To appear in *The Twenty-First National Conference on Artificial Intelligence* (AAAI), Boston, Massachusetts, July 2006.

Keeping in Touch: Maintaining Biconnected Structure by Homogeneous Robots

Mazda Ahmadi and Peter Stone
Department of Computer Sciences
The University of Texas at Austin
1 University Station C0500, Austin, TX 78712-0233
Email:{mazda,pstone}@cs.utexas.edu
http://www.cs.utexas.edu/~{mazda,pstone}

Abstract

For many distributed autonomous robotic systems, it is important to maintain communication connectivity among the robots. That is, each robot must be able to communicate with each other robot, perhaps through a series of other robots. Ideally, this property should be robust to the removal of any single robot from the system. In (Ahmadi & Stone 2006a) we define a property of a team's communication graph that ensures this property, called biconnectivity. In that paper, a distributed algorithm to check if a team of robots is biconnected and its correctness proof are also presented. In this paper we provide distributed algorithms to add and remove robots to/from a multi-robot team while maintaining the biconnected property. These two algorithms are implemented and tested in the Player/Stage simulator.

Introduction

Many applications of distributed autonomous robotic systems can benefit from, or even may require, the team of robots staying within communication connectivity. For example, consider the problem of multirobot surveillance (Parker 2002; Ahmadi & Stone 2006b), in which a team of robots must collaboratively patrol a given area. If any two robots can directly communicate at all times, the robots can coordinate for efficient behavior. This condition holds trivially in environments that are smaller than the robots' communication range. However in larger environments, the robots must actively maintain physical locations such that any two robots can communicate — possibly through a series of other robots. Otherwise, the robots may lose track of each others' activities and become miscoordinated. Furthermore, since robots are relatively unreliable and/or may need to change tasks (for example if a robot is suddenly called by a human user to perform some other task), in a stable multirobot surveillance system, if one of the robots leaves or crashes, the rest should still be able to communicate. Some examples of other tasks that could benefit from any pair of robots being able to communicate with each other, are space and underwater exploration, search and rescue, and cleaning robots.

We say that robot R_1 is *connected* to robot R_2 if there is a series of robots, each within communication range of the previous, which can pass a message from R_1 to R_2. It is not possible to maintain connectivity in the face of arbitrary numbers of robot departures: if there are any two robots that are not within communication of one another and all other robots simultaneously depart, the system becomes disconnected. Thus we focus on the property of remaining robust to any single failure under the assumption that the team can readjust its positioning in response to a departure more quickly than a second departure will occur. In order for the team to stay connected, even in the face of any single departure, it must be the case that every robot is connected to each other robot either directly or via *two* distinct paths that don't share any robots in common. We call this property *biconnectivity*: the removal of any one robot from the system does not disconnect the remaining robots from each other.

We tackle the problem of maintaing biconnectivity for multirobot systems by dividing it into three main steps: (1) Checking whether a team of robots is *currently* biconnected, (2) Maintaining biconnectivity should a robot be removed from (or added to) the team, and (3) Constructing a biconnected multirobot structure from scratch. To be applicable for teams of autonomous robots, all algorithms must be fully distributed.

Algorithms and analysis of step 1 are presented in detail in (Ahmadi & Stone 2006a). In this paper, we review the main ideas for that step. The main contribution of this paper is our approach to Step 2 assuming each robot works within a region and does not go out of its assigned region. Step 3—constructing a biconnected multirobot structure—remains as future work. Note that it is possible to achieve step 3, even if inelegantly, by having the robots move back to a base and disperse from there whenever they find that they are no longer biconnected.

For the purposes of this paper, we assume that robots have constant and identical communication ranges. This assumption applies in the case of homogeneous robot teams (or at least teams with homogeneous transmitters) such that the range is not dependent on a robot's battery level. This assumption allows us to assume the connection graph among robots is undirected: if robot A can send a message to robot B, then the reverse is also true. Extension of this work to the case where robots have heterogeneous communication capabilities is also a part of our future work plans.

