Distributed tree rearrangements for reachability and robust connectivity

Michael Schuresko[1] and Jorge Cortés[2]

[1] Department of Applied Mathematics and Statistics
University of California, Santa Cruz
mds@soe.ucsc.edu

[2] Department of Mechanical and Aerospace Engineering
University of California, San Diego
cortes@ucsd.edu

Abstract. We study maintenance of network connectivity in robotic swarms with discrete-time communications and continuous-time motion capabilities. Assuming a network topology induced by spatial proximity, we propose a coordination scheme which guarantees connectivity of the network by maintaining a spanning tree at all times. Our algorithm is capable of repairing the spanning tree in the event of link failure, and of transitioning from any initial tree to any other tree which is a subgraph of the communications graph.

1 Introduction

Given a group of robots with processing, motion, and communication capabilities executing a motion coordination algorithm, we address the following problem: how can we guarantee that the interaction graph induced by the inter-agent communication remains connected?

One strategy is to make custom modifications to each motion control algorithm to guarantee connectivity. It is desirable, however, to synthesize a general methodology that goes beyond a case by case study, and can be used in conjunction with any motion coordination algorithm. In this paper we take on this aim and propose an approach based on the preservation of a spanning tree of the underlying communication graph. The idea is to synthesize a distributed algorithm to agree upon "safe" re-arrangements of the spanning tree (i.e., re-arrangements that do not break connectivity) based on preferences specified by the motion coordination algorithm. Space constraints prevent us from presenting more than a rough sketch of our algorithm design and analysis results. A complete version of our discussion here can be found in [13].

Literature review. The fundamental importance of spanning trees to distributed algorithms motivate a vast collection of literature, see e.g., [7, 9], which explores their properties and designs algorithms to construct them. Spanning trees are especially crucial for the specific case of distributed computation over ad-hoc networks. For a survey of spanning tree repair algorithms for ad-hoc networks, see [4]. In cooperative control and robotics, several works have studied how to constrain the motion of the agents to preserve connectivity. For brevity, we only mention a few here. [1, 6, 5, 10] maintain the connectivity of the network by constraining robot motion to maintain a fixed set of links. The centralized solution proposed in [15] allows for a general range of agent motions. Many works [16, 3, 14, 12] control the algebraic connectivity of a robotic network.

2 Connectivity Maintenance Algorithm

The Connectivity Maintenance (CM) Algorithm is an algorithm to maintain a spanning tree of the communication graph. The intent is that if robot motion is constrained to not break any links of the spanning tree, the underlying graph will remain connected as well. An informal algorithm description follows.

> *[Informal description:]* Each robot maintains a reference to its parent in the spanning tree. At pre-arranged times, each robot is allowed to change its parent. Connectivity is preserved in the following way. Each robot keeps an estimate of its depth, i.e., distance from the root in the spanning tree. If no robot picks a robot of greater depth than its parent's, then no robot will pick one of its current descendants as a parent node. To allow robots to attach to potential parents of the same depth estimate, a tie-breaking algorithm based on UIDs is used to prevent potential formation of cycles, thus maintaining the spanning tree property.

The CM Algorithm should be coupled with two other algorithms:

- The first is a modification of the underlying motion coordination algorithm, modified to preserve the links of the spanning tree maintained by CM Algorithm. We refer to an algorithm which satisfies the constraints required for this role as one which is *motion compatible with* CM Algorithm. This algorithm also specifies which neighbors each agent would prefer to be connected to as an order relation. The relation we use in the simulations presented in Section 4 is roughly "each agent prefers to attach to agents which are closer to its position in physical space."
- The second algorithm is one which tells the robots to artificially increase their depth estimates at particular times. Doing so allows a robot of a lower actual depth to attach to a robot of a higher actual depth, at the expense of making the "depth estimates" diverge from the actual depth (hopefully only for short periods of time). Care must be taken to ensure that such an algorithm does not cause robot depth estimates to grow in an unbounded fashion. We specify a series of constraints on such an algorithm so that it still guarantees correctness of CM Algorithm. We refer to an algorithm which satisfies these constraints as one which is *depth compatible with* CM Algorithm. The simplest such algorithm, called Null Depth Increment Algorithm, never tells an agent to artificially increase its depth estimate.

