Coordinate-Free Boundary Nodes Identification by Angle Comparison in Wireless Sensor Networks

Linna Wei(✉), Xiaoxiao Song, and Xiao Zheng

School of Computer Science and Technology, Anhui University of Technology, Maanshan 243032, Anhui, China
linnawahut@gmail.com, xxiaosong_ahut@sina.com, Xzheng@ahut.edu.cn

Abstract. Identifying the boundary nodes of a wireless sensor network (WSN) is one of the prerequisites for healing coverage holes, which belongs to the main problems of QoS in a network. Most of the existing solutions for this problem rely on the usage of coordinates or a relatively even distribution of sensors on the underlying network. However, as equipping localization devices on sensors usually means considerably higher cost, coordinates are often unavailable in low-budget networks. And it is often hard to guarantee the distribution of nodes. Therefore, identifying the boundary nodes without coordinates still faces difficulties. In this paper, we propose a distributed algorithm to solve this problem. In our method, a checking node first finds out all the potential triangles that with vertices from the 1-hop neighbors of the node. Then, the sensor keeps triangles containing it by calculating angles. At last, triangles with gap edges are identified by angle comparison to avoid wrong identification of a boundary node which has a U shaped ring locates beside. Illustrated simulation results show the performance effectiveness of our method, especially in networks with random and uneven sensor deployment.

Keywords: Boundary node identification · Coordinate-free method
Angle comparison · Coverage problem · Wireless sensor networks

1 Introduction

The nodes in a WSN are responsible for sensing the events appear in the Region of Interest (ROI), collecting data, and transmitting them to background for data analyzing. Therefore, a WSN should have a sufficient number of nodes[1] to fully cover the entire ROI. Otherwise, events appear in places where sensors are missing would be omitted and further affects the results of data analysis.

The work is supported by the National Natural Science Foundation of China under Grant No. 61502010 and No. 61402009 and Natural Science Foundation of Anhui Province under Grant No. 1608085QF146.

[1] We interchangeably use the words 'node' and 'sensor' in this paper.

L. Wang et al. (Eds.): QShine 2017, LNICST 234, pp. 3–12, 2018.
https://doi.org/10.1007/978-3-319-78078-8_1

Sometimes, this situation may lead to fatal consequences. Such as a missing detection of moving troops on a battlefield or a neglect of the leakage of toxic gases [1].

Accordingly, it is essential that a WSN fully cover the ROI in the entire lifetime. However, nodes in it are often vulnerable. They are easily to be moved away from original positions by animals, destroyed by harsh environments, shut down by limited energy supply, or simply been settled outside of the ROI due to random deployment. If there are no extra sensors to replace the failed (or disappeared) ones, blank areas would arise and weaken the QoS of the network. Researchers usually refer those areas as 'coverage holes'. In order to fully cover the ROI (again), one often need to use redundant nodes to 'heal' the coverage holes, which means that moving additional sensors to cover the blank area in the ROI [2]. In order to perform this work, the network needs to know where the holes are. One method of finding them is to first identify the boundary nodes and then distinguish the border of the holes and the frontier of the network, without help from coordinates. In this paper, we propose a distributed algorithm using techniques of point in triangle test and angle comparison to solve the problem in networks where sensors are randomly deployed and coexist a number of small scattered coverage holes.

2 Related Work

The method of boundary node detection can be categorized into three main ways. The first one are the geometrical approaches. In these algorithms, all the nodes know their coordinates [3]. Thus, Voronoi Diagrams or Delaunay Triangles or anchor nodes can be used to help locating the boundary sensors [4]. Although, the calculations are easy, the mechanism requires extra cost for the usage of localization equipments. The second methods are statistical ones. They often assume that the distribution of nodes in networks follow certain statistical functions and have threshold values. The calculations in them are straightforward but they rely on the uniform distribution of nodes in networks and often require high density [5]. The topological methods are the third kind of solutions. They do not need coordinates of sensors and often only need connectivity information between nodes [6,7]. Due to limited information, most of them do not perform well in sparse networks, especially when executed distributively.

