Noname manuscript No.
(will be inserted by the editor)

A Localized Algorithm for Bi-Connectivity of Connected Mobile Robots

Shantanu Das · Hai Liu · Amiya Nayak · Ivan Stojmenović

the date of receipt and acceptance should be inserted later

Abstract Teams of multiple mobile robots may communicate with each-other using a wireless ad-hoc network. Fault-tolerance in communication can be achieved by making the communication network bi-connected. We present the first localized protocol for constructing a fault-tolerant bi-connected robotic network topology from a connected network, in such a way that the total movement of robots is minimized. The proposed distributed algorithm uses p-hop neighbor information to identify critical head robots that can direct two neighbors to move toward each other and bi-connect their neighborhood. Simulation results show that the total distance of movement of robots decreases significantly (e.g. about 2.5 times for networks with density 10) with our localized algorithm when compared to the existing globalized one. Proposed localized algorithm does not guarantee bi-connectivity, may partition the network, and may even stop at connected but not bi-connected stage. However, our algorithm achieved 100% success on all networks with average degrees ≥ 10, and over 70% success on sparse networks with average degrees ≥ 5.

Keywords : mobile sensors, robot networks, fault tolerance, distributed algorithm, localized movement control.

1 Introduction

With significant advancements in robotics technology and the emergence of a large number of applications for multi-robot systems, the problem of coordinating between a group of autonomous robots has become an issue of great importance. In such robot systems, coordination between individual robots is essentially accomplished through a wireless ad hoc network. For example, coordination of robotic relay stations was studied in [5] to maintain communication between an explorer and a base station. Application of mobile robotics is vast. Potential applications include military missions, unmanned space exploration, and data collection in sensor fields. But for such applications, coordination of a robot team in pursuit of common task is essential. Existing algorithms for mobile robots coordination are suitable for robots with no or very low failure rates. However, when robots are susceptible to failures, as in many applications, it is critical for robotic networks to incorporate the ability to sustain faults and operate normally. Communication faults in robot networks can be caused by hardware damage, energy depletion, harsh environment conditions and malicious attacks. A fault in a robot can cause stopping transmission tasks to others as well as relaying data to sink. Data sent by a robot will be lost if the receiving robot fails. So, a communication link failure on a route requires data to be re-routed. That is, in order to handle general communication faults, there should be at least two node-disjoint paths between each pair of robots in the network. A

This work was done while the first author was at the School of Information Technology and Engineering, University of Ottawa, Canada.

Shantanu Das
Department of Computer Science, ETH Zurich, Zurich, Switzerland E-mail: dass@inf.ethz.ch

Hai Liu · Amiya Nayak · Ivan Stojmenović
School of Information Technology and Engineering, University of Ottawa, Canada
E-mail: {hailiu,anayak,ivan}@site.uottawa.ca

Ivan Stojmenović
Electronic, Electrical & Computer Engineering, The University of Birmingham, United Kingdom

network is defined to be **bi-connected** if there exist two node-disjointed paths between any pair of nodes in the network, i.e., the removal of any node from the network leaves the network still connected. Therefore, bi-connectivity is the basic requirement for design of fault-tolerant networks [9].

In this paper, we focus on mobile robot networks and study movement control of robots to establish a fault-tolerant bi-connected network. The robot network is assumed to be connected, but not necessarily bi-connected. Achieving connectivity in a disconnected network is difficult due to the lack of communication between the disconnected parts. However, if the network is already connected, we can make it bi-connected (and thus fault-tolerant) by movement of selected robots. Recent work in [2] has shown that fault tolerance can be achieved through globalized robot movement control algorithm. It is a *centralized* algorithm that assumes one of robots or a base station has global information of the network. We focus on the *localized* version of movement control algorithm for building a fault-tolerant robot network. To the best our knowledge, this is the first work on localized movement control for fault tolerance of mobile robot networks.

The rest of the paper is organized as follows. Related work is introduced in Section 2. We propose a localized movement control algorithm to construct bi-connected mobile robot networks in Section 3. Results obtained from extensive simulations are provided in Section 5 to show the effectiveness of our algorithm. Finally we conclude our work in Section 6.

2 Related Work

Many topology control algorithms have been proposed to achieve network reliability in static networks. These algorithms cope with preserving fault tolerance by selecting certain links to neighbors in an already well connected network. The problem of adjusting the transmit power of nodes to create a desired topology in multiple wireless networks was studied in [10]. For static networks, two centralized algorithms were proposed to construct connected and bi-connected networks while minimizing the maximal transmission power of nodes. Two distributed heuristics were further proposed for mobile networks. The basic idea is to adaptively adjust node transmit power according to topological changes and attempt to maintain a connected topology with the minimum power. A more general case for k-vertex connectivity of wireless networks was studied in [7]. Both a centralized algorithm and a localized algorithm were proposed. Both above works assumed that nodes have uniform transmission range. That is, they focused on homogenous networks. Topology control in heterogeneous wireless networks was discussed in [7]. Two localized algorithms were proposed. It was proved that the topologies generated by the proposed algorithms preserve bi-connectivity of networks. An extension of cone-based topology control algorithm was proposed in [1]. Each node decides its own power based on local information about relative angle of its neighbors. It showed that a fault-tolerant network topology is achievable and transmission power of each node is minimized to some extent. The proposed algorithm can be extended to 3-dimensions. All these works on topology control to construct fault-tolerant networks by adjusting transmit power of nodes. Movement of nodes is not a controllable parameter even in the works where mobile networks are considered.

Significant amount of work has been done in coordinating teams of mobile robots or actors. However, little attention was paid to incorporate fault tolerance into these robotic networks. For example, Dynia et al. [5] studied the problem of maintaining communication between an explorer robot and base station by moving other robots along the path.

