Peer-to-Peer Networking and Applications (2019) 12:647–659
https://doi.org/10.1007/s12083-018-0656-y

Connectivity preserving obstacle avoidance localized motion planning algorithms for mobile wireless sensor networks

Md. Yeakub Hassan[1] · **Faisal Hussain**[2] · **Salimur Choudhury**[3]

Received: 30 October 2017 / Accepted: 20 April 2018 / Published online: 4 May 2018

Abstract

Mobile wireless sensor networks (MWSN) are better in terms of coverage and it plays an important role in ubiquitous wireless networks. We design Cellular Automaton (CA) based localized motion planning algorithms for mobile wireless sensors. We propose cellular automaton based algorithms for both dispersion and gathering problems. The dispersion algorithm is intended for self-deployment purpose with the goal of increasing the sensing coverage of the network. We apply a probabilistic approach that maximizes the network coverage as well as maintains the connectivity of the network. In addition, after finishing the dispersion, a gathering algorithm guides the sensors to round up to a single place for collection. It is noteworthy that both algorithms are synchronous which means that all sensors run algorithms in parallel at the same time. Moreover, our algorithms allow the sensors to avert obstacles in their path of movement. As cellular automaton functions depend on the local information about the network strictly, they are suitable for MWSN in practice. We evaluate the performance of our algorithm based on some defined metrics i.e., coverage, strongly connected coverage. We find that our dispersion algorithm maintains better coverage than state-of-the-art algorithm. Furthermore, in case of synchronous gathering, sensors get disconnected for some cases to form multiple clusters while using state-of-the-art algorithm, but our proposed gathering algorithm is always able to provide the connectivity.

Keywords Self-deployment · Synchronous gathering · Mobile wireless sensor network (MWSN) · Local algorithms · Cellular automaton · IoT · Obstacles

1 Introduction

Internet of Things (IoT) is one of the popular technology with enormous future possibilities. A wide range of applications of IoT, encourage the researchers to work on different aspects of this technology. Mainly IoT devices take smart decisions based on the collected data from the local environment. Moreover, it is necessary to accumulate the physical environmental data in case of infrastructural use of IoT, where Wireless Sensor Network (WSN) comes into play.

A number of sensor nodes formed a cooperated network, termed as WSN, where each node is equipped CPU, memory unit, transceiver, sensor array embedded [1]. Meanwhile, in MWSN, sensors are capable of movement, enabling them to self re-positioning in a dire situation to increase coverage or to track a target object. In a network, the area covered by its sensors is the coverage of that network. Various types of applications utilize sensor networking technology, like military and defense applications (e.g. battlefield surveillance, security operation, tracking border intrusion, intruder ship detection in sea, target tracking etc.), health care and medical applications (e.g. location tracking and activity monitoring of emergency respondents (ER), telecare system etc.), industrial applications (e.g. intelligent

This article is part of the Topical Collection: *Special Issue on Network Coverage*
Guest Editors: Shibo He, Dong-Hoon Shin, and Yuanchao Shu

✉ Salimur Choudhury
salimur.choudhury@lakeheadu.ca

Md. Yeakub Hassan
yeakub@cse.green.edu.bd

Faisal Hussain
faisalhussain@iut-dhaka.edu

[1] Department of Computer Science and Engineering, Green University of Bangladesh, Dhaka, Bangladesh

[2] Department of Computer Science and Engineering, Islamic University of Technology, Gazipur, Bangladesh

[3] Department of Computer Science, Lakehead University, Thunder Bay, ON, Canada

transportation system, smart car parking, smart homes and offices, industrial process, asset management etc.), environmental and agricultural applications (e.g. indoor air quality monitoring and fire detection, outdoor air pollutants monitoring, wildfire detection, precision agriculture, livestock monitoring etc.) and so on [2, 3].

Sensors are deployed depending on the type of application and normally the deployment can be in a random manner or follow a systemic approach [4]. Moreover, sensors can be deployed based on the area-priority concept for such regions that have different priority levels [5]. In a dynamic deployment scenario, sensors are deployed in one place densely which results in lower network coverage. To improve this coverage, dispersion algorithms are used to guide the sensors' movements. These algorithms can be divided into centralized, distributed and strictly localized. Various algorithms are proposed in [6–10] which are mainly centralized or distributed algorithms. However, these approaches are inapt in WSN as they are quite complex to handle since sensor nodes have low memory as well as low computational power [11].

On the other hand, after completing data collections from an area the sensors nodes are needed to be collected. So it is desired that sensor nodes will gather into one place for easy collection. This gathering method can be divided into two category synchronous gathering and asynchronous gathering [12]. In synchronous gathering, all sensor nodes move simultaneously. Localized motion planning algorithms are proposed for both sensor dispersion and gathering problem in [13]. However, these algorithms do not consider the presence of obstacle in the network. Similar gathering problem has been studied in robotics [14–16] and those algorithms are quite complex to implement in sensors.

Different problems in WSN is solved with the help of cellular automaton model in [4, 18–23, 25, 26, 39, 40] with low complexity in terms of memory and computation. Choudhury et al. in [20] propose a simple and strictly localized algorithm based on cellular automaton model that serves the dispersion problem. Additionally, Choudhury et al. in [12] propose an algorithm to solve the gathering problem to a common place. However, the model in [12, 20] does not include obstacles in the path of movement which makes it futile for practical scenarios.

In this paper, we design cellular automaton based strictly localized algorithms where sensors disperse themselves autonomously in order to maximize the network coverage by maintaining connectivity and after finishing dispersion they synchronously gather themselves at a single location. We consider that the system contains obstacles which need to be avoided by the sensors at the time of movements. We modify the algorithm in [20] at the commencement of our study of dispersion of sensor nodes which confirms that we need special rules to avoid these obstacles. Additionally, we also modify the algorithm in [12] and propose a synchronous gathering algorithm which guides these dispersed sensors to a common place. The main contribution of this research is a simple guiding algorithm for dispersion and gathering, in the presence of obstacles to improve the network coverage as well as maintain the network connectivity. Moreover It should be noted that as far we know, there is no CA based algorithm to solve the problem we are addressing that works in an obstacle prone system.

The rest of the paper is organized as follows. Section 2 overviews the state-of-the-art in WSN. In Section 3 we discuss some of the basics of cellular automata. In section 4 we describe the CA based motion planning algorithm. Section 5 defines the simulation tool and analyze the numerical results. Finally, Section 6 concludes the paper with possible future research directions.

2 Related works

Numerous research works have been done on the dispersion of MWSN. Various centralized and decentralized algorithms are proposed to solve dispersion problems like maximizing coverage of the network, object monitoring, strengthening the connectivity of the nodes etc. In [6], authors propose a "sweep coverage" technique where sensors periodically monitor some points of interest (POI) and make the coverage at each POI as time-dependent. Though a small number of sensors are enough to cover a larger area with this approach, it encumbers the sensors with a large amount of stored data as the sensors are unable to send data all the time. A game theoretic approach is discussed in[8] and utilizes the binary log-linear learning to maximize the coverage. They claim that their scheme asymptotically maximizes the number of covered nodes, provided that the agents have sufficient communication ranges.

An energy efficient minimum weighted trap coverage problem in wireless sensor networks is introduced in [27] where authors present a bounded approximation algorithm and prove that the problem is an NP-hard problem. Gupta et al. [7] study sensor positioning method in an anisotropic network topology and complex terrain. They resort to a stochastic sensor movement strategy to generate a trajectory of the sensors. On the other hand, a node placement strategy for ensuring connected coverage in sensor networks is addressed in [4] where two types of scenarios are considered. First one is to cover a certain region and the second one is to cover a fixed set of nodes. Du et al. [28] propose approximation algorithms for a minimum sensor connected coverage problem. Several algorithms for barrier coverage in sensor networks are also proposed in [29–31],

. However, these works are not same as the problem we are considering rather similar to our coverage problem. Virtual force approaches for sensor deployment in WSN are proposed in [10]. In virtual force approach, all sensors are considered as an actual coordinate point where a repulsive and attractive force acts among them. However, these approaches are computationally quite complex to handle in a WSN and also does not work for an unbounded area.