Beghdad et al. [8] suggested a distributed algorithm BDCIS that using 1-hop neighbors of a sensor. It requires less communication than most of the existing works and avoids false detection of boundary nodes with the existence of some U-shaped rings in the network. However, the critical cardinal value used in it may not fit for randomly deployed sensor networks with small holes. Dhanapala et al. [9] presented a distributed algorithm (we refer to it as VCSTPM) that uses geometric relationship for boundary detection. It uses hop distances from nodes to anchors to build a virtual coordinate system. Based on the system, the algorithm finds out triangles and performs Point in Triangle (PIT) test by calculating areas of them. But, when a U-shaped ring locates beside a checking sensor, this algorithm may misidentify a boundary node as an inner sensor.

3 The Neighborhood of a Sensor

We assume that each sensor node has an unique ID and they cannot move after deployment. Through communication, each sensor v obtains its one-hop-neighbor set N_v^1. A sensor v can obtain the value of an angle formed by it with other two nodes in N_v^1. And two connected sensors u and v can obtain the distance between them by evaluating the RSSI value, assuming there are no obstacles stand on the plain network. The communication radius of a sensor is no greater than twice the length of its sensing radius. We first define the terms of 'inner sensors' and 'boundary nodes' used in this paper. From the definition of inner nodes, we know that if a node v is an inner nodes it should be surrounded by a 'ring' formed by a number of other sensors.

Definition 1 (Inner Sensors). *Given a node v and the circumference c_v of its sensing area A_v, if c_v is fully covered by other sensors in the same network, v is an inner sensor.*

Definition 2 (Boundary Nodes). *Given a sensor v in a network, if v is not an inner sensor, v is a boundary node.*

3.1 Rings in the Neighborhood

Definition 3 (k-Hop Neighbor). *If node u is connected with node v and u is k-hop away from v, then u is a k-hop neighbor of v. The set N_v^k contains all the k-hop neighbors of v.*

Proposition 1. *Given a node v, if v is an inner node, it is surrounded by a ring R_v formed by the nodes in N_v^1.*

Proof. Assuming there is a ring $R_v^{'}$ surrounds a node v, one node $u^{'} \in N_v^2$ is in $R_v^{'}$. We remove $u^{'}$ from $R_v^{'}$, the remainder of it becomes a gap ring R_v^g. Two end points u_1 and u_2 of R_v^g together with node v and sensor $u^{'}$ construct a quadrilateral $Q_{u^{'}}$. There is a gap between node v and sensor $u^{'}$, which means a coverage hole in the ROI of $Q_{u^{'}}$. If else, node v and sensor $u^{'}$ should be directly connected and $u^{'} \in N_v^1$. When the quadrilateral $Q_{u^{'}}$ contains a hole, a part of circumference c_v is not covered by any sensor. This violates the definition of inner nodes. The same result also applies in a ring $R_v^{''}$ which contains any node $u^{''} \in N_v^k, k > 2$. □

From the above proposition, we know that one method to distinguish the inner nodes from the boundary sensors is to find out if there is a ring in N_v^1. However, there are some difficulties. The first one is we cannot affirm the number of nodes on the ring around sensor v (length of the ring). The second comes from the fact that it is hard to confirm if the ring we found out is surround sensor v (position of the ring), without coordinates. Trying to solve the first problem, we may examine that if all the nodes in N_v^1 can build a ring by finding out a Hamiltonian circle in a graph HNG_v^1. The graph is constructed by all the nodes

in N_v^1 and the connections between sensors. However, searching for a Hamiltonian circle in a graph is an NP problem, it is not promising to provide a solution under limited resources when $|N_v^1|$ is large in a dense network. Therefore, we put our focus on the second problem. If we first discover a ring R and then check out if R is surrounds v or locates beside v, it is difficult. But, if we only focus on searching a ring R surrounds v, it is relatively much easier.