Mobile robot network can be represented as a graph, where each node is a mobile robot and each edge denotes a communication link between a pair of robots. In a connected graph, a node is called a **critical node** if the graph is disconnected without the node. There are no critical nodes in a bi-connected graph. So, critical nodes are important in designing movement control algorithms to achieve bi-connected networks. Jorgic et al. [6] proposed an approach for localized p-hop critical node detection. To find if a node is critical in the network, a sub-graph of p-hop neighbors of the node is considered. From this sub-graph, the node itself and all its incident edges are excluded. If this resulting sub-graph of p-hop neighbors of a node is disconnected by excluding the node, then the node is critical. Since only local topological information is used, it is specified as p-hop critical node as it may not be globally critical. However, all the globally critical nodes are always p-hop critical for any value of p. As seen in Figure 1, the nodes A, B, and C are 2-hop critical nodes in the given network. We can also notice that nodes A and B in Figure 1 are only 2-hop critical nodes and are not globally critical. However, node C is globally critical in the network and is also 2-hop critical. Experiments showed that over 80% of locally estimated critical nodes and links are indeed globally critical [6].

Our problem is most related to the problem discussed by Basu and Redi [2], where movement control algorithms for fault-tolerant robot networks were proposed. In these algorithms, each mobile robot was as-

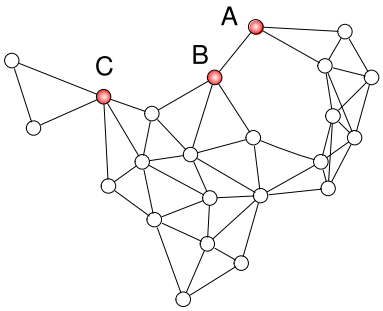


Fig. 1 An example of a network containing critical nodes.

sumed to be aware of global network topology. Based on the topological information, robots decide on their new position, which would thereby create a fault-tolerant network. The goal of these algorithms was to minimize the total distance travelled by all the robots. The authors further proposed an approximation algorithm for two dimensional cases. The basic idea was to divide a network into bi-connected blocks. The network is a block tree of these blocks. A block with maximum number of robots acts as the root of the tree. Algorithm works iteratively merging the blocks to form a single bi-connected block. Merging of the blocks is performed by block movement where each leaf block is moved towards its parent. If parent block is empty then leaf block is moved towards a critical node. After each iteration, robot connectivity is recalculated and block tree is reconstructed as well. However, the proposed algorithms require accurate and global information of entire network. It is applicable to only small size networks. For large scale networks, not only is global network information hard to obtain and maintain, but also the total distance of movements and the communication overhead on robots increase rapidly.

3 Localized Movement Control

In this section, we propose a localized movement control algorithm for fault tolerance of mobile robot networks. To the best of our knowledge, it is the first localized movement control algorithm to achieve bi-connected network topologies. For simplicity, we use a node to denote a mobile robot for the rest of paper. We assume that all nodes in the network have a common communication range r. We further assume that each node has information of its p-hop neighbors. It can be achieved by exchanging or relaying HELLO messages periodically within p-hops. To reduce exchange packets and collisions, we assume there is no RTS / CTS mechanism for transmissions of control packets. The network is assumed to be connected but not bi-connected. The problem of our concern is to control movement of nodes, such that the network becomes bi-connected. The objective is to minimize the total distance moved.

The distributed algorithm is executed at each node and starts as follows. At initialization stage, each node checks whether it is a p-hop critical node [6]. We define the **p-hop sub-graph** of a node as the graph which contains all nodes that are within p-hops from the node and all corresponding links. A node is said to be a **p-hop critical node** if and only if its p-hop sub-graph is disconnected without the node. Since each node is assumed to have knowledge of its p-hop sub-graph, it is able to determine whether it is a p-hop critical node. If a node finds itself a p-hop critical node, it broadcasts a *critical announcement* packet to all its direct neighbors.

To make the network bi-connected, all critical nodes should become non-critical by movement of nodes. Note that the movement of a node may create new neighbors, but it may also break some existing links. Since a critical node is the node that leaves its p-hop sub-graph disconnected without itself, breaking some current links of a critical node may cause disconnection of the network. However, for a non-critical node, the network remains connected if one of its current links is broken. Our basic idea of movement control is to move non-critical nodes while keeping critical nodes static (these nodes may later become non-critical). Depending on the number of critical neighbors of a critical node, we have the following three cases:

1. A critical node that does not have any critical neighbors.
2. A critical node that has exactly one critical neighbor
3. A critical node that has two or more critical neighbors.

We now consider each of these cases separately and describe the behavior of our algorithm in each scenario.

3.1 Case I: Critical node without critical neighbors

In this case, a node finds itself a p-hop critical node and does not receive any *critical announcement* packet from its neighbors. Since it is a critical node, its p-hop sub-graph can be divided into two (or more) disconnected sets (when the node is excluded from the graph). The basic idea is to select two neighbors from two such disjoint sets and move them towards each other until they become neighbors. Suppose distance between the two neighbors is d. Each node should move $(d - r)/2$ directly towards the other node to reach each other. To minimize the total distance of movement of nodes, two neighbors with the minimum distance d among all

possible pairs in the two sets are selected. The critical node sends these two neighbors a *movement control* packet containing their new locations. The two neighbors move to their new locations once the *movement control* packet is received. Note that a non-critical node may have several critical neighbors and it may receive multiple *movement control* packets from different critical nodes. Node IDs are used to assign priorities to critical nodes. Therefore, if a non-critical node receives more than one movement control packets, it always follows direction of the critical node having the largest ID. Note that there is no RTS/CTS mechanism in the network. The critical nodes with smaller IDs do not know and have no need to track the movement of their non-critical neighbors after sending *movement control* packets.

After movement of nodes, any node that loses a current neighbor, or finds a new neighbor, broadcasts a topology updated packet to its neighbors. This packet will be relayed hop by hop to reach p-hops neighbors of the sender. Each node receiving a topology update packet updates its p-hop sub-graph and checks its new status. A new iteration of movement control begins. The movement control algorithm for case 3.1 is illustrated with the following example.

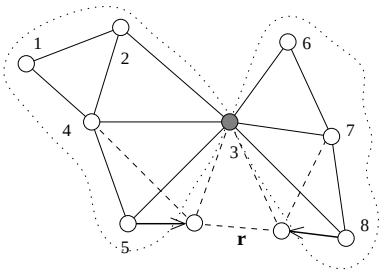


Fig. 2 Critical node without critical neighbor.