We show in [13] that CM Algorithm can recover from a wide variety of states (positions and topologies) resulting from link failures of the form "a link between two agents disappears who are instantly made aware of the link failure."

3 Reachability analysis: Cycle-detecting Depth Increment Algorithm

We introduce an algorithm which is *depth compatible with* CM Algorithm called Cycle-detecting Depth Increment Algorithm. When this algorithm is combined with CM Algorithm, the resulting strategy satisfies a very nice property: the combined algorithm can induce the constraint tree to match any tree, T_2, which is a subgraph of the communication graph. Specifically, if a tree, T_2, is a subgraph of the current graph, and every edge of T_2 is preferred by its parent, i, in T_2 to each of i's neighbors, then the tree stored by CM Algorithm will eventually become T_2.

An informal description of Cycle-detecting Depth Increment Algorithm is as follows.

> *[Informal description:]* Each robot stores a "start number", a "number of descendants" and a "mapping from child UID to child start number." At each round, in addition to the tree constraint info, each node sends the following info to each neighbor. If the neighbor is a child, it sends the appropriate entry in its mapping, or, if the child is not in the mapping, it sends its own start number. It always sends its "number of descendants." If the neighbor is not a child, it sends its own start number. With the messages received, each node updates its numbers in the following way. Its "number of descendants" is the sum of the "number of descendants" info received from each child, plus one (for itself). Its "start number" is the number its parent sends it. For each child it receives a message for, it adds an entry to its map that is indexed by that child's UID and has a value of "the sum, over all children with lesser UID, of the number of descendants of those children, plus one plus its own start number."

4 Simulations

Here we illustrate the performance of the Connectivity Maintenance Algorithm in several simulations. We combine the algorithm with Cycle-detecting Depth Increment Algorithm and the deployment algorithm presented in [2]. The proximity graph of the robotic network is the r-disk proximity graph. The deployment algorithm assumes that each robot has a sensor coverage disk. It moves the robots to maximize sensor coverage of a "region of interest" represented by a density function $\rho : \mathbb{R}^2 \mapsto \mathbb{R}$ given sensors which cover disks of radius r_d. We assume the robots have a maximum velocity of v_{speed}.

Links of the constraint tree are preserved via a modification of the procedure described in [1]. To preserve a link between two robots, we constrain the motion of the two robots to a circle of radius $\frac{r}{2}$ centered at the midpoint of the line between their positions. Because each robot has a "target" it moves towards, we can find the closest point to the target in the intersection of the circles generated by the constraint edges.

The algorithm resulting from the combination of the deployment algorithm with the Connectivity Maintenance Algorithm is executed in our Java simulation platform [11]. This platform provides a software implementation of the modeling framework introduced in [8]. Our results are shown in Figure 1.

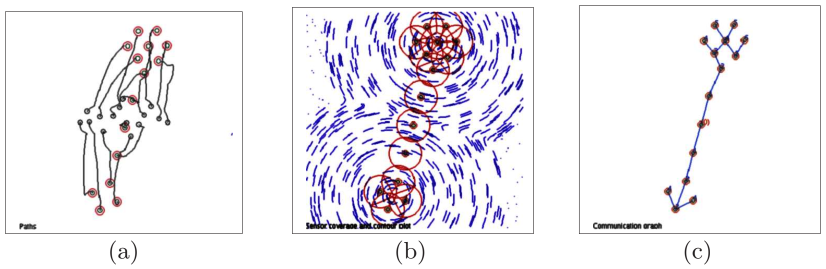

Fig. 1. The plots show an execution of Connectivity Maintenance Algorithm, showing (a) the paths taken by the robots, (b) a contour plot of the density field and the sensor coverage regions of the robots, (c) the final network constraint tree.

Figure 2 shows the evolution of the algorithm when repairing an initially disconnected tree.

