A Percolation Algorithm for Directional Sensor Networks

Mohammad Khanjary [a,*], Masoud Sabaei [b] and Mohammad Reza Meybodi [b]
[a] Computer Engineering Department, Science and Research Branch, Islamic Azad University, Tehran, Iran
[b] Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran

***Abstract*—In many applications of directional sensor networks especially border/barrier surveillance and intrusion detection applications, checking the establishment of the spanning clump/cluster of collaborated sensors that spans all over the region of interest is an essential task. Two sensors are collaborating sensors if their sensing ranges have been overlapped. When the spanning clump of collaborating sensors occurs, nothing could pass the region unless it will be detected by the network. In static networks (when orientation of sensor nodes is fixed), one time checking is needed. But in dynamic networks (when sensor nodes can adjust their orientations by using an algorithm) or hybrid networks (when some nodes are static and other nodes are dynamic), checking must be done periodically upon network topology changes. Therefore, it is very important to have an algorithm to test the establishment of spanning clump. In this paper, we present a distributed algorithm based on cellular automata to find out the establishment of spanning clump (percolation) in the region of interest by directional sensor networks consist of sensors with field-of-view angles between 0 and π. The proposed algorithm is distributed and works locally which makes it suitable for using in sensor networks. Moreover, the algorithm could be in other technical problems of sensor networks such as scheduling and topology control.**

I. INTRODUCTION

Sensing coverage is an important design issue in directional sensor networks which is related to covering the region of interest in a way that satisfies the especial needs of the application. Due to wide range of technical problems in *sensing coverage* and also its significance, a lot of researches have been focused on them. There are several reviews on different issues of coverage in directional sensor networks e.g. [1]-[2].

In other hand, *Percolation Theory* has been considered as a rigorous mathematical model for assessing coverage in wireless networks by researchers, recently. Generally, *Percolation Theory* describes the behavior of connected clusters in random graphs. Therefore, it has been considered to be used in wireless networks e.g. [3]-47] and also in sensor networks too e.g. [5]-[8].

Based on adjustability of sensor nodes, directional sensor networks could be classified to three categories: *Static*) when orientation of sensor nodes is fixed, d*ynamic*) when all sensor nodes are able to adjust their orientation and *Hybrid*) when some sensor nodes are able to adjust their orientation while others are not.

In this paper, we present a distributed algorithm to check the establishment of spanning clump (percolation) in the region of interest in directional sensor networks consist of sensors with field-of-view angle between 0 and π. The proposed algorithm works locally and could be used to check the establishment of spanning clump in different kinds of sensor networks, *static*, *dynamic* or *hybrid* which is very important especially for intrusion detection applications. Moreover, the proposed algorithm could be used for other technical problems of sensor networks such as topology control and scheduling.

A. Related works

Border/barrier coverage is used to detect mobile objects are entering into the boundary of a sensor network and its solution can be widely used in sensor barrier applications, such as intrusion detectors and border security. A sensor network is said to be a border coverage application if at least one spanning clump/cluster has been established in RoI in which nothing could pass the region unless it will be detected by the network.

In [9], Tao et al. considered strong barrier coverage using directional sensors where sensors can tune their orientations arbitrarily. They investigated the problem of finding appropriate orientations of directional sensors such that they can provide strong barrier coverage in one and two dimension.

Ma et al. [10] considered the minimum energy cost k-barrier coverage problem in static wireless sensor networks by transforming the problem into a minimum cost flow problem with side constraints and use Lagrangian relaxation technique. Li et al. [11] considered the weak k-barrier coverage problem in a belt region in which the height of region is less than diameter of sensors sensing range. A region is called to be weakly k-barrier covered ($k \geq 1$) if an intruder traveling along any orthogonal crossing path can be detected by at least k sensors.

Saipulla et al. [12] considered the barrier coverage when sensors are deployed along a line (e.g., sensors are dropped from an aircraft along a given path), they would be distributed along the line with random offsets due to wind and other environmental factors. Also, they studied sensor deployments along multiple lines and showed how barrier coverage is

* khanjary@srbiau.ac.ir

affected by the distance between adjacent lines and random offsets of sensors. Also, Liu et al. [13] considered constructing sensor barriers to detect intruders crossing a randomly-deployed sensor network on long strip areas of irregular shape without any constraint on crossing paths.

