes and the economical static nodes in it could not move away from their initial positions. In order to fill in the blank areas in the network caused by sensors' random deployment, we may deploy a few mobile sensors together with the static ones. These mobile nodes can relocate themselves to the holes [17]. The reposition of them requires the locations of the holes and this work needs guidance from boundary sensors.

In this paper, our algorithm picks out the network-boundary nodes in a nondeterministic three-dimensional surface WSN. In Sect. 2, we discussed the related works. Section 3 gives the network model we used for our algorithm. In Sect. 4, we explained the details of our method. Section 5 contains the simulation results. The advantages of our algorithm are the following:

- It is a distributed algorithm that only needs knowledge of the nodes in the 1-hop neighborhood of a sensor. After the communication procedure in the first step, a sensor executes the subsequent steps alone and is capable to make independent decisions on its role.
- It can solve problems in random networks with uneven density of nodes. It is also capable to identify boundary nodes in sparse three-dimensional WSNs.

2. Related Work

2.1 Boundary Detection in Planar Networks

The algorithms based on geometry often require the usage of Voronoi Diagrams or Delaunay Triangles [7], [8]. Li et al. [7] distinguished boundary nodes out from the inner ones by searching for all the empty circles after a Delaunay Triangulation of the network. Fang et al. [9] proposed an algorithm that picking out a hole in a network through communication among the sensors. A hole in it is defined as a circle built by the nodes with the local minia of a phenomena. This kind of methods are usually able to provide precise results but they need a prerequisite of the sensors' coordinates. The traditional ones of them is hard to be directly applied to a three-dimensional model as the missing of vertical coordinates would introduce errors. We will discuss this problem more specifically in Sect. 4.1.

The statistical methods tend to assume a particular distribution of sensors and need to carefully set one or more thresholds. Fekete et al. suggested a notion of 'Restricted Stress Centrality' in [10] to pick out the boundary nodes in a social network. Li et al. proposed an algorithm named as CDBD that uses the criteria of 'betweenness' and 'closeness' in the neighborhood of a node to decide its role [11]. In the method, the higher value of a sensor's betweenness or the lower value of its closeness indicates the higher possibility of it been a boundary node. Though this kind of algo-

rithms do not require sensors' locations, they need an even distribution of the nodes and this is difficult to be realized in a three-dimensional surface WSN. Besides, people also need to carefully set the thresholds for the vital criteria and vary them while changing the parameters of the networks.

The techniques using topology are adaptive to the networks with different models of sensor coverage and communication capability, they are mostly based on the connectivity between nodes [12]. Kroller et al. [13] presented a method that searches the 8-hop neighborhood of a node to find out if there is any structure of a 'flower' and uses augmenting cycles to identify the boundary sensors. Wang et al. [14] proposed a distributed algorithm in which there is a message flooding initiated from a random root sensor before the decision making on each node. The flooding brings in an extra time delay and communication load to the network. This kind of method either tries to find out a certain topological pattern in the entire network [18] or needs collaboration from a large number of nodes for preparation. When a sensor only has the information from its 1-hop or 2-hop neighbors, limited knowledge often forces it to search for a Hamiltonian Cycle in its local area, which means that it needs to solve a NP problem [19], not to mention the difficulties in determining the position and shape of the circle.

Beghdad et al. suggested an algorithm called BD-CIS [20] that does not depend on the finding of any Hamiltonian Cycle. It looks for a circle in the 1-hop neighborhood of a sensor by obtaining some closed paths of connected nodes from several independent sets. It seems that this method could be directly applied to a three-dimensional model but the core step of searching for all the independent sets with a threshold cardinality limits its usage. In a randomly deployed sensor network, a node may only have three or four 1-hop neighbors so it could be wrongly identified as a boundary point.

2.2 Boundary Detection in Volume Networks

Feng Li et al. proposed an on-line algorithm called UNFOLD to detect boundary nodes in a dynamic volume WSN [21]. The environment for using UNFOLD is a three-dimensional cloud in which each point is a sensor node. In the first step of it, each node exchanges information with its neighbors to obtain a local coordinate for a common viewpoint. Then, the generated coordinates are transformed to perform a Convexity Test. In step three, a decision is made to set the role of a given node based on whether the image of it is on a specific convex hull. UNFOLD can hardly be applied to the surface coverage model as each sensor in it stands on a convex hull formed by other nodes.

Zhou et al. suggested another method to solve the boundary node identification problem in this model [22]. It first constructs a tetrahedral structure to divide the entire surface of a network into some coarse triangles. Then, it seals them to exclude the inner sensors. The raw results obtained from the former two steps are then refined by a series of processes as the following: boundary landmark expansion,

boundary face split, and boundary surface thinning. Though it is a distributed method that only uses sensors' connectivity, it requires in each step the cooperation through communication between the nodes.

Zhou et al. also proposed a two-step localized algorithm for precise boundary detection [23]. In the first step, an Unit Ball Fitting (UBF) process is used to coarsely identify the boundary nodes. It is based on the idea that a boundary node should stand on the surface of an empty unit ball. In the second phase, an Isolated Fragment Filtering (IFF) step triggers the formerly picked out boundary nodes to send out local flooding packets to refine the results. When sensor density is high, the local message flooding would introduce heavy communication load into the network.

