A Self-Deployment Obstacle Avoidance (SOA) Algorithm for Mobile Sensor Networks

Bryan Sarazin bsarazin@bridgeport.edu
Department of Computer Science and Engineering
University of Bridgeport
Bridgeport, 06601, USA

Syed S. Rizvi srizvi@bridgeport.edu
Department of Computer Science and Engineering
University of Bridgeport
Bridgeport, 06601, USA

Abstract

A mobile sensor network is a distributed collection of nodes, each of which has sensing, computing, communicating, and locomotion capabilities. This paper presents a self-deployment obstacle avoidance (SOA) algorithm for mobile sensor networks. The proposed SOA algorithm provides full coverage and can be efficiently used in a complex, unstable, and unknown environment. Moreover, the SOA algorithm is implemented based on the assumption that nodes are randomly deployed near the sink where each node knows the location of the target. In proposed SOA algorithm, the nodes determine a partner node and link up effectively to form a node pair. A node pair which is closest to the target searches for the target with all other node pairs following the previous node. There are number of priority rules on which the mobility of sensor nodes is based. The SOA algorithm ensures that the nodes determine a path around any obstacles. Once a connection is established from the sink to the target, the node pair separates and starts providing the full coverage. The experimental verifications and simulation results demonstrate that the proposed algorithm provides three main advantages. First, it reduces the total computation cost. Second, it increases the stability of the system. Third, it provides greater coverage to unknown and unstable environment.

Keywords: Mobile nodes, Mobile networks, Self deployment, Sensor networks.

1. INTRODUCTION

The purpose of a mobile sensor network is to provide a reliable connection from sink to target and perform some form of information gathering. Wireless sensor networks provide different functions in a variety of applications including environmental monitoring, target tracking, and distributed data storage. A basic problem faced by the current sensor network is the need of an efficient deployment of sensor nodes that can provide the required coverage [1], [13]. In some situations, the tasks put forward higher requirements; they not only need a connection, but also require the connection to be efficient and secure. If the environment changes or a hostile environment can not guarantee the security of sensors, resulting in damage to sensors, or loss of contact with sensors, the entire system still has to ensure the realization of the most basic functions.

For instance, a mobile sensor network used in natural disaster relief such as earthquake, a safe route through hazardous terrain may need to be determined. The environment is complex and variable, and may continually change. There may be any number of unknown obstacles within this environment, with the possibility that they may shift or move. Therefore, in this defined area, we can not know the state of the environment, all sensors must be able to locate obstacles at run time and be able to negotiate them. The sensing and computation must be efficient [1] [2] since the response time is pressing in natural disaster relief. If it takes too long, the value of such a system is lost. This implies that, for each of the sensors to sense, perform computation, and then communicate with each other is inefficient [3] [11] [12]. Another case is in military applications such as target detection. The sensors should provide detection of the enemy in a given area. In this application, coverage is vital. If coverage criteria cannot be met, the enemy may not be detected, rendering the network virtually useless.

There are a number of problems associated with current mobile sensor networks. For instance, how can nodes provide sensing capability, how do we make computation and locomotion efficient, and how do the sensors create a stable connection while providing coverage? The proposed SOA algorithm provides solution to these problems. First, we assume that all nodes are randomly deployed near the sink. Each node has a priority based on its relative position to the sink, the target, and all other nodes. The nodes interact with each other to construct node pairs based on priority where each node pair effectively moving as a single node. Only the node pair with the highest target priority begins moving towards the target. The node pair with the second highest target priority follows the first pair and so on. Each node pair stays within communication range of the pair with higher target priority and higher sink priority. Only the node pair with the highest target priority performs computation to determine movement while the other node pairs simply follow the pair with higher priority. The proposed SOA algorithm shows a significant reduction in the number of computations that each sensor node has to perform in order to locate the position – thus it provides an efficient and faster way to calculate the position.

When the first node pair encounters an obstacle, it does three things. First, it calculates the range to the obstacle. Second, it determines the direction to avoid the obstacle. Third, it negotiates with the obstacle. Once the target is reached, the node pairs separate to provide coverage and connection reliability. We assume that the radius of the coverage area that each node provides is r whereas the amount of sensors in a combination (referred to as a pair) is assumed to be n. Taking these parameters into account, the whole mobile sensor network can cover an area of a width up to $n*r$.

Coverage criteria may be met by defining the number of nodes paired together. We can control the distance of separation and adjust this distance to meet our requirements. One of the nodes can keep communicating with all surrounding nodes, ensuring the connection is maintained even during the separation period (i.e., it shows a strong connection). Otherwise, the node can maintain a connection with at least two other nodes. The strong connection can make the mobile sensor network more stable and secure, because if one of the nodes is destroyed, its neighboring nodes can maintain communication with the other nodes. The strong connection could be used in a hazardous environment, such as on a battle field or in natural disaster relief. In this environment, the nodes could be easily damaged, but the mobile sensor network is pivotal, so it must keep working despite the loss of nodes.