Despite of static networks, in [14], Chen et al. studied the barrier gap problem in a line-based deployment and proposed a gap-finding algorithm and two gap-mending algorithms that rotate sensors to mend the gaps. Likewise, Wang et al. [15] considered hybrid sensor networks consist of static and mobile sensors while mobile sensors are used to fill the gaps and connect the static nodes. Also, in [16] He et al. presented a periodic scheduling algorithm in which each point along the barrier line is monitored periodically by mobile sensors.

In contrast to line-based deployment, in [17], He et al. extensively studied the curve-based deployment. By using a contracting map, they identified the characteristics of a curve deployment to be optimal and designed a sensor deployment algorithm that can provide close-to-optimal performance. In [18], DeWitt et al. studied the barrier coverage problem by repeating sleep/wakeup schedule that can provide the continuous operation of the maximum number of barriers in an energy harvesting sensor network.

In this paper, we consider randomly-deployed directional sensor networks in which nodes are homogeneous and can have field-of-view angle between 0 and π. First, we propose a method that each node can find out collaboration between itself and its neighbor nodes. Then, we propose a novel distributed algorithm based on cellular automata to check whether the spanning clump exists in the network or not.

The remainder of this paper is organized as follows: section II presents the terminology of the paper, section III presents the required fundamentals and our proposed algorithm, Section IV presents the simulation results and finally section V concludes the paper and presents the possible future works.

II. TERMINOLOGY

Let $X_\lambda = \{\xi_i : i \geq 1\}$ be a randomly deployed two-dimensional homogeneous directional sensor network where ξ_i represents the location of the sensor S_i [6].

Definition 1 (directional sensor nodes). Each directional sensor node i is denoted by a tuple $S_i(\xi_i, r, \varphi, \overrightarrow{\vartheta_i}, R)$ where ξ_i is the center, r is the sensing radius, φ is the angle of field-of-view ($0 < \varphi \leq \pi$), $\overrightarrow{\vartheta_i}$ is the orientation vector and R is the transmission radius of the sensor.

Definition 2 (sensing range). The sensing range of directional sensor node $S_i(\xi_i, r, \varphi, \overrightarrow{\vartheta_i}, R)$ is a circular sector which defined by

$$C_i(r, \varphi, \overrightarrow{\vartheta_i}) = \{\xi \in \mathbb{R}^2 : |\xi_i - \xi| \leq r \text{ and } \delta(\overrightarrow{\vartheta_i}, \overrightarrow{\xi_i\xi}) \leq \varphi/2\}$$

where $|\xi_i - \xi|$ stands for the Euclidean distance between ξ_i and ξ, $\delta(\overrightarrow{\vartheta_i}, \overrightarrow{\xi_i\xi})$ stands for angle between orientation vector of sensor ($\overrightarrow{\vartheta_i}$) and vector from point ξ_i to ξ (see Fig. 1) .

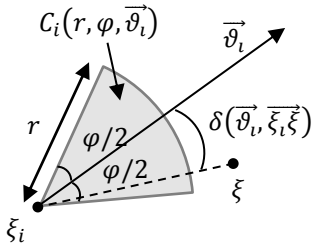

Fig. 1. Sensing range of directional sensor node S_i.

Definition 3 (collaborating sensors). Two directional sensors S_i and S_j are said to be collaborating only if there exist some points belong to sensing range of both (see Fig. 2).

$$Col(S_i) = \{S_j : \exists \xi \in \mathbb{R}^2, \xi \in C_i \text{ and } \xi \in C_j\}$$

Definition 4 (collaboration path). A collaboration path between two sensors S_i and S_j is a sequence of sensors $S_i, S_{i+1}, \dots, S_{j-1}, S_j$ such that any pair of sensors S_k and S_{k+1}, for $i \leq k \leq j-1$, are collaborating (see Fig. 3).

III. PERCOLATION ALGORITHM FOR DIRECTIONAL SENSOR NETWORKS

A. Localization in Sensor Networks

Location awareness is necessary in several applications of sensor networks. The localization of sensors can be done by using different manners. The simplest way is to equip all sensor nodes with a GPS receiver [19]. However, this solution is not usually feasible because of its high cost, high power consumption and environment constraint [20]. Moreover, the GPS fails in some situations such as underground, indoor and dense forest.