2.3 Surface Coverage

Zhao et al. proposed a WSN model in which nodes are deployed on a curved surface [16]. It focuses on the partial exterior of a three-dimensional entity rather than its inner part. They studied the problem of sensor deployment in it and calculated the number of nodes in need for a full coverage. By using the theories coming from Discrete Surface Ricci Flow and Centroidal Voronoi Partition, Jin et al. [24] provided an optimal node deployment strategy in this model.

Little work has been put to find out the boundary nodes in the WSNs for surface coverage. An algorithm suggested by Dhanapala et al. [25] uses Topology Preserving Map (TPM) to transform the connectivity between sensors into a virtual map and uses it to identify the boundary nodes. Although, the authors extended the usage of TPM into a couple of three-dimensional surface networks, it requires a number of anchor nodes and an uniform distribution of the sensors. As the virtual coordinate system is built based on hop-distance between sensors, it highly depends on the even distribution of nodes and could hardly deal with the problem in which nodes are randomly placed with points of different curvature. Not to mention that it needs to firstly broadcast messages in the entire network from several anchors to build the coordinate system, and there still remains problems related to the placement of the anchors.

3. Network Model

The RoI in our model is a curved surface and it is randomly deployed with wireless sensors. Each sensor obtains its three-dimensional Euclidean coordinate (x, y, z) through calculations of the surface localization algorithms [26], [27]. The sensors in the network are homogeneous ones with the same sensing range and communication radius. There is no strictly determined pattern for deploying them in our model. As our algorithm focuses on finding out the nodes on the edges on a curved surface and an edge could either be the frontier of a coverage hole or the boundary of the entire WSN, a few isolated nodes could be tolerated in the network. All of the nodes are static so they could not leave their initial positions after deployment. Each of them has an

unique ID. It could either be configured by the administrator or represented by its coordinate.

A network in our model is represented by a graph $G = (V, E)$. Each sensor s in the set of nodes S is a vertex $v \in V$. If two sensors u and v are located within each other's communication range, there is an undirected edge $e \in E$ between them in G . In the following part of this paper, we will represent a sensor simply by v . A sensor v collects data from the nodes in its 1-hop neighborhood $N_1(v)$. These data are stored by v in a set $Co_1(v)$ containing the nodes' IDs and their coordinates.

With our main purpose as boundary node identification, we would not discuss the aspects like sensors' scheduling or the implementation of communication between nodes in this paper.

Some of the terms in this paper are defined as the following:

Definition 1. (Inner Sensor) *Given a node v and its sensing area A_v on a curved surface, c_v is the circumference of A_v , if c_v is fully covered by a number of nodes that can directly communicate with v , v is an inner sensor.*

Definition 2. (Boundary Node) *If a sensor v is not an inner one, it is a boundary node.*

4. Identifying Boundary Nodes in Surface Coverage

Our solution is inspired by a simple idea, a sensor in our model is an inner node if it is fully surrounded by a couple of other ones. Otherwise, it is a boundary sensor. The sensors surrounding an inner node are the 1-hop neighbors of it that are located on the surface or in the body of a ball that using the inner sensor as its center and the sensor's communication range as its ball radius. Different from an unit ball been used in UBF in [23], the ball does not need to be fully filled with nodes, and the 1-hop neighbors of an inner sensor v roughly shape a latitude-circle-liked closed path r_v in the ball. The routes on the path are ensured by the connections between nodes.

Conversely, if we start from a 1-hop neighbor of v and move to the next stop by finding out another 1-hop neighbor of v that connects with the former one, the travel process should form a closed path. In the view of a ball B_v , if we use v as its center and move a semicircle plate along the z -axis from a random node $u \in r_v$ the trajectory of the plate should build a complete ball after the travel mentioned above. During the rotation of the plate, each 1-hop neighbor of v , even though it could be out of r_v , should be met with the plate. On the contrary, if we start from a 1-hop neighbor of a boundary sensor, we may stop the process halfway due to the impossibility of finding out a following node or discovering the final path to be an open one.

According to this idea, we suggest a distributed algorithm called "Circle Sweeping" that containing three steps in succession. It first projects the 1-hop neighborhood of a sensor to the plane. Then, it sorts the projected nodes according to their angles and finds out if there exists any ring formed by them. At last, the algorithm validates a circle to

Fig. 1 Three sensors on a curve and their projections.

(a) A planar view of the example network. It contains only three coverage holes.

(b) A 3D view of the example network. All sensors are boundary nodes.

Fig. 2 A planar view and the 3D one of an example network.

Fig. 3 A curve on the longitudinal profile of a RoI. The blue dot represents a sensor.

confirm that it is a ring surrounding the node.

4.1 Local Surface Planarization

As described in [16], if we only focus on the projected plane of a three-dimensional surface WSN, some holes on the curve would disappear (Fig. 1). When this happens, some sensors on the edges of the holes would be categorized as inner nodes. In an extreme case like the one depicted in Fig. 2, a few sensors are boundary nodes in the projected image, but in fact, all the nodes are boundary ones on the surface.

