A Directed Expansion Topology Control Algorithm for Mobile Ad-Hoc Networks

Marco Arguedas, Carlos Perez, Marco Carvalho
Institute for Human and Machine Cognition
40 S. Alcaniz St., Pensacola, FL 32502
Email: {marguedas,cperez,mcarvalho}@ihmc.us

Kelli Hoback, Wayne Kraus
Rockwell Collins
400 Collins Rd NE, Cedar Rapids, IA 52498
Email: {kahoback,wakraus}@rockwellcollins.com

***Abstract*—In this paper we present the directed expansion algorithm, a distributed and proactive topology control algorithm used for increasing the coverage of mobile nodes over a geographical area while maintaining network connectivity. The proposed approach is fully distributed and assumes a free-space wireless propagation model and no-obstacle navigation field. The algorithm relies on a virtual network backbone, and assumes the existence of a distributed mechanism for computing the network backbone. The proposed algorithm also maintains a fully connected topology by making the expansion task aware of the location and extent of the network backbone.**

We describe the directed expansion algorithm in detail, and present simulation results and performance analysis. Our results show the effectiveness of the proposed approach in expanding coverage over a geographical area.

I. Introduction

Physical placement of nodes in a MANET may not be a trivial task. Different circumstances, such as hostile environments where the mobile nodes cannot be air dropped or manually deployed, may prevent an optimal initial arrangement of the MANET nodes.

Mobile Ad-hoc based robotic networks rely on peer-to-peer communications support for coordination and information sharing. In such environments, scout and search tasks performed by some of the units are common, and critical for mission success. In order to support a persistent communications infrastructure during such missions, adaptive topology management mechanisms are necessary, either through a centralized coordination point or though distributed coordination.

In this paper we introduce the directed expansion topology control algorithm. The objective of this proactive algorithm is to mobilize the nodes in a MANET with the objective of extending the physical coverage of the network over a geographical area, while guaranteeing full network connectivity. This fully distributed algorithm assumes a free-space wireless propagation model and obstacle-free navigation, and it makes a best effort to maintain the network fully connected through the utilization of a network backbone. The algorithm assumes the existence of a backbone computation mechanism in the local node.

We provide quantitative result analysis of the algorithm performance, and we also explore the behavior of the algorithm under different configurations of the underlying backbone computation mechanism.

The paper is organized as follows: we first offer a detailed description of the directed expansion algorithm. Then we describe the experimental procedure with analysis of the experimental results, to conclude the paper with a discussion of the results and a conclusions and future work section.

II. Related Work

The problem of maximizing mobile sensor coverage on a given area and in a distributed manner has already been covered in [1], where the authors define two possible techniques for attacking the problem. First, an *Informative* approach is offered where neighboring nodes form *'coalitions'* and, upon exchanging node positions and laser position data, determines the coverage in the local vicinity of the terrain. In the second technique, nodes do not share any information but instead they only rely on *'vision'* information, and nodes are just *repelled* away from their neighbors without performing any local coverage analysis. These approaches, however, are only concerned with expanding node presence in a bounded area, and do not address concerns with possible intra-node communication rupture. An incremental approach is mentioned in [2], where nodes are deployed one at a time and the node most recently introduced makes use of previously gathered information to determine the ideal location for maximizing area coverage. Another effort [3] shows how robots can organize themselves to maximize the coverage of a bounded region, however the solution is based in the continuos creation, modification and destruction of network islands (and therefore allowing the lack of routes between any pair of nodes in the network). The issue of maximizing robot coverage has been explored widely as described in [4] [5] [6] [7], however most of these efforts concentrate in exploring geographical regions by 'sweeping' such areas, instead of trying to maximize robotic presence along a given area. A survey of mechanisms for maximizing coverage for a static MANET is offered in [8], however these are not suitable for situations where scout and search tasks and/or network reshaping are necessary for mission survivability.