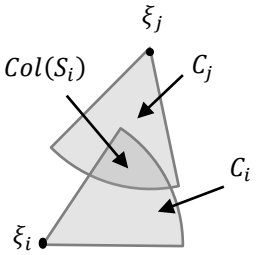

Fig. 2. Two collaborating directional sensor nodes.

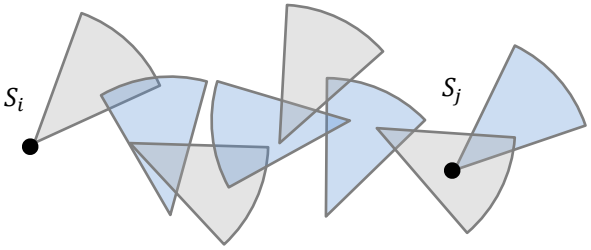

Fig. 3. Collaboration path between two sensors S_i and S_j.

Self-localization is an alternate solution for GPS, in which sensor nodes can estimate their position by using a localization protocol. Usually, these localization protocols use a few special nodes, called beacon nodes, which are assumed to

know their own locations (by using manual configuration, GPS or etc.) [21]. The beacon nodes (also called anchor nodes, seeds, references or landmarks) provide position information, in form of beacon messages, for blind nodes (also called unknown nodes, dumb nodes or targets). Blind nodes can use the position information of multiple close beacons to estimate their own positions.

Almost all existing localization protocols consist of two phases: 1) *distance/angle estimation*; 2) *position computation*. In *distance/angle estimation* phase, the most common range measurement techniques used to estimate distance or angle between two sensor nodes are time difference of arrival (TDOA), time of arrival (TOA), received signal strength indicator (RSSI), angle of arrival (AOA) and hop-count. In *position computation* phase, the position of blind nodes is estimated based on the available information of distance or angle and positions of beacon nodes. The commonly used techniques include trilateration, triangulation, bonding box, probabilistic approach and fingerprinting.

In this paper, we assume that nodes know their locations regardless of the technique which is used for localization.

B. How to Determine Collision between Sensors?

1) Finding the coordinates of arc points

Suppose sensor $S_i\left(\xi_i, r, \varphi, \overrightarrow{\vartheta_\iota}, R\right)$ where $(\xi_{i,x}, \xi_{i,y})$ is the coordinates of its center, r is its sensing radius, φ is its field-of-view angle, $\theta = \angle\overrightarrow{\vartheta_\iota}$ is the angle of its orientation vector and finally R is its transmission radius. Using basic concepts of trigonometry [22], we can derive the coordinates of its arc points. Consider two sample arc points ξ_i^{α} and $\xi_i^{-\beta}$ as it is shown in Fig. 4. By using *Cosine* and *Sine* definition, we can simply derive latitude and longitude of arc points, respectively, as shown in equation (1) and (2) for ξ_i^{α} and $\xi_i^{-\beta}$.

$$\begin{cases} \xi_{i,x}^{\alpha} = \xi_{i,x} + r.cos(\theta + \alpha) \\ \xi_{i,y}^{\alpha} = \xi_{i,y} + r.sin(\theta + \alpha) \end{cases} \quad (1)$$

$$\begin{cases} \xi_{i,x}^{-\beta} = \xi_{i,x} + r.cos(\theta - \beta) \\ \xi_{i,y}^{-\beta} = \xi_{i,y} + r.sin(\theta - \beta) \end{cases} \quad (2)$$

We will use these equations to transform the problem of collision detection between sensing ranges of two directional sensors to collision detection between line segments.