Consider the example shown in Figure 2, where node 3 in grey color is critical node and nodes 1, 2, 4, 5, 6, 7, 8 in white color are non-critical nodes. Suppose $p = 2$ in this example. Since node 3 is critical, its 2-hop sub-graph is divided into two disjointed sets $A = \{1, 2, 4, 5\}$ and $B = \{6, 7, 8\}$. Suppose distance of node 5 and 8 is the minimum among all possible pairs in these two sets, i.e. $d(5, 8) \leq d(x, y)$, $\forall x \in A, y \in B$. Node 3 computes new locations of node 5 and node 8 and sends *movement control* packets to them. Final locations of node 5 and node 8 are shown in Figure 2.

3.2 Case II: Critical node with one critical neighbor

In this case, there are two adjacent critical nodes and each critical node has only one critical neighbor. Note that, in such case, both nodes have non-critical neighbors, since otherwise node without any non-critical neighbor will be left with a single neighbor (the other critical node), which means that it will not be critical in the first place, according to the definition (which requires the existence of two disconnected components of neighbors after the node is removed).

Suppose the two adjacent critical nodes are node 4 and node 5, and ID of node 5 is larger than ID of node 4. Our basic idea is to let the critical node with larger ID, node 5 in this case, select one of its non-critical neighbors to move towards the other critical node, node 4. Similar to case 3.1, node 5 divides its p-hop sub-graph into two disjointed sets. Node 4 is contained in one of the two sets. Node 5 searches the other set and selects one of its non-critical neighbors that is the nearest to node 4. Suppose distance between the selected neighbor and node 4 is d. The selected neighbor should move distance $d - r$ to reach node 4 since critical nodes are not allowed to move, to avoid disconnection of networks. Node 5 computes new location of its moving neighbor and sends it a *movement control* packet. The neighbor moves to its new location after receiving the *movement control* packet. Similar to case 3.1, node ID's are used to break the tie when a non-critical node receives multiple movement control packets. After this move, one of critical nodes may become non-critical, and in the next iteration the movement control algorithm for case 3.1 may be applied. Similar to case 3.1, topology update and checking status operations will start after movement of nodes. The algorithm for case 3.2 is illustrated with the following example.

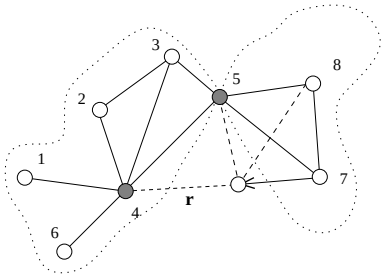


Fig. 3 A critical node with only one critical neighbor.

Consider the example in Figure 3, where nodes 4 and 5 in grey color are critical nodes and nodes 1, 2, 3, 6, 7, 8 in white color are non-critical nodes. Since ID of node 5 is larger than ID of node 4, node 5 leads movement control. Suppose $p = 2$ again. Node 5 divides its 2-hop sub-graph into two disjoint sets $A = \{1, 2, 3, 4, 6\}$ and $B = \{7, 8\}$. Suppose distance of node 4 and 7 is the minimum among all neighbors in B. That is, $d(4, 7) \leq d(4, x)$, $\forall x \in B$. Node 5 computes new loca-

tion of node 7, and sends it a *movement control* packet. Final location of node 7 is shown in figure 3.

Note that this movement may not resolve the problem completely. For example, in this case node 8 may become disconnected from node 7. However, this will be further considered in the next iteration of the same algorithm.

3.3 Case III: Critical node with several critical neighbors

In this case, some critical nodes have more than one critical neighbor. Note that each node sends a *critical announcement* packet to all its direct neighbors if it finds itself to be a p-hop critical node. After that, all nodes in the network know the status of their neighbors. We say that a critical node is **available** if it has non-critical neighbors and is **non-available** otherwise. Thus available critical nodes have non-critical neighbors that are able to move without causing partitioning. An available/non-available critical node broadcasts an *available/non-available announcement* packet to its neighbors. A critical node declares itself a **critical head** if and only if it is available and its ID is larger than the ID of any available critical neighbor, or has no available critical neighbors. Our basic idea for general cases is to use the pair wise merging strategy. Simulation results show that this strategy can efficiently and quickly construct a bi-connected network. Each critical head selects one of its critical neighbors to pair with. Any criterion for selecting will work. To be deterministic, we decide that available critical neighbor (if any) with largest ID is selected, or otherwise non-available critical neighbor with the largest ID. Then the movement control algorithm for case 3.2 is called for each pair to compute the new topology.

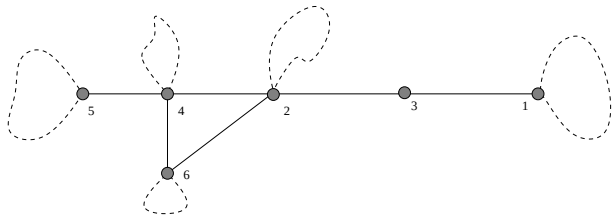


Fig. 4 Critical node with several critical neighbors.

Consider the example in figure 4, nodes 1, 2, 3, 4, 5, 6 in grey color are critical nodes (dashed block with a node is sub-graph of this node). Among these critical nodes, only nodes 1, 5, 6 are critical heads. Node 1 becomes a critical head since node 3 is non-available. Finally, there are three pairs: (1,3), (5,4) and (6,4), dominated by nodes 1, 5, and 6, respectively. Each critical head in a pair calls the movement control algorithm for case 3.2 to merge the pair. Note that some nodes may receive several new neighbors at once, like node 4 in this example.

One can expect that the network density would increase after merging. Pair-wise merging continues until all critical nodes become non-critical, i.e., the network becomes bi-connected. Note that a critical head dominates a pair to merge at each time. No action will be taken if there are no critical heads in the network. So the question that we need to answer is, whether there always exist critical heads in a network that is connected but not bi-connected.

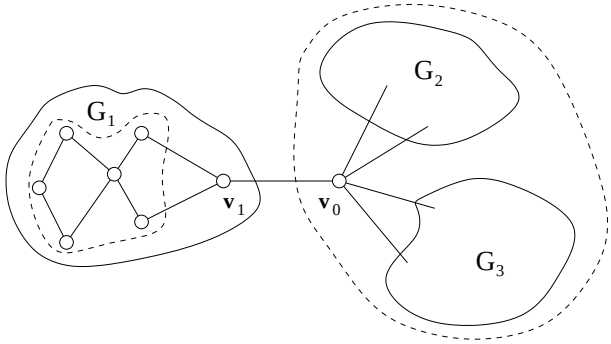


Fig. 5 Node v_1 partitions the graph into two disjoint components, one of which must be strictly smaller than G_1.