The fully distributed algorithm we propose is complementary to previous efforts and novel in that it provides higher-level mission support by focusing on a targeted (i.e. directed) network expansion while maintaining network connectivity at all times.

III. The Directed Expansion Topology Control Algorithm

The Directed Expansion algorithm (*dirExp*) is a distributed topology control algorithm designed to maximize and extend the presence of the nodes in a MANET over a given geographical area while maintaining a route between any two nodes in the network. The algorithm assumes the existence of the following features on each node:

- *Geographical localization and advertisement mechanism*: nodes must share a common frame of reference, and each node needs to be aware of its current position (e.g., via a GPS sensor or some other localization mechanism such as the internal odometer on the mobile component of the node). At the same time, nodes must be able to disseminate their position information to their 1-hop neighbors.
- *A backbone computation mechanism*: the proposed approach assumes the existence of a distributed mechanism for building a network backbone that is executed independently of and in parallel to the directed expansion task, such as CDS [9]. The backbone building mechanism must also provide an API that allows *dirExp* to ask nodes whether they or their 1-hop neighbors are or not part of the backbone. Another desirable feature of the network backbone computation mechanism would be the ability for nodes to temporarily stop the computation. Although not a requirement, this feature would enable an optional mode of operation of the directed expansion algorithm where the state of the backbone remains static during the execution of *dirExp*.
- *Communications range detection mechanism*: nodes must provide an API that allows *dirExp* to determine whether the node is still in communication range to each one of the one-hop neighbors, or whether the node has already lost or is about to lose communication with the neighboring nodes.

The directed expansion algorithm is comprised of two stages: the *directional expansion* stage, and the *neighborhood expansion* stage. As a general rule, and in order to maintain a stable backbone, nodes in either stage of the algorithm are only allowed to participate in the expansion process if they are marked as not being part of the backbone. This means that nodes will remain static as long as they are marked as backbone nodes.

During the *directional expansion* stage, the objective of the algorithm is to mobilize the node in the specified expansion direction, while making sure that the node stays connected to the network backbone at all times. The direction of expansion is specified as an angle relative to the X-axis of the frame of reference, and must be provided by a higher level component (i.e. a human or a higher level application). To guarantee network connectivity to the backbone in this stage, the algorithm constantly monitors the list of 1-hop neighbors and computes and stores the list of the closest backbone neighbors. Upon computing the list, *dirExp* checks whether further movement

```
Input: $n_i$ (this node)
Input: $N$ (set of neighbors of node $n_i$)
Input: $B$ (set of nodes in backbone)
Input: $r$ (angle for moving)
Input: $s$ (speed for moving)
Input: $d_b$ (minimum distance desired to the closest backbone
       node)
Input: $d_n$ (minimum distance desired to any neighbor)
$c \leftarrow$ null;
if $n_i \in B$ then
    $dx \leftarrow 0$;
    $dy \leftarrow 0$;
else
    if $B \cap N \neq \emptyset$ then
        $c \leftarrow$ null;
        for $n \in B \cap N$ do
            if $c =$ null $\vee$ dist$(n_i, n) \leq$ dist$(n_i, c)$ then
                $c \leftarrow n$;
            end
        end
        if dist$(n_i, c) < d_b$ then
            $dx \leftarrow \cos(r)$;
            $dy \leftarrow \sin(r)$;
        else
            $dx \leftarrow 0$;
            $dy \leftarrow 0$;
            for $n \in N - (B \cap N)$ do
                if dist$(n_i, n) < d_n$ then
                    $dx \leftarrow dx + (n_i(x) - n(x))/$dist$(n_i, n)$;
                    $dy \leftarrow dy + (n_i(y) - n(y))/$dist$(n_i, n)$;
                end
            end
            $u \leftarrow \sqrt{dx^2 + dy^2}$;
            $dx \leftarrow dx/u$;
            $dy \leftarrow dy/u$;
        end
    else
        if $c \neq$ null then
            $dx \leftarrow (c(x) - n_i(x))/$dist$(n_i, c)$;
            $dy \leftarrow (c(y) - n_i(y))/$dist$(n_i, c)$;
        else
            $dx \leftarrow 0$;
            $dy \leftarrow 0$;
        end
    end
end
SetVelocity $(s \cdot dx, s \cdot dy)$;
```