On the other hand, a CA based approach is presented in [19] where it is argued that cellular automaton model can be used with success to simulate a large network, verifying the properties in a quick and objective way. CA based algorithms use only local information in the algorithm which suits very well for WSNs applications. In [11], a cellular automaton based localized algorithm is proposed to improve the coverage of the network with a minimum number of moves utilizing dynamic deployment. Their movement depends on the position of neighboring sensors along four quadrants and two predefined parameters. Mobile dispersion algorithm, based on cellular automaton is proposed in [20] where the sensors are placed randomly in an obstacle-free field and allow them to disperse. Three transition rules drive their algorithm to maximize the coverage and strengthening the connectivity. In [32], authors propose an algorithm based on cellular automaton to monitor mobile objects. The aim of their algorithm is not to improve the coverage rather maximize the time of monitoring targeted objects.

A fundamental problem for gathering autonomous robots in one place is discussed in [33] where neighbors of a robot are calculated using a constant euclidean distance. Similar to this problem, an asynchronous gathering of robots is also discussed in [34]. Choudhury et al. [12] propose a CA based synchronous gathering localized algorithm for mobile wireless sensor networks where the authors place sensors in a 2-D grid randomly by maintaining connectivity and gathers these sensors into one location. However, obstacles in the path of movement was not considered here. Ando et al. in [15] and Degener et al. in [14] work on the gathering problem in the study of robotics. In [15], a local algorithm is proposed where each sensor determines the smallest enclosing circle that contains all of its neighbors and moves towards the center of that circle. However, computing this circle is complex compared to any CA based algorithm [12]. Degener et al. [14] prove that running time taken by a sensor is tight which is $O(n^2)$ where n is the number of sensors. In [13], localized motion planning algorithms are proposed for both sensor dispersion and gathering problem where authors do not consider the presence of obstacle in the network. Saadatmand et al. [35] proposed a CA based local algorithm for sensor gathering in obstacle free field which has the worst case complexity of $O(n)$ where n is the number of sensors.

Moreover, in recent years, a number of problems are solved by cellular automaton in the field of WSN. In [36], a non-volatile two-dimensional cellular automaton model is proposed to analyze the space-time dynamics of a WSN. Sang et al. [37] propose a moving object tracking CA based model in a distributed mobile wireless sensor network. Furthermore, "The lightweight 2-dimensional (2-D) cellular automata based symmetric key encryption" algorithm (L2D-CASKE) is proposed in [23]. Intruder detection system based on cellular automaton theory is presented in [17] which is scalable, self-organized and easily implementable. This system utilizes periodic wake sensor barrier (wave) that sweep the sensor field and able to endure frequent communication failures, obstacles, and sensor failures. A cellular automaton based key management scheme, CAB is proposed in [24] which allows sensors to establish pairwise keys during any stage of the network operation.

3 The cellular automaton model

A Cellular Automaton (CA) model is a biologically inspired model which is used in different physical systems. We can model a two dimensional cellular automaton, following Moore neighborhood [38], as a two dimensional grid. Each cell of the grid can be in a state from a finite set of states. The states of the cells change after discrete time interval and changes depend on the local information rather than the global information. We define a cellular automaton as a quadruple, $CA = \{C, Q, \delta, N\}$. Here C represents the cells of the grid, Q represents the states of the cells, δ represents the rules for transition of the automation and last of all, N is the neighborhood of a cell. At any moment t, each cell $c \in C$ has a state $q \in Q$. At time $t + 1$, the state of every cell is determined by the transition rules (δ) and state of the neighboring cells. Cells having less Chebyshev distance than the radius i conform the neighbor N of a cell c. In a typical cellular automaton, all cells check the status of its neighbors and change their state according to the transition rules synchronously.

4 Problem formulation and algorithms

We consider a 2-D cellular automaton model where a set of homogeneous sensors Σ and some random obstacles are deployed. The sensors can move within the 2-D grid which is an unbounded area. Our goal is to disperse these sensors in a fashion where they can maximize the area coverage by maintaining connectivity and finally gather all these sensors at a single location so that we can use them for later purpose. In the following subsections, we discuss the characteristics

of the cellular automaton model (cell, state, neighborhood), existing localized algorithms and obstacle avoidance motion planning algorithms.

4.1 Cell

A collection of cells are arranged in a 2-D square grid where one square is considered as one cell. We consider that a cell can contain more than one sensor as a typical CA cell can hold multiple mobile sensors. The sensors can move to one cell in one time step and, hence, they can move in the positive and negative x-direction as well as in the positive and negative y-direction.

4.2 State

We define states of each cell as a triplet (M, S_x, S_y). Here M denotes the types of a cell as "sensors", "empty" and "obstacles" and S_x, S_y denotes the previous movement position of a sensor in x-direction and y-direction respectively. A set of types $M \in \{0, 1, 2\}$ is defined for a cell where 1 denotes the presence of sensors in the cell, 2 represents the presence of obstacles in the cell and 0 represents an empty cell. For $M \in \{0, 2\}$, we replace S_x, S_y with $\emptyset$ as neither an empty cell nor a cell with obstacle holds sensors. Hence, we define $S_x, S_y \in \{0, 1, -1, \emptyset\}$ for a cell where 0 indicates that the sensor in this cell is staying in the same cell from previous time step, 1 and -1 implies that the sensor in this cell is moved from positive and negative x and y-direction respectively. The state of each cell changes in discrete time steps and it depends on a set of self-deployment rules of mobile sensors. In Fig. 1, we have shown the positions to which the sensor s allowed to move in positive and negative x and y-direction. The sensor can move to the positions of a_+, b_+ and c_+ in positive x-direction and a_-, b_- and c_- in negative x-direction. Similarly, positions of positive and negative y-direction are a_-, a_0, a_+ and c_-, c_0, c_+ respectively.

4.3 Neighborhood

The simplest neighborhood in our cellular automaton model for any given cell is the cell itself and its eight adjacent neighbors. A sensor has communication radius R_c and within this communication radius, all cells are considered as neighbors of the cell of that sensor. Two sensors are directly connected if they are placed within their communication radius and the network is connected if any two sensors can be connected by a path of connected sensors. The sensors also have sensing radius R_s which specifies the monitoring function of the network and the cells within this sensing radius is covered by the sensors. The area covered by the largest connected sensors components of the network

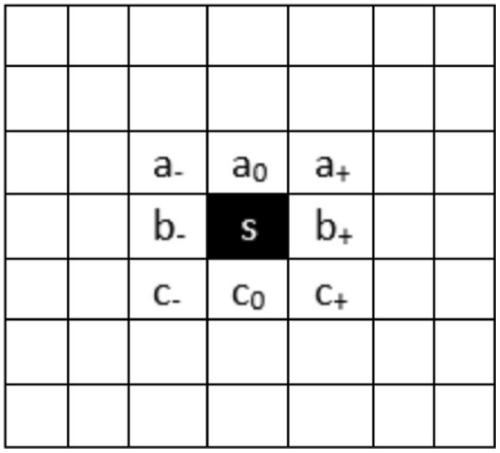


Fig. 1 Positions to which the sensor s allowed to move in x and y-direction

is called the total coverage of the network. In order to avoid long chains in the network and maintain the strong connectivity among sensors, we consider scenarios where communication radius is greater than sensing radius [20].