The rest of the paper is organized as follows. First, we present related work, followed by the graph theory background and assumptions about the investigated multirobot systems. The next section presents an overview of the distributed algorithms to detect if the robots are biconnected. Next, the algorithms to add or remove robots to/from the biconnected structure are provided, followed by empirical results and a conclusion.

Related Work

In recent years, the problem of maintaining connectivity with ad-hoc networks has been studied extensively in the field of mobile robotics. However, to the best of our knowledge, none of the previous methods ensure connectivity in the face of robot failures.

Assuming robots do not fail, some of the previous methods focus on creating connected structures. Howard et al. (Howard, Matadd, & Sukhatme 2002) provide useful heuristics for creating connected structure. However their method is not robust to robot removal and furthermore it is centralized. Nguyen et al. (Nguyen *et al.* 2003) make the robots follow a leader robot in a line, thus making a connected structure.

The methods that do consider the possibility of robots failure do not ensure connectivity. Ulam and Arkin (Ulam & Arkin 2004) provide four different methods of restoring connectivity in the event of robot failure in multirobot teams. Although inspiring, their results show that none of the methods ensure connectivity. In the method presented by Vazquez and Malcolm (Vazquez & Malcolm 2004), each robot tries to be neighbor of at least one other robot, which does not necessarily result in a connected structure.

Anderson et al. (Anderson, Simmons, & Goldberg 2003) construct and maintain a connected structure using a central entity which is assumed to be able to communicate with any robot. All of our algorithms are distributed, and we consider it impractical to assume that there is a center that can communicate with any robot.

Although we are not aware of any previous use of biconnected structures for multirobot systems, biconnected graphs are an old concept in graph theory. Typical related work in graph theory is on algorithms to find a biconnected component in a graph with optimal time complexity, in dynamic graphs, or in a restricted subclass of all graphs (e.g. (Westbrook & Tarjan 1992)). In all these cases, the algorithms are either centralized, or if distributed, each node has full knowledge of the whole graph. Some work in distributed computing is closer in spirit to our work, however a main difference between their problem statement and ours is that in distributed computing (e.g. (Swaminathan & Goldman 1994)), any node can send a message to any other node. That is, the nodes are not restricted to send messages only through the existing edges of the graph.

Preliminaries

We first provide some graph definitions and theorems which will be used later in the paper. For basic graph definitions, such as vertex, edge, neighbor, path and loop please see (Diestel 1997). Later in the section, definitions and assumptions which are specific to our multirobot system will be presented.

Definition 1. Internally vertex-disjoint paths. *Two paths between v_1 are v_2 are internally vertex-disjoint if they have no vertices in common except v_1 and v_2.*

Definition 2. Biconnected graph. *If in graph G, after removing any vertex, it is possible to find a path from any vertex to any other one, the graph G is said to be biconnected.*

Definition 3. Doubly connected vertices. *In graph G, we say vertex v_1 and v_2 are doubly-connected iff there are two or more internally vertex-disjoint paths between v_1 and v_2.*

Lemma 1. *Undirected graph $G(V, E)$ is biconnected if and only if any two vertices $v_1, v_2 \in V$ are doubly-connected.*

Proof. It is a special case of Menger's Theorem (See Theorem 3.3.1 of (Diestel 1997)). □

Note that in undirected graphs one vertex being doubly-connected to all other vertices is not a sufficient condition for the graph to be biconnected. For an example see Figure 1 where v is doubly-connected to all other vertices, but removing v makes the graph disconnected. In the following theorem we show that in an undirected graph if there are two vertices that are doubly-connected to all other vertices, then the graph is biconnected.

Theorem 1. *Undirected graph $G(V, E)$ is biconnected if and only if there exists two distinct vertices $v_1, v_2 \in V$ such that both v_1 and v_2 are doubly-connected to any other vertex in V.*

Proof. For a proof see (Ahmadi & Stone 2006a). □

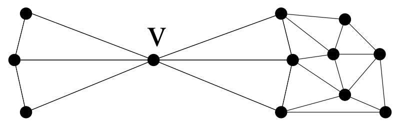


Figure 1: V is doubly-connected to all other vertices, but the graph is not biconnected.