The controversial result in the above example rises from the reason that a line on the longitudinal profile of the RoI in our model is a curve (Fig. 3). As there exists an angle between the curve and the horizontal line, the length of any line segment on the curve is shortened after projection.

Fig. 4 The 1-hop neighborhood of a sensor in 3D/2D. Projecting a sensor v (red dot) and all the points in N_v^1 (blue dot) on a 3D surface to the plane. The 1-hop neighbors of v are contained in a 3D sphere built by a semicircle plate that meeting with each 1-hop neighbor during its rotation.

Therefore, the interval distances between sensors on the surface are cut down on the plane. When an angle grows large enough, the space between two apart nodes on the surface disappears on the projected plane and it is viewed as if they were connected.

From the above discussion, we learn that the behavior of “eliminating the heights of nodes” modifies the connections between sensors. Therefore, if we preserve the original topology of a WSN, we could find a way to solve our problem in a two-dimensional fashion. In order to minimize potential communication loads, we limit our algorithm to run in the 1-hop neighborhood of a sensor. The process of shadowing the 1-hop neighborhood of a sensor to the plane (Fig. 4) is named as Local Surface Planarization. In this step, the connections between sensors are kept as their original ones on the curved surface and we still name the projected nodes as their original ones.

4.2 Circle Finding

After the first step, the local area of a sensor is flattened on the plane. This area contains the sensor itself and the nodes in its 1-hop neighborhood. Based on our intuitive idea, if there exists a closed path built by the points around the sensor, it is an inner node, otherwise it stands on a boundary line. Thus, our next step is to find out a circle in the planar local area.

Lemma 1. *Starting from a node u in the 1-hop neighborhood N_v^1 of an inner sensor v , we rotate a semicircle plate P along the z -axis with v as the center, if P meets a point $q \in N_v^1$ and q connects with u , we draw a line segment $s_{u,q}$ between u, q . After the rotation, we project the set of the line segments to the plane. Then there exists a sectionalized ring R_v around v .*

Proof. We assume v is an inner node without any closed ring on the planar graph. This situation happens when the rotation of P stops at a node $l \in N_v^1$ and l, u are not connected. The plate P stops before turning 360° and its trajectory can not build a complete ball. Therefore, v is a boundary

node. $\square$

An immediate idea on searching for a closed path in a graph is by finding out a Hamiltonian Cycle in it. However, this means that we need to solve a NP problem. Though we can use an approximation algorithm to solve this problem, as the distribution of sensors are random in the network, it is time consuming when there appears dense nodes in some areas. Another thought would be looking for a circle by performing Depth-First-Search in the graph. But in this way, there would be too many useless closed paths interfere with our work so that we have to distinguish a qualified one out from a number of circles.

As we only need to locate one circle that is around an inner node and it does not need to pass all its 1-hop neighbors, points that do not stand on any closed path should not participate in the process of circle finding. We build a graph G_v^1 to represent the local area of v after local surface planarization. A point in G_v^1 is either a node in N_v^1 or v . By collecting N_u^1 from each $u \in N_v^1$, there is an edge e_{pq} connecting two nodes p and q in G_v^1 if they are connected. Then we prune G_v^1 by removing any node with less than 2 neighbors together with any edge connected to it.

Lemma 2. *After pruning, only the sensors on loops stay in G_v^1 .*

Proof. It is obvious that any node with only one neighbor could not stand on a loop. If there is any sensor with a degree of 1 exists in G_v^1 , it should stand on the end of an open path and it would be removed together with the edge connected with it in the pruning process. Assuming there exists an open path in G_v^1 after pruning, as the sensor on the end of the path has a degree of less than 2, the pruning process should not have stopped. $\square$

At this point, our work is to find out a circle from the remaining part in G_v^1 . We solve it by simulating the process of rotating a semicircle plate P in the local area of v . Starting from a node u in N_v^1 , the first sensor q met by P is the closest one to it and the angle between u and q that using v as the vertex is the smallest. Using q as the new starting point, another node $r \in N_v^1$ should be met by P after rotating for an angle of α_v^u and r is the closest sensor next to q . The plate P keeps moving along with the z axis around v until it reaches to the last sensor in N_v^1 that it has not met before. Using v as the apex and u, r as the other two vertexes, the angle $\angle uvr$ keeps growing if we fix the initial starting point u . According to this, we transform the coordinate system of G_v^1 into a local one by occupying v as the origin. In this system, we calculate an angle for each node in N_v^1 . An angle α_v^u of sensor u ($u \neq v$) equals to the value of $\angle uvp$, p is a random point on the positive x axis. Then, we record the value of α_v^u of u with its ID in a set M_v . Next, we sort the items in M_v according to the value of their angles so that we could visit the sensors in an order. We search for any circle in N_v^1 on the ordered sequence by sliding a Connection Window along a line of M_v .