4.4 Existing localized algorithms

A cellular automaton based mobile algorithms *CAMPA* for sensor dispersion and synchronous gathering are proposed in [20] and [12] where an obstacle-free field is considered to deploy mobile sensors. Three local rules to define the dispersion of the sensors in each of the time steps are discussed in [20]. To gather all these sensors at a single location, a synchronous gathering algorithm is proposed in [12]. The local rules for the dispersion of the sensors are : i) Rules for Movement of Sensors ii) Rules for Blocking Movement and iii) Rules for Moving back. In the following subsections, we describe these three rules and synchronous gathering rules with their behavior while facing obstacles. However, details explanation of these rules is discussed in [20] and [12] respectively.

4.4.1 Rules for movement of sensors

CAMPA algorithm determines the movement of a sensor s based on the weighted number of neighbors of s. The weights for the neighbors of s depends on the distance between neighbors and s. As distance gets increased between a neighbor and s, the amount of weight of that neighbor gets lower. As an example, in case of $R_c = 3$, the weights of the neighbors of s at distance 1, 2, 3 are 4, 2, 1 respectively. Let, the sum of weights of neighbors of s in the negative x-direction and positive x-direction are w_{x_-} and w_{x_+}, respectively. Similarly, the sum of weights of neighbors of s in the negative y-direction and positive

y-direction are w_{y_-} and w_{y_+}, respectively. The decision of movement of s in the x-direction is defined as

(i) if $w_{x_-} = w_{x_+}$ then, s does not move in the x direction
(ii) if $w_{x_-} > w_{x_+}$ then, s moves to the positive x direction
(iii) if $w_{x_-} < w_{x_+}$ then, s moves to the negative x direction

Here, movement in y- direction is calculated analogously.

4.4.2 Rules for blocking movement

Two rules for blocking movement are introduced in *CAMPA* algorithm to prevent the network from losing connectivity. First one is $R_c - 1$ blocking rule where a sensor s does not move in the positive x-direction if it does not see any neighbors within distance $R_c - 1$ in the negative x-direction and other three directions follow the same rule. Another one is square blocking rule where if two sensors are connected diagonally with the distance of R_c and all other cells of the square with this diagonal are empty, then this two sensor does not move any of the places outside of that square. In case of $R_c - 1$ and square blocking rule, if multiple sensors are placed into one cell then only one sensor applies the blocking rule and others apply the movement rule to move any other directions.

4.4.3 Rules for moving back

As we are concerned about the connectivity of the network, *CAMPA* algorithm introduces a move back rule which decreases the hop-distance between individual sensors to ensure the strong connectivity of the network. In move back rule, a sensor remembers only whether or not there were neighbors in the direction opposite to the current movement. At any given time t, before moving to the positive x-direction, a sensor s remembers if there are any neighbors in the negative x-direction. If the sensor does not see any neighbor in the negative x-direction at $t + 1$ time but it had at least one neighbor at time t in negative x-direction, then the sensor moves back to the negative x-direction. Move back in y-direction is calculated correspondingly. In case of multiple sensors in one cell, move back rule is applied for only one of them and rest of them apply movement rule to move any direction.

In *CAMPA* algorithm, these rules are giving good result in terms of coverage if there is no obstacle in the field. However, in case of obstacles, these rules do not perform well to maximize the coverage. In the following example in Fig. 2, all the sensor are failed to move any directions in *CAMPA* algorithm (cell marked as red contains obstacles). Here, some cells are in state 0 where sensors can move and increase the coverage as well as maintain the connectivity, but *CAMPA* algorithm fails to make these movements. In this paper, we propose obstacle-avoidance movement rules where sensors can move around obstacles and increase the total network coverage by maintaining network connectivity.

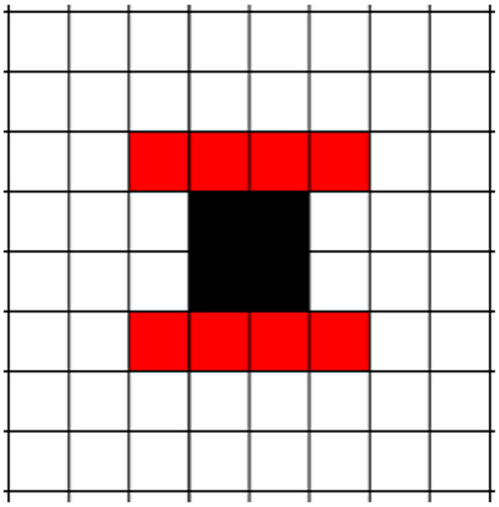


Fig. 2 Sensors are failed to move in any directions for *CAMPA* algorithm

4.4.4 Rules for synchronous gathering

A large number of sensors are dispersed to increase the coverage of the network but these sensors need to move independently to a single location where they can be collected for later use. A cellular automaton based synchronous gathering rules are discussed in [12] where a set of mobile sensors are deployed sparsely in a two-dimensional obstacle free grid and their movement rules gather all the sensors into one cell. In this movement rule, if a sensor sees any neighbors within distance R_c in the positive x-direction and does not see any neighbors within distance R_c in negative x-direction then the sensor moves towards the positive x-direction. If a sensor does not see any neighbors within distance R_c in both positive and negative x-direction or sees at lest one neighbor within distance R_c in both positive and negative x-direction then the sensor does not move in the x-direction. The same rules apply to the y-direction. In this rule, when the enclosing sensors make a 2×2 square or a 1×2 rectangle, the sensors enter into a cycle. Only for this cases, if a sensor sees a sensor directly to the positive x-direction and no sensor in negative y-direction, then the sensor moves directly to the positive x-direction. Similarly, if a sensor sees another sensor directly to the negative y-direction and no sensor in directly or diagonally (c_+) positive x-direction, then the sensor moves directly toward negative y-direction. Finally, if a sensor sees another sensor diagonally (c_+) towards positive x-direction

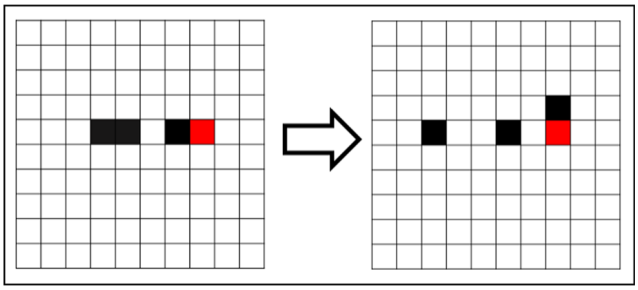


Fig. 3 Obstacle avoidance movement in the only positive x-direction

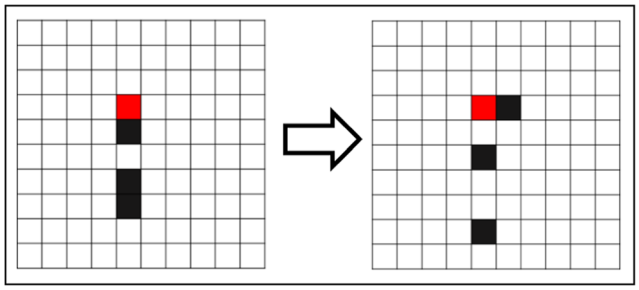


Fig. 4 Obstacle avoidance movement in the only positive y-direction

then the sensor moves diagonally (c_+) to the positive x-direction. Nevertheless, details description of these rules is discussed in [12].

4.5 Obstacle avoidance movement rules

The main contribution of this paper is discussed in this section. We propose an obstacle avoidance movement rules for both sensor dispersion and gathering movement where the rule is applied in three different scenarios. These scenarios are discussed in the following.

