Biconnected Structure for Robotic Sensor Networks

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}

Extended Abstract

The use of multi-robot systems for surveillance and exploration tasks has been studied extensively in the robotics community. However, the relationship between multi-robot surveillance systems and sensor networks has not been fully explored. In the surveillance task, robots have to collaboratively sense a given area to detect events of interest. Although some problems such as path planning are unique to the multi-robot setting, the communication issues and data fusion problems are common to both multi-robot and sensor network systems.

Like sensor networks, many applications of distributed autonomous robotic systems can benefit from, or even may require, the team of robots staying within communication connectivity. 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 multi-robot 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 do not 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.

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 (e.g (Howard, Matadd, & Sukhatme 2002)).

Assuming robots do not fail, some of the previous methods focus on creating connected structures (e.g (Howard, Matadd, & Sukhatme 2002)) However, the methods that do consider the possibility of robot failure do not ensure connectivity (e.g. (Ulam & Arkin 2004)).

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 (e.g. (Tarjan & Vishkin 1984)), in dynamic graphs, or in a restricted subclass of all graphs In all these cases, the algorithms are either centralized, or if distributed, each node has full knowledge of the whole graph. Also, the related work in the static sensor networks use centralized approaches to set the position of the sensors. 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.

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. We have also developed algorithms for the case where robots have heterogeneous communication capabilities. Although the algorithms are more

complicated, but the basic ideas are the same as presented in this extended abstract.

We tackle this problem 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 multi-robot structure from scratch. To be applicable for teams of autonomous robots, all algorithms must be fully distributed.

We have focused on fully achieving and analyzing steps 1 and 2. Step 3 remains as future work.

The first step, checking for biconnectivity, 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.

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 that are in a common loop with 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.

The basic idea for filling CR_R lists is for the robots to sign 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 signed 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. The details of the algorithm (who/when should send initiating messages, and when the robot initializes the CR_R list) is beyond this extended abstract. The algorithm is fully analyzed and completes in $2Nc$ seconds, where N is the maximum number of robots in the system, and c is the maximum period from the time that robot R_1 receives message *X*, until its neighbor robot R_2 receives the processed (possibly signed) version of message *X* from R_1.

The basic idea for filling DCR_R for robot R, is to find the robots that are in a common loop with R. We have proved that if there are two robots that are in a common loop with any other robot, the robot graph is biconnected.

When robot R receives a message that has been signed by itself (i.e. R), it knows the robots that have signed the message after the R signature are in a common loop with R, and should be added to DCR_R. The designed algorithm finds the complete DCR_R list in nc seconds, where n is the number of robots in the system.

Two neighbor robots check if they are in a common loop with any other robot ($DCR_R = CR_R$), and after communicating with eachother they will know if the robot graph is biconnected.

Assuming that the multirobot structure has the biconnected property, here we consider the problem of adding and removing robots to/from the structure. For now, we assume that each robot has a base position (which can change over time) and does not go farther than r units from its base. Two robots are considered neighbors, if they can communicate from anywhere within r radius of their bases.

We further assume that robots send their base position information to other robots, but because of localization errors and communication delays, the information of other robot base positions may not be accurate.

If S is a biconnected multirobot structure, and robot R wants to join the structure, it needs to choose a base location that would make it the neighbor of two other robots. In that way, the newly constructed structure is also biconnected. From among the locations that will be a neighbor to two robots, robot R should choose the one that is *best*, based on a task-specific evaluation function.

After removal of a robot, the newly generated multirobot structure may no longer be biconnected.

Let $G_r(V_r, E_r)$ be the subgraph of the robot graph, where V_r is the neighbors of r. After removal of r, if G_r is biconnected, with basic graph analysis, we can deduce that the robot graph is still biconnected. If all robots move d units towards the ex-position of r, no previously existent neighbor relationship is destroyed. Thus if moving d units is enough for the neighbors of r to get connected, then all robots moving d distance units towards the ex-position of r will make the robot graph biconnected again.

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 (or one of them), decide how much they should get closer to r (d distance units) in order to get connected. Since the robots do not have complete information of other robots' positions, they can not compute d without any errors. Because of that, after the robots get closer to r for d units, they check if they are biconnected by the biconnectivity check algorithm. This process continues until the robots know they are biconnected based on the checking biconnectivity algorithm.

Acknowledgment

This research was supported in part by NSF CAREER award IIS-0237699 and ONR YIP award N00014-04-1-0545.

References

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.

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*.

Tarjan, R., and Vishkin, U. 1984. Finding biconnected componemts and computing tree functions in logarithmic parallel time. In *25th Annual Symposium on Foundations of Computer Science, 1984.*, 12–20.

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*.