Definition 3. (Connection Window) A connection window CW_v is an unfixed-sized sliding window without any

Algorithm 1 Circle Finding

```

1: At each node  $v$ :
2: Set  $v.isBoundary = false$ .
3: Set  $FindCircle = false$ .
4: Send ("Hello") to the 1-hop neighbors of  $v$ .
5: Create  $N_v^1$ .
6: if  $|N_v^1| < 3$ : then
7:    $v.isBoundary = true$ .
8: end if
9: Send ("1-hop neighbor") to the 1-hop neighbors of  $v$ .
10: Receive  $N_u^1$  from the neighbors, create  $G_v^1$ .
11: Prune  $G_v^1$ .
12: for Each node  $u \in G_v^1$ , calculate the number of neighbors of  $u$  in  $G_v^1$ .
    do
13:   if Less than 2: then
14:     Remove  $u$  from  $G_v^1$ .
15:   end if
16: end for
17: if Number of points in  $G_v^1$  is less than 4: then
18:    $v.isBoundary = true$ .
19: end if
20: Calculate an angle  $\alpha_v^u$  for each node  $u$  in  $G_v^1$  except for  $v$ , obtain a set  $M_v$ .
21: Sort  $M_v$  according to the size of the angle of each sensor.
22: Set the first node  $i$  in  $M_v$  as the starting point.
23: while  $FindCircle = false$  and  $i$  is not the last one indexed in  $M_v$ : do
24:   Find the connection window  $CW_v^i$  of  $v$  from  $i$  in the sorted set  $M_v$ :
25:   Use the last node  $j$  in  $CW_v^i$  as the new starting point.
26:   Slide the connection window  $CW$  on  $M_v$ .
27:   if  $CW$  can not move forward: then
28:     Get the last element  $e$  in  $CW$ .
29:     if  $i$  and  $e$  are connected: then
30:        $FindCircle = true$ .
31:       Record the sensors encountered by  $CW$  during its movement.
32:     else
33:        $i =$  the next node of  $i$  in  $M_v$ 
34:     end if
35:   end if
36: end while
37: if  $FindCircle = false$  then
38:    $v.isBoundary = true$ .
39: end if

```

predetermined starting point. It contains a subset of the sensors in N_v^1 . For the first node s in CW_v^s , it is connected with any other node $t \in CW_v^s, t \neq s$.

We find a connection window CW_v^i of v from the first node i in the sorted set M_v . For any node in CW_v^i , it is connected with i ; but for any one in M_v that is excluded from CW_v^i , it is disconnected with i . Assuming j is the last node in CW_v^i , the sensor k that is indexed behind j in M_v is disconnected with i . The mutual connection between two sensors in CW_v^i is ensured by the original topology of the WSN. Then, we move the connection window CW_v^i that starts from i to j toward a new position that starts from j to g and rename it as CW_v^j , g is the last node in CW_v^j . The size of the connection window varies during the movement. If it shrinks to a small one that contains only one node, the movement stops. A connection window CW_v^r with only one sensor means that if there is any node connected with r it is indexed ahead of r in M_v . After the sliding, we check the connectivity between the initial starting point i and the stop point r . If they are

connected, we obtain a ring R_v . Otherwise, we find another connection window CW_v^t of v from the second node t in M_v and slide it. This process is repeated again and again until we successfully find out a circle or we start the sliding of a connection window from the last node in M_v .

Theorem 1. *If the position of a connection window on M_v changes for at least once during its movement from a start point s to an end point t and s, t are connected, there exists a closed path in G_v^1 .*

Proof. As the position of the connection window CW_v^s changes, there exists at least a middle sensor m between s and t on M_v . From the definition of a connection window we know both of the pairs of sensors $s - m$ and $m - t$ are connected, m is the last node in CW_v^s and the first one in CW_v^m . After the connection window sliding from the position of CW_v^s to the one of CW_v^m the movement stops at node t , t is the last sensor in CW_v^m . Since $s - m$, $m - t$ and $t - s$ are connected, there exists a close path built by the sensors s, m and t in G_v^1 . $\square$

After obtaining R_v , we can not guarantee that v is an inner sensor as the searching of R_v is guided by the structure of G_v^1 . Therefore, we need a further step to confirm the validity of R_v .

4.3 Circle Verification

In this step, we validate if R_v is a circle surrounding v . With the help of coordinates, we divide the nodes on R_v into four quadrants in the local coordinate system of v . If the sensors are gathered together in a single quadrant or scattered in two consecutive ones, we obtain the result that R_v stands at one side of v . Next, we focus on situations in the remaining cases. As M_v is a sorted set, the sensors in each quadrant $Q_i^v, i \in \{1, 2, 3, 4\}$ are also placed in order.

Definition 4. (Quadrant Head) A quadrant head is a sensor u in Q_i^v that has the minimum angle a_u^v .

Definition 5. (Quadrant Tail) A quadrant tail is a sensor u in Q_i^v that has the maximum angle a_u^v .

Definition 6. (Consecutive Quadrants) For any two quadrants in the local coordinate system of v , they are consecutive ones if none of them is lack of node.