4.5.1 Scenario 1

In this scenario, a sensor s wants to move only positive x-direction (position of b_+) and gets blocked by obstacles. In this case, our movement rule considers the movement in y-direction to avoid the obstacle. As the movement on b_+ is blocked, our algorithm immediately checks the other two positions in positive x-direction (a_+ and c_+) for the movement of the sensor. If these positions are not blocked by an obstacle or any blocking rules then, the sensor chooses any of this positions randomly. It is noted that, our algorithm is localized and choosing any position between two available positions generates same results in terms of network coverage and connectivity. If both the positions (a_+ and c_+) are blocked then, the sensor does not move in any directions. Figure 3 shows an example of obstacle avoidance movement in only positive x-direction where cell marked as red contains obstacles. Movement in the only negative x-direction is determined analogously.

4.5.2 Scenario 2

In this scenario, a sensor s wants to move only positive y-direction (position of a_0) and gets blocked by an obstacle. For the movement of the sensor, our movement rule checks the other two positions in positive y-direction (a_- and a_+) and chooses one position randomly if they are not blocked by an obstacle or any blocking rules. Figure 4 shows an example of obstacle avoidance movement in the only positive y-direction. Movement in the only negative y direction is calculated correspondingly.

4.5.3 Scenario 3

In this case of obstacle blocking scenario where a sensor s wants to move in both positive x and positive y-direction (position of a_+) and gets blocked by an obstacle. Here, our movement rule considers the other two positions in positive y-direction and positive x-direction (a_0 and b_+ respectively) and chooses one position randomly if they are not blocked by an obstacle or any blocking rule. Movement in a_-, c_+ and c_- is determined in the same fashion. Figure 5 depicts an example of obstacle avoidance movement in positive x and positive y-direction.

4.6 Obstacle avoidance motion planning algorithms

Obstacle avoidance motion planning algorithm for sensor dispersion and synchronous gathering is presented in Algorithm 1. In this algorithm, at each time step, all the

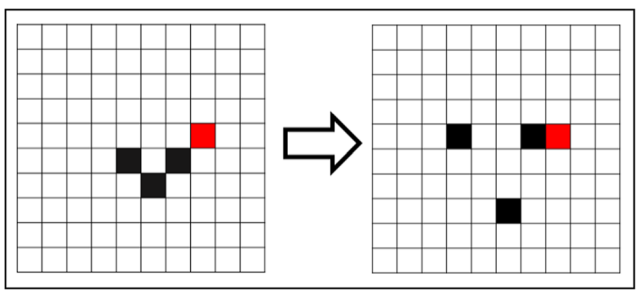


Fig. 5 Obstacle avoidance movement in both positive x and y-direction

sensors run the algorithm locally in order to determine their next position. Thus, the algorithm runs synchronously at all sensors. At the beginning of the Algorithm 1, all the sensors disperse themselves until they reach at a time period $t_{threshold}$. After $t_{threshold}$ time period, all the sensors synchronously gather themselves into one location. In Algorithm 1, dispersion of all the sensors is executed in line 2–8 where we call Algorithm 2 at each time period to calculate the dispersion rule for each of the sensors. The sensors within the radius of communication R_c are considered as neighbors of a sensor. In Algorithm 2, each sensor checks whether it breaks the connectivity with previous sensors at the previous state using the *"Moveback"* rules. If the *"Moveback"* condition is satisfied in line 2 of Algorithm 2, a sensor steps back to its previous state. It is noteworthy that, at the initial stage there is no previous state thus no move back action occurs. Mainly this move back action strengthens the connectivity. If the *"Moveback"* condition is not satisfied (i.e. connectivity has not broken) then Movement rule is applied in line 5 of Algorithm 2. However, that calculated position might be an obstacle. In that case, the next location is updated based on the rule for the obstacles in line 7 of Algorithm 2. Even after that, a sensor may end up in a position where it can not move due to the blocking rules. In line 10 of Algorithm 2, next location is checked, whether it is blocked by any blocking rule. Only if the next location is not blocked, the sensor moves to the next location.

Algorithm 1 Obstacle Avoidance Motion Planning Algorithm

1: **procedure** RUNMOTIONPLANNINGALGORITHM(Σ, $t_{threshold}$) ▷ Σ is a set of sensors and $t_{threshold}$ is the dispersion time period for all sensors
2: **while** $t < t_{threshold}$ **do** ▷ Sensor dispersion until $t_{threshold}$ time period
3: **for** $s \in \Sigma$ at each time period t **do**
4: S_x, S_y is the previous location of s
5: $runDispersionAlgorithm(s, S_x, S_y)$
6: update current location and previous location of s
7: **end for**
8: **end while**
9: **for** each sensor s at each time period t **do** ▷ Synchronous gathering of all sensors
10: $runGatheringAlgorithm(s)$
11: update current location of s
12: **end for**
13: **end procedure**

Algorithm 2 Obstacle Avoidance Sensor Dispersion Algorithm

1: **procedure** RUNDISPERSIONALGORITHM(s, S_x, S_y) ▷ Dispersion movement of sensor s
2: **if** *Moveback* condition is satisfied for s **then**
3: Move back to previous location (S_x, S_y)
4: **else**
5: Calculate next location (S'_x, S'_y) for s using *Movement Rule*
6: **if** Next location (S'_x, S'_y) is an obstacle **then**
7: Calculate (S'_x, S'_y) for s by following cases for obstacles
8: **end if**
9: Apply *Blocking Rule* on (S'_x, S'_y)
10: **if** next location (S'_x, S'_y) is not blocked **then**
11: Move s to next location (S'_x, S'_y)
12: **end if**
13: **end if**
14: **end procedure**

At $t_{threshold}$ time period, all the sensors are dispersed to maximize the network and we need to gather all these sensors into one cell so that we can use them for later purpose. In Algorithm 1, synchronous gathering of all the sensors is executed in line 2–8 where we call Algorithm 3 at each time period to calculate the gathering rule for each of the sensors. A *"synchronous gathering movement rule"* is applied in line 2 of Algorithm 3 to calculate the next location for a sensor s. However, this calculated position might be an obstacle. In that case, the next location is updated based on the rule for the obstacles in line 4 of Algorithm 3. If the calculated next location is also blocked by obstacle then the sensor does not move to any directions.

Algorithm 3 Obstacle Avoidance Synchronous Gathering Algorithm

1: **procedure** RUNGATHERINGALGORITHM(s) ▷ Gathering movement of sensor s
2: Calculate next location (S'_x, S'_y) for s using *Synchronous Gathering Rule*
3: **if** Next location (S'_x, S'_y) is an obstacle **then**
4: Calculate (S'_x, S'_y) for s by following cases for obstacles
5: **if** next location (S'_x, S'_y) is not blocked **then**
6: Move s to next location (S'_x, S'_y)
7: **end if**
8: **end if**
9: **end procedure**

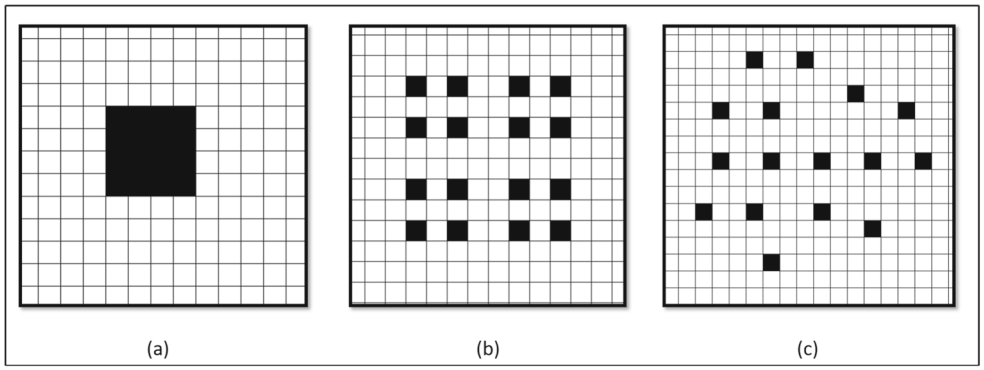


Fig. 6 Comparison of final configurations with 16 mobile sensors (R_c, $R_s = 3, 1$): **a** Initial configuration **b** Final configuration after the deterministic approach of our dispersion algorithm (coverage 100) **c** Final configuration after the probabilistic approach of our dispersion algorithm (coverage 144)

4.7 Analysis

We consider a deterministic and probabilistic approach for the sensor dispersion algorithm. In the deterministic approach, one sensor checks all the rules at each time period to determine whether it should move from the current cell or not. In the probabilistic approach, each sensor verifies the rules with some probability at each time period and probabilistic approach in dispersion algorithm provides better coverage than the deterministic approach similar to [20]. However it takes more time than the deterministic approach to reach a final configuration. In Fig. 6, an example of deterministic and probabilistic approach of our proposed dispersion algorithm is shown where probabilistic approach outperforms the deterministic approach in terms of coverage. However, in case of synchronous gathering we consider a deterministic approach to gather all the sensors into one cell.

