# Boundary Detection in Dynamic Wireless Sensor Networks using Convex Hull Techniques

Tapas Kumar Mishra, Jayadeep Sadhu, and Arun Kumar
Department of Computer Science and Engineering
NIT Rourkela, Odisha, 769008
mishrat@nitrkl.ac.in, 217cs1096@nitrkl.ac.in, kumararun@nitrkl.ac.in

*Abstract* - **Perimeter protection aims to detect intrusions: these intrusions may range from agricultural fields to army base camps. The main problem concerning perimeter protection is edge or boundary detection. In this paper, two boundary detection algorithms have been proposed that use convex hull techniques to find the boundary of a static and dynamic wireless sensor network (WSN). Whenever a boundary node on a static WSN has to be replaced, our method chooses a new neighbor such that the overall area covered by the boundary is maximized. In the dynamic case, we demonstrate that the boundary of a dynamic WSN can be maintained in $O(log^2 n)$ time, where $n$ is the number of sensor nodes in the network, maximizing the coverage area of the network at all times.**

## I. INTRODUCTION

*Wireless sensor network* (WSN) refers to a group of dedicated sensors distributed in a region for studying the physical conditions of the environment, i.e. monitoring, collecting and recording the environmental parameters. The environmental conditions of interest are temperature, sound, pollution levels, humidity, wind, and so on. A central facility is used for organizing the data captured by WSNs. Each node in a sensor network includes (1) a small battery, (2) a sensing module, (3) a microprocessor to process data read by the sensors, (4) a memory to temporarily hold sensor data, and (5) an interface for communication among network nodes. Typically, a WSN consists of hundreds or thousands of these wireless sensors and the low cost helps in such deployment.

Boundary detection in WSNs act as tool for perimeter protection; consider the following concrete example. When an agricultural field is attacked by pests and insects, it is desirable to know which areas are affected such that effective decisions for spraying of pesticides and insecticides can made in real time. In order to achieve that, UAVs are used to spread sensors over the field. Once the sensors land in the field, (a) the sensors organize themselves to form the network, (b) start sensing, collecting the required data, and (c) dispatching them back to the sink for effective decision-making. The sensor nodes right on the boundary act as indicators for the point of an event. These sensor nodes on the boundary of the network usually have sudden changes in readings of the parameters compared to other sensors not in the boundary. Thus, the location of these boundary nodes is sufficient to solve the problem of boundary detection for sensor networks. There are four broad classes of boundary detection algorithms.

*Localized boundary detection*: The data collected from the neighboring nodes of any sensor node act as indicator for the location of a sensor node: whether it is positioned on or near a boundary. It is challenging to design localized boundary detection algorithms; however the energy consumption is very low as compared to other algorithms due to localized communications, see [1], [2] for detail.

*Centralized boundary detection*: Every sensor node in the network share their data with the sink (which is more powerful as compared to other nodes in the network). It examines these data and precisely locates the boundary. However, the data collection cost can be cumbersome if the region of interest spans a huge geographical area, refer to [3] for detail.

*Hierarchical boundary detection*: Since there is a trade off between energy consumption and accuracy, in order to accommodate these, several hierarchical boundary detection algorithms have been proposed. The WSN is organized into multiple levels of sensor nodes. The child sensor nodes send the data gathered to the parent nodes, which estimates the boundary based on the data collected.

*Distributed algorithms*: These algorithms dynamically detect and track large-scale phenomena; we present a distributed algorithm as proposed in [7] shortly, see [4], [5], [10], [11], [12], [13] for other details.

Renold, A.P and Chandrakala. S et al. [7] propose a *distributed* algorithm and a *centralized algorithm* which try to find the perimeter of the network in different ways. Upon failure of a sensor node on the perimeter, these methods do not focus to obtain maximum possible area at all times. In this paper, methods to handle the network upon failure of a node, i.e, the residual energy of a node falling below the required threshold or the sensor node moving away from the network etc. have been discussed in detail. Moreover, when nodes are mobile, while all the previous algorithms fail, our proposed algorithms works smoothly. After inserting a node into the network or deleting a node from the network, our proposed method efficiently updates the boundary using a divide and conquer approach.

This paper proposes two boundary detection algorithms using convex hull techniques to find the boundary of a static and dynamic WSN. The proposed algorithm addresses the drawbacks of [7] by optimizing the perimeter detection algorithm, so that the sensor network can cover the maximum possible area at all times. Also, the proposed work is extended to compute the perimeter of the WSN even if the nodes are mobile.