2) Collision detection between two line segments

Given line segment A (from point (A_x^1, A_y^1) to point (A_x^2, A_y^2)) on infinite line A and line segment B (from point (B_x^1, B_y^1) to point (B_x^2, B_y^2)) on infinite line B, a collision occurs on line segment A if $u_A \in [0,1]$ where u_A is calculated by equation (3). Likewise a collision occurs on line segment B if $u_B \in [0,1]$ where u_B is calculated by equation (4) [23].

$$u_A = \frac{(B_x^2 - B_x^1)(A_y^1 - B_y^1) - (B_y^2 - B_y^1)(A_x^1 - B_x^1)}{(B_y^2 - B_y^1)(A_x^2 - A_x^1) - (B_x^2 - B_x^1)(A_y^2 - A_y^1)} \quad (3)$$

$$u_B = \frac{(A_x^2 - A_x^1)(A_y^1 - B_y^1) - (A_y^2 - A_y^1)(A_x^1 - B_x^1)}{(B_y^2 - B_y^1)(A_x^2 - A_x^1) - (B_x^2 - B_x^1)(A_y^2 - A_y^1)} \quad (4)$$

Therefore, if $u_A \in [0,1]$ and $u_B \in [0,1]$ simultaneously, then a collision occurs between line segments. The pseudo code of collision detection between two line segments has been shown in Fig. 5.

3) Collision detection between two sectors

Given sensors $S_i\left(\xi_i, r, \varphi, \overrightarrow{\vartheta_\iota}, R\right)$ and $S_j\left(\xi_j, r, \varphi, \overrightarrow{\vartheta_j}, R\right)$ where $(\xi_{i,x}, \xi_{i,y})$ and $(\xi_{i,x}, \xi_{i,y})$ are coordinates of their centers, $\theta_i = \angle\overrightarrow{\vartheta_\iota}$ and $\theta_j = \angle\overrightarrow{\vartheta_j}$ are the angle of their orientation vectors and, r, φ and R are their sensing radius, field-of-view angle and transmission radius, respectively (as shown in Fig. 4), we are interested in finding a method to check whether collision occurred between their sensing range or not.

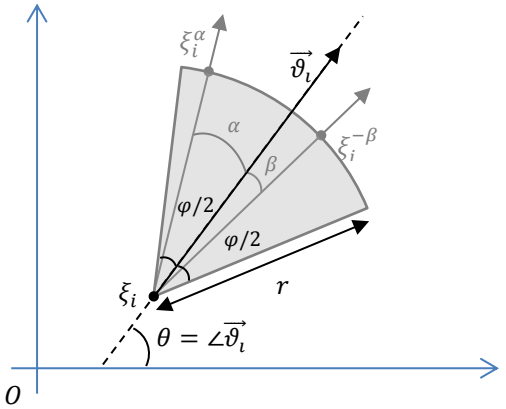

Fig. 4. Deriving coordinates of two sample arc points (ξ_i^{α} and $\xi_i^{-\beta}$) in a typical directional sensor node.

```
% Algorithm of Collision Detection between
% Two Line segments - CDbTLS
function CDbTLS(A¹, A², B¹, B²)
    u_A = ((B_x²−B_x¹)(A_y¹−B_y¹)-(B_y²−B_y¹)(A_x¹−B_x¹)) / ((B_y²−B_y¹)(A_x²−A_x¹)-(B_x²−B_x¹)(A_y²−A_y¹))
    u_B = ((A_x²−A_x¹)(A_y¹−B_y¹)-(A_y²−A_y¹)(A_x¹−B_x¹)) / ((B_y²−B_y¹)(A_x²−A_x¹)-(B_x²−B_x¹)(A_y²−A_y¹))
    if (0 ≤ u_A ≤ 1) and (0 ≤ u_B ≤ 1)
        return yes;
    else
        return no;
    end if
end function
```

Fig. 5. Algorithm of collision detection between two lines.