Algorithm 1: Directed Expansion Algorithm (one iteration)

can get the node disconnected from the backbone. If such is the case, the algorithm just stops node movement; otherwise the node is free to continue with the current movement pattern. If at some point in time the node encounters itself with no backbone neighbors, the node is asked to move towards the closest backbone neighbor computed in previous iterations.

By now, nodes have stretched out as far as possible from the nodes in the backbone. However, because of the nature of the node movement in the first stage (the movement is mostly rectilinear, with corrections in case of disconnection to the backbone), nodes attached to the external backbone nodes will most likely end up *clumped* together. Figure 1b shows how

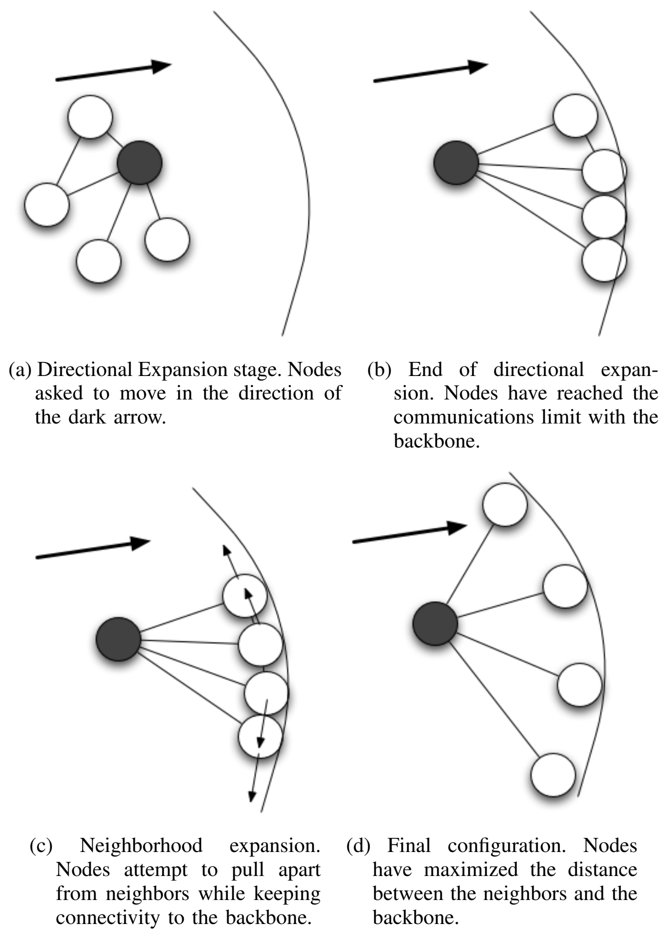
(a) Directional Expansion stage. Nodes asked to move in the direction of the dark arrow.
(b) End of directional expansion. Nodes have reached the communications limit with the backbone.
(c) Neighborhood expansion. Nodes attempt to pull apart from neighbors while keeping connectivity to the backbone.
(d) Final configuration. Nodes have maximized the distance between the neighbors and the backbone.

Fig. 1: Directed Expansion algorithm for a simple topology. Backbone nodes are in darker color.

nodes are clustered together and sub-optimally distributed on the particular geographical area.

To maximize terrain coverage, and minimize possible interference due to node proximity, the next phase of the directed expansion algorithm, *neighborhood expansion*, attempts to push a node away from its neighbors while attempting to maintain connectivity to the backbone. To do this, the algorithm computes the list of all the 1-hop non-backbone neighbors that are closer than a certain acceptable distance d_n. Then, *dirExp* computes the sum of normalized direction vectors between the local node and each of the members of the list. The resultant direction vector is then reversed and used to set the new target direction of the node.