We look at our multirobot system as a graph, such that its vertices are robots and edge $(v_1 v_2)$ exists in the graph iff the robot corresponding to v_1 can communicate directly to the robot corresponding to v_2 (i.e. v_1 and v_2 are in communication range of each other). A formal definition of robot graph follows.

Definition 4. Robot graph *$RG(V, E)$ is a graph, where its vertices (V) are the robots and $(v_1, v_2) \in E$ iff corresponding robots to v_1 is a neighbor of corresponding robot to v_2. Size of V (i.e. number of robots in the multirobot team) is called n in this paper.*

Assumption 1. *Robots are aware of the maximum number of robots in the system, which can be considerably higher than the actual number of robots. The maximum number is called N throughout the paper.*

Assumption 2. *Robots have identical communication capabilities.*

As a result of the above assumption, the neighbor property is symmetric, and the robot graph is undirected.

Definition 5. Connected. *We say robot R_1 and robot R_2 are connected, when in the corresponding robot graph, there is a path between R_1 and R_2.*

Assumption 3. *Each robot has a unique and ordered ID. For robot X its ID is called X.id.*

Next, the definition and assumption regarding the communication between robots is provided.

Definition 6. Message, stamped message. *For our purposes,* a message*, which is used for robot communication, is a string in format $(T, (S))$, where T indicates the type of the message, and S is a list of robot* stamps*. Message $(T, (S))$ is said to be* stamped *by robot R iff $R.id \in (S)$. Robot R stamps message $(T, (S))$ by generating new message $(T, (S, R.id))$.*

Assumption 4. *When called for by the protocol, robots relay messages to one another. Robots start processing received messages, as soon as they get them. The maximum period from the time that robot R_1 receives message* X*, until its neighbor robot R_2 receives the processed (possibly stamped) version of message* X *from R_1 is c seconds.*

Algorithms to Check Biconnectivity

As mentioned in the introduction, checking for biconnectivity is the first step towards the overall goal of achieving and maintaining a biconnected multirobot structure. It is an important step, because the robots must be able to detect if they are not biconnected, so that they can take remedial actions to restore *biconnectivity* before they lose *connectivity*. The remedial actions could be as simple as all robots moving back to a base and dispersing from there.

Note that the biconnected property is a global property of the multirobot system: robots cannot determine whether or not it holds from purely local information. For example see Figure 1, where the graph is not biconnected, and the robots associated with the nodes on the right side of the graph need global information about the nodes on the left side to know that the whole structure is not biconnected.

In our approach, each robot R, maintains two lists:

- CR_R *(connected robots)*: the list of robots that are connected to R.
- DCR_R *(doubly-connected robots)*: the robots doubly-connected to R.

Each robot R first fills the CR_R list, then using that, the DCR_R list is computed. Finally with the help of the DCR_R list, it detects if the robot graph is biconnected.

In the rest of this section, we first provide an algorithm (CR-FILL) for filling CR_R, then another algorithm (DCR-FILL) is presented which fills the DCR_R lists with the help of the already computed CR_R lists. Afterwards an algorithm which checks the biconnectivity with the help of the computed DCR_R lists is provided. All these algorithms are distributed and each robot runs them independently.

CR-FILL

In this subsection, we provide an algorithm for filling the CR_R list. That is for robot R, it finds the robots that are connected to it.

The basic idea is for the robots to stamp and pass messages in the system. In this way, if there is a path of $r_0 \rightarrow r_1 \rightarrow r_2 \rightarrow R$, between r_0 and R, robot R will receive a message that is stamped by r_0, r_1 and r_2. Thus it will know that it is connected to those robots, and will add them to the CR_R list.