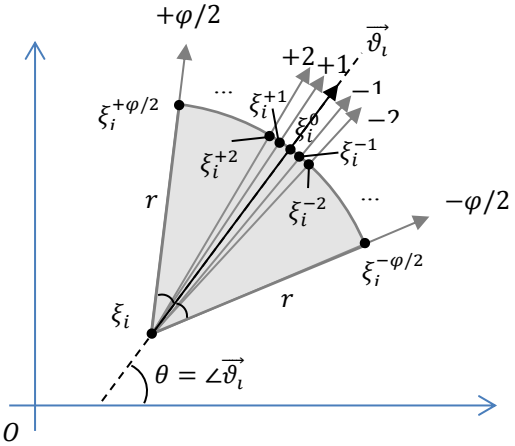

Fig. 6. Breaking the sensing range (sector) to φ line segments.

```
% Algorithm of Collision Detection between Two Sectors - CDbTS
% ξ_i: center and θ_i: angle of orientation vector of sensor i
% ξ_j: center and θ_j: angle of orientation vector of sensor j
% r: sensing radius and φ: field-of-view angle of both sensor i and j
function CDbTS(ξ_i, θ_i, ξ_j, θ_j, r, φ)
    if sqrt((ξ_{i,x} − ξ_{j,x})^2 + (ξ_{i,y} − ξ_{j,y})^2) ≤ 2r
        for m = 0 to φ/2
            derive coordinates of ξ_i^{+m} and ξ_i^{−m} by equation (1) and (2)
            for n = 0 to φ/2
                derive coordinates of ξ_i^{+n} and ξ_i^{−n} by equation (1) and (2)
                if ((CDbTLS(ξ_i, ξ_i^{+m}, ξ_j, ξ_j^{+n}) or (CDbTLS(ξ_i, ξ_i^{+m}, ξ_j, ξ_j^{−n}) or
                   (CDbTLS(ξ_i, ξ_i^{−m}, ξ_j, ξ_j^{+n}) or (CDbTLS(ξ_i, ξ_i^{−m}, ξ_j, ξ_j^{−n}))
                   return yes;
                end if
            end for
        end for
    end if
    return no;
end function
```

Fig. 7. Algorithm of collision detection between two sectors.

Because sensing range of each directional sensor is a circular sector (due to definition 2), if distance between sensors is more than twice the sensing radius, collision will never occur. So, the checking of percolation only will be done for sensors with less distance.

Each sector has two adjacent sides and an arc. To check the collision between two sectors, it is needed to check collision between all possible pair of them. It means we need to check nine possible collisions. Moreover, collision detection between an arc and a line segment and also between two arcs mathematically is very complicated.

Therefore, to solve the problem we transform the problem of collision detection between two sectors to collision detection between φ line segments. Consider Fig. 6 as a typical scheme of sensing range of a directional sensor. As it is shown, a sector with field-of-view angle φ (in degrees) could be broken to φ line segments for each one degree (from ξ_i to ξ_i^0, from ξ_i to ξ_i^{+1} and from ξ_i to ξ_i^{-1}, … , from ξ_i to $\xi_i^{+\varphi/2}$ and from ξ_i to $\xi_i^{-\varphi/2}$). To have a collision between two sectors, at least one line segment from first sector must have collision with at least one line segment from second sector. Therefore, it is enough to check all possible collisions. Each line segment could have collision with all line segments of the other sector. Therefore, φ^2 situations must be checked in worst case. As soon as a collision detected, the checking process will be ended by positive answer for collision between sensors while for negative result all situations must be checked. The pseudo code for collision detection between two sectors has been shown in Fig. 7.

C. The algorithm: PADSN

Given a randomly-deployed directional sensor network which all nodes know their location, the proposed algorithm includes four phases, the *Initialization phase*, the *Neighbor detection phase* the *State update phase* and the *Decision phase*. Assuming each node has a ***status*** variable, a ***region*** variable and a ***percolation table***, in this section we will explain these phases.

1) Initialization phase

Consider Fig. 8. There are three regions called *Start Boundary Region*, *Main Region* and *End Boundary Region* which are predetermined regions. Each sensor node by using its location information recognizes that it is belonged to *Start Boundary Region*, *End Boundary Region* or *Main Region* and sets its ***region*** variable. Then, all sensors in *Start Boundary Region* will set their ***status*** variable to **PERCOLATED**. All other sensors (sensors in *End Boundary Region* and *Main Region*) will set their ***status*** variable to **NON-PERCOLATED**.

2) Neighbor detection phase

In this phase, each sensor node collects the information of its neighbors through message exchange. Each node broadcasts its location and also its angle of orientation vector (θ) to its neighbors and receives the broadcasted information of its neighbors. By using this information, each node creates its ***percolation table***. A typical ***percolation table*** for node i has been shown in table I. Each row of this table is related to one of the neighbors and includes two fields: ***Neighbor's Address*** and a ***Collaboration Flag***. The address of the neighbor which is a unique identifier will be stored in ***Neighbor's Address*** and ***Collaboration Flag*** shows whether this neighbor has collaboration with node i or not? Collaboration between node i and its neighbors will be examined locally and distributedly by CDbTS function which is shown in Fig. 7. At the end of this phase, all nodes created their ***percolation table***.

