A Distributed Cellular Automaton Algorithm for Barrier Formation in Mobile Sensor Networks

Huw Lloyd
School of Computing, Mathematics and Digital Technology
Manchester Metropolitan University
Manchester, United Kingdom
huw.lloyd@mmu.ac.uk

Mohammad Hammoudeh
School of Computing, Mathematics and Digital Technology
Manchester Metropolitan University
Manchester, United Kingdom
m.hammoudeh@mmu.ac.uk

***Abstract*—There is growing interest in the application of wireless sensor networks to the problem of monitoring international borders. In this application, *barrier coverage* is essential in order to ensure that intrusion events are detected. The use of mobile sensors has the potential to enable barrier coverage to be achieved and maintained in hostile environments where the orderly deployment of sensors is impossible. In this paper, we present a distributed cellular automaton based algorithm for autonomous deployment of mobile sensors to achieve barrier coverage. We compare the algorithm with an existing, state-of-the-art algorithm and show that our proposed algorithm achieves barrier coverage with competitive or improved energy cost, and with a communication overhead that is orders of magnitude less. For dense deployment scenarios, our algorithm uses up to six times less energy than the state-of-the-art algorithm.**

***Index Terms*—keywords**

I. INTRODUCTION

Wireless Sensor Networks (WSN) are an important technology for a wide range of monitoring applications, and interest has recently grown in the use of WSN for monitoring international borders [1]–[4]. Border monitoring applications are typically concerned with establishing *barrier coverage* rather than *area coverage* of the sensed region; the aim is to detect entities crossing the region of interest, and this is guaranteed as long as the region is covered by a continuous chain of sensors with overlapping sensing areas (*Strong Barrier Coverage*, [5], [6]); *Strong k-Barrier Coverage* is the existence of k independent strong barriers [5]. Recent work has focused on the construction of strong k-barriers using networks of *mobile* nodes [7], [8], which can move to fill coverage gaps. Ideally, a practical algorithm should be *distributed* in nature, so that no central node is required to coordinate the sensors, and so that the algorithm is scalable to arbitrarily large deployments.

Cellular Automata (CA) present a model of computation which is inherently local and distributed; however CA are capable of generating rich and complex *global* behaviour from simple *local* rules [9]. In this paper, we present a novel CA-based algorithm for the formation of strong $k-$barriers in mobile WSNs. The algorithm is distributed and decentralized; nodes rely only on messages exchanged with neighboring nodes and local information about their surroundings. We show that the energy performance of our algorithm is competitive with the best existing algorithms in most cases, and superior in cases where the nodes are deployed in a dense initial configuration. Furthermore, the communication overhead of the proposed algorithm is shown to be very low compared to the existing state of the art. Although CA have been explored previously in the context of mobile WSNs [10], [11], to the best of our knowledge this is the first application of CA to the problem of barrier formation with mobile sensors.

II. RELATED WORK

Two recent works represent the state of the art in distributed algorithms for barrier formation: MobiBar [7] and DDABC [8]. MobiBar is a distributed algorithm which achieves the maximum level of k-barrier coverage with the available sensors, and is shown by the authors to outperform competing distributed approaches, and to achieve performance close to centralized approaches. Two deployment scenarios are considered: *random*, in which nodes are initially distributed randomly over the Region of Interest (ROI) and *dense*, in which all the nodes are initially clustered in a small area. The algorithm achieves maximum barrier coverage in both scenarios. In MobiBar, barriers are formed in a way that fills the ROI with overlapping nodes starting at the *baseline*, b_0 (the border) and then filling barriers $b_1, b_2, \ldots$ at increasing distances from the border.

Distributed Deployment Algorithm for Barrier Coverage (DDABC) [8] is a distributed algorithm for barrier formation that achieves near-optimal performance with respect to movement distance of nodes. In contrast to MobiBar, DDABC does not form barriers at the border, but aims to fill the ROI with an evenly-spaced grid of nodes which form barriers covering the entire ROI. In order to do this, the ROI is divided into *clusters* of 2×2 cells. A drawback of this algorithm is that each cluster must have at least one node present at the beginning of the algorithm, or this cluster will not receive any nodes in the final deployment. In the simulations presented by [8], this is always the case, since they consider random deployment with relatively large numbers of nodes. However DDABC is likely to fail in the dense deployment scenario, and in random deployments with small numbers of nodes. The formation of clusters places a restriction on the communication radius which needs to be much larger than for MobiBar. The simulations in [8] use a communication radius of 80m,

compared to 15m for MobiBar. However, for the scenarios given (random deployment with large numbers of sensors), DDABC is shown to outperform MobiBar.