A helper algorithm must run continuously on all the robots to help the CR-FILL and DCR-FILL (to be presented in the next section) algorithms. With that algorithm, all robots continually stamp and pass biconnected type messages that they receive. Any robot r, which receives $(\text{“}CR\text{”}, (S))$, checks the content of S, and if $r.id \notin S$ it stamps the message and send it out. That is, it sends message $(\text{“}CR\text{”}, (S, r.id))$. If $r.id \in S$, it does not send any message because stamping and sending it would lead to a duplicate ID in $(S, R.id)$. For an overview of this algorithm see Algorithm 1.

Algorithm 1 Message passing algorithm which robots continually run

upon receiving a message of form $(\text{“}CR\text{”}, (S))$ **do**
  if $R.id \notin (S)$ **then** robot R broadcast message $(\text{“}CR\text{”}, (S, R.id))$.
end upon

The implementation details of the CR-FILL algorithm, which involves when to send the initiating message, and when to set the CR_R to empty is provided in (Ahmadi & Stone 2006a). It can be proven that the CR-FILL algorithm can be completed within $3Nc$ seconds (recall that N is the maximum number of robots).

DCR-FILL

In this subsection, DCR-FILL, an algorithm to fill the DCR lists is presented. It is assumed that the message passing algorithm (Algorithm 1) is running continually by all robots.

The basic idea for filling DCR_R for robot R is to find the robots that are in a common loop with R. When the robot graph is undirected (Assumption 2), there is a loop including both R and R' iff two internally vertex-disjoint paths (Definition 1) exist between R and R'. In this case, R and R' are doubly-connected (Definition 3). According to Algorithm 1, robots pass stamped messages around. When robot R receives a message that has been stamped by itself (i.e. R), it knows the robots that have stamped the message after the R stamps are in a common loop with R, and should be added to DCR_R.

Robot (r) starts by broadcasting message $(\text{“}DCR\text{”}, (R.id))$, which will be heard by all of its neighbor robots. Upon receiving message $(\text{“}DCR\text{”}, (S))$, if this is the first time to receive a “DCR” message, it resets DCR_r to empty (initializing). Then it checks the content of (S). If its own ID *is* in the stamp part of the message (S), it can represent (S) as $(S_1, r.id, S_2)$. If S_2 includes more than one vertex, it means that there is a loop and the robot adds all the IDs in S_2 to DCR_r. If S_2 includes only one vertex, it means that the robot has gotten back a message from a robot that it has just sent a message to, and should be ignored. Algorithm 2 presents the pseudocode of this algorithm.

Algorithm 2 Pseudocode for DCR-FILL algorithm

Time 0: robot R broadcasts $(\text{“}DCR\text{”}, (R.id))$.
upon receiving a message of form $(\text{“}DCR\text{”}, (S))$ **do**
  if this is the first time to receive a “DCR” message **then**
    reset DCR_R to empty
  end if
  if $R.id \in (S)$ **then**
    split (S) to $(S_1, R.id, S_2)$
    if $size(S_2) > 1$ **then** add the IDs in S_2 to DCR_R
  end if
end upon

We claim for robot R, the DCR-FILL algorithm sets the correct DCR_R list within nc seconds.

Theorem 2. *For any robot R, the* DCR-FILL *algorithm finds the full list of doubly-connected robots (DCR_R) within nc seconds.*

Proof. See (Ahmadi & Stone 2006a) □

Biconnectivity Check

After running CR-FILL and DCR-FILL consecutively, the CR and DCR lists will be accurate. Notice that both algorithms for filling the CR and DCR lists finish within a known time limit. Thus the robots should wait $3Nc$ seconds, and afterwards CR and DCR lists will be accurate. For robot r if CR_r and DCR_r are equal, it means that r is doubly-connected to all the robots that it is connected to. By Theorem 1, we know if there are two robots r_1 and r_2 that are doubly-connected to all other robots, then the robot graph is biconnected. Also, we know by Lemma 1 that if there is a robot that is not doubly-connected to all other robots, the robot graph is not biconnected. Thus if the robot and one of its neighbors is doubly-connected to all other robots, the robot knows that the robot graph is biconnected. Also if the robot or one of its neighbors is not doubly-connected to all other robots, it will know that the robot graph is not biconnected.