If during the execution of the neighborhood expansion phase the node were to lose connectivity to the backbone, *dirExp* will immediately fall back into the directional expansion stage, where the algorithm should attempt to recover from the backbone disconnection. Algorithm 1 describes each iteration of the directed expansion algorithm.

The presence of the mobile nodes as they traverse the terrain will cause changes in network topology. Such changes in topology may trigger changes to the 'initial' backbone. Depending on the elected backbone computation algorithm (such as CDS [9]), these changes in the backbone have the potential side effect of allowing nodes that initially were part of the backbone to become 'regular' (non-backbone) nodes, thus allowing such nodes to move according to the rules of the directed expansion algorithm.

As modifications to the initial structure of the backbone may or may not be desirable, we have also explored the implications of allowing the modifications to the backbone structure while performing the directed expansion. We discuss such exploration in the following sections.

IV. Experimental procedure

To evaluate the algorithm performance, we have implemented the directed expansion algorithm in the NS-3 [10] network simulator. We executed simulations for network sizes of 10, 20 and 40 nodes.

In all the simulations, the initial position of the nodes have been chosen using a random rectangle position allocator from NS-3. At the beginning of the simulations, the network is fully connected, and the directed expansion algorithm has been configured to try and expand the network on a 45-degree angle in all the cases.

We also developed an NS-3 implementation of CDS [9] as the backbone computation mechanism for the simulations, and ran the simulations in two different modes: one in which we allowed the CDS computation to take place during the execution of the directed expansion (called 'CDS-enabled'), and a second mode (called "ocked-CDS') where, upon placing the nodes on their initial positions, we gave the CDS algorithm 15 seconds to stabilize, and once the CDS backbone was stable, we stopped the computation of the algorithm and 'locked' the state of each node: belonging or not to the backbone. In the 'locked-CDS' mode, the backbone status of the nodes remains unchanged while the node is in the *directional expansion* phase. However, once the node enters the *neighborhood expansion* phase, the computation of the CDS algorithm is resumed[1]. The directed expansion simulation is only started after 'locking' the state of the nodes.

For each network size we ran 10 simulations (per CDS mode) with 10 different random seeds for the initial node position. The random seeds were the same for the simulations of same network size with different CDS mode, so for each network size we ran a total of 20 simulations with 10 distinct initial topologies per network size. Figure 2 shows the initial and final network topologies for one of the 20-node simulations. We chose to calculate d_n (Algorithm 1) as 60% of the average distance between a node and its neighbors (including

[1]As the moving nodes reach the outermost layers of the network, it is necessary need to 'unlock' the computation of the CDS algorithm in order to allow for new nodes to become part of the backbone, and thus allow the network to continue expanding in the required direction. Note that by doing this, new nodes will join the original backbone, however, the backbone nodes that were originally 'locked' in place remain in their original positions and maintain all of their initial backbone neighbors as well. Notice also that nodes will always remain static once they are marked as CDS-backbone nodes, regardless of their initial CDS status.

the backbone neighbors). In all the scenarios, the simulation time was 15 minutes (900 seconds).

V. Experimental results

We are interested in analyzing the performance of *dirExp* in both increasing the terrain coverage and in increasing the presence the nodes in the specified direction. We evaluate the performance of the expansion part of the algorithm by using two different metrics: *terrain coverage* and *network longitude*.

Assuming the nodes have unit-disc coverage (with chosen radius of 150m), we compute the *terrain coverage* as the summation of the areas covered by all the nodes for both the initial and final topologies for each of the simulated network sizes. Table II illustrates how, in both configurations, the algorithm showed an important increase (greater than 40%) in area coverage in all the cases. Under the null hypothesis that the average change in coverage is the same for both configurations of the *dirExp* ('CDS-enabled' vs. 'locked-CDS'), the p-values for the t-test are all under 0.02 for all network sizes. Given that both averages are statistically different we can assume that the 'locked'-CDS version of the algorithm provides a better coverage improvement.