Lemma 1 *Any globally connected network has globally non-critical nodes.*

Proof The proof is by contradiction. Let us assume, on the contrary, that all nodes in the connected graph G are globally critical nodes. Thus, for any node v of graph G, the removal of v partitions G into at least two disjoint components. We define $N(v, G)$ as the number of nodes in the smallest component by removing v from graph G. Suppose v_0 is the node whose removal results in the smallest component, i.e. $N(v_0, G) \leq N(v, G)$ for any $v \in G$. Let G_1 denote the smallest component after removing v_0 (see Figure 5). That is, $N(v_0, G) = |G_1|$. We arbitrarily select one of v_0's neighbors in G_1, say v_1. Note that v_1 is critical node. Removing v_1 results in at least two disjoint components. One of these components that contains v_0 is guaranteed to contain all nodes in $(G \setminus G_1)$. It is because G_1 and $((G \setminus G_1) \setminus \{v_0\})$ are two disjoint components, and there is no edge between v_1 and any node in $G \setminus G_1 \setminus \{v_0\}$. Therefore, the number of nodes in the component that contains v_0 is at least $|G \setminus G_1|$. It means that any other component that does

not contain v_0 has a size smaller than $|G_1|$. That is, $N(v_1, G) < |G \setminus (G \setminus G_1)| = |G_1| = N(v_0, G)$. It contradicts the assumption that $N(v_0, G) \leq N(v, G)$ for any $v \in G$.

Note that this theorem is not valid if applied to p-hop criticality. That is, there are connected networks without any p-hop non-critical nodes. Consider a large ring for example. All nodes in this ring are globally non-critical. However, for small p (less than half ring size), all nodes are critical, and thus such rings do not have p-hop non-critical nodes.

Theorem 1 *If the network is globally connected but not bi-connected then it has a p-hop critical head for sufficiently large p.*

Proof Since the network is not bi-connected, it has globally critical nodes. Each globally critical node v has $N(v, G)$ as defined in the proof of Lemma 1. Let v_0 be such a node which minimizes $N(v, G)$. Let G_1 denote the smallest component after removing v_0 (see Figure 5). We arbitrarily select one of v_0's neighbors in G_1, say v_1. We observe that v_1 is not a globally critical node. Otherwise, following the proof of Lemma 1, we will get a contradiction. Therefore v_1 is globally non-critical-node, and thus p-hop non-critical for sufficiently large p. Then v_0 is p-hop critical head and theorem is proven.

Note that, however, Theorem 1 may not be true when p is a fixed small number. Consider, for example, two large rings with a single common node v_0. The network is not bi-connected because of node v_0. However all nodes are p-hop critical and therefore none of them is available, and the network does not have critical heads.

3.4 An example

We now present a concrete example to illustrate the proposed algorithm. For this example, we use a small field of size 20m x 20m with only 12 mobile nodes. The nodes were placed randomly, until we obtained a connected but not bi-connected network. The initial locations of the robots is shown in Figure 6(a). Notice that there are three critical nodes (marked with a darker color) in this example, nodes 0, 4 and 9. Each of these nodes has only non-critical neighbors (i.e. Case I applies here). Thus, according to our algorithm, each of these critical nodes asks some of their neighbors to move to new locations. Node 0 asks node 5 and node 7 to move towards each-other, node 4 asks nodes 3 and 10 to move towards each other, and node 9 asks nodes 2 and 11 to move towards each other. Thus, after the first iteration, nodes 4 and 9 are no longer critical, but node 5 becomes critical due to its movement (see Figure 6(b)). At this stage, there are two critical nodes (nodes 0 and 5) connected to each other. In the next iteration, node 5 (which is the larger one in the pair) asks one of its neighbors (node 1) to move toward node 0 converting it into a non-critical node (Figure 6(c)). Finally in the third iteration, the only remaining critical node (node 0) asks its two non-critical neighbors (nodes 6 and 7) to move towards each other, and now the network becomes fully bi-connected as shown in Figure 6(d). Thus, the algorithm achieves bi-connectivity in only three iterations (in this example) and in each iteration only a few nodes are moved.

4 Maintaining Connectivity

As seen in the example from the last section, the movement of robots may sometimes break existing links in the network. The important question is whether such movements can cause the network to become disconnected. Notice that, in our algorithm, we move only non-critical. Controlled movement of a single non-critical node will never cause disconnection of the network. However, the network may be disconnected when multiple non-critical nodes move concurrently. For example consider the network represented by the graph G in Figure 7.

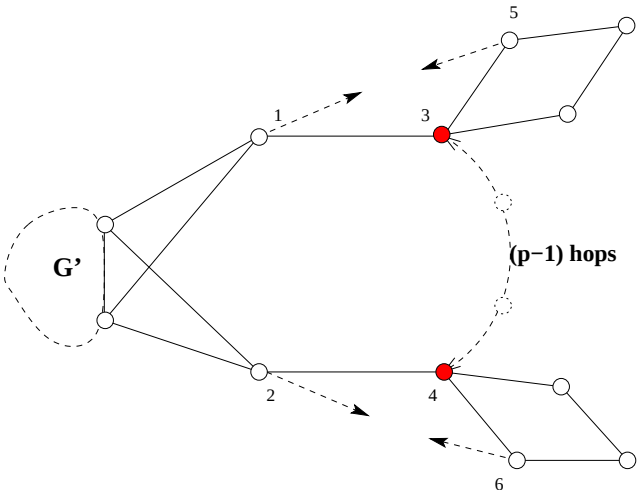


Fig. 7 The simultaneous movement of node-1 and node-2 disconnects the graph G.