The remainder of this paper addresses the problem of boundary detection while avoiding shortcomings of [7] and is organized as follows. Section II discusses various methods for boundary node replacement. Section III provides results boundary detection in a dynamic wireless sensor network. Section IV discusses the results obtained followed by the Conclusion.

## II. Boundary detection in a Static Wireless Sensor Network

In what follows, we discuss the distributed convex hull algorithm as given in [7] .

A location request (LOC_REQ) message is sent by the *initiator* node to every other node in the network. The nodes reply with their corresponding location information. Then, the coordinates are sorted by the initiator node based on their $x$-coordinates and the node with least $x$-coordinate value is informed to begin the formation of convex-hull. Among all neighbors of the node (with least $x$-coordinate value), the convex-hull construction process chooses the node which falls in the counter clockwise direction of the current node. This iteration continues until all the nodes in the boundary of the network are discovered.

The only difference between Distributed and Centralized algorithms is that, in Distributed algorithm, the residual energy of the sensor nodes is checked after the formation of the boundary whereas, in Centralized algorithm, residual energy is checked prior to the boundary formation. For computing the convex hull, Jarvis march is used which has a running time of $O(n*h)$ , where $h$ is the number of nodes in the convex hull. The complexity of Distributed algorithm is same as of Jarvis march algorithm. Once, the convex hull of the network is formed, the algorithm monitors the residual energy of each sensor node on the perimeter if it meets the required threshold. If the sensor node's energy falls below this threshold, one needs to find a new neighbor to replace the failure node.

The proposed neighbor selection algorithms are as follows.

### A. *Nearest neighbor method (NN)*

In this method, euclidean distance is used as a metric to determine the nearest neighbor to the convex node that has to be replaced.

### B. *Minimum sum of angles 1 (MSoA 1)*

Among the neighbor nodes of the convex node, the node which gives the value of $(\theta 1 + \theta 2)$ the minimum is selected as shown in the Fig. 1.

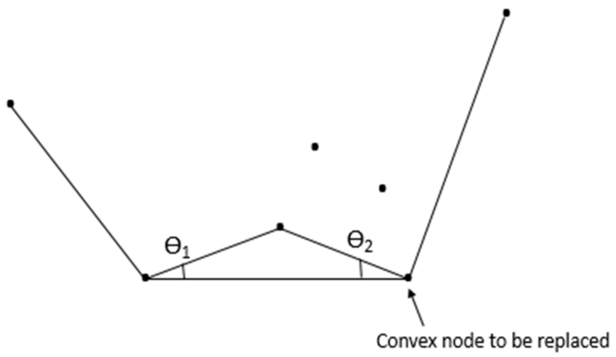

Fig. 1. Neighbor selection using minimum sum of angles 1

### C. *Minimum sum of angles 2 (MSoA 2)*

Among the neighbor nodes of the convex node, the node which gives the value of $(\theta 1 + \theta 2)$ the minimum is selected as shown in the Fig. 2.

### D. *Areas method*

Here, among the neighbor nodes of the convex node, the node which yields maximum convex hull area is selected. This method will yield the maximum possible area in all scenarios.

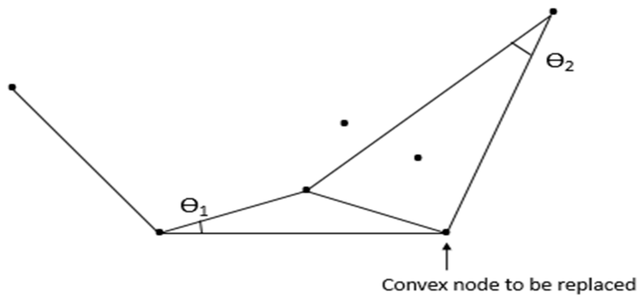

Fig. 2. Neighbor selection using minimum sum of angles 2

## III. Boundary detection in a Dynamic Wireless Sensor Network

Mobility of the sensor nodes in a WSN can be handled using dynamic convex hull strategies when analyzing the perimeter of the network. Unlike static method, where area optimization techniques are used to find the maximum possible area c ed by the sensor nodes, dynamic convex hull gives the maximum possible area by default. The authors in [9] demostrate how to maintain the convex hull of a set of $n$ nodes in $O(log^2 n)$ time after inserting a node into the WSN or deleting a node from it in a divide and conquer approach. Convex hull is maintained by considering it as a union of two structures: *lc-hull* and the *rc-hull*. The lc-hull (rc-hull) is a convex arc which begins at the rightmost point of highest $y$-coordinate and ends at the rightmost point of lowest $y$-coordinate and tightly bounds the set from the left (respectively, from the right). The whole sensor network can be thought of as a union of it's lc-hull and rc-hull.