Lemma 3. *Given two consecutive quadrants Q_i^v and Q_j^v with nodes ($j = i+1$ or $j = i-1$ is not required as a quadrant may be empty), we obtain a head (or tail) node u from one quadrant and a tail (or head) sensor p from the other. If u and p are not connected, R_v locates beside v . Otherwise, if the angle difference between a_u^v and a_p^v is greater than 180° , R_v locates beside v .*

Proof. We use Fig. 5 to illustrate. A sensor pair u, p or u, q is obtained from a ring R_v . If the circle is large, we use the pair of u, p . If it is small, we use the pair of u, q . Sensor u is connected with node q , not p . It is clear in the figure that if R_v is large and locates beside v , u and p are not connected. Otherwise, v stands inside of R_v . If R_v is small and u, q are connected, as R_v locates at one side of v , the angle difference

Fig. 5 A circle stands next to a sensor. The black node is sensor v , the blue dot p on the top left corner is a node on the circle and the diamond one u represents another. The grey dot q on the lower left corner is a substitute of p in a different situation.

between a_u^v and a_q^v is greater than 180° . Otherwise, v should be surrounded by R_v . $\square$

Algorithm 2 Circle Validation

```

1: At each node  $v$  that has a  $R_v$ :
2: Set  $ValidCircle = true$ .
3: Divide the nodes on  $R_v$  into four sets that each one belongs to a quadrant.
4: for Each quadrant  $Q_k^v$  with at least one node: do
5:   Obtain a quadrant head  $qh_k$  and a quadrant tail  $qt_k$  of  $Q_k^v$ .
6:   for Any pair of two consecutive nodes  $i$  and  $j$  on  $M_v$  in  $Q_k^v$ : do
7:     if  $i$  and  $j$  are disconnected: then
8:        $ValidCircle = false$ .
9:     end if
10:   end for
11: end for
12: if  $ValidCircle = true$ : then
13:   for Any two consecutive quadrants  $Q_i^v$  and  $Q_j^v$  that has sensors: do
14:     Get the quadrant head  $qh_i$  and the quadrant tail  $qt_j$  (or the quadrant tail  $qt_i$  and the quadrant head  $qh_j$ ),  $i, j$  are two consecutive sensors on  $M_v$ .
15:     if  $qh_i$  and  $qt_j$  (or  $qt_i$  and  $qh_j$ ) are disconnected: then
16:        $ValidCircle = false$ .
17:     end if
18:     Calculate the angle difference  $\alpha$  between  $a_{qh_i}^v$  and  $a_{qt_j}^v$ .
19:     if  $\alpha > 180^\circ$  then
20:        $ValidCircle = false$ .
21:     end if
22:   end for
23: end if
24: if  $ValidCircle = false$  then
25:    $v.isBoundary = true$ .
26: end if

```

Theorem 2. *If R_v passes circle validation, it is a ring surrounding v .*

Proof. We assume there is a ring R stands next to the sensor v , but it passes circle validation. From Fig. 5 we know that a ring locates next to v could have a shape like the letter ‘U’. We pick the two nodes i and j on the two ends of the ‘U’ shaped circle. The two sensors i and j are disconnected, otherwise R is a circle that surrounding v . If i and j appear in the same quadrant, the ring could not pass the validation as the two nodes are disconnected. If i and j are sensors at two different quadrants, it would also fail at the test due to the same reason. If the ring R does not look like ‘U’ and simply locates next to v , we could find a line l that separates

the plane of G_v^1 into two half, one contains v and the other has R . We find two nodes i and j on R that are nearest to l . The sensors i and j should fall into two distinct quadrants. As i and j are both on the other side of l , the angle difference between them is greater than 180° , so that R should fail at the test. $\square$

In Circle Sweeping, any sensor v only needs to communicate with its 1-hop neighbors in local surface planarization. The communication load in this step is at most $O(n^2)$, $n = |N_v^1|$. After obtaining G_v^1 , each single sensor v can identify its role in the network independently through a series of steps with the complexity of at most $O(n^2)$.

5. Simulation Results

We generate several networks in Matlab to perform our simulations. We obtain three-dimensional networks by modifying the parameters in the “Peak” function and deploying sensors randomly on the smoothly curved surfaces. All the networks are perturbed ones managed by the grids on the projected plane. In each area on the curved surface that corresponds to a grid on the plane, a number of k nodes are randomly deployed with a distance of at most σ away from the grid center, k varies according to the settings of the networks.

5.1 Visual Results

We first get two groups of visual results to show the effectiveness of Circle Sweeping.

5.1.1 Single Algorithm

We generate a 40×40 three-dimensional surface network with 3143 sensors (Fig. 6(a)) and run Circle Sweeping on it. The shape of the testing network is a relatively complex figure that contains sharp corners and inward and outward angles which is designed to show the effectiveness of our algorithm. Figure 6(b) shows the result. From it we can

see that Circle Sweeping successfully depicts the nodes on the boundary of the entire network and the ones on the edge of the inner hole. We notice that there are some boundary nodes gathered together at the bottom in Fig. 6(b), it seems that these sensors are wrongly identified inner ones. But another look at the same position in Fig. 6(a) ensures us that these nodes are the boundary sensors in the network caused by a relatively sparse node density on the three-dimensional local surface.