The overview of the biconnectivity check algorithm is shown in Algorithm 3. The *initiator* robot (which can be any robot who wants to check biconnectivity) starts by sending a ("CHECK-REQUEST",()) message to its neighbors to ask them to check if they are doubly-connected to other robots or not. Upon receiving a ("CHECK-REQUEST",()) the other robots run biconnectivity check (unless they are already running it) as non-initiators (skipping line 3 of Algorithm 3). Note that multiple robots can run the biconnectivity check algorithm in parallel.

If the robot is doubly-connected to all other robots, it sends the message ("DC-TRUE", ()) to all its neighbor robots, and a ("DC-FALSE",()) message otherwise. If the robot is doubly-connected to all other robots and receives a ("DC-TRUE", ()) message, it knows that the robot graph is biconnected. Otherwise (if it is not biconnected to all other robots, or receives a ("DC-FALSE", ()) message) it knows that the robot graph is not biconnected. Since the initiator and its neighbors should run the biconnectivity check, the total time needed for the biconnectivity check to complete is $6Nc + 2c$ seconds.

Note that c is ideally on the order of milliseconds, though in practice it may be difficult to guarantee such small bounded transmission times. In such cases, the algorithms as is may become impractical for large teams of fast-moving (so that connectivity changes quickly) robots. Space and time complexity of the biconnectivity check is of $O(n)$ for each received message. In the worst case, each robot should deal with $(n-1)!$ messages in each time period, but the worst case does not happen in real applications. For a realistic scenario with 35 robots, each robot on average has to deal with 69 messages in each time period. (Ahmadi & Stone 2006a)

Algorithm 3 Pseudocode for biconnectivity check algorithm. It returns *true* if the robot graph is biconnected, and *false* otherwise.

1: run CR-FILL and DCR-FILL in parallel, and wait $3Nc$ seconds for them to be finished.
2: **if** (initiator) **then** send message ("CHECK-REQUEST",())
3: **if** $size(DCR_R) = size(CR_R)$ **then**
4:     send message ("DC-TRUE",())
5: **else**
6:     send message ("DC-FALSE",())
7:     return false;
8: **end if**
9: **if** a message of form ("DC-FALSE", ()) is received **then** return false;
10: **if** a message of form ("DC-TRUE", ()) is received **then**
11:     **if** $size(DCR_R) = size(CR_R)$ **then** return true;
12: **end if**

Adding and Removing Robots from a Biconnected Structure

In the previous section a fully distributed algorithm for checking whether a team of robots is biconnected is presented. As presented in the introduction, that is the first of the three steps to realizing the overall goal of achieving and maintaining communication connectivity in a team of robots. This section presents how a team can *maintain* biconnectivity when team members are added or removed.

Here we assume that each robot works within a region (which can change over time) and does not go out of its assigned region. This is a common practice in many multirobot systems which use task allocation techniques for coordination (e.g.(Ahmadi & Stone 2006b)). Two robots are considered neighbors if they can communicate from anywhere in their assigned regions.

We further assume that robots send the position information of their assigned regions to other robots, but because of localization errors and communication delays, the position information of other robots' regions, especially those at a far distance, may not be accurate.

Adding Robots to a Biconnected Structure

In this section the problem of adding robots to a biconnected structure is considered. Assuming each robot is able to cover a limited sized region, the goal of the robots is to cooperatively cover as much area as possible while staying biconnected.

If S is a biconnected multirobot structure, and robot R wants to join the structure, it needs to choose a region that would make it the neighbor of two other robots. The next theorem shows by doing so, it will be doubly connected to all other robots, and the robot graph remains biconnected.