Here node 3 and node 4 are connected by a path of $p-1$ hops (shown as a dashed line in the figure). Note that nodes 3 and 4 are globally critical and thus would be identified as p-hop critical nodes. For $p \geq 2$, nodes 5 and 6 are identified as non-critical. For $p \geq 3$, nodes 1 and 2 are also identified as non-critical. This may

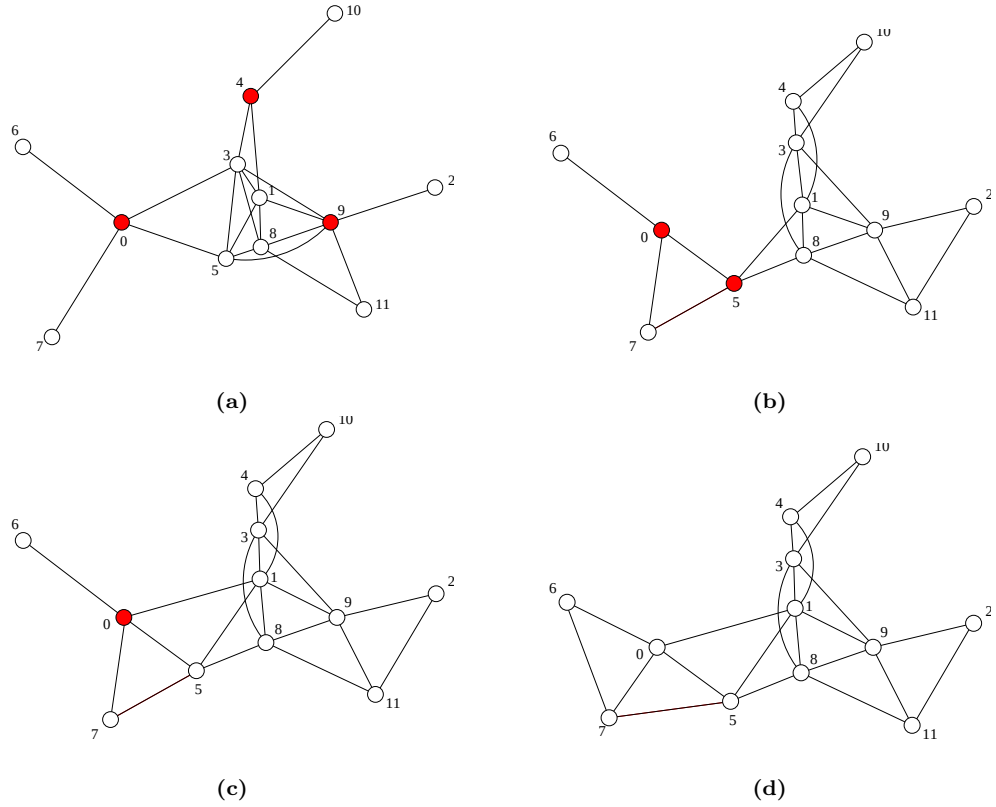


Fig. 6 An illustrative example

not be immediately evident, but can be easily verified. Consider for example the p-hop neighborhood of node 1 (with $p \geq 3$). Notice that there are two distinct paths of length at most p from node 1 to node 4. In other words, there are cycles within the p-hop neighborhood subgraph of node 1. All the neighbors of node-1 are part of some such cycle. Thus, node 1 identifies itself as locally non-critical. A similar case holds for node 2.

Let us assume that $p \geq 3$ and consider how our algorithm behaves in this case. According to our algorithm, node 3 which is critical will ask the non-critical nodes 1 and 5 to move towards each other. Similarly node 4 which is also critical may ask nodes 2 and 6 to move towards each-other. Each of these movements in itself will not cause disconnection of the network. However if nodes 1 and 2 both move at the same time (i.e. in the same iteration) then the links between the subgraph G' and these two nodes (nodes 1 and 2) would be simultaneously broken. Thus, the subgraph G' would be disconnected from the rest of the graph.

A *cut* in a connected graph G is a set of edges $C = \{e_1, e_2, \ldots, e_t\}$ such that $G \setminus C$ is disconnected. To prevent the network from becoming disconnected, we need to ensure that all edges belonging to a cut are not simultaneously broken. Note that when the nodes

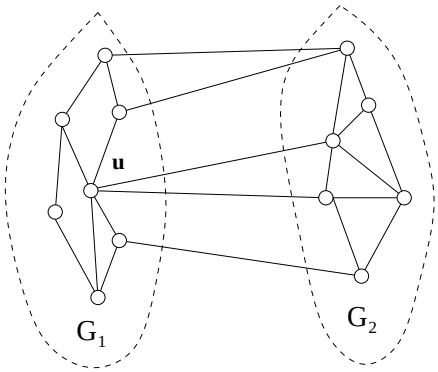


Fig. 8 The cut in the graph G divides it into subgraphs G_1 and G_2.

have only p-hop information, it is not possible to identify the cuts in the network. A node can only identify the local cuts in the p-hop subgraph. However, all local cuts are not global cuts for the network.

Lemma 2 *Any global cut in a graph includes at least one p-hop local cut, for any value of p.*

Proof Suppose G is graph having a global cut $C = \{e_1, e_2, \ldots, e_t\}$ which divides the graph into two components G_1 and G_2 (see Figure 8). Let u be a vertex in G_1 incident to edge e_1. Consider the set C_u of edges

in the p-hop neighborhood of u that belong to C. Note that C_u forms a local cut in the p-hop neighborhood subgraph of u i.e. C_u is a p-hop local cut. However, C_u is a subset of C by definition. This proves the result of the lemma.

Based on the above result, one approach to guarantee connectivity would be to ensure that no local cuts are broken during the movement. Thus, whenever a critical node decides to ask some robot to move, it would first have to communicate and obtain a permission from all other nodes in the p-neighborhood before asking any robot to move. For example, in Figure 7, node 3 needs a permission from node 4 before asking robot 1 to move. So, there would be a large communication overhead for each decision taken by a robot. For instance, for a network of average degree 10 and $p = 3$, a node needs to communicate with 1000 nodes before reaching any decision. Further this approach does not guarantee the progress of the algorithm. We thus use a much simpler algorithm where each node takes a decision based on the available local information without any further communication. This algorithm does not guarantee preservation of connectivity in all cases, but it is much more efficient. In fact, it is hardly possible for any localized algorithm to maintain connectivity while moving nodes, without incurring significant communication costs. However, it is encouraging that, in our simulations, the proposed localized algorithm was always successful on construction of bi-connected network topologies, provided that the initial network was sufficiently dense.