We examine the same example of Fig. 2 for our obstacle avoidance dispersion movement rules. Figure 7 shows the final configuration of Fig. 2 for proposed algorithm. It is clear that in our proposed movement rule sensors can move around the obstacles and increase the network coverage.

Theorem 1 *The synchronous gathering rules never increase distance among the sensors with the presence of obstacles.*

Proof In synchronous gathering rule at any given time t, a sensor moves closer to another sensor because it moves away from the empty direction. If this movement is blocked by an obstacle, the sensor looks for another cell in the same direction (away from empty direction). Here, consider the scenario of Fig. 8 where the next location of sensor 1 and 3 is in $1^{'}$ (south-east) and $3^{'}$ (north-west) respectively. If the cell labeled with $1^{'}$ is blocked by an obstacle then the scenario matches with the scenario 3 of obstacle avoidance movement rule. According to the scenario 3 of obstacle avoidance movement rule, the sensor should move to any cell between directly right or directly below from its current location. If the sensor moves directly right, then the distance between sensor 2 and 1 remains same. The distance between 2 and 1 decreases if sensor 1 moves directly below from its current location. It is noted that, if all three potions are blocked then the sensor remains in its current cell. Movement of sensor 3 is also calculated in the same

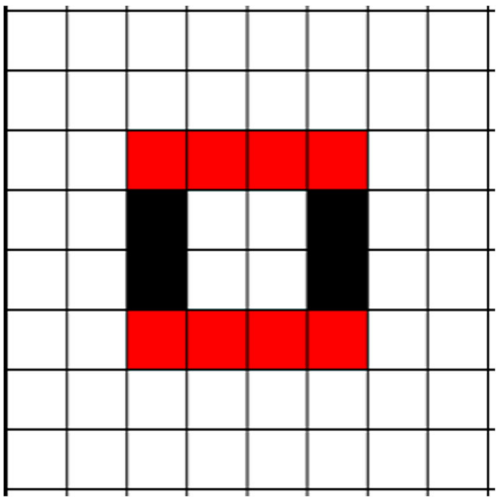


Fig. 7 Final configuration of Fig. 2 for proposed algorithm

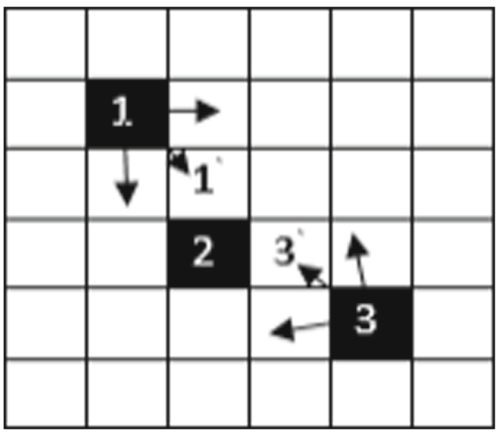


Fig. 8 Scenario for synchronous gathering

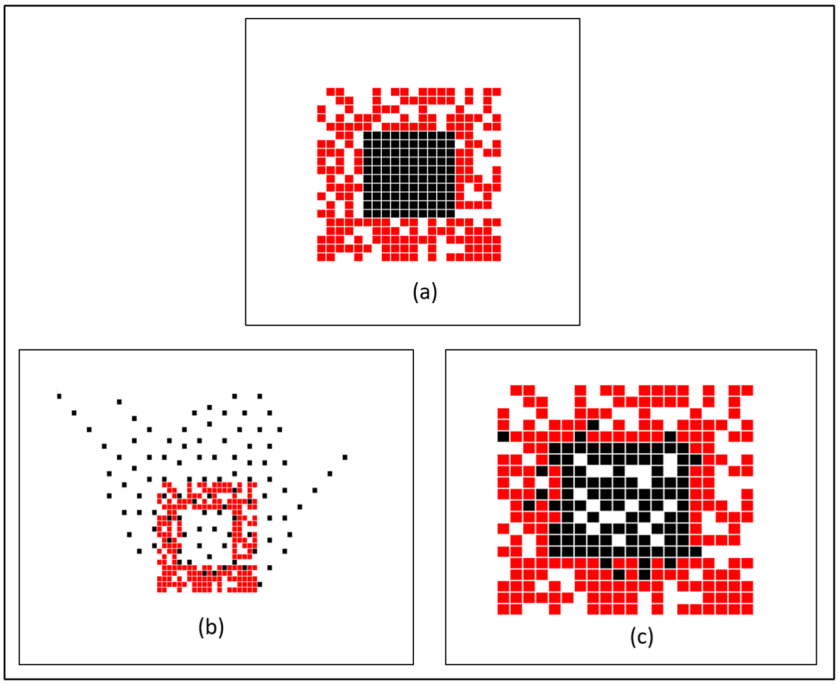


Fig. 9 Comparison of final configurations with 100 mobile sensors: **a** Initial Configuration **b** Final Configuration of Proposed Algorithm **c** Final Configuration of *CAMPA*

fashion. Similarly, for other scenarios, we can show that a sensor always moves opposite to the empty region with the presence of obstacles. Since a sensor never moves to an empty direction, it never increases the distance with other sensors. □

5 Performance evaluation

The aim of this paper is to propose a decentralized algorithm that disperses sensors to maximize the area coverage as well as maintain the connectivity of the network and then gathers all the sensors into one location after reaching to the final configuration. All the sensors maintain a threshold time $t_{threshold}$ which determines the time of dispersion of each sensor. It is noted that all the sensors start the execution of gathering rules at the same time. We measure the performance of dispersion algorithm using the coverage and connectivity of the network. The performance of synchronous gathering algorithm is measured based on the state of the cells. It is noted that synchronous gathering algorithms run in deterministic approach [12]. We consider different scenarios to evaluate the coverage of the network where (R_c, R_s) is $(3, 1)$ and $(3, 2)$. In some cases, the sensor deployment can create long chains to maximize the coverage of the network which is not useful in practice. Therefore, we determine the performance of our algorithm in terms of the *strongly connected coverage* of a network