We then compute the *network longitude* by first obtaining the 'center of gravity' of the network as the average of the positions of all the nodes. The *longitude* of the network is defined as the distance between the computed 'center of gravity' and the furthermost node in the network. In all the simulated cases, the directed expansion algorithm showed an average increase in network longitude of at least 54% (with a maximum average increase of 124%), as shown by Table I. All the t-test gave a minimum p-value of 0.05 (and a maximum p-value of 0.228), which do not allow us to determine if the maximum distance between the center of the network and the furthest node is affected by changes in the backbone due to re-computation of the CDS algorithm.

To determine the general orientation of the network, we use a linear regression algorithm to find the function that best fits the positions of the nodes in the terrain. The slope of such function serves as an indicator of the general orientation of the network (with respect to the positive X-axis). For each simulated scenario we then computed the ratio between the reorientation the network was subject to (basically, the difference in degrees from the initial slope to the final slope in the simulation) and the total reorientation the network could have experienced (that is, total number of degrees from the initial to the user requested slope). As Table III shows, in average, after the 15 minutes of simulation the slope of the network was reoriented at least midway (that is, at least 50%) between the initial and target slopes in all cases. Under the null hypothesis that for both modes of the directed expansion algorithm the slope correction factor is the same, the p-values for the t-tests are less than or equal to 0.026 for the smaller network sizes (10 and 20 nodes). However, the p-value for the t-test on the 40 node network is 0.49. Even though for smaller network sizes the mean slope correction values are statistically different, the means for the larger networks are not. We can conclude that, at least for smaller network sizes, the directed expansion algorithm finds a benefit from 'locking' the CDS computation upon start.

TABLE I: Average improvement in area coverage after executing the directed expansion algorithm

	CDS Enabled		Fixed CDS		
Nodes	**Avg**	**StDev**	**Avg**	**StDev**	p**-value**
10	63%	26.25	77%	34.67	0.01275
20	71%	20.36	88%	15.84	0.00493
40	40%	4.57	56%	5.26	0.00001

TABLE II: Average increase in network longitude after for both configurations of the directed expansion algorithm.

	CDS Enabled		Fixed CDS		
Nodes	**Avg**	**StDev**	**Avg**	**StDev**	p**-value**
10	105%	53.20	124%	62.27	0.22839
20	92%	29.78	106%	28.89	0.12266
40	54%	33.65	62%	25.92	0.05534

TABLE III: Average slope correction performance for both configurations of the directed expansion algorithm

	CDS Enabled		Fixed CDS		
Nodes	**Avg**	**StDev**	**Avg**	**StDev**	p**-value**
10	50.41%	27.63	71.99%	28.68	0.0262
20	67.06%	17.38	82.11%	9.22	0.0108
40	50.32%	8.41	63.81%	7.43	0.4901

VI. Discussion

In the previous section, we proved that there is a statistical difference between the two modes of execution of *dirExp* in regards of the maximization of the area of coverage.

The condition on which the CDS computation was executed did not seem to have an effect on increasing the *longitude* of the network. We attribute this behavior to the following: in the 'locked-CDS' version of the algorithm, nodes move freely in the direction of expansion. However, these non-backbone nodes move along the side of the outermost layers of the network may move into a position where there is only one backbone node in the neighborhood. If this occurs, the node will enter the *neighborhood expansion* phase and re-enable the CDS computation. The CDS computation may cause the node to become a backbone node if another moving node enters the neighborhood. The natural derivation of this effect is a 'thickening' of the backbone. Keep in mind that in this mode of the algorithm, once a node becomes part of the backbone, the CDS computation is 'locked' again and the node remains as a backbone node for as long as the directed expansion algorithm is executed.

