Autonomous Biconnected Networks of Mobile Robots

Jesse Butterfield
Department of Computer Science
Brown University
Providence, RI 02912-1910
jbutterf@cs.brown.edu

Karthik Dantu
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
dantu@usc.edu

Brian Gerkey
Artificial Intelligence Center
SRI International
Menlo Park, CA 94025-3493
gerkey@ai.sri.com

Odest Chadwicke Jenkins
Department of Computer Science
Brown University
Providence, RI 02912-1910
cjenkins@cs.brown.edu

Gaurav S. Sukhatme
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
gaurav@usc.edu

Abstract— **Groups of robots can be used in a coordinated fashion to achieve goals that individual robots cannot. One of the key requirements for this is being able to communicate amongst themselves in a timely and robust manner. This capability has been assumed as available in many multi-robot solutions but has not received enough attention in research. We approach the subproblem of moving from a connected network of robots to achieving biconnectivity. Biconnectivity provides both robustness to change in links and better bandwidth for communication by providing multiple paths to the destination. We take two approaches to the same problem - one a deterministic graph-based approach and the other a Markov random field approach. Both have their advantages. Preliminary results indicate much promise in both these directions.**

I. Introduction

When trying to create a multi-hop wireless network between two fixed terminals using mobile robots, an interesting sub-problem is taking a singly connected network and repositioning the robots to make the network biconnected. Since a biconnected graph remains connected even if you remove one of the vertices, these networks would be robust to any single robot failure. We are taking two possible approaches to solving this problem, which we plan to implement on identical robots in identical settings to get a head to head comparison. First the problem can be studied from an graph theory approach, where global information about the graph is compiled and a strategy is devised. In Section 2 we will explain the algorithm we use. The second method takes a probabilistic approach, using a robot's local sensing and messages from only its neighbors in the network to come to a local decision about the optimal action it should take. We will explain in Section 3 how we use Markov random fields and belief propagation to accomplish this. The first technique has advantages in that its behavior is deterministic. Hence, for the given problem, if the global network information is sensed correctly the algorithm will be guaranteed to make the graph biconnected or tell you it's not possible. In the second approach there are no guarantees, but its probabilistic nature makes it possible for it to succeed even with some wrong or incomplete information. Also using this general Markov random field structure makes adapting this technique to other problems easier.

II. Related Work

The problem of achieving biconnectivity has received some research attention previously. [1] proposes an algorithm to check for biconnectivity by listing all the doubly connected robots in a distributed fashion. It also proposes an algorithm to fix a biconnected network of robots when robots are added or deleted on an existing biconnected network. This work, however, does not propose any algorithms to create a biconnected network when one does not exist.

[2] does a more theoretical analysis on the problem of minimizing motion to achieve biconnectivity. They have a linear programming formulation of the problem in the 1-dimensional case. They also propose a block-tree movement heuristic to achieve biconnectivity in 2-dimensional topologies.

The key difference in our work in the algorithmic section and the above two pieces of work is the lack of assumption of accurate localization of the robots. Our work is based on off-the-shelf robots with no special equipment for accurate ranging/localization. We use the radio to determine gradients of signal strength and thereby estimate the relative bearing of the robots. Correspondingly, our algorithms work with angles as opposed to position. Also, [1] does not have algorithms to create biconnected networks from a connected network while [2] assumes global knowledge of location of the robot.

III. Algorithmic approach

A. Introduction

A network of robots can be visualized as a unit-disk graph [3] wherein nodes are mobile. We assume that each node is a off-the-shelf robot (like an iRobot create) and has ability to communicate wirelessly using a 802.11 type radio. It is also assumed that the radio has a mechanism to measure the received signal strength as is the case with

WIOPT 2008, 1st – 3rd Apr 2008, Berlin, Germany.
 ISBN 978-963-9799-18-9
DOI 10.4108/ICST.WIOPT2008.3245

most off-the-shelf 802.11 cards. The key observation is that a connected graph can be split into biconnected components using Tarjan's algorithm [4]. Note that there are distributed algorithms to check biconnectivity citemazda-aaai06. Shown below in Fig. 1 is a graph and Fig. 2 shows its biconnected components.