In this work, we compare our proposed algorithm against the results for MobiBar presented in [7]. While DDABC outperforms MobiBar in some circumstances, MobiBar has wider applicability, can function in a wide range of deployment scenarios, and does not require long-distance communication. In Section IV, we repeat the scenarios simulated by [7] using our algorithm.

III. CELLULAR AUTOMATON ALGORITHM

The system is decentralized and distributed, so that all nodes run the same algorithm. Space is notionally divided into square *cells*. Each node maintains information about the state of the cell in which it currently resides and immediate neighbouring cells. Information about node movements and positions is communicated through the communication protocol described in Section III-B. Each node updates a finite state machine (FSM, described in section III-C) to determine its next action; this FSM makes use of a CA (described in section III-D) to determine local node movements.

A. Network Model

We assume a rectangular ROI which is delineated by $0 \leq x \leq w$, $0 \leq y \leq h$; the border to be monitored lies on the line $y = 0$ (see Figure 1). We assume that the sensors have a uniform sensing radius r_s and communication radius r_c, and that the sensors are equipped with GPS receivers which enable them to determine their positions. Following [7], we divide the region of interest into square *cells* of size Δ, with $\Delta = 2r_s - 2\epsilon$, where ϵ is the maximum error in position measurement. In this way, if all the cells in a line across the Border of Interest (BOI) are occupied by a node at the centre, barrier formation is guaranteed even in the presence of positional errors. A cell is defined by two indices (i, j) with $x \leq i\Delta < (i+1)\Delta$ and $j\Delta \leq y < (j+1)\Delta$. The choice of origin here is arbitrary (that is, the cell $i = 0, j = 0$ is not special), and in a practical application all transformations that map the GPS coordinates onto x, y values are equally valid. Other than knowing the direction of the border, the nodes require no knowledge of the dimensions of the ROI, as long as they have some means of determining locally whether a neighboring cell is a valid member of the ROI or not.

Our cellular automaton requires that a sensor within a cell must be able to communicate with any node in the *Moore neighborhood* (that is, in its own cell and the eight surrounding cells). This places a limitation on the communication radius, $r_c \geq 2\sqrt{2}\Delta$. Finally, we assume that all nodes possess some property (UID) with which they can be uniquely identified.

B. Communication Protocol

The communication protocol (Table I) provides the messages required in order for the distributed system to maintain each node's local knowledge of the positions and UIDs of its peers in the Moore neighborhood of its current cell.

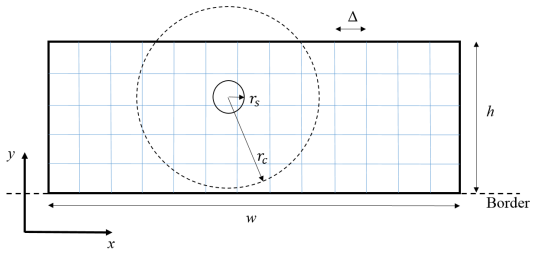

Fig. 1. Geometry of the region of interest and network model.

TABLE I
COMMUNICATION PROTOCOL

Message ID	Payload	Meaning
`initial_pos`	x, y, UID	Broadcast message of initial position of a node.
`arrival`	x, y, UID	Node UID has arrived in a new cell at position (x, y).
`welcome`	x, y, UID	Message sent in reponse to an `arrival` message, reporting the details of an existing node in the neighborhood.
`leave`	UID	Node UID is moving to another cell.
`move`	x, y	Message sent to instruct another node to move to a new position (x, y).

C. Finite State Machine

We assume that each node has access to absolute time. The algorithm proceeds in time *phases*, each of duration τ_{phase}. We require that τ_{phase} is greater than the maximum time for a mobile node to travel between cells, i.e. $\tau_{\text{phase}} > 3\sqrt{2}\Delta/(2v)$, where v is the movement speed of a node. For typical values ($\Delta = 10\text{m}, v = 0.5\text{ms}^{-1}$), $\tau_{\text{phase}} > 42\text{s}$. Since this time is large compared to expected discrepancies between individual nodes' clocks we assume that all nodes will simultaneously be in the same phase for the majority of τ_{phase}.