The described backbone 'thickening' causes the area coverage metric to increase, as there is a larger distribution of nodes throughout the direction of expansion. Figure 2(b) illustrates a network with a 'thickened' backbone. When using 'normal' CDS computation, nodes are allowed to switch back to a non-backbone state, thus allowing them to resume their mobility

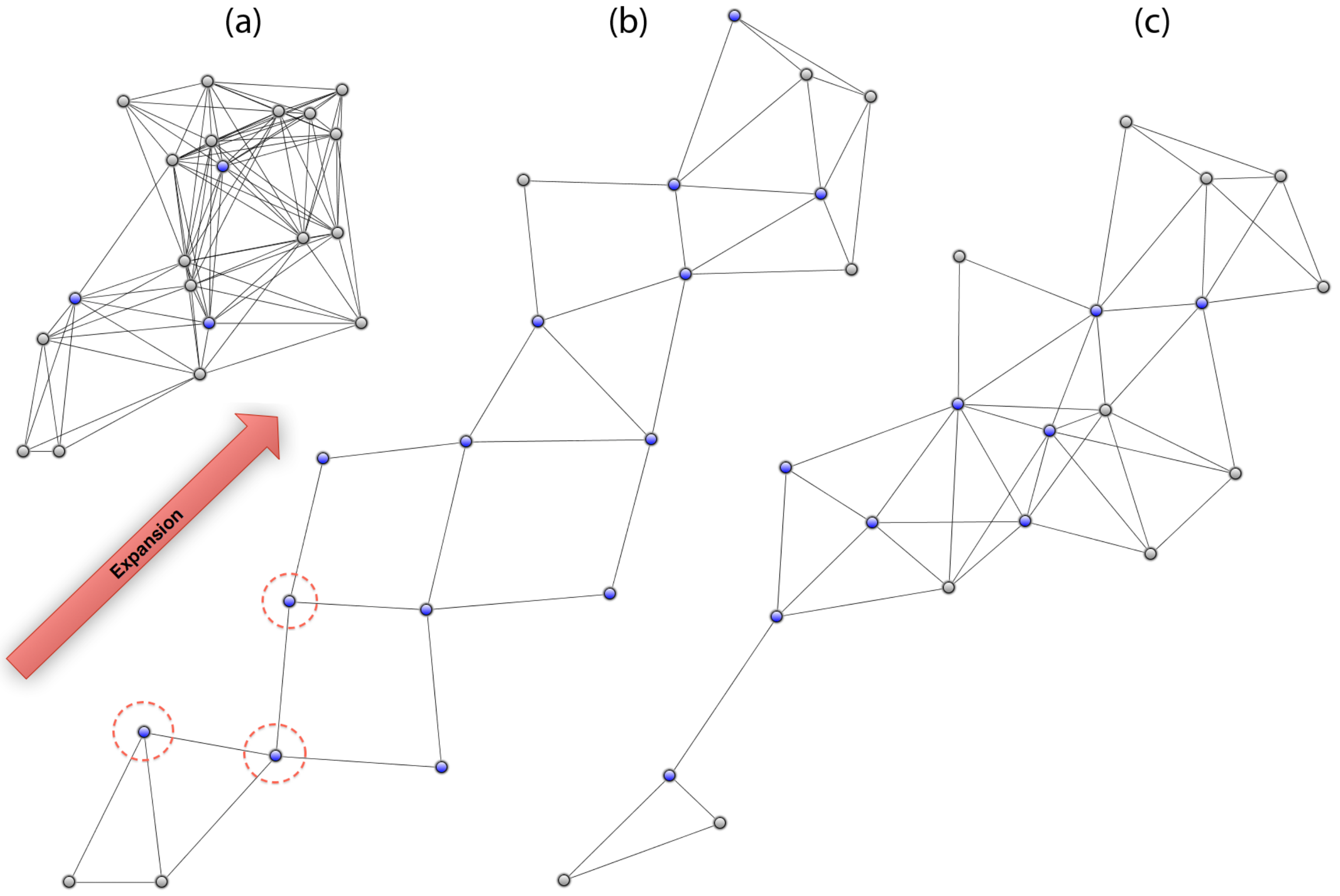

Fig. 2: Topologies for 20 Nodes simulation. (a) Initial state. (b) Final state in 'locked' CDS mode (circled nodes show that the original CDS nodes kept their initial position). (c) Final state, regular CDS mode.