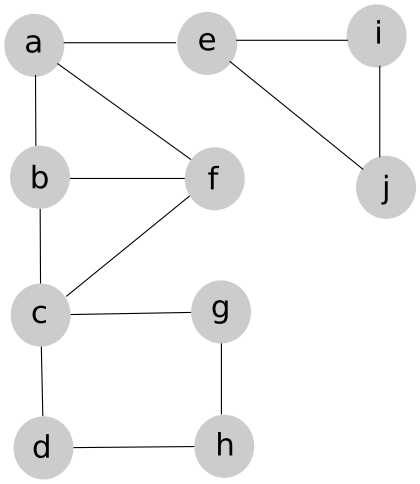

Fig. 1. A sample graph

These biconnected components are connected either via a vertex or an edge. These nodes are called *articulation points*. In Fig. 2, nodes *a-e* constitute two ends of an edge that is an articulation point and node *c* is a vertex articulation point. The next observation is that the only step to be undertaken in connecting the two biconnected components across articulation points is to add one edge between the two biconnected components. If we were to assume equal radii of communication for all the robots, then nodes B and C have to subtend an angle of $60°$(Fig. 3). The articulation point now commands the two nodes to approach each other.

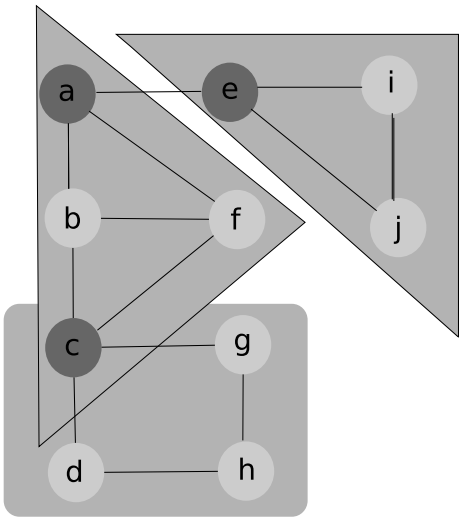

Fig. 2. Biconnected components of the sample graph with articulation points

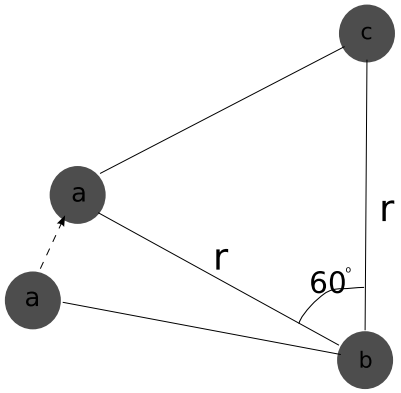

Fig. 3. Movement heuristic to connect two biconnected components

B. Algorithm

Our algorithm works in two stages. Every node makes a list of its neighbors. These neighbor lists are then sent to a cluster head that is either chosen in advance or elected on the fly. The cluster-head then computes the biconnected component(s) using a slightly modified Tarjan's algorithm. It then communicates the biconnected component information about each edge to its vertices. It also informs the articulation points whether they are edge or vertex articulation points. This concludes stage I involving the cluster head. The action now shifts to the articulation points. The articulation points list their neighbors in sorted order of the angle subtended in its local coordinate system. It picks the two edges (and corresponding neighbors) who are next to each other in the list, belong to different biconnected componets and closest in terms of angle subtended at the articulation point (if there are more than one). The articulation point then asks the two nodes to move "towards" each other. This is elaborated in subsection III-D.