5.1.2 Multiple Algorithms' Comparison

Next, we generate a 20×20 three-dimensional surface network with 938 sensors and run Circle Sweeping, the statistical based algorithm CDBD [11], and the topology based method BDCIS [20] on it. We compare these three algorithms as they are all running on the 1-hop neighborhood of a single sensor, output decisions by each node individually, and all of them do not require any behavior of Hamiltonian Cycle finding. The testing network has a structured outer boundary and a major inner coverage hole with curved edge. The boundary of the coverage hole has one inward and two outward obtuse angles, and one outward acute angle. Each grid has 5 sensors so the average node degree is 16.38 and the density of network is neither sparse nor too dense. Thus, we could obtain an overview of the performance of the three algorithms.

Figure 7(a) shows a three-dimensional visual output of Circle Sweeping. We can see that the three-dimensional view could not well present the results, so we show them in the two-dimensional view (Fig. 7(b) to Fig. 7(d)). Figure 7(b) displays several clear boundary lines of the network and the inner coverage holes. Circle Sweeping has successfully identified the corners and the boundary lines. Figure 7(c) shows that the boundary nodes picked out by CDBD are not enough. The edge lines are break in several areas which means that the nodes on the edges are wrongly identified as inner ones. Figure 7(d) indicates that BDCIS selected too many boundary nodes, the edge lines in it are thick and many of them are intersecting, which means that a lot of inner nodes are wrongly identified as boundary sensors.

5.2 Statistical Results

The above two groups of simulations give us a visual impression of our algorithm. For the statistical results we run another two groups of tests to obtain the correctness and effectiveness of Circle Sweeping. We add the algorithm based on Delaunay Triangulation in two-dimensional networks [7] (represented by PDT) to further support our opinion that the geometric strategies used in two dimensional networks are no longer effective in the three-dimensional ones.

5.2.1 Algorithms' Comparison

We generate ten 20×20 three-dimensional surface networks

(a) A 3D random sensor network with 3143 sensors. (b) The result of running Circle Sweeping on the left network. The red dots are the boundary nodes picked out by the algorithm, and the blue crosses are the inner ones.

Fig. 6 Boundary node identification of Circle Sweep on a random network.

Fig. 7 Visual comparison of the outputs from three algorithms. The magenta line segments show the connectivity between boundary sensors.

Fig. 8 Average node degree of the 10 simulation networks with the number of nodes per grid ranges from 1 to 10.

with their shapes the same as the one used in Sect. 5.1.2. In these networks, each area on the curved surface that corresponds to a grid on the plane, a number of k nodes are randomly deployed, k ranges from 1 to 10. Accordingly, the average node degree in each of the network ranges from 2.71 to 33.65 (Fig. 8). We test the four algorithms on these random networks with diverse sensor density to obtain their performances.

Figure 9 shows the correctness of the four algorithms. The Y axis of it represents the percentage of the boundary nodes been correctly identified by each of the method. We see from the figure that correctness of Circle Sweeping is above 85 percent when k is no more than 4. This value lowered as the sensors became denser and denser but still higher than the results of PDT and CDBD. It rises as we sort the nodes according to the angles we defined in Sect. 4.2. Due to the increasing of the number of nodes in the network, there may exist multiple sensors with the same angle but different connection with their neighbors. Therefore, the sliding of Connection Windows are disturbed in Circle Finding. We also see that PDT output a good result when $k = 1$. It is because the three-dimensional network is a rather sparse one when $k = 1$, the connectivity between nodes are well kept when the network is projected to the two-dimensional plane. However, when sensor density increases its correctness dropped sharply. The line of CDBD in Fig. 9 well explains its character, it is an algorithm useful for the networks with high sensor density. It only picked out 20 percent of the

Fig. 9 The correctness of the four algorithms.

Fig. 10 The effectiveness of the four algorithms.

boundary nodes of the test networks when they are sparse and performed better when they became denser.

It seems that BDCIS is the best one for identifying boundary nodes among the four algorithms in Fig. 9 as the correctness of it is always higher than 90 percent. However, Fig. 10 indicates that the effectiveness of the method is low. It obtained the result by overly picking out boundary nodes. The Y axis of Fig. 10 represents the percentage of nodes that are real boundary ones which have been effectively identified by each of the method. We can see that CDBD also has the same problem in the three-dimensional environments. In networks where sensors are not evenly deployed, the high dependence on the statistical inter-relation between nodes made its performance getting lower and lower. The statistical results support our visual ones in Sect. 5.1.2, Circle Sweeping outperforms the other three in identifying boundary nodes in nondeterministic three-dimensional surface WSNs.

Fig. 11 The correctness and effectiveness of Circle Sweeping when the positioning error of each node ranges from 1% to 20%.