## Data structure

A concatenable queue data structure is used to store the lc and rc-hulls. A concatenable queue can process the following operations: Insert, Delete, Search, Concatenate, Split. A 2-3 Tree can be used to implement a concatenable queue. A 2-3 Tree is a tree in which every node that is not a leaf has 2 or 3 children, and every path from the root to a leaf is of the same length.

The co-ordinates of the sensor nodes are sorted by their y-coordinate and are assigned to the leaves in left-to-right order. To dynamically maintain a convex hull in a 2-3 Tree, at each internal node, the following extra information other than the points which constitute to lc-hull and rc-hull are stored.

1. f($\alpha$) = a pointer to the father of $\alpha$ (if any)
2. lson($\alpha$) = a pointer to the left son of $\alpha$,
3. rson($\alpha$) = a pointer to the right son of $\alpha$,
4. max($\alpha$) = the largest y-value of the points in the subtree of lson($\alpha$),
5. The points 's' and 't' where the bridge connects the lc-hulls of its children. The line segment $\overline{st}$ is called bridge B.
6. A balance factor that indicates the difference in heights of the left and right subtrees (its value is -1 if the left subtree is bigger and 1 if the right subtree is bigger).

## Construction of the lc-hull

The sensor network is splitted into two halves, A and C, by a horizontal line (bridge) parallel to the x axis. The lc-hull of A, the lc-hull of C and the bridge b between the hulls forms the the lc-hull of the complete network (see Fig. 3). This gives a recursive definition of the lc-hull.

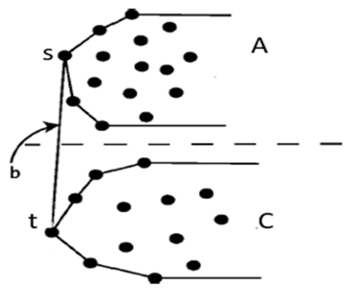

Fig.3. Constructing the lc-hull of a set of points.

In order to obtain the lc-hull of the parent node, one needs to determine the two points p and q on the lc-hulls of A and C respectively. This is acheived by drawing a tangent to both the lc-hulls of A and C as shown in the Fig. 4. But for an efficient practice, a binary search is performed on both the lc-hulls. $p = p_{[m/2]}$ and $q = q_{[n/2]}$ act as common tangent or not. If it doesn't form a common tangent, the algorrithm determines on which segment to continue the search and neglect the other part as shown in the Fig. 5.

Repeatedly choosing $p$ and $q$ in the middle of the remaining parts of A and C enables the algorithm to find u and d, hence B, in $O(logn)$ time.

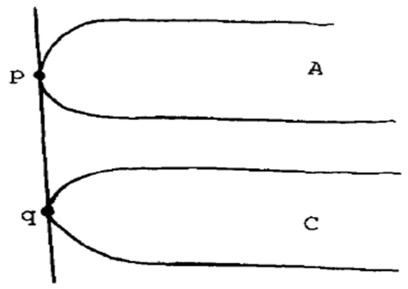

Fig.4. Case where the bridge is a common tangent to A and C.

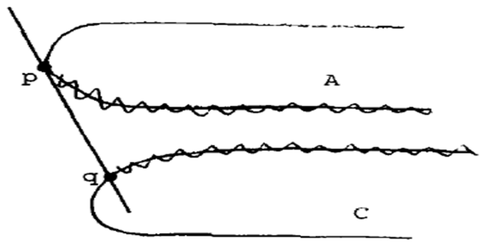

Fig.5. A case where the bridge is not a common tangent to A and C.

## Dynamically maintaining the Sensor Network

Whenever one tries to insert/delete a point in a dynamic convex hull, two routines (DOWN, UP) are invoked which update the hull. Using the same procedures, one achieves the dynamic maintainance of the sensor network.

Firstly, for every point that is inserted into the network, the algorithm finds a place in the tree to insert, which in our case is a binary search tree (BST). Each node in the BST is a concantenable queue, which has lc-hull and rc-hull of their children nodes.