TABLE I
PERCOLATION TABLE OF SENSOR NODE i

Neighbor's Address	Collaboration Flag
j	yes
k	no
l	yes
m	no

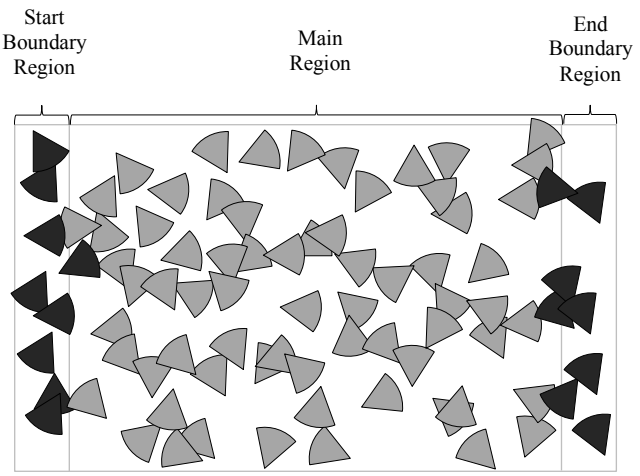

Fig. 8. A typical directional sensor network.

3) State update phase

In each iteration, sensors which newly changed their ***status*** variable to **PERCOLATED** send UREQ (update request) to their neighbors. In first iteration, only sensors in *Start Boundary Region* have ***status*** variable equal to **PERCOLATED.** Therefore, the algorithm will be started by them. The UREQ packet includes the address of sender. Each neighbor node which receives UREQ and its ***status*** variable is equal to **NON-PERCOLATED**, looks up in its ***percolation table*** for the sender's address and if the ***collaborating flag*** for the sender is ***yes***, it will change its ***status*** variable to **PERCOLATED**. If ***status*** of the UREQ receiver node was **PERCOLATED** or the sender was not a collaborating node due to its ***percolation table***, nothing will be done and the ***status*** variable will be unchanged.

In next iteration, the nodes which just changed their ***status*** variable to **PERCOLATED** in previous iteration create the UREQ packet including their address and send it to their neighbors and so on. Sending the UREQ packet by sensor nodes which their ***status*** variable just turned to **PERCOLATED** will be done only one time.

4) Decision phase

The algorithm will be terminated in one of two following situations:

1) In recent iteration, one of nodes in *End Boundary Region* received the UREQ packet and changed its status to **PERCOLATED**. It means a percolation path between at least one of nodes in *Start Boundary Region* and at least one of nodes in *End Boundary Region* (recent node) exists and percolation occurs. This node creates USTOP (update stop) packet and broadcasts it to its neighbors. USTOP packet includes the ***address*** of creator and a flag called ***percolation***. Because percolation occurred in this situation, ***percolation*** flag will be set to ***yes***. Upon receiving the USTOP packet, each node broadcasts it to its neighbors again and stops the broadcasting of UREQ packet (if any exists) or changing its orientation (in case of dynamic and hybrid networks).
2) A timer that was set for the algorithm execution by *Start Boundary Region* sensors gets time out. It means the percolation path (spanning clump) does not exist and the algorithm will be stopped. Therefore, *Start Boundary Region* nodes create the USTOP packet including their ***address***, set its ***percolation*** flag to ***no*** and broadcast it through the network to let other nodes know that there is not any percolation path. In case of *dynamic* or *hybrid* networks, this could help adjusting algorithms, to modify the position of nodes or their sensing orientation/radius and check the percolation again.

IV. Simulation

To check the accuracy of our proposed algorithm, we simulated the proposed algorithm for four different field-of-view angles, $\varphi = \pi$, $3\pi/4$, $\pi/2$ and $\pi/4$, and compared it with result we achieved in [6] for aligned-orientation directional sensor networks (ALODSNs). In [6], we used a centralized algorithm which was used a matrix to derive whether percolation occurs in the network or not. To compare the results, we used the equal parameters for simulation. The charts of this comparison have been shown in Fig. 9 to Fig. 12. In these figures, the horizontal axis represents the density of nodes and the vertical axis represents percent of percolation occurred in 500 times simulation for each step.