2. PROBLEM FORMULIZATION

The goal of this research work is to develop an algorithm for self-deployment of a mobile sensor network which has the ability to build an uninterrupted wireless connection between the sink and the target while at the same time provides coverage to a certain area within an unknown environment. To achieve this goal, we use the moving algorithm for self-deployment of a mobile sensor network.

The moving algorithm is based on the connection built between multiple nodes, communication range, and the direction of movement of each node. Each node finds a suitable position in the unknown environment to ensure successful deployment. The nodes should have the ability to determine movement without needing a constant connection with other nodes. If the node has enough self-direction, it makes node communication more efficient because it does not need to maintain constant communication. Each node may only communicate with the other nodes within its communication range since the communication between nodes should be efficient as possible. However, each node has the ability to communicate with the sink via multi-hop communication. The nodes use this multi-hop communication system to report obstacle position if known, target position if known, and its own position.

An obstacle may exist in one of the two possible states. The obstacle may be a safe distance from the node. In this case, the node broadcasts its location and keeps moving. In the other case, the obstacle is in the path of the node. The node broadcasts the location of the obstacle and navigates it. Self-organization allows the following nodes (i.e., nodes immediately behind the higher priority nodes) to navigate the obstacle without performing any computation (i.e., these nodes simply follow the path of a higher priority node).

Before we present the proposed SOA algorithm, it is worth mentioning some of our key assumptions and notations we use in the proposed algorithm.

- Locomotion (i.e., each node has the capability of movement).
- Communication (i.e., each node can communicate with the other nodes within the communication rage).
- Observation (i.e., each node can detect potential obstacles and the target).
- Position detection (i.e., each node can detect its position such as using a GPS system)
- For the sake of the simulation results, we shall assume that the sink knows its position and the position of the target. This prevents the nodes from attempting to scan the entire environment in order to detect the target.
- We shall also assume that the target is detectable by each node and does not have the capability of movement. Also, we assume that the potential obstacles are present within the paths (i.e., no obstacle is too large to avoid).

3. MOVING AND PRIORITY RULES FOR SOA ALGORITHM

Mobile sensor networks (MSNs) have received considerable research attention over the last decade because of their ease of deployment without the need of any fixed infrastructure [14]. Due to its highly dynamic nature and network topology, one of the fundament challenges in MSN is the design of self deployment algorithms that can enable the sensor nodes to organize themselves while at the same time maintain a consistent connection with the other deployed nodes and provide a coverage, so that the sensor nodes can communicate with each other within their respective communication range.

Several self-deployment algorithms have been suggested for MSNs over the past few years [3] [9] [11] [15]. The proposed SOA algorithm is the extension of the obstacle avoidance algorithm proposed by Takahashi et. al [3]. However, our SOA algorithm differs from the algorithm proposed by [3] since the proposed SOA algorithm not only avoids the obstacles but also provides coverage to sensor nodes which is a significant improvement over the algorithm suggested by [3].

The algorithm is based upon a number of priority and moving rules. The priority rules for a node *n* establish the priority rules for all objects which include the sink, target, and all other nodes.

These priority rules are as follows:

- **Priority rule I:** priority-s is settled to the node which is nearest to node *n* and closer to the sink. If there are no nodes closer to the sink than node *n*, priority-s is settled to the sink.
- **Priority rule II:** priority-t is settled to the node which is nearest to node *n* and closer to the target. If there are no nodes closer to the target than node *n*, priorities-t is settled to the target.
- **Priority rule III:** It is not permitted that priority-t is settled to an object for which priority has already been settled.

It should be noted that the stable connection area is defined as the area within which node *n* can effectively communicate. Taking this into consideration, the moving rules can be defined as follows:

- **Moving rule I:** Node *n* moves to the stable connection area of priority-s and keeps this condition. If node *n* cannot move to that area, it moves to the nearest position in the area it can reach. In this case, the Moving rule I is not satisfied.
- **Moving rule II:** Node *n* moves to the stable connection area of priority-t and keeps this condition with maintenance of Moving rule I. If node *n* cannot move to that area, node *n* moves to nearest position in the area it can reach. In this case, the Moving rule II is not satisfied.
- **Moving rule III:** The higher priority rule preferentially gets executed. Moving rule II is executed only after the Moving rule I is satisfied.

Also, the obstacle avoidance algorithm used is the Virtual Force Field (VFF) [13] method. Any obstacle acts as a virtual repulsive force against any node once it has been detected.

4. SELF-DEPLOYMENT OBSTACLE AVOIDANCE (SOA) ALGORITHM

We assume every node is initially deployed near the sink as shown in Fig. 1.

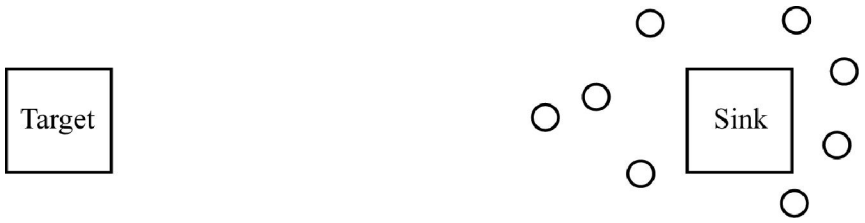

FIGURE 1: Initial Deployment of Nodes.