Assume for the sake of demonstration that we are trying to insert/delete a sensor node whose $y$-coordinate is $k$, then the following steps take place when $k$ is inserted.

1. The location of insertion of the node is determined.
2. Once the position is known (say $x$), the path from root to $k$ is determined.
3. While we traverse down the tree in the search of $x$, at every node, we split the node (concatenable queue)

along its bridge. Thus, the bridge divides the lc-hull of its children into left subtree and right subtree.

4. Left and right subtrees are glued back to its children and the search is continued.

5. Steps 3 & 4 are repeated until '$x$' is reached.

6. Once $x$ is reached, after inserting $k$ at $x$, we traverse back to the root, aiming to re construct the lc-hulls that were split during the insertion process. This step is essential because, once the sensor node is inserted, updating the lc and rc-hulls determines if the inserted node belongs/does not belong to the boundary of the existing network.

7. While re-constructing, a check is performed to know whether the tree is balanced or not at each node. Next, we find the new bridge for the two lc-hulls, join them by determining the bridge as shown in the Fig. 3, so that the lc-hull of the parent has the lc-hulls of the updated children node(s).

8. The process is exited once we reach the root.

This completes the procedure when a point is being inserted into the existing network; deleting a node from the network is similar to insertion.

## IV. Results

The Areas(sq. units) obtained, boundary formation (ms) time using Distributed algorithm with number of nodes as 10 is given in Table.1. The Areas(sq. units) obtained in , boundary formation (ms) time using Centralized algorithm with number of nodes as 30 is given in Table. 2.

Table I

AVERAGE AREAS,TIMES FOR DISTRIBUTED CONVEX-HULL ALGORITHM.

| No. of nodes | 30 | |
|---|---|---|
| Avg over 20 times | Avg area | Avg time |
| NN | 481.33 | 0.34 |
| MSoA1 | 502.33 | 0.71 |
| MSoA2 | 502.33 | 0.51 |
| Areas | 504.66 | 0.72 |

Table II

AVERAGE AREAS,TIMES FOR CENTRALIZED CONVEX-HULL ALGORITHM.

| No. of nodes | 30 | |
|---|---|---|
| Avg over 20 times | Avg area | Avg time |
| NN | 481.33 | 0.06 |
| MSoM1 | 502.33 | 0.07 |

To demonstrate the perimeter detection in the case of a dynamic sensor network, we've assmued 10 sensor nodes deployed at (0.6,5.1), (5.3,2.4), (11,1.1), (12.9,3.1), (13.8,7.3), (13.2,11.9), (9.5,14.9), (6.7,15.25), (4.4,14), (2.1,11.1). The convex-hull formed using these nodes are shown in the Fig. 6.

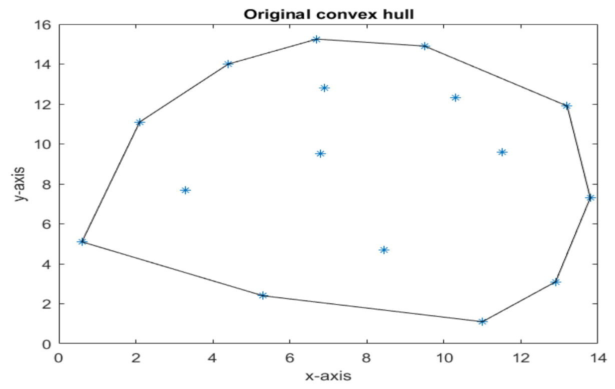

Fig. 6. Original perimeter of the sensor network.

Upon adding the sensor at the point (2,2.5) the sensor network is updated as shown in the Fig. 7.

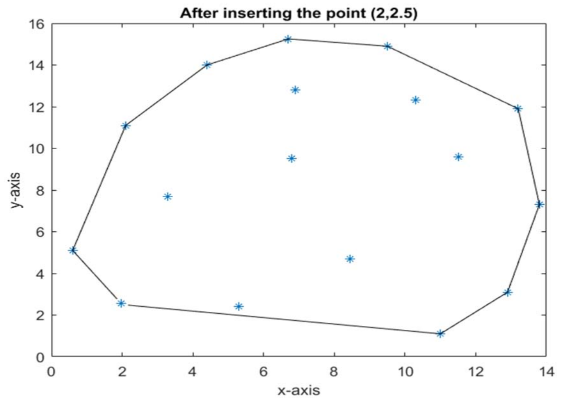

Fig.7. Sensor network after adding a node at (2,2.5).