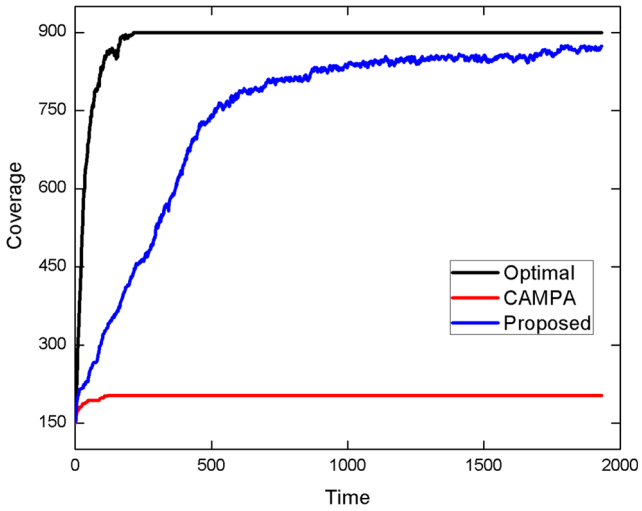


Fig. 10 Comparison of the total coverage for different probabilistic algorithms (Scenario of Fig. 9)

Table 1 Comparison of the no of cells in the state of having sensors in synchronous gathering rules with 100 mobile sensors

Time	Proposed	CAMPA
2000	100	100
2010	67	85
2020	41	60
2030	23	36
2040	14	19
2050	1	6
2100	1	6
2200	1	6

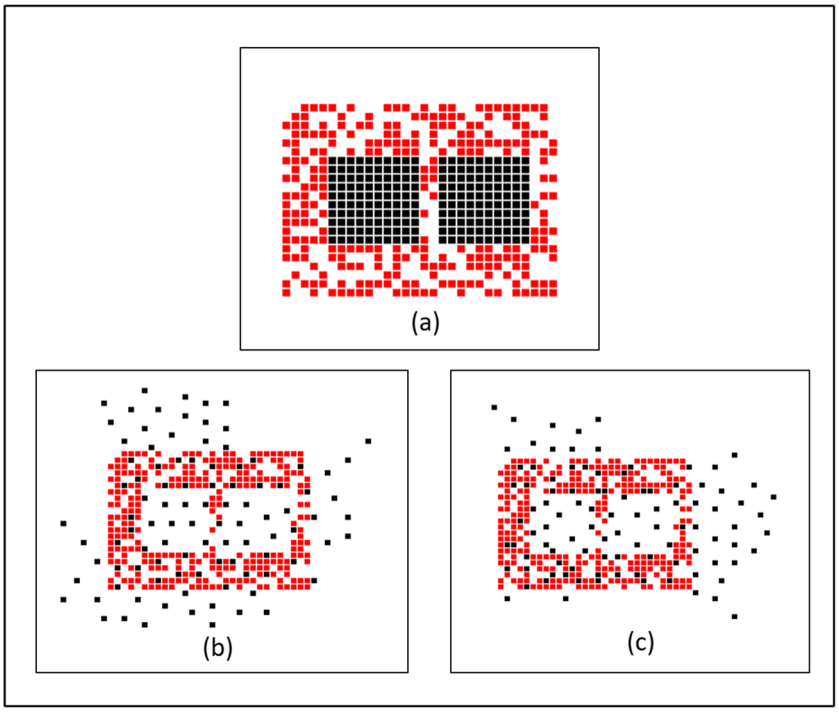


Fig. 11 Comparison of final configurations with 200 mobile sensors: **a** Initial Configuration **b** Final Configuration of Proposed Algorithm **c** Final Configuration of *CAMPA*

($aSCC$). The value of $aSCC$ consisting of n sensors is the ratio between the average coverage of the network ($aCOV$) and the average hop distance of the network (aHD) [20]. Here, the average hop distance of a network (aHD) consisting of n sensors is

$$aHD = \frac{\sum_{s_1 \neq s_2} hd(s_1, s_2)}{n(n-1)}, \tag{1}$$

where $hd(s_1, s_2)$ represents the length of the shortest path of sensors connecting s_1 and s_2. We compare the result of our algorithm with the *CAMPA* algorithm. It is noted that we distribute obstacles within maximum possible coverage area randomly. We apply the probabilistic approach with probability 0.1, 0.2 and 0.3 where 0.1 probability takes more time than 0.2 probability to reach the final configuration and 0.3 probability has more chances of sensors being disconnected. In Fig. 6, performance comparison between deterministic and probabilistic approach is discussed. In case of sensor deployment, probabilistic approach provides better network coverage than the deterministic approach [20]. We find that the probabilistic approach with 0.2 probability gives a better result than all other variants. We implement all the algorithms using the programming language python (Version 3.6). Next, we describe some of

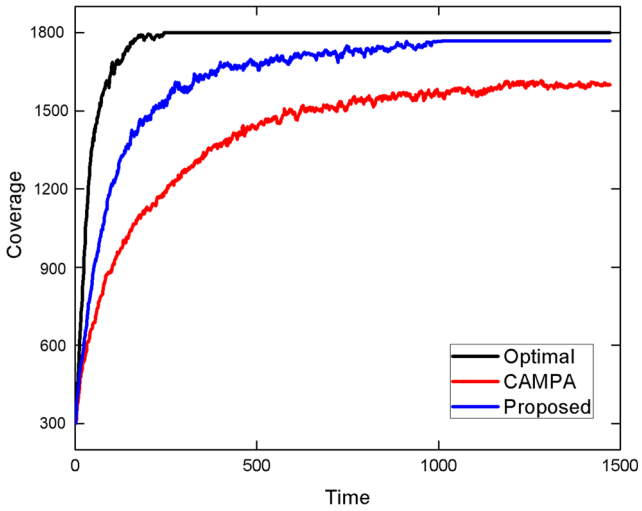


Fig. 12 Comparison of the total coverage for different probabilistic algorithms (Scenario of Fig. 11)

Table 2 Comparison of the no of cells in the state of having sensors in synchronous gathering rules with 200 mobile sensors

Time	Proposed	CAMPA
1500	200	200
1510	170	176
1520	147	164
1550	122	143
1600	102	122
1620	86	93
1650	58	69
1700	23	29
1750	1	29
1780	1	29
1800	1	29

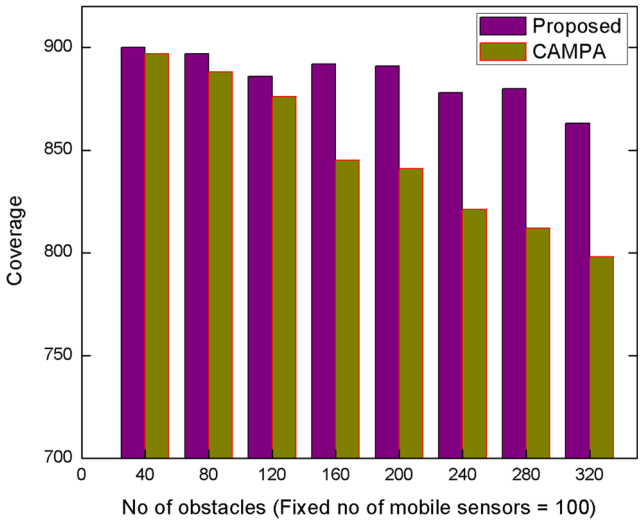


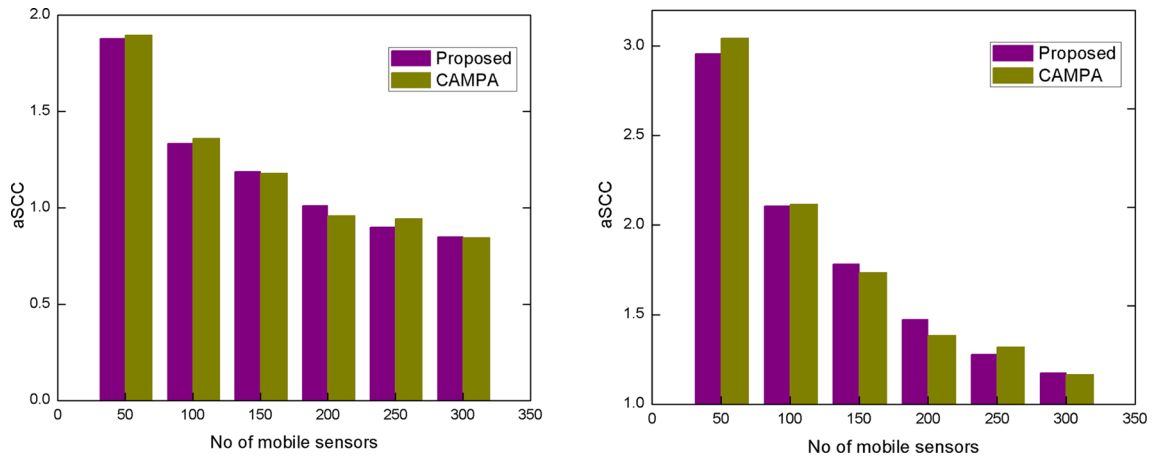
Fig. 13 Comparison of coverage of final configuration between two probabilistic algorithms for various amount of obstacles with 100 mobile sensors

the input scenarios and results of our algorithm. It should be noted that all results are calculated based on 20 experiments.