1) Node Data: Each node maintains the following data: `x, y` (the current position), `UID`, `i_phase` (the current phase, given by $\lfloor t/\tau_{\text{phase}} \rfloor$), `neighbors` (an array of lists of `node` data structures which each contain `x, y, UID` for the cell's neighbours), and `state` (the current state of the state machine – one of `STATE_START, STATE_IDLE, STATE_RUNCA` or `STATE_MOVE`).

2) Start State: In the start state a node initializes the empty `neighbours` data structure, adds itself to the list for the central cell, then sends a broadcast `initial_pos` message before setting the state to `STATE_IDLE`.

3) Idle State: The idle state is a loop which runs while the state remains idle. Messages are handled first; on receiving `initial_pos` or `arrival` messages, new nodes are added to the `neighbours` data and a `welcome` message is sent to the source of any `arrival` message. If a `move` message is received, the target position is saved, a broadcast `leave` message is sent, and the state is set to `STATE_MOVE`. Finally, if $\lfloor t/\tau_{\text{phase}} \rfloor >$ `i_phase`, then `i_phase` is incremented and `state` is set to `STATE_RUNCA`.

4) Run CA State: In this state (Algorithm 1), the cellular automaton is run using the data stored in `neighbors`. The details of the CA algorithm are given in Section III-D; for the purposes of this section, it can be assumed that the CA will return a list of changes (called `change_list`). This list, which may be empty, contains key-value pairs in which the key identifies a cell in the neighborhood, and the value is the net change in the node count. This phase also deals with moving nodes to their final positions in the centre of a cell when there is only one node in the cell.

Algorithm 1 RUNCA State

```
Obtain change_list from CA
Initialize empty list move_list
for each c(key, value) in move_list do
    if c.value< 0 then
        Find c1, the nearest cell in move_list with c2.value> 0
        c.value ← c.value+1
        c2.value ← c2.value−1
        Append (c,c2) to move_list
    end if
end for
if UID = first UID in cell list then
    for each (c0, c1) in move_list do
        Find first node in cell c0 which has not already moved, and send it a move message to move to the centre of cell c1
    end for
end if
if move_list empty AND node has not moved for two stencil phases then
    send move message to self, with target centre of cell
end if
state ← STATE_IDLE
```

5) Move State: In this state, the node moves to its target position defined in the `move` message which initiated the state. On reaching the target, the node broadcasts an `arrival` message and sets its state to `STATE_IDLE`.

D. Cellular Automaton

We restrict any given iteration of the automaton to updating a set of cells whose neighbourhoods do not overlap. This *stencil* of cells to be updated is moved after each phase, so that each cell is updated once every cycle of nine phases. The Moore neighborhood, and the update stencil are illustrated in Figure 2.

1) Stencil Update: The stencil is updated by considering the (i, j) indices of the cell to be updated, along with the current value of `i_phase` for the node requesting the update. Cell (i, j) is in the stencil if $i \equiv$ `i_phase` $\bmod 3$ and $\lfloor j/3 \rfloor \equiv$ `i_phase` $\bmod 3$. The CA update rule is only run for cells which are currently in the stencil.

2) Update Rule: Let the N be the number of cells in the Moore neighbourhood, and $\bar{n}$ be the mean number of nodes per cell, rounded down to an integer. The excess over the mean, e is the remainder on dividing N by $\bar{n}$. We define the *distribution order*, D, in which the e excess nodes are distributed among the cells of M, as

$$D = \begin{cases} S, SE, SW, W, C, E, NE, N, NW & \bar{n} = 0,\ j \text{ even} \\ S, SW, SE, E, C, W, NW, N, NE & \bar{n} = 0,\ j \text{ odd} \\ N, NW, NE, E, C, W, SW, S, SE & \bar{n} > 0,\ j \text{ even} \\ N, NE, NW, W, C, E, SE, S, SW & \bar{n} > 0,\ j \text{ odd} \end{cases}$$

The update rule which gives the updated values n'_c for the cells $c \in M$ is then:

1) Set $n'_c = \bar{n}\, \forall\, c \in M$.
2) Increment n'_c by one for the first e cells in D which are also in M.

That is, the excess e is distributed in the cell order given by D.