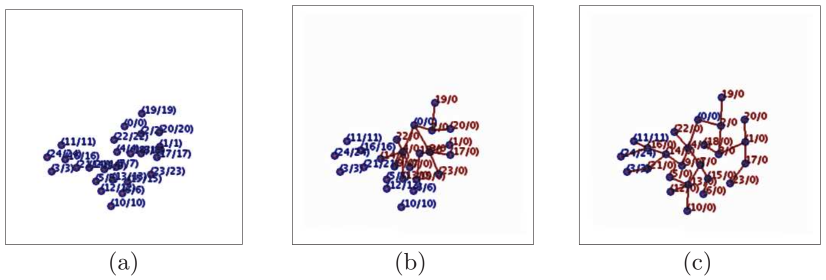

Fig. 2. Progress of repair starting with an initially disconnected constraint tree. Agents are labeled by "agent id/root id" : those in blue have not yet completed repair.

5 Conclusions and future work

We have designed a tree-rearrangement algorithm for connectivity with reachability and repair capabilities. The algorithm can be shown to be provably correct, and is easily composable with other motion coordination algorithms. Future work will include understanding how the resulting trees of our algorithm compare to minimum spanning trees, developing systematic ways to encode preference rearrangements in connection with other coordination algorithms, and exploring the properties of the algorithm in conjunction with other proximity graphs, such as the visibility graph.

Acknowledgments

This research was supported in part by NSF CAREER Award ECS-0546871.

References

1. H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, 1999.
2. J. Cortés, S. Martínez, and F. Bullo. Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM. Control, Optimisation & Calculus of Variations*, 11(4):691–719, 2005.
3. M. C. de Gennaro and A. Jadbabaie. Decentralized control of connectivity for multi-agent systems. In *IEEE Conf. on Decision and Control*, pages 3628–3633, San Diego, CA, December 2006.
4. F. C. Gaertner. A survey of self-stabilizing spanning-tree construction algorithms. Technical report, Ecole Polytechnique Fdrale de Lausanne, 2003. Available electronically at `http://infoscience.epfl.ch/search.py?recid=52545`.
5. A. Ganguli, J. Cortés, and F. Bullo. Multirobot rendezvous with visibility sensors in nonconvex environments. *IEEE Transactions on Robotics*, 2008. Accepted. To appear.
6. J. Lin, A. S. Morse, and B. D. O. Anderson. The multi-agent rendezvous problem. Part 1: The synchronous case. *SIAM Journal on Control and Optimization*, 46(6):2096–2119, 2007.
7. N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1997.
8. S. Martínez, F. Bullo, J. Cortés, and E. Frazzoli. On synchronous robotic networks – Part I: Models, tasks and complexity. *IEEE Transactions on Automatic Control*, 52(12):2199–2213, 2007.
9. D. Peleg. *Distributed Computing. A Locality-Sensitive Approach*. Monographs on Discrete Mathematics and Applications. SIAM, 2000.
10. K. Savla, G. Notarstefano, and F. Bullo. Maintaining limited-range connectivity among second-order agents. *SIAM Journal on Control and Optimization*, 2007. Special issue on "Control and Optimization in Cooperative Networks." (Submitted Nov 2006) To appear.
11. M. D. Schuresko. CCLsim. a simulation environment for robotic networks, 2008. Electronically available at http://www.soe.ucsc.edu/˜mds/cclsim.
12. M. D. Schuresko and J. Cortés. Distributed motion constraints for algebraic connectivity of robotic networks. In *Journal of Intelligent and Robotic Systems*, 2008. Special issue on "Special Issue on Unmanned Autonomous Vehicles." Submitted.
13. M. D. Schuresko and J. Cortés. Distributed tree rearrangements for reachability and robust connectivity. *SIAM Journal on Control and Optimization*, 2009. Submitted.
14. P. Yang, R. A. Freeman, G. Gordon, K. M. Lynch, S. Srinivasa, and R. Sukthankar. Decentralized estimation and control of graph connectivity for mobile sensor networks. In *American Control Conference*, Seattle, WA, 2008.
15. M. M. Zavlanos and G. J. Pappas. Controlling connectivity of dynamic graphs. In *IEEE Conf. on Decision and Control and European Control Conference*, pages 6388–6393, Seville, Spain, December 2005.
16. M. M. Zavlanos and G. J. Pappas. Potential fields for maintaining connectivity of mobile networks. *IEEE Transactions on Robotics*, 23(4):812–816, 2007.