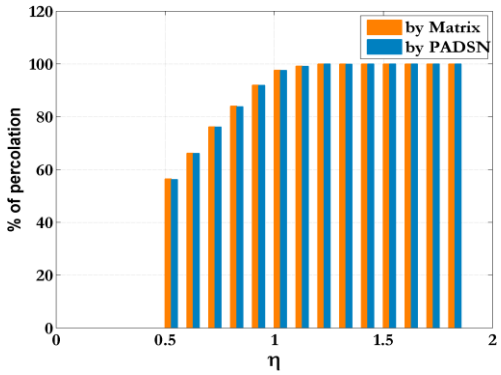

Fig. 9. Comparison between simulation chart of ALODSNs with $\varphi = \pi$ [6] and simulation chart of PADSN.

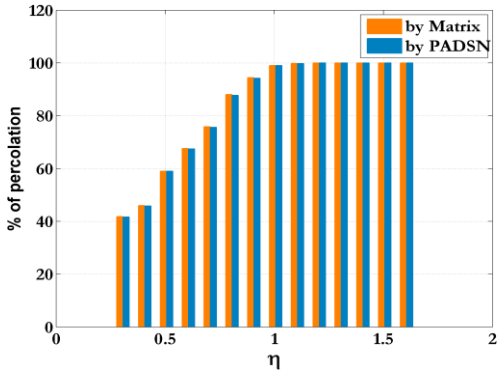

Fig. 10. Comparison between simulation chart of ALODSNs with $\varphi = 3\pi/4$ [6] and simulation chart of PADSN.

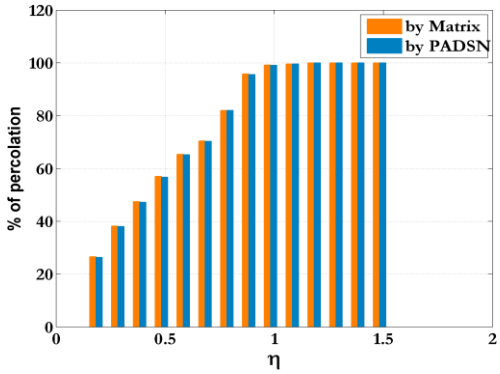

Fig. 11. Comparison between simulation chart of ALODSNs with $\varphi = \pi/2$ [6] and simulation chart of PADSN.

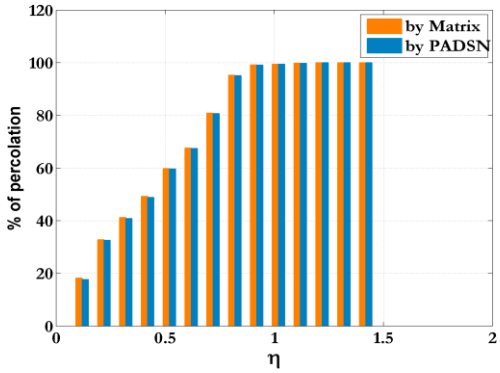

Fig. 12. Comparison between simulation chart of ALODSNs with $\varphi = \pi/4$ [6] and simulation chart of PADSN.

As it is seen, the proposed algorithm works well and differences (caused by asymptote situations between sectors which will be interpreted as no collaboration) between results is highly negligible (less than 1%).

V. Conclusion and Future Works

In this paper, we introduced an algorithm based on cellular automata for checking the percolation in directional sensor networks for all field-of-view angles of sensors between 0 and π which is very important especially for intrusion detection applications. The proposed algorithm is distributed and works locally which make it suitable for using in different kinds of directional sensor networks such as static, dynamic and hybrid. Also, the proposed algorithm could be used in other technical problems of sensor networks such as scheduling and topology control.

Possible future works include similar algorithm for inhomogeneous sensor network and three-dimensional sensor networks.

References

[1] M. A. Guvensan and A. G. Yavuz, "On coverage issues in directional sensor networks: A survey," Ad Hoc Networks, vol. 9, no. 7, pp. 1238-1255, 2011.

[2] Y. Charfi, N. Wakamiya and M. Murata, "Challenging issues in visual sensor networks," IEEE Wireless Communications, vol. 16, no. 2, pp. 44-49, 2009.