4.1 Determination of Connection Priority

First, the sink receives the position information of all nodes. Then the sink determines the relative distance between each node and the target, and each node and the sink.

4.2 Determination of Partner Node

Each node determines its partner node based on Priority rule II. For instance, the node with the highest priority-t partners with the second highest priority-t (Fig. 2), this continues until all nodes are paired. Once two nodes are partnered, they are closed enough to assume that they can move as one pair node.

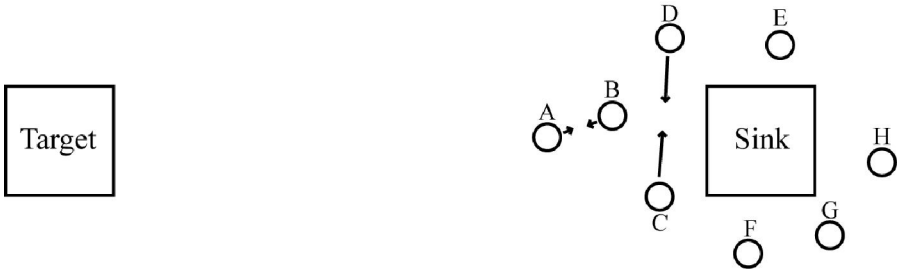

FIGURE 2: Formation of Node Pairs.

Information node *n* has		**Relative Distance**		
node ID number	position	to node *n*	to target	to sink
target	(X_t, Y_t)	D(t, *n*)	-	D(t, s)
sink	(X_s, Y_s)	D(s, *n*)	D(s, t)	-
Node 1	(X_1, Y_1)	D(1, *n*)	D(1, t)	D(1, s)
Node 5	(X_5, Y_5)	D(5, *n*)	D(5, t)	D(5, s)
Node 3	(X_3, Y_3)	D(3, *n*)	D(3, t)	D(3, s)
Node *n*	(X_n, Y_n)	-	D(*n*, t)	D(*n*, s)
Node 2	(X_2, Y_2)	D(2, *n*)	D(2, t)	D(2, s)
Node 6	(X_6, Y_6)	D(6, *n*)	D(6, t)	D(6, s)

Table 1: Node *n*'s Information about Position and Relative Distance

The distance (*d*) between two nodes, *a* and *b*, is shown using the following expression: $d(a,b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$ where x and y are the x-axis and y-axis coordinates in the constellation diagram. The complete information and relative distance for an arbitrarily node n is shows in Table 1.

4.3 Decision of Moving Direction

Each node pair moves toward its target based on the priority order. Based on the relative distance between the center point to the target, the node which is nearest to target gets the highest priority-t where as the node nearest to the sink gets the highest priority-s. The node determines its movement based on the location of the node-pair with higher priority-t. This location is determined as follows (see Fig. 3).

$$\frac{x_c - x_a}{x_b - x_a} = \frac{d}{d_a} \tag{1}$$

and

$$\frac{y_a - y_c}{y_a - y_b} = \frac{d_a}{d} \tag{2}$$

where

$$d = \frac{r}{\sqrt{2}} \qquad d_a = v \tag{3}$$

All node pairs begin moving toward the target following the established moving and priority rules. The node pair with the highest priority-t moves directly toward target. The node pair with the second highest priority-t directly follows the highest priority-t node pair and so on (see Fig. 4).

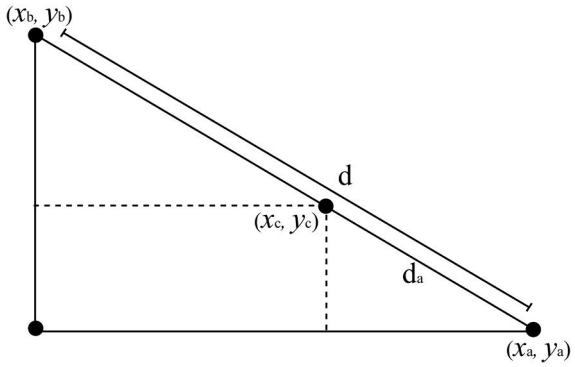

FIGURE 3: Determination of Movement Direction.