in the direction of expansion. This causes more nodes to reach the furthermost backbone nodes in the direction of expansion and a clumping of nodes in that area. Even though the *neighborhood expansion* stage of the directed expansion algorithm should spread the area of coverage, node concentration in certain areas is higher than in the case where the CDS was 'locked', and so is the intersection of the area of coverage of the nodes.

In both modes of expansion, nodes that move within the internal 'layers' of the network have better chances of expanding the furthest. Because at the edge of the network in the direction of expansion both versions of the algorithm work in a similar manner, we can explain the lack of statistical difference between the two modes of operation when using the longitude metric to evaluate the algorithm.

On a separate note: one could argue that the drop in performance in slope correction shown in large networks is not related to the algorithm itself, but rather to the test scenario. The 40-node tests showed that the slope of the network was corrected in the right direction, however one must remember that as the number of nodes grows, there is a reduction in the speed at which the network can be 'reshaped'. We believe that, had the 40-node tests given more simulation time, the performance in slope correction could have been equal to or better than for the smaller network sizes.

It is also of interest to determine how the evaluated metrics improve over time. Figures 3, 4, and 5 show the evolution over time of area coverage, angle of slope and network *longitude*, respectively. Notice that while the area covered shows almost a linear increase with time, the angle of slope and the network expansion start to exhibit an asymptotic behavior. This last observation allows us to think of a possible optimization where the expansion process is stopped after reaching 'acceptable' thresholds in network slope and network expansion.

Figures 2(b) and 2(c) show some nodes towards the bottom-left corner of the network that did not move in the direction of expansion. These nodes entered the *neighborhood expansion* phase of the algorithm upon reaching the limit of the communications range to the closest backbone node, thus stopping movement. However, these nodes could have moved in the direction of expansion and still stay within communications range with a backbone node. Future versions of the algorithm need to include heuristics to account for this situation.

VII. Conclusions and Future Work

In this paper we have introduced the directed expansion algorithm. We have shown that the algorithm succeeds both in improving network coverage and increasing the network presence in a given direction. The algorithm showed a higher improvement in area coverage when allowed to control the underlying backbone computation mechanism.

The current design of the algorithm does not optimize the mobility pattern of the nodes. Nodes may traverse long distances in order to expand the network coverage. We believe a more integral approach where the directed expansion algorithm also drives the computation of the network backbone would allow for a better utilization of resources.

Our future efforts include evaluating the behavior and performance of the directed expansion algorithm with new backbone computation mechanisms. We may also attempt to investigate the effects of relaxing the rules that stop the backbone nodes from moving.

References

[1] M. Batalin and G. Sukhatme, "Spreading out: A local approach to multi-robot coverage," in *Proc. of the 6th Internat. Symposium on Distributed Autonomous Robotic Systems*. Citeseer, 2002, p. 373382.

[2] A. Howard, M. Matarić, and G. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Autonomous Robots*, vol. 13, no. 2, pp. 113–126, 2002.

[3] A. Winfield, "Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots," *Distributed autonomous robotic systems*, vol. 4, pp. 273–282, 2000.

[4] N. Hazon and G. Kaminka, "Constructing spanning trees for efficient multi-robot coverage," in *Proc. ICRA*. Citeseer, 2006, pp. 1698–1703.

[5] H. Choset, "Coverage for robotics–A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 113–126, 2001.

[6] C. Kong, N. Peng, and I. Rekleitis, "Distributed coverage with multi-robot system," in *ICRA06: Proceedings of the 2006 IEEE International Conference on Robotics and Automation*. Citeseer, 2006, pp. 2423–2429.