3.2 Triangles in the Neighborhood

Theorem 1. *Given a node v, if v is an inner node, there is a triangle T_v formed by the nodes on ring R_v surrounds v.*

Proof. From Proposition 1, we obtain a ring R_v surrounds v. There are at least three nodes on R_v, otherwise no ring can be formed. We randomly pick out three sensors u_1, u_2, and u_3 on R_v to construct a triangle T_{test}. We enumerate all the potential triangles by finding out all combinations of the vertices on R_v. For each test triangle, we do Point in Triangle (PIT) test to check if it contains v. If none of them fits, the area of R_v does not cover v, R_v locates beside v. □

Corollary 1. *Given a node v and a set $S_v = N_v^1 - \bigcup R_v$, if v is an inner node, either the elements in S_v forms no triangle or none of the triangles contains v.*

Proof. Assuming there exists a triangle $T^{'}$ built by nodes in S_v contains v, nodes on the three vertices of $T^{'}$ form a ring R_v^*. As $R_v^* \subseteq \bigcup R_v$, any sensor $u^{'}$ on the vertex of $T^{'}$ is not in S_v. □

From the above we know, if we obtain a triangle T_{test} built by nodes on N_v^1 that contains v, we may identify v as an inner node. The uncertainty comes from the possibility that there may exist an 'U' shaped ring locates beside the checking node. Two ends of the U-ring with another points on it may form a triangle that containing v.

3.3 Difference Between Triangles

Definition 4 (Gap Edge). *Given a U-shape ring R_v^U constructed by the nodes in N_v^1 of sensor v, the connection between two ends $p_{R_v^U}$ and $q_{R_v^U}$ of R_v^U provides an edge e_{pq} that filling the gap of the U-ring. Such edge e_{pq} is a gap edge.*

A triangle T_{test} built with a gap edge e_g may mislead an algorithm that identifying inner nodes by searching triangles containing a checking sensor. Thus, we need to remove these triangles (Fig. 1).

Lemma 1. *Given a boundary sensor v, a triangle T_v containing v is formed by the nodes on a U-ring R_v locates beside the node. Gap edge E_v on T_v divides area around v into two parts A_v^1 and A_v^2. Nodes u_1, u_2, and u_3 are the three vertices on T_v. Among them, u_1 and u_2 are the end points of E_v. The shortest path between u_1 and u_2 is on the same side of the area that containing T_v.*

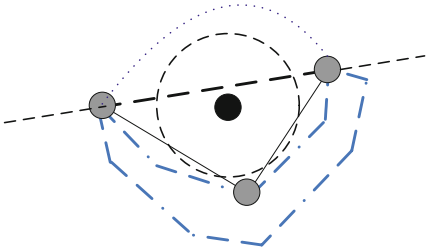


Fig. 1. An 'U' shaped ring locates beside the checking node (colored black). Three nodes on the U-ring (colored gray) build a triangle that containing the checking sensor. There is also a gap edge (black dotted line) built by two end nodes on the U-ring. The edge divides area around the checking sensor into two parts. There should be no shortest path between the end points of the gap edge on the opposite side of the U-ring.

Proof. We assume area A_v^1 contains T_v. If the shortest path $P_{u_2}^{u_1}$ between u_1 and u_2 is in area A_v^2, there exists at least one node p_v on the path. Then, the two edges other than E_v on T_v form a ring $R_v^{'}$, together with the path $P_{u_2}^{u_1}$. The area of $R_v^{'}$ includes the area of T_v. Since T_v contains v, the ring $R_v^{'}$ contains v. Node v is an inner sensor. □

Theorem 2. *Given a boundary sensor v, a triangle T_v containing v is formed by the nodes on a U-ring R_v located beside the node. Edge e_v on T_v is the gap edge. Nodes u_1, u_2, and u_3 are the three vertices on T_v. And the former two nodes are end points of e_v. Angle $\angle u_2u_1u_3$ is a side angle of T_v. Randomly pick a node u_r on the shortest path $P_{u_2}^{u_1}$ between u_1 and u_2. The value of angle $\angle u_ru_1u_3$ is no greater than the value of $\angle u_2u_1u_3$.*

Proof. The two end points on E_v divide the U-ring R_v into two parts R_v^1 and R_v^2. Since R_v is U-ring, the length of R_v^1 and R_v^2 is hardly to be equal. We assume R_v^1 is the longer one. If u_3 is on R_v^1 and $P_{u_2}^{u_1}$ is on R_v^2, it is easy to verify the result. The same situation holds when u_3 is on R_v^2 and $P_{u_2}^{u_1}$ is on R_v^2. □

According to the above analysis, we propose our distributed coordinate-free boundary nodes identification algorithm based on point in triangle test and angle comparison.