Now, Upon deleting the node at (11,1.1) the sensor network is updated as shown in the Fig. 8.

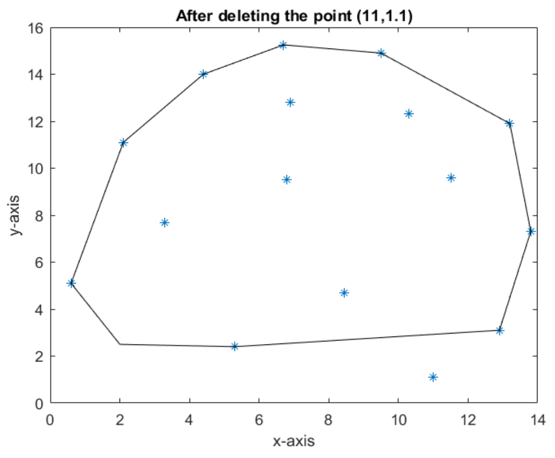

Fig. 8. Sensor network after deleting the node at (11,1.1).

## CONCLUSION

This paper has presented two boundary detection algorithms using convex hull techniques to find the boundary of a static and dynamic WSN. The paper has presented and addressed a few drawbacks of by optimizing the perimeter detection algorithm, so that the sensor network can cover the maximum possible area at all times. The work has been extended to compute the perimeter of the WSN even if the nodes are mobile. In future, we plan to extend the work to vehicular adhoc networks (VANETs) and analyse the results in detail.

## REFERENCES

[1] Krishna K. Chintalapudi and Ramesh Govindan, 2003. “Localized edge detection in sensor fields”. Ad-hoc Networks Journal.
[2] Min Ding, Dechang Chen, KaiXing, and Xiuzhen Cheng, 2005. “Localized fault-tolerant event boundary detection in sensor networks”. IEEE INFOCOM.
[3] Jie Liu, Patrick Cheung, Leonidas Guibas, and Feng Zhao, 2002. “A dual-space approach to tracking and sensor management in wireless sensor networks”. International Conference on Mobile Computing and Networking, pages 131–139.
[4] David Moore, John Leonard, Daniela Rus, and Seth Teller, 2004. “Robust distributed network localization with noisy range measurements”. Proceedings of the sec ond ACM Conference on Embedded Networked Sensor Systems.
[5] Anand Panangadan and Gaurav S. Sukhatme, 2005. “Data segmentation for region detection in a sensor network”. International Conference on Dis- tributed Computing in Sensor Systems (DCOSS).
[6] Misra, S., and Jain, A., 2011. “Policy controlled self-configuration in unattended wireless sensor networks”. J. Netw. Comput. Appl., 34(5), Sept., pp. 1530–1544.
[7] Renold, A. P., and Chandrakala, S., 2017. “Convex-hull-based boundary detection in unattended wireless sensor networks”. IEEE Sensors Letters, 1(4), Aug, pp. 1–4.
[8] Jarvis, R., 1973. “On the identification of the convex hull of a finite set of points in the plane”. Information Processing Letters, 2(1), pp. 18 – 21.
[9] Overmars, M. H., and van Leeuwen, J., 1981. “Maintenance of configurations in the plane”. Journal of Computer and System Sciences, 23(2), pp. 166 – 204.
[10] Kumar, A. , Shwe, H. , Wong, K. and Chong, P., 2017. "Location-Based Routing Protocols for Wireless Sensor Networks: A Survey", Wireless Sensor Network, 9, pp. 25-72. doi: 10.4236/wsn.2017.91003.
[11] Kumar, A., Zhao, M., Wong, K-J., Guan, L. Y., Chong, Peter H. J., 2018. "A Comprehensive Study of IoT and WSN MAC Protocols: Research Issues, Challenges and Opportunities", IEEE Access, vol. 9, pp. 76228 - 76262.
[12] Shwe, H-Y., Wang, C., Chong, Peter H. J., Kumar, A., 2013. "Robust cubic-based 3-D localization for wireless sensor networks", Wireless Sensor Network, vol. 5, pp. 169-179.
[13] Zhao, M., Kumar, A., Ristaniemi, T., Chong, Peter H. J., 2017. "Machine-to-Machine Communication and Research Challenges: A Survey", Wireless Personal Communications - Springer, vol. 97, Issue 3, pp. 3569–3585