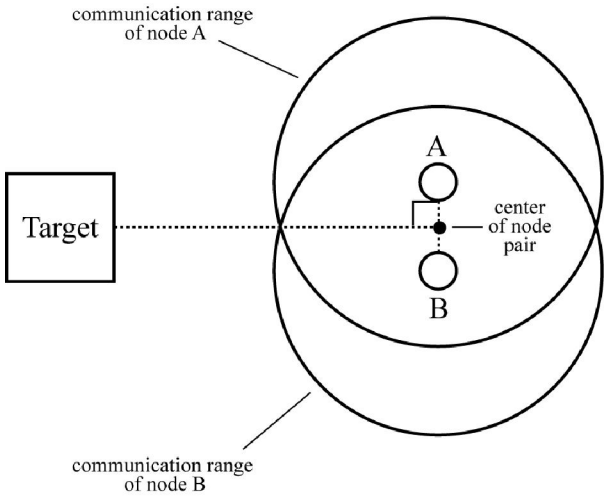

FIGURE 4: Setup of a Node Pair

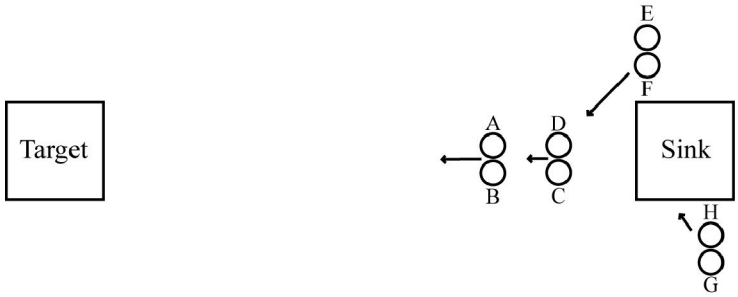

FIGURE 5: Navigation of an Obstacle by a Node Pair

Fig.5 shows the navigation method that will be discussed later in detail. After each time interval, each node pair communicates its location, and each node pair recalculates its destination based on the calculations in (3) (4) and (5).

The node pair with the highest priority-s can not break the link with the sink. When it reaches the stable connection edge, it moves to the nearest position in the area that it can reach without breaking the connection with the sink.

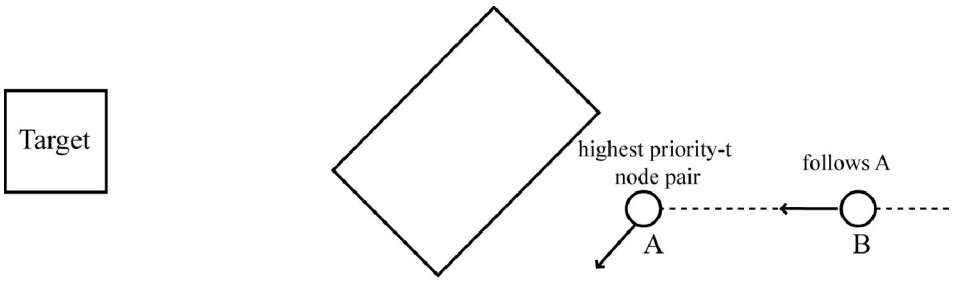

FIGURE 6a: Highest Priority-t Node Pair (*A*) Encounters Obstacle. The Next Node Pair (*B*) Simply Follows Node Pair *A*.

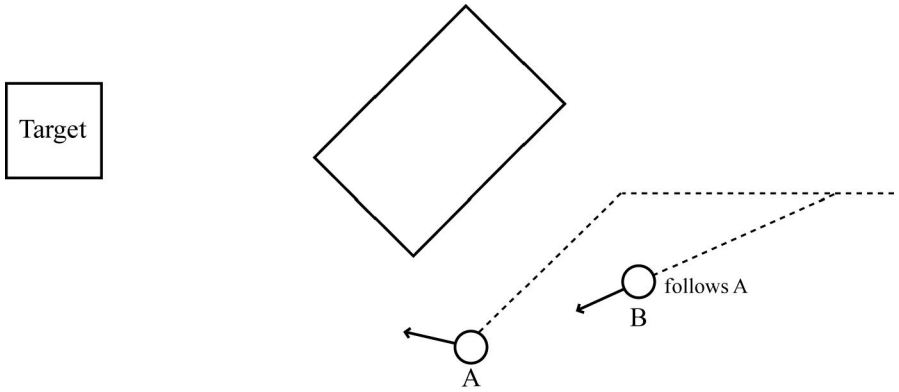

FIGURE 6b: Node Pair *A* has Negotiated Obstacle. Node Pair *B* has Simply Followed A.

When a node pair reaches the stable connection edge, it ceases movement in order to maintain its connection with the higher priority-t node pair, or the higher priority-s node pair, or both. When the highest priority-t node pair reaches the target the connection is built.

4.4 Obstacle Violation

We shall assume the obstacle is rectangular in shape. When the node pair detects the obstacle it calculates the edge position. If the obstacle does not impede the path to the target, it broadcasts the obstacle's location and continues moving. If the obstacle does block the path, the node pair attempts to move around it (Fig. 5). The node pair's direction of movement is parallel to the surface of obstacle while still close enough to detect the obstacle. The node pair continues to move this direction until it determines it can move safely in the direction of the target. The worst-case scenario occurs when obstacle runs perpendicular to the node pair's path to the target. The node pair moves around the obstacle in a predetermined direction.

When the highest priority-t node pair changes its direction of movement, the path of the next node pair automatically updates. This occurs because each node pair follows the higher priority-t node pair (Fig. 6a and 6b).

4.5 Partner Separation

The algorithm to determine separation is essential in order to ensure the full coverage and the ability to communicate with as many neighboring nodes as possible. After a connection between

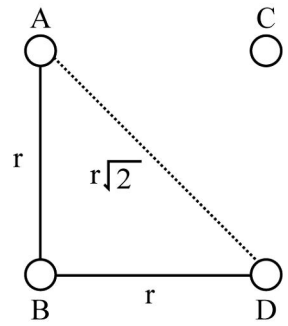

FIGURE 7a: The Maximum Distance between Nodes is *r*. Node *A* can Communicate with Node *B* and Node *C* but not Node *D*.