4 Boundary Nodes Identification

Our distributed algorithm contains three parts. It first finds out all the potential triangles constructed by the 1-hop neighbors of a checking sensor. Then, it executes PIT for each triangle to test if it contains the node. At last, the triangles containing gap edges are removed by angle comparison.

In the algorithm, each sensor is initially set as an inner node. A node v collects the 1-hop neighbors of it by communication with directly connected

neighbors. In this process, if $|N_v^1| < 3$, no ring can be built by nodes in the 1-hop neighborhood of v, thus v can be directly set as a boundary node. After the execution of this algorithm, each node v obtains a list of triangles, the vertices on them are the nodes in N_v^1.

4.1 Point in Triangles Test

Next, we do PIT test for each item in L_T to examine if it contains the checking sensor.

Algorithm 1. Point in triangle test by angles

```
Input: node v, triangle T, angle φ = 0.
Output: true or false.
Initially, output = false;
for each edge e in T: do
    Get the two end points p and q on e.
    Obtain value of the angle ∠pvq.
    φ = φ + ∠pvq.
end for
if φ = 2π then
    output = true.
end if
```

Since sensor v can obtain the value of an angle formed by it with other two nodes in N_v^1 and a triangle T in L_T is built by the nodes in N_v^1, for any two nodes p and q on T, we can obtain value of angle $\angle pvq$. We add up the value of the angles in T. If T contains v, the result should be 2π. Otherwise, T does not contain v. This is ensured by axiomatic geometry.

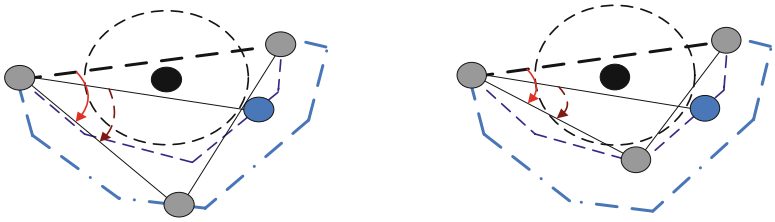
Each triangle T that contains node v is added into a list L_T^c. Then, we remove the items in L_T^c that containing gap edges (Fig. 2).

4.2 Remove the Triangle that with a Gap Edge

In the last step, we remove all the invalid triangles in L_T^c. We use a Test of Triangle Validity (TTV) algorithm to check the validity of each triangle in L_T^c. After running TTV, if $|L_T^c| > 0$, which means there exists at least one valid triangle, the checking sensor is an inner node. Otherwise, it is a boundary node.

Definition 5 (Valid Edge). *Given an edge e, if e is not a gap edge, e is a valid edge.*

Definition 6 (Valid Triangle). *Given a triangle T, if all the edges in T are valid edges, T is a valid triangle.*



(a) The third vertex on the triangle containing the gap edge is on the longer segment of the divided U-ring.

(b) The third vertex on the triangle containing the gap edge is on the shorter segment of the divided U-ring.

Fig. 2. Angle comparison in triangles with gap edge. The blue dot is the sensor selected randomly on the shortest path between two nodes on the end of a gap edge. (Color figure online)

Algorithm 2. Test of triangle validity

1: Input: node v, triangle T.
2: Output: valid or invalid.
3: Initially, the validity V_T of T is $V_T = valid$.
4: **for** each edge e in T: **do**
5: Set the initial validity V_e of e, $V_e = valid$.
6: Get the two end points p and q on e.
7: Get the third vertex w on T.
8: Calculate value of the angle $\angle qpw$ by the Cosine Theorem.
9: Find the shortest path P_q^p between node p and q within the set N_v^1.
10: Pick a random node u on path P_q^p.
11: Calculate value of the angle $\angle upw$ by the Cosine Theorem.
12: **if** $\angle upw \leq \angle qpw$ **then**
13: $V_e = invalid$.
14: **end if**
15: $V_T = V_T \wedge V_e$.
16: **end for**

In step 8 and 9 of TTV, we use the Cosine Theorem to calculate the value of an angle and this requires values of distance between two nodes. Given node p and q, if they are directly connected, by evaluating the RSSI value between them, d_{p-q} can be obtained. If node p and q are not directly connected, we can calculate d_{p-q}, as $p \in N_v^1$ and $q \in N_v^1$. We first obtain the distance d_{p-v} between node p and v. Then, we get the distance d_{q-v} between node q and v. At last, with the help of angle $\angle pvq$, we calculate the distance d_{p-q}.