5 Performance Analysis

We tested the performance of our algorithm in a simulated environment and analyzed its efficiency with respect to the distance traveled metric. We also performed comparisons with the existing algorithm that uses global information [2] henceforth called the globalized algorithm. In all our simulations, the proposed algorithm was 100% successful, achieving bi-connectivity of the network within a few iterations, in most cases.

5.1 Simulation Environment

The results presented in this paper are based on simulations performed at the application layer, assuming an ideal MAC layer underneath, with no communication loss and instantaneous delivery of messages. In the following, we use n to denote the size of the network i.e. the number of nodes in the network. The network density d is measured as the average degree of a node in the network. It depends both on the network size and on the area of the sensing field. For most of the experiments, we maintained network density of $d \approx 10$ (i.e. an average of 10 neighbors per node). We performed experiments varying the number of nodes in the sensor field while scaling the sensor field size accordingly. We considered sensor fields with an area from 300 m^2 (for $n = 10$) to 3000 m^2 (for $n = 100$) and the communication range of all the nodes were set to 10m. Nodes were placed randomly within the sensor fields, at the rate of 1 node per $30m^2$ which ensures an average of around 10 neighbors per sensor node. We also evaluated the performance of our algorithm for various values of p, by varying the knowledge range of the robots while keeping the network size fixed. Finally we also performed some simulations on networks of smaller densities. This was done by scaling the area of the sensing field while keeping the network size constant.

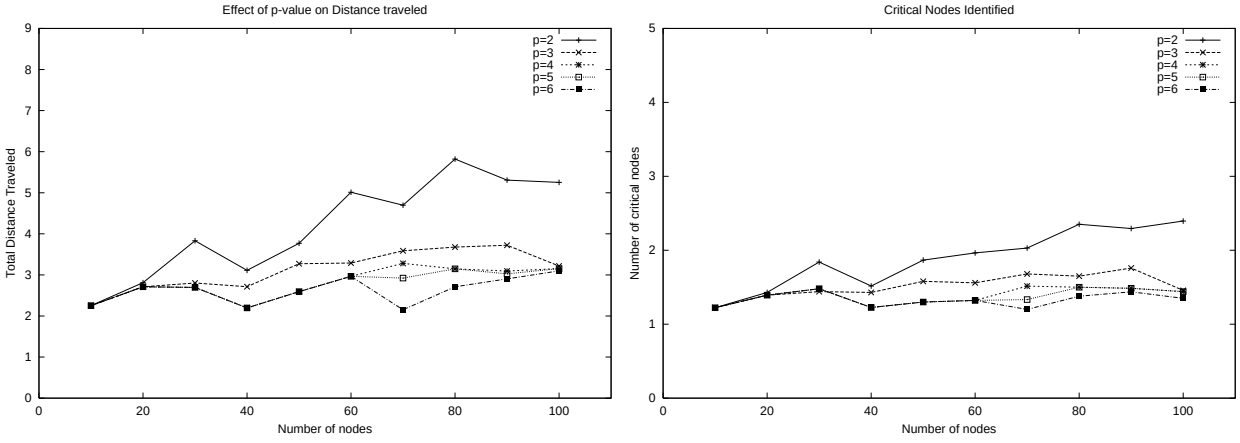
The networks used in the simulations were generated by randomly placing nodes within the sensing field. From such randomly generated networks, we selected the ones which were connected but not bi-connected and this set of networks were used in our experiments. For each experiment and each set of parameters, we executed the algorithm on 100 different networks and the averaged results are presented below. We assumed that each mobile robot has initial knowledge of its own position and can obtain information from its p-hop neighbors. We also assumed that a robot can freely move from one position to another within the sensor field (i.e. there are no physical obstacles in the sensor field).

5.2 Effect of Knowledge Range

The value of p affects the performance of our algorithm to a great extent. If p is small, each node has knowledge about its immediate neighborhood only. Many nodes which are not globally critical may be identified as critical nodes within the p-hop neighborhood.

This results in some unnecessary movement of robots which may be avoided by increasing the knowledge range (i.e. increasing p). We tested our algorithm for various values of p ranging from 2 to 6, on networks obtained using the method described above. Other parameters were kept same as mentioned in section 5.1. Thus, we used networks on size $n = 10$ to $n = 100$, keeping the network density reasonably fixed at $d \approx 10$.

The simulation results for various values of p are shown in Figure 9. The figure also show the number of nodes that were identified as critical at the beginning of the algorithm. As expected, using a larger p-value increases the efficiency of the algorithm because less



(a) **(b)**

Fig. 9 Total distance traveled and number of critical nodes of our protocol for various values of p and n, for $d = 10$

nodes are identified as critical (see Figure 9(b)) and thus, the algorithm makes less movement. However notice that the amount of communication between nodes grows as a factor of Δ^p as the value of p increases (where Δ is the average degree of the network). Thus, we would like to choose a value of p which is as small as possible, without compromising too much on the efficiency of the algorithm in terms of the movement of robots. The results obtained from simulations show that there is large difference in efficiency of the algorithm when using 2-hop information as compared to the case when $p \geq 3$. So, it is best to use the value of $p = 3$ to optimize both the amount of movement and the amount of communication performed during the algorithm.