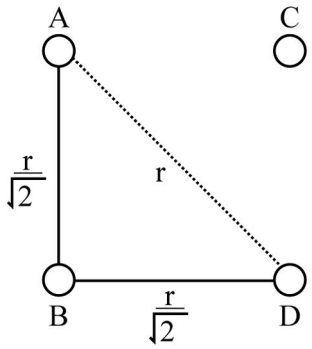

FIGURE 7b: Node *A* may Communicate with Node *D*.

the target and the sink is built, the node pairs separate to cover more area and also create a more reliable connection. The maximum allowable separation distance *r* is defined by the communication range of the nodes. In Fig. 7a, node *A* can communicate with nodes *B* and *C* but not node *D* because its distance is greater than *r*. We can ensure node *A* may communicate with node *D* by reducing the distance between node *A* and node *B* and also node *B* and node *D* (see Fig. 7b for complete illustration). System parameters along with their definitions are presented in Table 2. Specifically, the distance between nodes *A* and *B* can be defined in (4)

$$d_{(b,c)} \leq r \tag{4}$$

In order to achieve this, the distance from *A* to *B* must be:

$$d_{(b,c)} = \frac{r}{\sqrt{2}} \tag{5}$$

Using by the Pythagorean Theorem:

Parameters	Definitions
a	Distance from P_a to P_b
c	Distance from P_c to P_b
P_a	Position of Node a defined by (x_a, y_a)
P_b	Position of Node b defined by (x_b, y_b)
P_c	Position of Node c defined by (x_c, y_c)
r_a	Broadcast range of Node a
r_c	Broadcast range of Node c

TABLE 2: Definition of Parameters to Determine Separation

$$r = \sqrt{\left(\frac{r}{\sqrt{2}}\right)^2 + \left(\frac{r}{\sqrt{2}}\right)^2} \tag{6}$$

Equation (6) gives ideal location of the separation node. It is calculated based on the location of node *A* and node *C*. The distance between node *A* and node *B* is displayed in (7) and the distance between node B and C should be no greater than *r*. In order to determine the location to which the separation node moves, a number of calculations are performed as follows:

$$a^2 + h^2 = r_a^2 \tag{7}$$

$$c^2 + h^2 = r_c^2 \tag{8}$$

$$a = \frac{r_a^2 - r_c^2 + (a+c)^2}{2(a+c)} \tag{9}$$

$$P_{center} = P_a + \frac{a(P_c - P_a)}{a+c} \tag{10}$$

$$x_b = x_{center} + \frac{h(y_c - y_a)}{a+c} \tag{11}$$

$$y_b = y_{center} + \frac{h(x_c - x_a)}{a+c} \tag{12}$$

and

$$x_b = x_{center} - \frac{h(y_c - y_a)}{a+c} \tag{13}$$

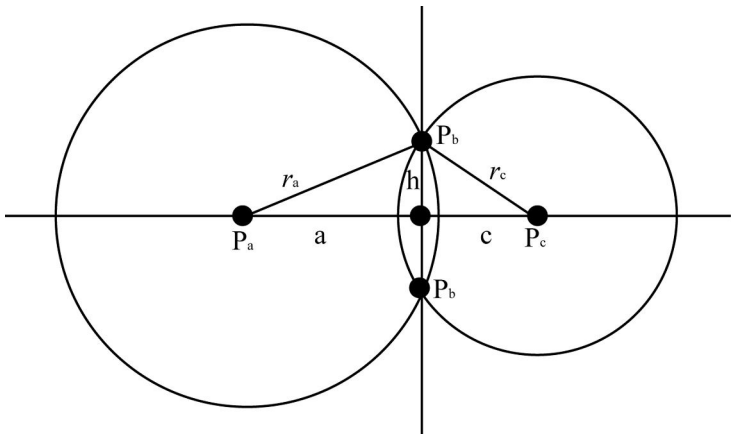

FIGURE 8: Determination of Location which Separating Node should move.

$$y_b = y_{center} - \frac{h\left(x_c - x_a\right)}{a + c} \tag{14}$$

Finally, the direction of separation is based on the location of the obstacle. Equations (11) and (12) give two points for which the separation node may move. This movement of nodes is shown in Fig. 8. We can determine which point is based on their distance from the obstacle. The separation node moves to the point whose distance to the obstacle is less. Once the separation has taken place, this system has satisfied the requirements of the mobile sensor network. It has determined a safe path from the sink to the target, detected any obstacle in its path, and provided coverage of the environment.

5. EXPERIMENTAL VERIFICATIONS AND PERFORMANCE ANALYSIS

This section presents the performance analysis of the proposed SOA algorithm. Before we present our simulation results, it is worth mentioning some of our key assumptions and simulation environment.