We place 100 mobile sensors in the middle of the grid as in Fig. 9 where the value of (R_c, R_s) is (3, 1). Figure 9 also shows the final configuration of *CAMPA* and our proposed algorithm. In case of *CAMPA* algorithm, most of the sensors get blocked and can not avoid obstacles to maximize the coverage. In our proposed algorithm, sensors move around the obstacles due to the obstacle avoidance movement rule and maximize the total network coverage. Figure 10 shows the coverage of the algorithms through time. We calculate the optimal coverage considering *CAMPA* algorithm in an obstacle-free field. Therefore, it is clear that total coverage for our algorithm is very close to the optimal coverage as well as provides much better result than the *CAMPA* algorithm. We measure the performance of synchronous gathering algorithm using the number of the cell which contains mobile sensor at each time period. In Table 1 we show the total number of cells that contains sensors in the 2-D grid for our proposed algorithm and *CAMPA* algorithm. At time 2000 all the sensors are in their final configuration for dispersion algorithm. After that, they initiate the synchronous gathering and all the sensors try to gather into one cell. In our proposed algorithm finally, one cell remains which contains all the sensors which indicate that all the sensors are gathered in one location. Meanwhile, in *CAMPA* finally all the sensors move to 6 different cells. From Fig. 9 we can see that for *CAMPA* algorithm most of the sensors remain inside the free spaces covered by obstacles. Therefore, for synchronous gathering algorithm most of the sensors able to meet in one location as in the final configuration they meet in 6 different locations.

To get different scenario we place 200 sensors (as Fig. 11) in the center divided into two clusters but in communication range. Here, for *CAMPA* algorithm, sensors placed on the bottom left side fail to avoid the obstacles in the final configuration but in our proposed algorithm sensors avoids the obstacles and maximizes the coverage. Figure 12 shows the coverage rate for this scenario where our algorithm provides better coverage than *CAMPA* algorithm. Table 2 shows the performance between proposed algorithm and *CAMPA* algorithm for synchronous gathering problem. Here, in our proposed algorithm all the sensors are able to meet in one cell whereas *CAMPA* fails to gather them in one location due to the presence of obstacles. From Tables 1 and 2 it is clear that our algorithm outperforms *CAMPA* algorithm in terms of synchronous gathering problem.

Figure 13 shows the comparison of coverage of final configuration between two algorithms for various amount of obstacles with the same initial configuration of 100



(a) $R_c = 3$ and $R_s = 1$ (b) $R_c = 3$ and $R_s = 2$

Fig. 14 Comparison of aSCC between two probabilistic algorithms for various numbers of mobile sensors

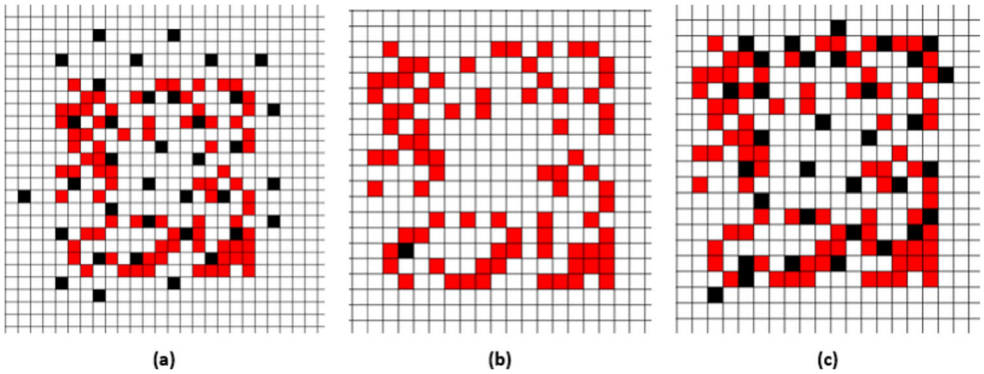


Fig. 15 Comparison between synchronous gathering algorithms with 100 mobile sensors: **a** Initial Configuration **b** Final Configuration after gathering using Proposed Algorithm **c** Final Configuration after gathering using *CAMPA* algorithm

mobile sensors. Here, in case of *CAMPA* algorithm, the total coverage of final configuration gets decreased with the increasing amount of obstacles, but our proposed algorithm maintains better coverage than *CAMPA* algorithm in this cases.

Figure 14a and b show the comparison based on the metric of average strongly connected coverage (aSCC) with $R_c, R_s = (3, 1)$ and $R_c, R_s = (3, 2)$ respectively. These figures indicate that our proposed algorithm maintains pleasant result with compare to *CAMPA* algorithm. It should be noted that in some cases, sensors got disconnected to form multiple clusters while using *CAMPA* algorithm, but our proposed algorithm does not face that type of problem. In some cases, sensors are more strongly connected in *CAMPA* algorithm as most of the sensors cannot pass through obstacles and remain close to each other.

In Fig. 15 we show the final configuration after completing synchronous algorithm for both proposed algorithm and *CAMPA* algorithm. Here, we distribute all the sensors randomly in a 2-D grid and run only synchronous gathering portion of both the algorithms. Figure 15b and c shows the final configuration of our proposed algorithm and *CAMPA* algorithm respectively. Our algorithm gathers all the sensors into one cell whereas *CAMPA* algorithm fails to gather them due to the presence of obstacles. Summarizing above simulation results, we can conclude that our proposed algorithm outperforms existing algorithms in both the coverage maximization and synchronous gathering problem.

6 Conclusion and future work

We consider a locomotion problem in presence of obstacles for MWSN where the main goal is to increase the network coverage while maintaining the connectivity of the network and finally gather all the sensors into one location. A cellular automaton based model is considered where mobile sensors are placed in a 2-D square grid. We propose a probabilistic approach in our coverage maximization algorithm where mobile sensors can avoid obstacles and disperse them to maximize the coverage area of the network. Our experiment shows that our algorithm outperforms existing cellular automaton based algorithm and provides result near to optimal result. We also propose a deterministic synchronous algorithm where all the sensors move around the obstacle and meet in one location. However, our algorithm provides a better result than the existing algorithm for synchronous gathering problem. In our future work, we want to study the behavior of the system where obstacles are also in motion.

Acknowledgment Authors of this paper are grateful to the anonymous reviewers for their constructive comments and meticulous review about this work which led the authors to an improvement of the work.