4.3 Discussion

Performance Evaluation. Each sensor v in our distributed algorithm sends out and receives $O(n)$ messages, $n = |N_v^1|$. The computational complexity of it

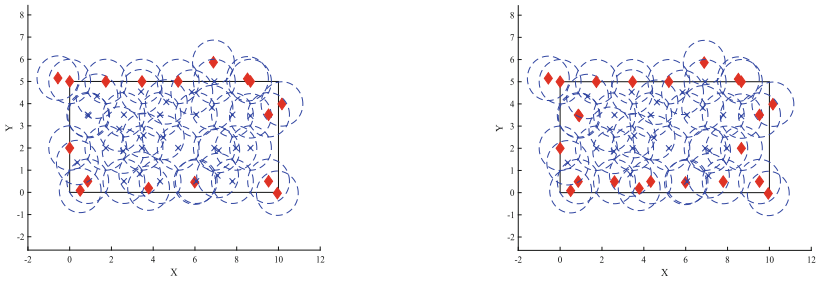
for each sensor v contains three parts. The first one comes from enumerating triangles in N_v^1, it is $O(n^3), n = |N_v^1|$. The second part is PIT, it contributes $O(n), n = |L_T|$. The main contribution of computational complexity in TTV comes from the search of a shortest path between two nodes. Depends on the breadth first algorithm used on the graph HNG_v^1 constructed by all the nodes in N_v^1 and the connections between sensors, the complexity of TTV can be up to $O(mn^2)$, $m = |L_T^c|$, n is the number of vertices in HNG_v^1.

Algorithm Improvement. We do not need to visit all the triangles in L_T^c. If L_T^c contains at least one valid triangle, the checking sensor is an inner node. Thus, the algorithm only needs to visit the triangles in L_T^c one after another, if it meets a valid triangle, it stops.

Due to the lack of coordinates, it is hard to locate triangles containing the gap edges precisely by TTV, especially in a network with random sensor deployment. And interferences may be brought in TTV by the accuracy problem in floating point calculation. In order to refine the result, if a sensor is set as an inner node after running TTV and it has at least two 1-hop neighbors as boundary nodes, it checks the number of connected components of a graph built by its boundary neighbors. If the graph has more than one connected components, then the checking sensor should be a boundary node.

5 Simulations

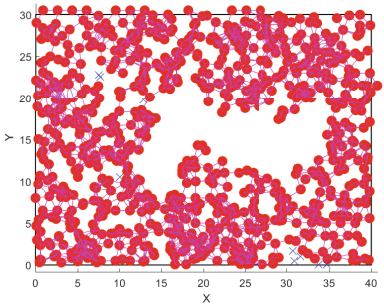
We first build a 5 * 10 small ROI and randomly deploy 48 nodes in it. The network fully covers the ROI. Figure 3(a) shows that due to random position of sensors our algorithm without refinement has a little difficulty on fully distinguish the boundary nodes. After refinement, the defects can be compensated.



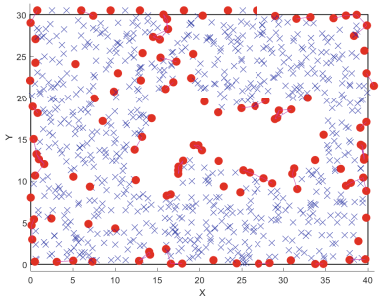
(a) Algorithm without refinement. (b) Algorithm with refinement.

Fig. 3. Boundary nodes identification on a small random network. Red diamonds show the boundary nodes.

In order to further test the performance of our algorithm, we build a 40 * 30 ROI containing 848 randomly deployed nodes. Among these sensors, 524 of them



(a) BDCIS performance. Boundary nodes are overly picked out.



(b) VCSTPM performance. Inner nodes are overly identified.