5.1 Simulation Environment

The unknown environment is defined to be a square with sides equaling 800m. The origin point (0, 0) is located in the uppermost left corner. Each node is represented as a black square and both the sink and the target are represented by a larger square. The sink is designated by a blue square and the target is represented by a green square. A large obstacle is placed within the field, which is represented by a red square. Each node is capable of sensing and communicating within its communication range designated by *r* (in meters). Nodes may communicate with nodes outside of its range via a multi-hop communication system. For the simulation, the range is 80m. Each node also has the capability of movement which is designated by *v* (in meters). Simulation will capture data after each 1 m/s (i.e., time is simulated in 1 second intervals). The initial state of the environment is shown in Fig. 9 and Table 3.

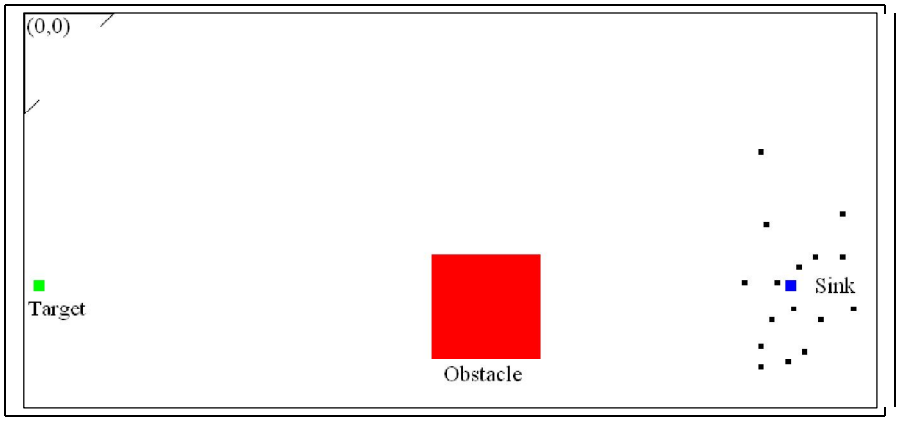

FIGURE 9: Initial State of the Simulation Environment.

Parameters	Definitions	Values
n	Number of mobile nodes	16
V	Speed of mobile node	1.0(m/s)
S	Sink position	(700,375)
T	Target position	(10,375)
$D_{(S,T)}$	Distance between sink and target	690m
R	Communication range	114m

TABLE 3: Initial State of the Simulation Environment with Simulation Parameters

5.2 Symbols Definition

A node is denoted by *n*. The sink is represented by *S*, the target *T*, and the obstacle *O*. Within the environment shown in Fig. 9, all objects are represented by an (x, y) grid coordinates.

Coverage is the quality of service by which the wireless mobile sensor network is measured. The nodes must be placed as efficiently as possible within the environment so they may communicate with neighboring nodes and also provide maximum coverage. For the sake of simulation, the distance between nodes is the metric by which the system is evaluated. We examine the distance between a sample node and the node it follows during the deployment. We also examine the distance to the node following it. If this distance becomes greater than *r* at any point, the nodes have lost communication.

Ideally, the distance between the nodes can be calculated using (5) as described earlier. Also, as the nodes separate, the distance of the separation node and its partner is important. The distance to neighboring node is equally important. If this distance exceeds *r*, communication between nodes is lost.

5.3 Simulation Results

Our mathematical model was simulated using Java. We sampled the information from node 2 in 10 second intervals. In order to maintain communication with nodes 0 and 4, the distance cannot at any point be greater than 114 m. As shown in Appendix 1, the distance between nodes 0 and node 4 never exceeds that distance. From this, we can identify that node 2 has maintained communication with both nodes 0 and node 4 during the entire simulation. The distance information is illustrated in Fig. 10b and also presented in Table 4 (see Appendix 1).

Also the distance between neighboring nodes should not exceed 114 m in order to maintain communication. In the final state of the simulation, this is achieved as shown in Appendix 1. A

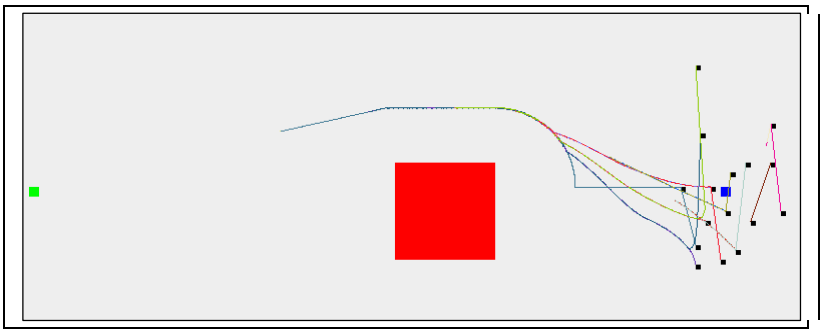

FIGURE 10a: Simulation during Nodes Movement.

state of the simulation is shown in Fig. 10a. The state of the simulation before node separation is shown in Fig. 11. Finally, the final state of the simulation is shown in Figure 12.