References

1. Martínez J-F, Garcí A-B, Corredor I, López L, Hernández V, Dasilva A (2007) Qos in wireless sensor networks: survey and approach. In: Proceedings of the 2007 Euro American conference on Telematics and information systems. ACM, New York, p 20
2. Aldeer MMN A summary survey on recent applications of wireless sensor networks. In 2013 IEEE Student Conference on Research and Developement, Putrajaya, Malaysia, pp 485–490, vol 2013
3. Fernández-Lozano J, Gomez-Ruiz J, Martín-Guzmán M, Martín-Ávila J, Carlos SB, García-Cerezo A (2017) Wireless sensor networks for urban information systems: Preliminary results of integration of an electric vehicle as a mobile node. In: Iberian Robotics conference. Sevilla. Springer, Spain, pp 190–199
4. Kar K, Banerjee S (2003) Node placement for connected coverage in sensor networks. In: WiOpt'03: Modeling and Optimization in Mobile. Ad Hoc and Wireless Networks, Sophia Antipolis, p 2
5. Ateś E, Kalayci TE, Uğur A (2017) Area-priority-based sensor deployment optimisation with priority estimation using k-means. IET Commun 11(7):1082–1090
6. Cheng W, Li M, Liu K, Liu Y, Li X, Liao X (2008) Sweep coverage with mobile sensors. In: 2008 IEEE international symposium on parallel and distributed processing, Miami, FL, USA, pp 1–9
7. Gupta V, Jeffcoat DE, Murray RM (2006) On sensor coverage by mobile sensors. In: 45th IEEE conference on decision and control. IEEE, San Diego, pp 5912–5917

8. Yazıcıoğlu AY, Egerstedt M, Shamma JS (2013) A game theoretic approach to distributed coverage of graphs by heterogeneous mobile agents. IFAC Proceedings Volumes 46(27):309–315
9. Yu X, Huang W, Lan J, Qian X (2012) A novel virtual force approach for node deployment in wireless sensor network. In: 2012 IEEE 8th international conference on distributed computing in sensor systems, Hangzhou, China, pp 359–363
10. Rout M, Roy R (2016) Dynamic deployment of randomly deployed mobile sensor nodes in the presence of obstacles. Ad Hoc Netw 46:12–22
11. Choudhury S, Akl SG, Salomaa K (2012) Energy efficient cellular automaton based algorithms for mobile wireless sensor networks. IEEE, Paris, pp 2341–2346
12. Choudhury S, Salomaa K, Akl SG (2015) Cellular automaton based localized algorithms for mobile sensor networks. Int J Unconv Comput 11:417–447
13. Munir A, Uzzaman S, Hossen MS, Choudhury S, Alam M (2016) Localized motion planning algorithm for mobile wireless sensor networks. Int J Unconv Comput 12:363–391
14. Degener B, Kempkes B, Langner T, Meyer auf der Heide F, Pietrzyk P, Wattenhofer R (2011) A tight runtime bound for synchronous gathering of autonomous robots with limited visibility. In: Proceedings of the 23rd annual ACM symposium on Parallelism in algorithms and architectures. ACM, New York, pp 139–148
15. Ando H, Suzuki I, Yamashita M (1995) Formation and agreement problems for synchronous mobile robots with limited visibility. In: Proceedings of the 1995 IEEE international symposium on intelligent control. IEEE, Monterey, pp 453–460
16. Bhagat S, Chaudhuri SG, Mukhopadhyaya K (2016) Fault-tolerant gathering of asynchronous oblivious mobile robots under one-axis agreement. J Discrete Algoritms 36:50–62
17. Baryshnikov YM, Coffman E, Kwak KJ (2008) High performance sleep-wake sensor systems based on cyclic cellular automata. In: International conference on information processing in sensor networks, 2008 IPSN'08. IEEE, St. Louis, pp 517–526
18. Choudhury S, Salomaa K, Akl SG (2012) A cellular automaton model for wireless sensor networks. J Cell Autom 7(3):223–241
19. Cunha RO, Silva AP, Loureiro AA, Ruiz LB (2005) Simulating large wireless sensor networks using cellular automata. In: Proceedings of the 38th annual symposium on simulation. IEEE Computer Society, San Diego, pp 323–330
20. Choudhury S, Salomaa K, Akl SG (2014) Cellular automaton-based algorithms for the dispersion of mobile wireless sensor networks. Int J Parallel Emergent Distrib Syst 29(2):147–177
21. Choudhury S (2017) Cellular automata and wireless sensor networks. Springer International Publishing, Cham, pp 321–335
22. Li W, Zomaya AY, Al-Jumaily A (2009) Cellular automata based models of wireless sensor networks. In: Proceedings of the 7th ACM international symposium on Mobility management and wireless access. ACM, New York, pp 1–6
23. Kumar KJ, Reddy KCK, Salivahanan S (2011) Novel and efficient cellular automata based symmetric key encryption algorithm for wireless sensor networks. Int J Comput Appl 13(4):30–37
24. Teymorian AY, Ma L, Cheng X (2007) Cab: A cellular automata-based key management scheme for wireless sensor networks. In: IEEE Military Communications Conference, 2007. MILCOM 2007. IEEE, Orlando, pp 1–7
25. Torbey S, Akl SG (2012) Reliable node placement in wireless sensor networks using cellular automata. In: International conference on unconventional computing and natural computation. Springer, Berlin, pp 210–221
26. Rashid N, Choudhury S, Salomaa K (2018) Localized algorithms for redundant readers elimination in rfid networks, International Journal of Parallel. Emergent and Distrib Syst 0(0):1–12
27. Chen J, Li J, He S, He T, Gu Y, Sun Y (2013) On energy-efficient trap coverage in wireless sensor networks. ACM Trans Sensor Netw 10(1):2
28. Du YL, Wu L (Nov 2017) Connected sensor cover and related problems. Peer-to-Peer Netw Appl 10(6):1299–1303
29. He S, Gong X, Zhang J, Chen J, Sun Y (2014) Curve-based deployment for barrier coverage in wireless sensor networks. IEEE Trans Wirel Commun 13(2):724–735
30. He S, Shin DH, Zhang J, Chen J, Sun Y (2016) Full-view area coverage in camera sensor networks: dimension reduction and near-optimal solutions. IEEE Trans Veh Technol 65(9):7448–7461
31. He S, Shu Y, Cui X, Wei C, Chen J, Shi Z (2017) A trust management based framework for fault-tolerant barrier coverage in sensor networks. In: 2017 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, San Francisco, pp 1–6
32. Choudhury S, Salomaa K, Akl SG (2015) Cellular automata and object monitoring in mobile wireless sensor networks. In: 2015 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, New Orleans, pp 1344–1349
33. Poudel P, Sharma G (2017) Universally optimal gathering under limited visibility. In: International Symposium on Stabilization, Safety, and Security of Distributed Systems. Springer, Boston, pp 323–340
34. Flocchini P, Prencipe G, Santoro N, Widmayer P (2005) Gathering of asynchronous robots with limited visibility. Theor Comput Sci 337(1-3):147–168
35. Saadatmand S, Moazzami D, Moeini A (2016) A cellular automaton based algorithm for mobile sensor gathering. J Algorithms and Comput 47(1):93–99
36. Yu Q, Jiang W, Leng S, Mao Y (2015) Modeling wireless sensor network based on non-volatile cellular automata. IEICE Trans Commun 98(7):1294–1301
37. Ko SK, Kim H, Han YS (2013) A ca model for target tracking in distributed mobile wireless sensor network. In: 2013 13th international conference on control, automation and systems (ICCAS 2013), Busan, Korea, pp 1356–1361
38. Garzon MH (2012) Models of massive parallelism: analysis of cellular automata and neural networks. Springer Science & Business Media, Berlin
39. Munir A, Hossen MS, Choudhury S Localized load balancing in RFID systems. In 2016 International Conference on Theory and Practice of Natural Computing, Japan, pp 34–45
40. Munir A, Hossen MS, Choudhury S CARRE: cellular automaton based redundant readers elimination in RFID networks. In: 2016 International Conference on Communications (ICC), Malaysia, pp 1–6