5.3 Comparison with the Globalized Algorithm

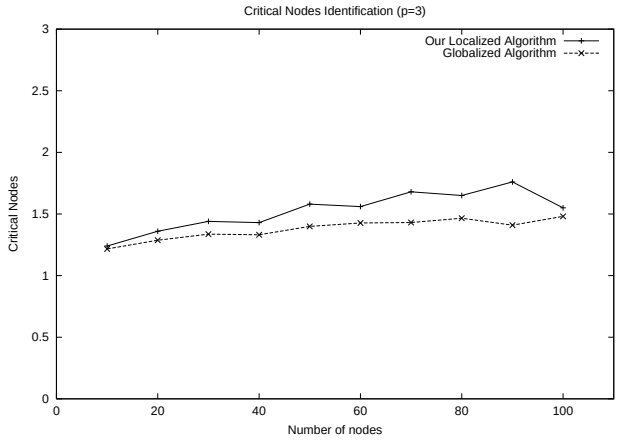
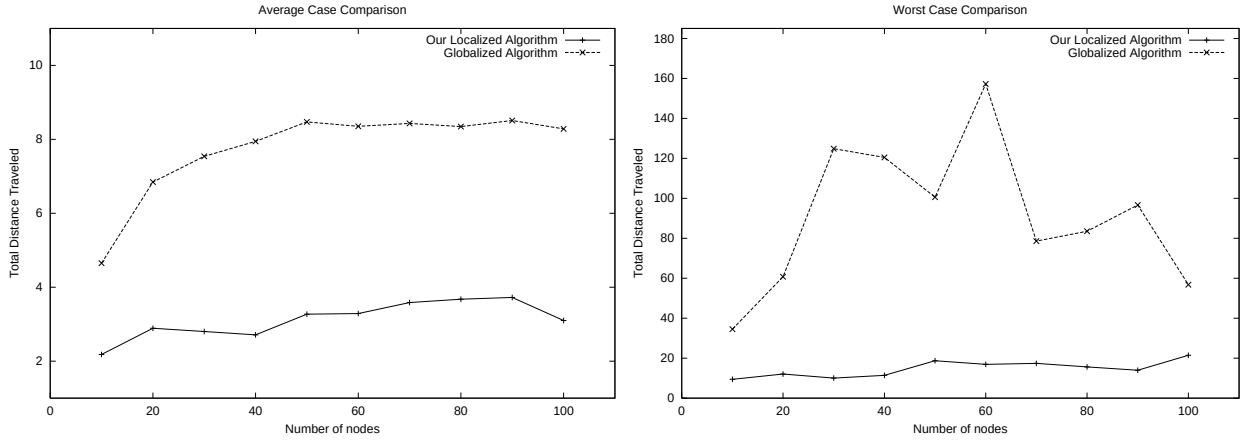


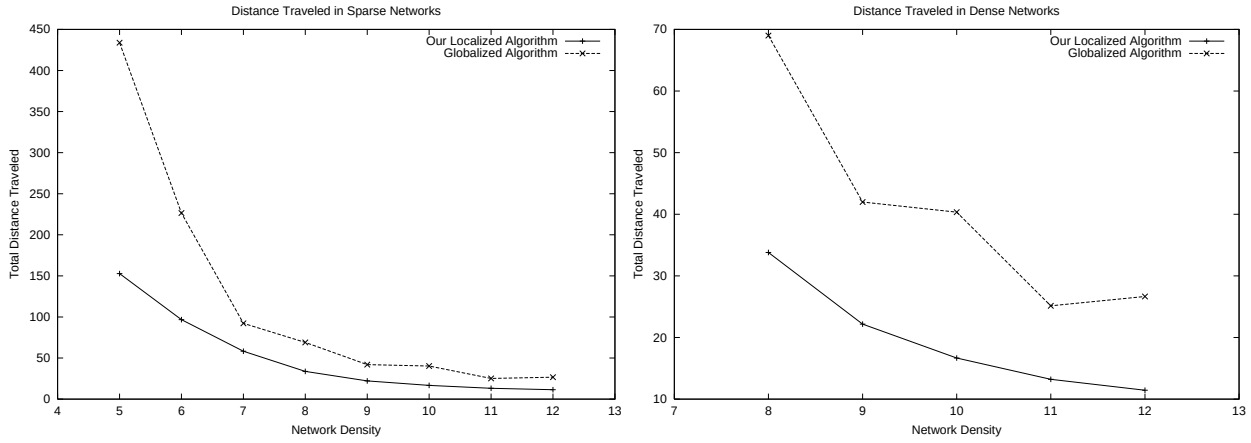
Fig. 11 Critical node identification by the two algorithms in networks of various sizes

We now present results of comparison between the performance of our algorithm with that of the globalized algorithm. For these set of experiments, we used the value of $p = 3$ which gives the best results, as mentioned in the previous section. As before, we considered networks of size $n = 10$ to $n = 100$, with constant network density of 10. In each case, we executed both the globalized algorithm and our localized algorithm (with p set to 3) on the same set of networks obtained using the method described above. Figures 10 (a) and (b) show the average and worst case behavior of the two algorithms, in terms of the distance traveled metric. Our proposed algorithm outperforms the globalized algorithm significantly in all cases. In our algorithm, in each iteration, individual nodes are moved towards each other instead of moving blocks of nodes together as in the case of the globalized algorithm. This results in great performance improvement as can be seen from the simulation results. Notice that, since our algorithm uses only local information to identify the critical nodes, it would identify more nodes as critical nodes as compared to the globalized algorithm. Figure 11 shows the average values for the number of nodes that were initially identified as critical by our localized algorithm using local information up to $p = 3$ hops. This compares favorably with the actual number of critical nodes for the same networks as computed by the globalized algorithm. In other words, our localized algorithm did not identify too many globally non-critical nodes as critical, which suggests that using only local information is not too harmful in general, for identifying critical nodes.



(a) **(b)**

Fig. 10 Comparison of the two algorithms in networks of various sizes (but constant network density $d = 10$). (a) Average case (b) Worst case



(a) **(b)**

Fig. 13 Comparison of the two algorithms in (a) networks of various densities (b) networks of high density only.

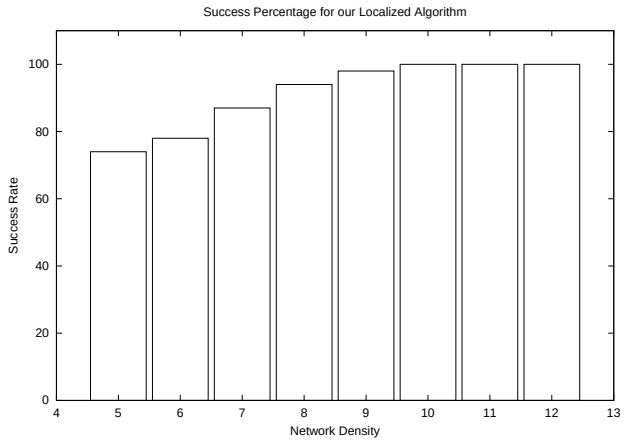


Fig. 12 Percentage rate of success of our algorithm on networks of various densities (but fixed size n=100)

5.4 Performance on sparse networks

When the network is sparse, distances between the robots are larger. So, robots are expected to move more for sparser networks. We performed some experiments to test how our algorithm scales to this situation. We created networks of various densities ranging from 5 to 12 and compared the performance of the two algorithms on these networks. For this set of experiments, the network size was fixed at $n = 100$ and the area of the sensor was scaled appropriately to obtain networks of various densities. All other parameters were same as before. The value of p was set to 3, as explained earlier.