[7] N. Hazon and G. Kaminka, "On redundancy, efficiency, and robustness in coverage for multiple robots," *Robotics and Autonomous Systems*, vol. 56, no. 12, pp. 1102–1114, 2008.

[8] M. Cardei and J. Wu, "Coverage in wireless sensor networks," *Handbook of Sensor Networks*, 2004.

[9] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," in *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*. ACM, 1999, p. 14.

[10] T. Henderson, M. Lacage, G. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM Demonstration*, 2008.

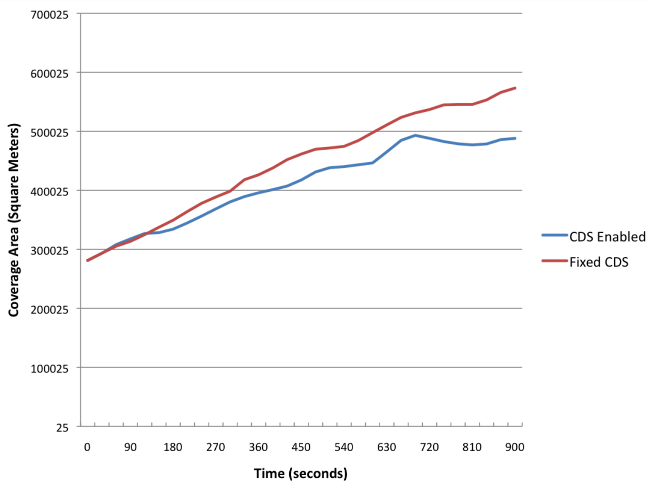

Fig. 3: Evolution of the area coverage in time, for a 20-node simulation.

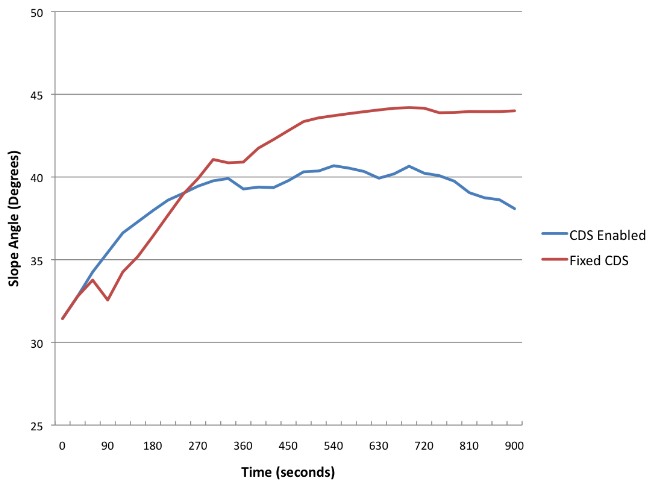

Fig. 4: Evolution of the network slope in time, for a 20-node simulation.

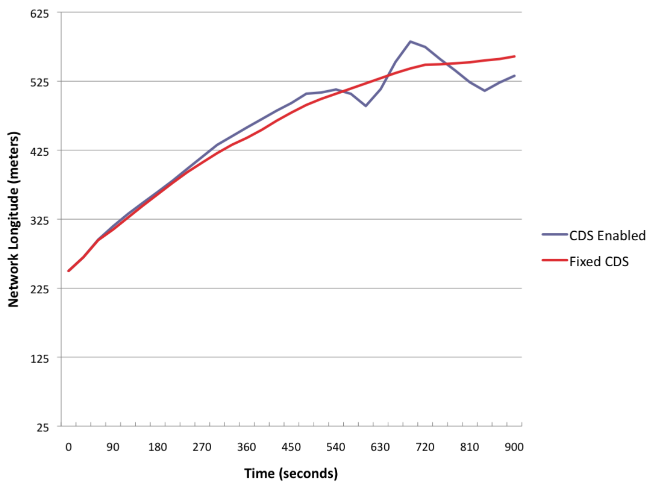

Fig. 5: Evolution of the overall network *longitude* in time, for a 20-node simulation.