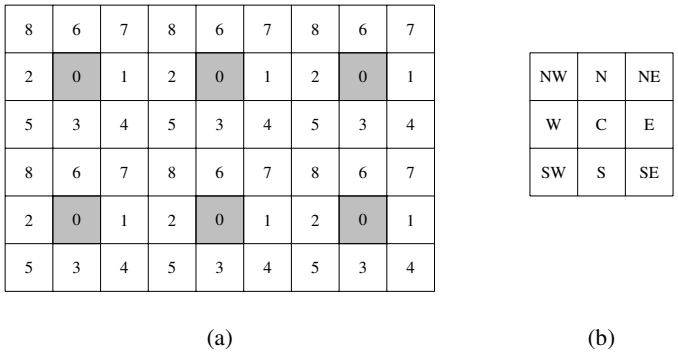

Fig. 2. (a) A section of the CA stencil. Numbers in cells refer to the values of `i_phase` $\bmod 9$ in which cells are active. The shaded cells highlight the configuration of the stencil when `i_phase` $\equiv 0 \bmod 9$. (b) Labelling of the cells in the Moore neighborhood.

IV. EVALUATION

The algorithm has been implemented in a Python simulation (the full source code may be downloaded from https://github.com/huwlloyd-mmu/wsn_ca). Following [7], we run experiments with two different deployment scenarios; *random*, in which the sensors are distributed randomly over the ROI, and *dense*, in which the sensors are distributed randomly within a small region at the bottom left corner of the BOI ($0 < x < \Delta, 0 < y < \Delta$). We use the same parameters as [7] ($w = 400$m, $h = 100$m, $r_s = 5$m, $\epsilon = 0.5$m, $\Delta = 9$m). The communication radius is set to 30m. This is larger than the value of 15m used by [7] although well within the maximum communication range of typical sensor network nodes [12]. We use the same model of energy consumption as [7]. The cost of receiving a message is 1 unit of energy, sending a message costs 1.25 units, movement of 1m uses 300 times the energy of sending a message (375 units), and starting or stopping a movement uses the same energy as 1m of movement. For each configuration, we determined average quantities from 50 simulation runs. In all cases, the algorithm converged to a solution with the maximum number of barriers formed.

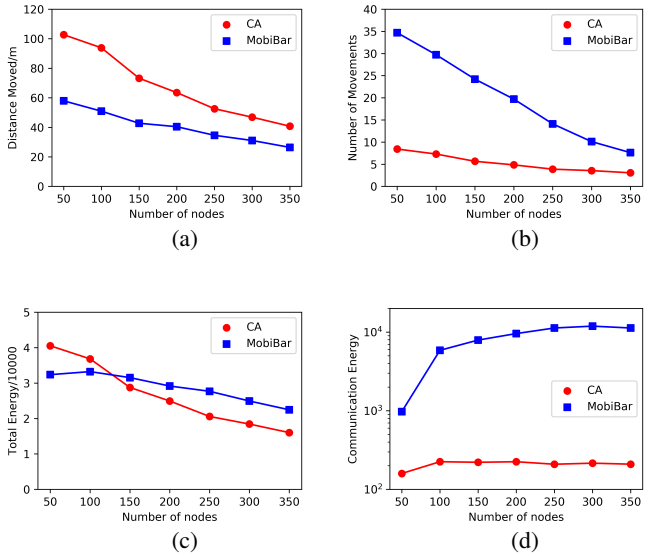

Fig. 3. Random deployment, distance (a), number of moves (b), total energy used (c) and energy used in communication (d)

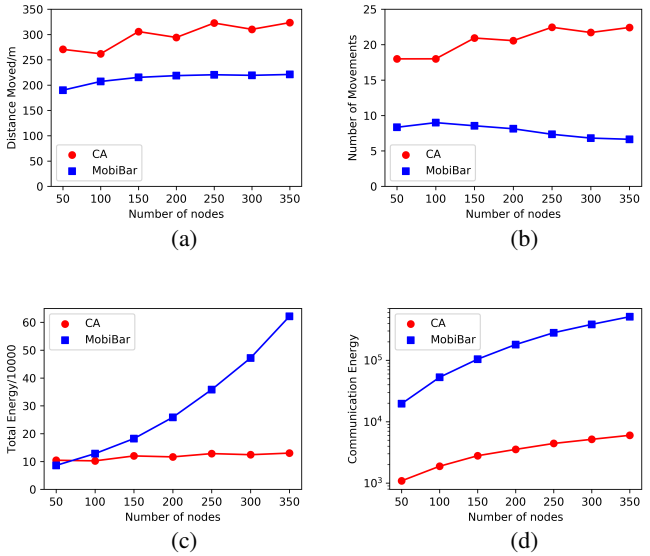

Fig. 4. Dense deployment, distance (a), number of moves (b), total energy used (c) and energy used in communication (d)