Theorem 3. *If graph $G(V, E)$ is biconnected, then graph $G'(V', E')$ is also biconnected if $V' = V + \{v_1\}$ and E' includes all the edges in E plus at least two edges that connect v_1 to two of the vertices of V.*

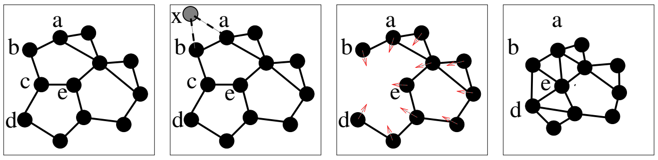
Proof. Assume that in G', v_1 is neighbor of v_i and v_j. There are two paths of v_iv_1 and $v_i - v_jv_1$ between v_i and v_1, so v_i and v_1 are doubly connected. Since G is biconnected, v_i is also doubly connected to all the other vertices in V'. Same argument holds for v_j, and there are two vertices (v_i and v_j) that are doubly connected to all other vertices in V'. Thus based on Theorem 1 the graph G' is biconnected. □

See Figure 2(b), where robot x is added to the biconnected structure of Figure 2(a). Since x is a neighbor of both a and b, based on Theorem 3 the new structure is also biconnected. Note that in the figure, each region is represented by a node.

When robot R wants to join a biconnected structure, from among the locations that will be a neighbor to two or more robots, robot R should choose the one that is *best* based on a task-specific evaluation function. For our evaluation criterion, where the multirobot system should cover as much surface as possible, the new robot should choose a region that has the minimum intersection with the other robots' assigned regions while being the neighbor of at least two robots. Computing which region has the minimum intersection with all the other regions even for the case where the regions are all exactly known is a time consuming process. Instead we use a heuristic to find a near-optimal position for the region of the newly added robot.

Specifically, the new robot picks the candidate position that maximizes the sum of the *distances* to the two closest robots' regions, where distance between two regions is defined as the distance between the regions' centers. The *center* of a region is defined as the point-wise average of the points in that region[1]

A prerequisite of the above mentioned method of choosing a region is for the new robot to know the position of the center of the other robots' regions. Recall that we have assumed that each robot has an approximation of other robots' region (which has gained through communication). The new robot starts by sending a position request to other robots, and they respond with the approximate position of the center of the regions of all robots. Using the approximation of the center points of all robots' regions, the new robot chooses a region that maximizes the sum of the distances to the two closest robots' (nr_1 and nr_2) regions. The new robot goes directly towards nr_1. After getting close to nr_1, it again requests position information to get the region positions of nr_1 and nr_2 more accurately. Note that nr_1 is expected to have more current and accurate information about the positions of its own and its neighbors' assigned region than a robot located far away using delayed and noisy information. Afterwards it chooses its exact region position such that it is a neighbor of both nr_1 and nr_2 and maximizes the heuristic distance-based criterion provided above.



(a) (b) (c) (d)

Figure 2: (a) An initial biconnected structure (b) Addition of robot x to the team (c) Robot c leaves the structure. The arrows show the direction that the other robots need to move in order to restore a biconnected structure. (d) newly constructed structure after removal of c from the initial structure.

Removing Robots from a Biconnected Structure

After removal of a robot, the newly generated multirobot structure may no longer be biconnected. In this section, an outline of a distributed algorithm for the robots to maintain the biconnected structure after removal of a robot is provided.

After removal of a robot r from a biconnected structure, if ex-neighbors of r are doubly connected, the newly generated structure is also biconnected. The following theorem provides a more formal description.

Theorem 4. *Let $G(V, E)$ be a biconnected graph, and $r \in V$ be a vertex of the graph. Graph $G'(V', E')$ is constructed from G by removing vertex r. Let $G_r(V_r, E_r)$ be a subgraph of G, where V_r are the neighbors of r, and E_r is the set of edges that are in E between the vertices of V_r. If G_r is biconnected, then graph G' is also biconnected.*

[1]Theoretically it is possible that the center of a region is not in the region, which does not cause any problems.