Unfortunately our localized algorithm was not always successful for sparse networks (i.e. for networks with average degree less than 10). Figure 12 shows the success rate of our algorithm on networks of different densities. The algorithm was tested on 100 ran-

domly generated connected but non-bi-connected networks (for network densities $d = 5$ to 12) and we counted the number of times our algorithm was successful in achieving biconnectivity within 50 iterations. As shown in the figure, the algorithm was successful for all test runs on networks with average degree 10 or more. It is encouraging to see that even for the sparsest networks studied ($d = 5$), the algorithm succeeded in 74% of the cases.

In terms of the distance traveled metric, our localized algorithm still outperforms the globalized algorithm on the successful runs. Figure 13 shows the distance traveled by the robots during successful executions of our algorithm as compared to those for the globalized algorithm executed on the same set of networks. As the network becomes sparser, the values for the distance traveled metric increases drastically as shown in Figure 13(a). The same results are shown at a different scale in Figure 13(b) to highlight the fact that differences in performance of the two algorithms are still significant if we consider only the dense networks (e.g. for d=10, the globalized algorithm performs 2.5 times worser than our localized algorithm). These experimental results show that our algorithm is more efficient than the globalized algorithm in all cases (considering only the successful runs). In fact for sparse networks (i.e. network densities 5 or 6), there is a significant improvement in the performance of our algorithm with respect to the globalized algorithm.

6 Conclusions and Future Work

In this paper, we proposed a localized movement control algorithm to construct a fault-tolerant mobile robot network. We presented simulations results to show the effectiveness of our algorithm and its efficiency in terms of success rate and the total distance traveled by the robots. The simulations results for randomly generated connected networks show that our localized movement control algorithm significantly outperforms its globalized counterpart. It is interesting to note that, in most cases, the use of local information (in fact, information about 3-hop neighbors only) is sufficient to convert the network to a bi-connected one in an efficient manner. Thus, global information about the network is not necessary to achieve bi-connectivity. The results shown in this paper are for networks obtained by randomly scattering robots on fixed region and then selecting those which satisfy the condition of connectivity and non-bi-connectivity. On this class of graphs, the proposed algorithm was successful in the construction of bi-connected topologies in most cases, while failing only for a few sparse graphs.

In future, we would like to identify the classes of networks for which our algorithm fails and propose improvements to deal with these difficult cases. An interesting open problem is to determine if there exists any localized algorithm that guarantees bi-connectivity starting from any connected network.

The localized algorithm proposed in this paper achieves fault-tolerance by converting a connected network to bi-connected one. In case the original network is disconnected, the algorithm can be used to make each connected component fault-tolerant. However, the problem of constructing a connected and fault-tolerant network starting from a disconnected network is much more difficult and would be considered in the future.

We are also investigating on applications of mobile robots in (possibly heterogeneous) sensor networks, as data collectors and actors. In these applications, sensor field coverage is an important metric for the algorithm and would be the topic for future work. In particular, both globalized and localized algorithms may suffer from tendency of mobile robot to bi-connect by moving toward the center of the network. This may leave border area in sensor networks unattended by any robot. Therefore, another criterion will be added in the future research, the preservation of area coverage and certain functionalities, while attempting to bi-connect.

Acknowledgements

This research is supported by NSERC Strategic Grant STPSC356913-2007B, and the UK Royal Society Wolfson Research Merit Award.

References

1. M. Bahramgiri, M. Hajiaghayi, and V. Mirrokni. “Fault-tolerant and 3-dimensional distributed topology control algorithms in wireless multi-hop networks”, Proc. IEEE Int. Conf. on Computer Communications and Networks (ICCCN02), pp. 392–397, 2002.
2. P. Basu and J. Redi, “Movement control algorithms for realization of fault-tolerant ad hoc robot networks”, IEEE Network, 18(4):36-44, 2004.
3. R. Chow, and T. Johnson, “Distributed Operating Systems & Algorithms”, Addison Wesley Longman Inc., 1997.
4. J. Cheriyan, S. Vempala, and A. Vetta, “An Approximation Algorithm for the Minimum-Cost k-Vertex Connected Subgraph”, SIAM J. Computing 32(4): 1050-1055, 2003.
5. M. Dynia, J. Kutylowski, P. Lorek, and F.M. auf der Heide, “Maintaining Communication Between an Explorer and a Base Station” , IFIP Conf. on Biologically Inspired Cooperative Computing (BICC’06), vol. 216, pp. 137-146, 2006.
6. M. Jorgic, I. Stojmenovic, M. Hauspie, D. Simplot-Ryl, “Localized Algorithms for Detection of Critical Nodes and Links for Connectivity in Ad Hoc Networks ” , Proc. 3rd Ann. Med. Ad-Hoc Workshop (Med-Hoc-Net), pp. 360-371, 2004.

7. N. Li, and J.C. Hou, “Topology Control in Heterogeneous Wireless Networks: Problems and Solutions” , Proc. IEEE INFOCOM, pp. 232-243, 2004.
8. N. Li, and J.C. Hou, “FLSS: A Fault-Tolerant Topology Control Algorithm for Wireless Networks”, Proc. 10th Ann. Int. Conf. on Mobile Computing and Networking, pp. 275-286, 2004.
9. H. Liu, P.J. Wan, X. Jia, “Fault-Tolerant Relay Node Placement in Wireless Sensor Networks”, Proc. COCOON, pp. 230-239, 2005.
10. R. Ramanathan and R. Rosales-Hain, “Topology Control of Multihop Wireless Networks using Transmit Power Adjustment”, Proc. IEEE Infocom, vol.2, pp. 404-413, 2000.
11. D. Vass, A. Vidcs, “Positioning Mobile Base Station to Prolong Wireless Sensor Network Lifetime” , CoNEXT 2005 Student Workshop, pp. 300-301, 2005.
12. O. Younis, S. Fahmy, and P. Santi, “Robust Communications for Sensor Networks in Hostile Environments”, Proc. 12th IEEE Int. Workshop on Quality of Service (IWQoS), pp. 10-19, 2004.