6. CONCLUSION & FUTURE WORK

This paper presented a new algorithm that can effectively deploy the sensor nodes by avoiding obstacles (if any) between the source and target. The simulation results demonstrated that the self-deployment algorithm is successful. Moreover, the system is able to negotiate an unknown environment, an obstacle, detect a target, and deploy to provide maximum coverage of the environment. It ensures the connection between the nodes is not lost by maintaining the distance between the nodes. The proposed SOA algorithm is an improvement over current algorithms. By pairing the nodes at the beginning of the deployment, this allows the most efficient deployment time from the sink to the target. While other algorithms provide efficient deployment with regards to time, SOA algorithm provides this, and also increases the amount of coverage of the environment. Also, SOA algorithm ensures that a greater area of coverage can be achieved when the nodes separate. While other algorithms provide effective coverage of an environment, our

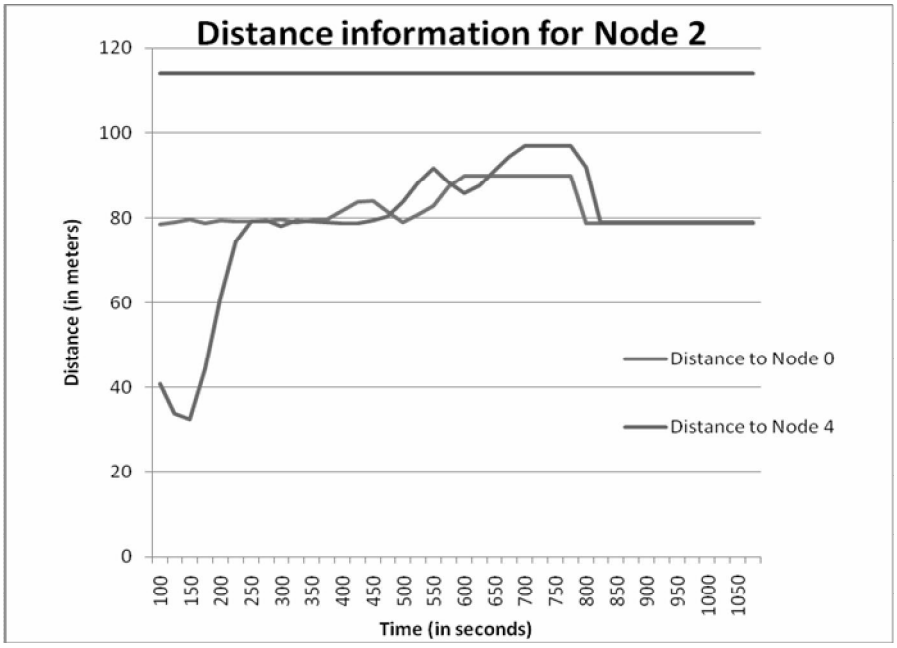

FIGURE 10b: Distance Information for Node 2 during the Entire Simulation

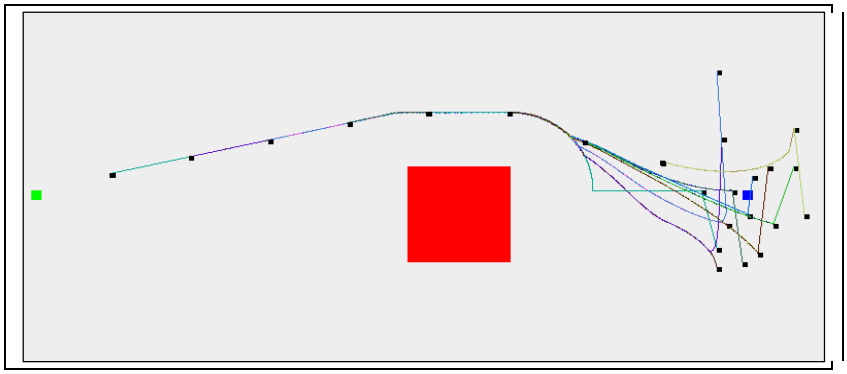

FIGURE 11: State of Simulation before Node Pair Separation.

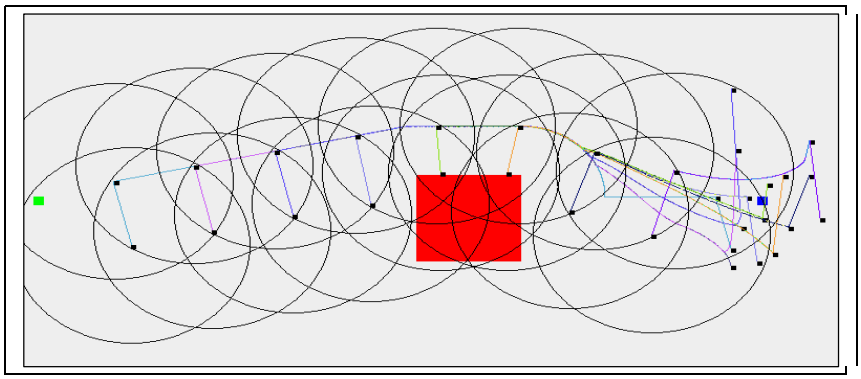

FIGURE 12: Final State of Wireless Sensor Network.

proposed algorithm ensures the ability to provide coverage quickly, by initially pairing nodes. It may be possible, in the future, to show that the mobile sensor network is more efficient when more nodes are added into the network. If more nodes are added to a node pair, it takes less of the networks resources to deploy the nodes. Only one node in the node pair must communicate and perform computation during the deployment of the network. Moreover, the proposed SOA algorithm provides fast deployment of nodes to targets since the priority after the pairing of nodes is to reach the target as efficiently as possible.