[3] I. Glauche, W. Krause, R. Sollacher and M. Greiner, "Continuum percolation of wireless ad hoc communication networks," Physica A, vol. 325, pp. 577-600, 2003.

[4] A. Jiang and J. Bruck, "Monotone percolation and the topology control of wireless networks," in Proc. IEEE INFOCOM, 2005, pp. 327-338.

[5] H. M. Ammari and S. K. Das, "Integrated coverage and connectivity in wireless sensor networks: A two-dimensional percolation problem," IEEE Transactions on Computers, vol. 57, no. 10, pp. 1423-1434, 2008.

[6] M. Khanjary, M. Sabaei and M. R. Meybodi, "Critical density for coverage and connectivity in two-dimensional aligned-orientation directional sensor networks using continuum percolation," IEEE Sensors Journal, vol. 14, no. 8, pp. 2856-2863, 2014.

[7] H. M. Ammari and S. K. Das, "Critical density for coverage and connectivity in three-dimensional wireless sensor networks using continuum percolation," IEEE Transactions on Parallel and Distributed Systems, vol. 20, no. 6, pp. 872-885, 2009.

[8] L. Liu, X. Zhang and H. Ma, "Optimal density estimation for exposure-path prevention in wireless sensor networks using percolation theory," in Proc. IEEE INFOCOM, 2012, pp. 2601-2605.

[9] D. Tao, S. Tang, H. Zhang, X. Mao and H. Ma, "Strong barrier coverage in directional sensor networks," Computer Communications, vol. 35, no. 8, pp. 895-905, 2012.

[10] H. Ma, D. Li, W. Chen, Q. Zhu and H. Yang, "Energy efficient k-barrier coverage in limited mobile wireless sensor networks," Computer Communications, vol. 35, no. 14, pp. 1749-1758, 2012.

[11] L. Li, B. Zhang, X. Shen, J. Zheng and Z. Yao, "A study on the weak barrier coverage problem in wireless sensor networks," Computer Networks, vol. 55, no. 3, pp. 711-721, 2011.

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

[13] B. Liu, O. Dousse, J. Wang and A. Saipulla, 2008. "Strong barrier coverage of wireless sensor networks," in Proc. ACM MobiHoc, 2008, pp. 411-420.

[14] J. Chen, B. Wang, W. Liu, L. T. Yang and X. Deng, "Rotating directional sensors to mend barrier gaps in a line-based deployed directional sensor network," IEEE Systems Journal, vol. PP, no. 99, pp. 1-12, doi: 10.1109/JSYST.2014.2327793.

[15] Z. Wang, J. Liao, Q. Cao, H. Qi and Z. Wang, "Achieving k-barrier coverage in hybrid directional sensor networks," IEEE Transactions on Mobile Computing, vol. 13, no. 7, pp. 1443-1455, 2014.

[16] S. He, J. Chen, X. Li, X. S. Shen and Y. Sun, "Mobility and intruder prior information improving the barrier coverage of sparse sensor networks," IEEE Transactions on Mobile Computing, vol. 13, no. 6, pp. 1268-1282, 2014.

[17] S. He, X. Gong, J. Zhang, J. Chen and Y. Sun, "Curve-based deployment for barrier coverage in wireless sensor networks," IEEE Transactions on Wireless Communications, vol.13, no. 2, pp. 724-735, 2014.

[18] J. DeWitt, S. Patt and S. Hongchi, "Maximizing continuous barrier coverage in energy harvesting sensor networks," in Proc. IEEE ICC, 2014, pp.172-177.

[19] N. Labraoui, M. Gueroui and M. Aliouat, "Secure dv-hop localization scheme against wormhole attacks in wireless sensor networks," Transactions on Emerging Telecommunication Technologies, vol. 23, no. 4, pp 303–316, 2012.

[20] A. Savvides, C. H. and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in Proc. ACM MOBICOM, 2001, pp. 166-79.

[21] L .N. Donggang, N. Peng and D. Wenliang, "detecting malicious beacon nodes for secure location discovery in wireless sensor networks," in Proc. IEEE ICDCS, pp. 609-619, 2005.

[22] http://mathworld.wolfram.com/Trigonometry.html, 2014.

[23] C. Ericson, "Real-time collision detection," Morgan Kaufmann, 2004.