5.2.2 Algorithm Performance in the Presence of Positioning Errors

As Circle Sweeping is a geometric algorithm that uses the coordinates of nodes, we test the performance of it in the presence of positioning errors. We use a 20×20 three-dimensional surface network with 370 sensors, a number of 2 nodes are randomly deployed in each grid. We use a sparse network as the results in Sect. 5.2.1 already showed that we need to improve the performance of Circle Sweeping in dense sensor networks. The random location errors of each sensor ranges from 1% to 20%. The red line in Fig. 11 presents the effectiveness of Circle Sweeping and the blue line shows the correctness of it. We see that the correctness of the method is relatively steady when sacrificing its effectiveness. Thus, Circle Sweeping is tolerable with minor positioning errors of the sensors.

6. Conclusion

In this paper, we study the problem of identifying boundary nodes in a three-dimensional surface WSN. We use an algorithm that solve the problem in the 1-hop neighborhood of a sensor. The method that we named as Circle Sweeping contains three steps: local surface planarization, circle finding, and circle validation. This distributed algorithm projects the 1-hop neighborhood of a sensor to the plane, finds out if there is any circle on the plane and then checks whether it is surrounding the sensor. Simulation results show the correctness and effectiveness of it. Our future work will focus on improving its performance in dense sensor networks with high average node degree.

Acknowledgments

The work is supported by the National Natural Science Foundation of China under Grant No.61502010 and No.61672038 and Natural Science Foundation of Anhui Province under Grant No.1608085QF146. We would like to thank the anonymous referees for their valuable comments and helpful suggestions.

References

- [1] B. Gu, Y. Chen, H. Liao, Z. Zhou, and D. Zhang, "A Distributed and Context-Aware Task Assignment Mechanism for Collaborative Mobile Edge Computing," *Sensors*, vol.18, no.8, 2018.
- [2] Z. Zhou, H. Liao, B. Gu, K.M.S. Huq, S. Mumtaz, and J. Rodriguez, "Robust Mobile Crowd Sensing: When Deep Learning Meets Edge Computing," *IEEE Network*, vol.32, no.4, pp.54–60, 2018.
- [3] K. Latif, N. Javaid, A. Ahmad, Z.A. Khan, N. Alrajeh, and M.I. Khan, "On energy hole and coverage hole avoidance in underwater wireless sensor networks," *IEEE Sensors Journal*, vol.16, no.11, pp.4431–4442, 2016.
- [4] M.R. Senouci, A. Mellouk, and K. Assoune, "Localized movement-assisted sensor deployment algorithm for hole detection and healing," *IEEE Transactions on Parallel and Distributed Systems*, vol.25, no.5, pp.1267–1277, 2014.
- [5] N. Dziengel, M. Seiffert, M. Ziegert, S. Adler, S. Pfeiffer, and J. Schiller, "Deployment and evaluation of a fully applicable distributed event detection system in wireless sensor networks," *Ad Hoc Networks*, vol.37, no.P2, pp.160–182, 2016.
- [6] N. Ahmed, S.S. Kanhere, and S. Jha, "The holes problem in wireless sensor networks: a survey," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol.9, no.2, pp.4–18, 2005.
- [7] W. Li and W. Zhang, "Coverage hole and boundary nodes detection in wireless sensor networks," *Journal of Network and Computer Applications*, vol.48, pp.35–43, 2015.
- [8] M. Saoudi, F. Lalem, A. Bounceur, R. Euler, M.-T. Kechadi, A. Laouid, M. Bezoui, and M. Sevaux, "D-LPCN: A distributed least polar-angle connected node algorithm for finding the boundary of a wireless sensor network," *Ad Hoc Networks*, vol.56, pp.56–71, 2017.
- [9] Q. Fang, J. Gao, L.J. Guibas, "Locating and bypassing holes in sensor networks," *Mobile Networks and Applications*, vol.11, no.2, pp.187–200, 2006.
- [10] S.P. Fekete, M. Kaufmann, A. Krooller, and K. Lehmann, "A new approach for boundary recognition in geometric sensor networks," *Proceeding of the 17th Canadian Conference on Computational Geometry*, pp.82–85, 2005.
- [11] X. Li, S. He, J. Chen, X. Liang, R. Lu, and S. Shen, "Coordinate-free distributed algorithm for boundary detection in wireless sensor networks," *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, Houston, pp.1–5, 2011.
- [12] L. Sitanayah, A. Datta, and R. Cardell-Oliver, "Heuristic algorithm for finding boundary cycles in location-free low density wireless sensor networks," *Computer Networks*, vol.54, pp.1630–1645, 2010.
- [13] A. Kroller, S.P. Fekete, D. Pfisterer, and S. Fischer, "Deterministic boundary recognition and topology extraction," *SODA '06 Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, Miami, pp.1000–1009, 2006.
- [14] Y. Wang, J. Gao, and J.S.B. Mitchell, "Boundary recognition in sensor networks by topological methods," *MobiCom '06 Proceedings of the 12th annual international conference on Mobile computing and networking*, Los Angeles, pp.122–133, 2006.
- [15] S.M.N. Alam and Z.J. Hass, "Coverage and connectivity in three-dimensional networks with random node deployment," *Ad Hoc Networks*, vol.34, pp.157–169, 2015.
- [16] M.-C. Zhao, J. Lei, M.-Y. Wu, Y. Liu, and W. Shu, "Surface coverage in wireless sensor networks," *Proceeding of IEEE INFOCOM 2009*, Rio de Janeiro, pp.109–117, 2009.
- [17] Z. Zhou, J. Feng, B. Gu, B. Ai, S. Mumtaz, J. Rodriguez, and M. Guizani, "When Mobile Crowd Sensing Meets UAV: Energy-Efficient Task Assignment and Route Planning," *IEEE Transactions on Communications*, vol.66, no.11, pp.5526–5538, 2018.
- [18] O. Saukh, R. Sauter, M. Gauger, and P.J. Marrón, "On Boundary Recognition Without Location Information in Wireless Sen-