7. REFERENCES

[1] Y. Liang, C. Weidong, X. Yugeng. "*A review of control and localization for mobile sensor networks*". In Proceedings of the Sixth World Congress on Intelligent Control and Automation (WCICA 2006), pp. 9164-9168, Dalian, China, 2006.

[2] T. Jindong, X. Ning. "*Integration of sensing, computation, communication and cooperation for distributed mobile sensor networks*". In Proceedings of the IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, pp**.** 54- 59, 2003.

[3] J. Takahashi, K. Sekiyama, T. Fukuda. "*Self-Deployment algorithm of mobile sensor network based on connection priority criteria*". Proceedings of 2007 International Symposium on Micro-Nano Mechatronics and Human Science (MHS2007), pp. 564-569, 2007.

[4] M. Singh, M. Gore. "*A solution to sensor network coverage problem*". In Proceedings of the 2005 IEEE International Conference on Personal Wireless Communications, (ICPWC)**,** pp. 77-80, January, 2005.

[5] R. Tynan, G. DavidMarsh, D. O'Kane. "*Interpolation for wireless sensor network coverage*". In Proceedings of the Second IEEE Workshop on Embedded Networked Sensors, pp. 123-131, 2005.

[6] M. Cheng, L. Ruan, W. Wu. "*Achieving minimum coverage breach under bandwidth constraints in wireless sensor networks*". In Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 2638- 2645, 2005.

[7] S. Ram, D. Majunath, S. Iyer, D. Yogeshwaran. "*On the path coverage properties of random sensor networks*", IEEE Transaction on Mobile Computing, 6(5): 494-506, 2007.

[8] P. Pennesi, C. Paschalidis. "*Solving sensor network coverage problems by distributed asynchronous actor-critic methods*". In Proceedings of the 46th IEEE Conference on Decision and Control, pp. 5300-5305, 2007.

[9] N. Aziz, A. Mohemmed, D. Sagar. "*Particle swarm pptimization and voronoi diagram for wireless sensor networks coverage optimization*" In Proceedings of the International Conference on Intelligent and Advanced Systems, pp. 961-965, 2007.

[10] J. Kanno, J. Buchart, R. Selmic, V. Phoha, "*Detecting coverage holes in wireless sensor networks*". In Proceedings of the 2009 17th Mediterranean Conference on Control and Automation, pp.452-457, Thessaloniki, Greece June 2009.

[11] Y. Li and Y. Liu, "*Energy saving target tracking using mobile sensor networks*". In Proceedings of the IEEE International Conference on Robotics and Automation**,** pp. 674-679, April 2007.

[12] S. Zhang, J. Cao, L. Chen, D. Chen. "*Locating nodes in mobile sensor networks more accurately and faster*". In Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, (SECON '08), pp. 37-45, San Francisco, CA, 2008.

[13] J. Lu, T. Suda. "*Differentiated surveillance for static and random mobile sensor networks.* IEEE transactions on wireless communications, 7(11): 4411-4423, 2008.

[14] A. Rai, S. Ale, S. Rizvi, A. Riasat. "*New methodology for self localization in wireless sensor networks*". Journal of Communication and Computer, 6(11): 37-44, 2009.

[15] S. Rizvi and A. Riasat, "Use of self-adaptive methodology in wireless sensor networks for reducing energy consumption," *IEEE International Conference on Information and Emerging Technologies (IEEE ICIET-2007)*, pp. 1 – 7, July 06-07, 2007.

Time	Distance to Node O	Distance to Node 4
25	78.47	40.8
50	78.89	33.82
75	79.63	32.47
100	78.86	44.56
125	79.37	61.21
150	79.3	74.44
175	79.21	79.19
200	79.19	79.33
225	79.68	78.13
250	78.9	79.46
275	79.35	79.24
300	79.61	78.99
325	81.64	78.82
350	83.74	78.64
375	84	79.52
400	81.53	80.25
425	79.04	83.79
450	80.7	88.21
475	82.77	91.7
500	87.42	88.38
525	89.85	85.87
550	89.85	87.68
575	89.85	90.96
600	89.85	94.48
625	89.85	96.9
650	89.85	96.9
675	89.85	96.9
700	89.85	96.9
725	78.85	91.9
750	78.85	78.9
775	78.85	78.9
800	78.85	78.9
825	78.85	78.9
850	78.85	78.9
875	78.85	78.9
900	78.85	78.9
925	78.85	78.9
950	78.85	78.9
975	78.85	78.9
1000	78.85	78.9

Appendix 1: TABLE 4: Distance Information for Node 2