Fig. 4. Boundary nodes identification on a random network. (Color figure online)

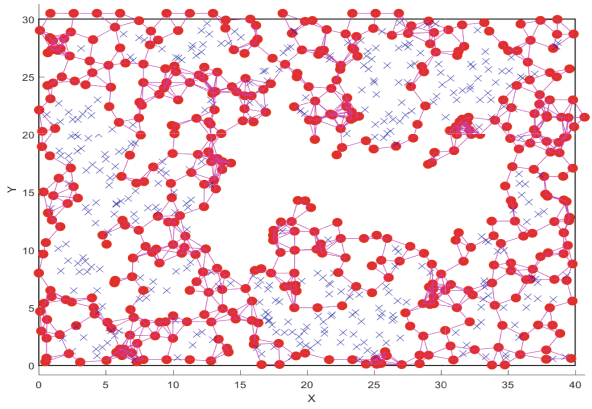


Fig. 5. Our algorithm performance. Most of the boundary nodes and inner ones are correctly distinguished. (Color figure online)

are boundary nodes. In this network, there is an irregular shaped coverage hole at the center, together with several small holes randomly scattered in it. In the simulation results, the red dots are the boundary nodes and the light blue crosses are the inner ones. Our algorithm correctly identified 450 boundary sensors and 304 inner nodes (Fig. 5). The correctness rate is 85.8% for the boundary nodes and 93.8% for the inner ones. The boundaries of most of the small holes are also correctly depicted. However, we see from Fig. 4 that BDCIS overly picked out too many boundary nodes (Fig. 4(a)) and VCSTPM overly identified too many inner sensors (Fig. 4(b)). Though both of them show their efficiency in distinguishing the boundary of the center hole and the frontier of the entire network, they can hardly recognize the random small holes. Due to lack of space, other simulation results are omitted in this part.

6 Conclusions

In this paper we studied the problem of identifying boundary nodes without coordinates in a distributed fashion. We analyzed the feature of an inner sensor and proved that each inner node has a triangle containing it. We extracted the difference between triangles containing inner nodes and the ones containing boundary nodes next to U-rings. According to these analyses, we presented a three-step algorithm to identifying the boundary sensors. The effectiveness of it has been verified by simulation results. Our future work will be focus on improving this algorithm, reducing the usage of angles, and studying problems in three dimensional environments.

References

1. Shu, L., Mukherjee, M., Wu, X.: Toxic gas boundary area detection in large-scale petrochemical plants with industrial wireless sensor networks. IEEE Commun. Mag. **54**, 22–28 (2016)
2. Senouci, M., Mellouk, A., Assnoune, K.: Localized movement-assisted sensor deployment algorithm for hole detection and healing. IEEE Trans. Parallel Distrib. Syst. **25**, 1267–1277 (2014). ISSN: 1045-9219
3. Fang, F., Gao, J., Guibas, L.: Locating and bypassing holes in sensor networks. Mobile Netw. Appl. **11**, 187–200 (2006)
4. Li, W., Zhang, W., Sneddon, I.N.: Coverage hole and boundary nodes detection in wireless sensor networks. J. Netw. Comput. Appl. **48**, 35–43 (2015)
5. Fekete, S.P., Kaufmann, M., Krooller, A., Lehmann, K.: A new approach for boundary recognition in geometric sensor networks. In: Proceedings of the 17th Canadian Conference on Computational Geometry, pp. 82–85 (2005)
6. Kroller, A., Fekete, S.P., Pfistere, D., Fischer, S.: Deterministic boundary recognition and topology extraction. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, SODA 2006, Miami, pp. 1000–1009 (2006)
7. Wang, Y., Gao, J., Mitchell, J.: Boundary recognition in sensor networks by topological methods. In: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking, MobiCom 2006, Los Angeles, pp. 122–133 (2006)
8. Beghdad, R., Lamraoui, A.: Boundary and holes recognition in wireless sensor Networks. J. Innov. Digit. Ecosyst. **3**(1), 1–14 (2016)
9. Dhanapala, D., Jayasumana, A., Mehta, S.: On boundary detection of 2D and 3D wireless sensor networks. In: 2011 IEEE Global Telecommunications Conference - GLOBECOM 2011, Houston, pp. 1–5 (2011)