- sor Networks,” *ACM Transactions on Sensor Networks*, vol.6, pp.20:1–20:35, 2010.
- [19] F. Yan, P. Martins, and L. Decreusefond, “Connectivity-Based Distributed Coverage Hole Detection in Wireless Sensor Networks,” 2011 IEEE Global Telecommunications Conference - GLOBECOM 2011, Houston, pp.1–6, 2011.
- [20] R. Beghdad and A. Lamraoui, “Boundary and holes recognition in wireless sensor networks,” *Journal of Innovation in Digital Ecosystems*, vol.3, no.1, pp.1–14, 2016.
- [21] F. Li, J. Luo, C. Zhang, S. Xin, and Y. He, “UNFOLD: uniform fast on-line boundary detection for dynamic 3D wireless sensor networks,” *Proceeding of the 12th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '11)*, New York, Article 14, 11 pages, 2011.
- [22] H. Zhou, H. Wu, and M. Jin, “A robust boundary detection algorithm based on connectivity only for 3D wireless sensor networks,” *Proceeding of IEEE INFOCOM 2012*, Orlando, pp.1602–1610, 2012.
- [23] H. Zhou, S. Xia, M. Jin, and H. Wu, “Localized and Precise Boundary Detection in 3-D Wireless Sensor Networks,” *IEEE/ACM Transactions on Networking*, vol.23, no.6, pp.1742–1754, 2015.
- [24] M. Jin, G. Rong, H. Wu, L. Shuai, and X. Guo, “Optimal surface deployment problem in wireless sensor networks,” *Proceeding of IEEE INFOCOM 2012*, Orlando, pp.2345–2353, 2012.
- [25] D.C. Dhanapala, A.P. Jayasumana, and S. Mehta, “On boundary detection of 2-d and 3-d wireless sensor networks,” *Proceeding of 2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, Houston, pp.1–5, 2011.
- [26] Y. Yang, M. Jin, and H. Wu, “3D surface localization with terrain model,” *Proceeding of IEEE INFOCOM 2014*, Toronto, pp.46–54, 2014.
- [27] J.-K. Lee, Y. Kim, J.-H. Lee, and S.-C. Kim, “An efficient three-dimensional localization scheme using trilateration in wireless sensor networks,” *IEEE Communications Letters*, vol.18, no.9, pp.1591–1594, 2014.

Linna Wei received the B.E. degree and the Ph.D. degree from College of Computer Science and Engineering, University of Electronic Science and Technology of China. She is currently a lecturer with the School of Computer Science and Technology, Anhui University of Technology. Her research interests include computer networks and wireless sensor networks.

Xiaoxiao Song received the MEng degree from the school of Computer Science and Technology, University of Anhui Technology in 2018. She is currently with the College of Information Engineering, Chaohu University. Her research interest include computer networks.

Xiao Zheng received the B.E. degree from Anhui University, PR China, and the Ph.D. degree in computer science and technology from Southeast University, PR China. He is currently a professor at the School of Computer Science and Technology, Anhui University of Technology. His research interests include computer networking, service computing, mobile cloud computing and machine learning.

Xuanguo Wu received the Ph.D. degree from the school of Computer Science and Technology, University of Science and Technology of China, Hefei, China, in 2013. He is currently an associate professor with the school of Computer Science and Technology, University of Anhui Technology. His research interests include computer networks and privacy protection.

Guan Gui received the Dr. Eng. degree in information and communication engineering from University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2011. From October 2009 to March 2014, he joined Tohoku University as for research assistant as well as postdoctoral research fellow, respectively. From April 2014 to October 2015, he was an Assistant Professor in Akita Prefectural University. Since November 2015, he has been a professor with Nanjing University of Posts and Telecommunications (NJUPT), Nanjing, China. He is currently engaged in research of deep learning, compressive sensing, and advanced wireless techniques. He is a senior member of Institute of Electrical and Electronics Engineers (IEEE). Dr. Gui has been serving editors of *Security and Communication Networks* (2012–2016), *IEEE Transactions on Vehicular Technology* (2017), *IEEE Access* (2018) and *KSII Transactions on Internet and Information Systems* (2017). He received several best paper awards such as ICC2014, ICC2017 and VTC2014-Spring.