Proof. Assuming G_r is biconnected, we prove any two vertices $v_i, v_j \in V'$ that are doubly connected in G, are also doubly connected in G'. If none of the two vertex-disjoint paths between v_i and v_j include r, the proof is trivial. Since the paths are vertex-disjoint, only one of them can include r. Assume the two vertex-disjoint paths between v_i and v_j are $v_i \ldots f_1 \ldots r \ldots l_1 \ldots v_j$ and $v_i \ldots f_2 \ldots l_2 \ldots v_j$, where f_1 and f_2 are the first, and l_1 and l_2 are the last vertices respectively from V_r in the respective paths. Since G_r is biconnected, there is a loop which includes f_1, f_2, l_1 and l_2. In the loop, there is a path (call it P_1) between f_1 and one of l_1 or l_2 (without loss of generality assume it is l_1) that does not include f_2 or l_2. Both f_2 and l_2 are in the loop but not in path P_1, so there is a path P_2 between them (part of the loop) that is vertex-disjoint from P_1. By substituting P_1 and P_2 in the two paths between v_i and v_j, the two vertex-disjoint paths which do not include r are generated, and the theorem is proved.

$\square$

Using the above theorem we now show how to reconstruct the biconnected structure after removal of a robot. We claim if all robots move d units towards the ex-position of r, no previously existent neighbor relationship is destroyed. The reason is simple: if a group of points all move towards a base, the distance between each two point decreases. Thus if moving d units is enough for the neighbors of r to become biconnected, then all robots moving d distance units towards the ex-position of r makes the robot graph biconnected again. See this process from Figure 2(a) to Figure 2(c), and to the newly constructed biconnected structure in Figure 2(d). Note that a cycle graph is biconnected. In practice, usually the neighbors of r form a cycle by getting closer to the ex-position of r.

When robot r is removed, its neighbors are notified (either by an explicit message from the robot which is about to quit, or by not hearing from it for several seconds). The neighbors of r should decide how much they should get closer to r (d distance units) in order to get connected. For that purpose each robot stores the list of neighbors, and list of neighbors of neighbors which can be computed during the message passing algorithm. Thus when robot r is removed, all its neighbors know that they are a neighbor of a removed robot, and furthermore they know which robots are neighbors of r. The robot with the lowest ID between the neighbors of the removed robot r, assumes the *leadership* and computes the d value. For computing the d value, the leader robot starts from $d = 0$ and increases it in discrete steps until the subgraph of neighbors of r after moving d units towards ex-position of r becomes biconnected. For checking if the neighbor robots are biconnected, the leader robot checks if two of the neighbor robots are doubly connected to all other robots. Typically, there should only be a few robots which are neighbor of r, thus running this algorithm does not take a long time.

After the leader robot has computed the d value, it sends a movement message intended for all robots, the message is ("MOVE" (*pos* d) ()), where *pos* is the ex-position of r based on the leader robot information. Each robot upon receiving a ("MOVE" (*pos* d) ()) message for the first time, sends out the same message out, and moves d unit towards *pos*.

Since the leader robot does not have complete information of other robots' positions, the graph that it uses for checking biconnectivity is not accurate, and the d value may not be correct. As a result, after the robots get closer to r for d units, the leader robot checks if they are biconnected with the biconnectivity check algorithm provided in the previous section. If not biconnected, the leader robot sets d' to a constant heuristic positive value, and send a message to other robots to move an additional d' units towards the ex-position of r. This process continues until the leader robot knows that they are biconnected based on the checking biconnected algorithm.

Experimental Results

In this section we investigate the effects of the provided algorithms for maintaing biconnectivity on the total area that robots are able to sense. The algorithms for maintaining biconnectivity which also includes the checking biconnectivity algorithm is implemented in the Player/Stage (Gerkey, Vaughan, & Howard 2003) simulator. The environment is assumed to be an open infinite environment (relative to sensing and communication ranges of the robots).

In all experiments, the robots assume responsibility for a circular region. The size of the circular regions are constant for all robots. Combining the biconnected ideas with more elaborate task assignment methods remains for future work. The cr ratio is defined as the ratio between the communication range and the size of the robots' regions. Two different scenarios are considered. In the first, cr ratio is set to 4, and in the second, cr ratio is 2. Notice that when the communication range is 2 times bigger than the robot regions, only adjacent robot regions are considered neighbors.