Figures 3 and 4 show the results of the simulations compared to results taken from [7]. In the random deployment scenario (Figure 3), the total energy consumption of both algorithms is comparable. However, the CA algorithm uses less energy in communication by an order of magnitude; the algorithm is far less dependent on communication and therefore may hold an advantage over MobiBar when conditions are difficult for wireless communication. Figure 4 shows the results for the dense deployment. The difference between the two algorithms is clearer in this scenario. The energy consumption for the CA algorithm remains roughly constant, whereas there is a sharp increase in energy requirement for MobiBar as the density of nodes increases. In this scenario, the energy consumed by MobiBar is dominated by communication. We conclude that in the dense deployment scenario, the CA algorithm has a clear advantage over MobiBar. It should also be noted that DDABC would likely fail in this scenario.

V. CONCLUSION

In this paper, we have presented a novel CA-based algorithm for deployment of mobile sensors to form barriers. The algorithm is decentralized and distributed (and hence scalable), relatively simple to implement, and gives the maximum possible number of barriers for a given number of sensors. We used simulations of the algorithm to make comparisons with the state-of-the-art algorithm MobiBar. We find that our algorithm is competitive with MobiBar in random deployment scenarios, but outperforms MobiBar by a significant margin in dense deployment scenarios. Moreover, in all scenarios the communication overhead of the proposed algorithm is very low compared to MobiBar (in some cases 100 times less) which would make the algorithm more suitable for use in situations in which communication is difficult.

REFERENCES

[1] M. Hammoudeh, F. Al-Fayez, H. Lloyd, R. Newman, B. Adebisi, A. Bounceur, and A. Abuarqoub, "A wireless sensor network border monitoring system: Deployment issues and routing protocols," *IEEE Sensors Journal*, vol. 17, no. 8, pp. 2572–2582, April 2017.

[2] C. Komar, M. Y. Donmez, and C. Ersoy, "Detection quality of border surveillance wireless sensor networks in the existence of trespassers' favorite paths," *Computer Communications*, vol. 35, no. 10, pp. 1185 – 1199, 2012.

[3] T. Yang, D. Mu, W. Hu, and H. Zhang, "Energy-efficient border intrusion detection using wireless sensors network," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, pp. 1–12, 2014.

[4] Z. Sun, P. Wang, M. C. Vuran, M. A. Al-Rodhaan, A. M. Al-Dhelaan, and I. F. Akyildiz, "Bordersense: Border patrol through advanced wireless sensor networks," *Ad Hoc Netw.*, vol. 9, no. 3, pp. 468–477, May 2011.

[5] S. Kumar, T. H. Lai, and A. Arora, "Barrier coverage with wireless sensors," in *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '05. New York, NY, USA: ACM, 2005, pp. 284–298.

[6] B. Liu, O. Dousse, J. Wang, and A. Saipulla, "Strong barrier coverage of wireless sensor networks," in *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '08. New York, NY, USA: ACM, 2008, pp. 411–420.

[7] S. Silvestri and K. Goss, "Mobibar: An autonomous deployment algorithm for barrier coverage with mobile sensors," *Ad Hoc Networks*, vol. 54, pp. 111 – 129, 2017.

[8] T. G. Nguyen and C. So-In, "Distributed deployment algorithm for barrier coverage in mobile sensor networks," *IEEE Access*, vol. 6, pp. 21 042–21 052, 2018.

[9] S. Wolfram, *A New Kind of Science*. Champaign, Ilinois, US, United States: Wolfram Media Inc., 2002.

[10] S. Choudhury, K. Salomaa, and S. G. Akl, "Cellular automata and object monitoring in mobile wireless sensor networks," in *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, March 2015, pp. 1344–1349.

[11] S. Choudhury, "Cellular automata and wireless sensor networks," *Emergence, Complexity and Computation*, vol. 24, pp. 321–335, 2017.

[12] G. Anastasi, A. Falchi, A. Passarella, M. Conti, and E. Gregori, "Performance measurements of motes sensor networks," in *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '04. New York, NY, USA: ACM, 2004, pp. 174–181.