In both experiments, the maximum number of robots in the environment is 10. If the number of robots in the structure is 10, in an episode a robot is randomly chosen to be removed. Also, if there is no robot left in the structure, a robot is added. When there are 1 to 9 robots one robot is randomly added or removed. There are five robots in the first episode.

Each experiment consists of 100 episodes. S is defined as the sum of the area that all the robots currently in the structure can cover. For the case where cr ratio is 2, on average 73% of S is covered, and for the case where cr ratio is 4, on average 91% of S is covered. That is, the robots remain biconnected while covering 73% and 91% of the total area that they could cover not worrying about being connected.

To show the strength of the provided heuristic for choosing the region of the newly added robot, for the case of $cr = 2$, we also experimented with using the region with minimum intersection with other robots regions. Note that finding the region with minimum intersection with other regions is only possible in simulation where the speed of the simulator can be decreased, and the algorithms are running centrally on a PC. If robots use the region with minimum intersection with other robots' regions instead of the provided heuristic for choosing the region of the newly added robot, for cr equal to 2, on average 75% of S is covered. This result shows that using the heuristic does not significantly affect performance.

Conclusion and Future Work

In this paper, we defined and argued the need for biconnected multirobot structures. A distributed algorithm for checking biconnectivity is presented, proven correct, and analyzed theoretically. Algorithms to maintain a biconnected structure in the event of addition or removal of a robot are presented and tested in the Player/Stage simulator.

Opportunities for future work include relaxing our assumption that robots have identical communication capabilities, thus making the robot graph a directed graph; and creating algorithms to construct a biconnected structure from scratch. Overall, the work presented in this paper makes important steps towards enabling teams of distributed robots to reason about their ability to remain in communication contact while executing a joint task.

Acknowledgments

We would like to thank Kurt Dresner and Nick Jong for their valuable comments on an earlier version of this paper. This research was supported in part by NSF CAREER award IIS-0237699 and ONR YIP award N00014-04-1-0545.

References

Ahmadi, M., and Stone, P. 2006a. Keeping in touch: A distributed check for biconnected structure by homogeneous robots. In *The 8th International Symposium on Distributed Autonomous Robotic Systems*.

Ahmadi, M., and Stone, P. 2006b. A multi-robot system for continuous area sweeping tasks. In *Proceedings of International Conference on Robotics and Automation (ICRA), to appear.*

Anderson, S.; Simmons, R.; and Goldberg, D. 2003. Maintaining line of sight communications networks between planetary rovers. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*.

Diestel, R. 1997. *Graph Theory*. New York: Springer.

Gerkey, B.; Vaughan, R.; and Howard, A. 2003. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the International Conference on Advanced Robotics*, 317–323.

Howard, A.; Matadd, M.; and Sukhatme, G. 2002. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots, Special Issue on Intelligent Embedded Systems* 13(2):113–126.

Nguyen, H.; Pezeshkian, N.; Raymond, M.; Gupta, A.; and Spector, J. 2003. Autonomous communication relays for tactical robots. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*.

Parker, L. E. 2002. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots* 12(3):231–255.

Swaminathan, B., and Goldman, K. J. 1994. An incremental distributed algorithm for computing biconnected components (extended abstract). In *Proceedings of the 8th International Workshop on Distributed Algorithms*.

Ulam, P., and Arkin, R. 2004. When good comms go bad: Communications recovery for multi-robot teams. In *Proceedings of 2004 IEEE International Conference on Robotics and Automation*.

Vazquez, J., and Malcolm, C. 2004. Distributed multirobot exploration maintaining a mobile network. In *Proceedings of Second IEEE International Conference on Intelligent Systems*.

Westbrook, J., and Tarjan, R. E. 1992. Maintaining bridge-connected and biconnected components on-line. *Algorithmica (Historical Archive)* 7(1):433–464.