Algorithm 1 Phase I: Compute biconnected components

```
N ⇐ neighbors of this node
if I am server then
    while i < Num.Nodes do
        List[i] ← receive neighor list from i
    end while
    Compute biconnected components and articulation points
    Classify articulation points as edge and vertex
    Send this info back to the corresponding nodes
else
    Send neighbor list to server
    Receive biconnecitivity component information
end if
```

C. Angle estimation

Since we do not have any sensors from which we can get the relative bearing of one robot from another, we need a way to estimate the same. We use the radio signal strength for this purpose. Our test setup had two iRobot create robots. Each

Algorithm 2 Phase II: At articulation points

$A \Leftarrow$ Angles of all neighbors
Sort(A)
if $edge - articulation - point$ **then**
  Assuming $e \Leftarrow$ edge articulation point in neighbor set N, pick i such that
  i is either before or after e in A
  i subtends the smaller angle
else if $vertex - articulation - point$ **then**
  From A, pick (i, j) such that
  (i, j) are adjacent to each other
  edges to i and j belong to different biconnected components
  Angle subtended by edges to i and j is the least among nodes satisfying the first two conditions
end if

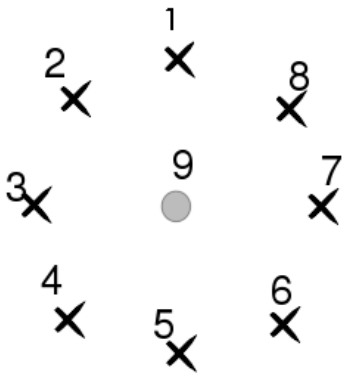

Fig. 4. Pattern to sample signal strength around the robot

robot has an x86-based ebox with a EMP mini-pci wireless card based on the atheros chipset. Both robots are fitted with the APXtender omnidirectional antenna. For the experiment, we keep one robot stationary and make the other robot sample the signal strength according to the pattern(Fig. 4). At each point in the pattern, the robot samples the signal strength every second for ten seconds (ten samples in total) to ensure that the signal strength has settled. We then average the signal strengths for that point. We varied the distance between the two robots from ft to 35ft at intervals of 5ft and repeat the experiment four times at each position for redundancy. Based on these measurements, the line of three collinear points with maximum gradient is assumed to be the bearing of the neighboring robot.

Fig. 5 shows the angle estimation error as a function of distance as measured between two robots. Overall, the error is between 20-30 degrees with a standard deviation of 20 degrees.

D. Achieving biconnectivity

Given the angle estimates, the next step is to move the robots such that biconnectivity is achieved. From our algorithm, we want to move robots that are neighbors of an articulation point but belong to different biconnected components closer. However, the key issue here is the fact that we have only bearing information and no range information. Given such a scenario, it is hard to predict the exact location (relative or absolute) of the robots with respect to each other. Hence, we have come up with a coarse rule for the movement of these robots. At the articulation point, we figure out which quadrant each of these robots are. Depending on this, the robots are asked to move toward each other along one of the axes if they are in different quadrants(Fig. 6) in the coordinate frame of the articulation point. If both the nodes are in the same quadrant, they are asked to move diagonally towards each other (Fig. 7).

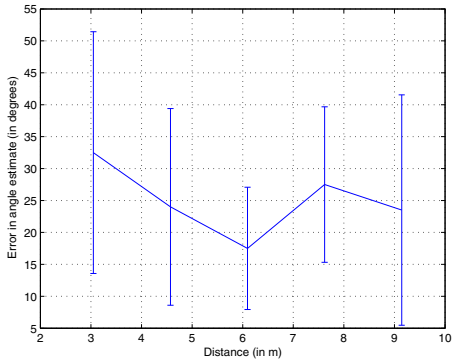

Fig. 5. Angle estimation error as a function of distance

E. Simulations

We are in the process of testing out our algorithm in Player/Stage [5]. Preliminary results indicate that our algorithm works reasonably well. Seen below is a screenshot of the algorithm in action with four nodes on Stage(Fig. 8). This is the case where the nodes from the different biconnected components are in different quadrants wrt to the articulation point. The direction of motion as instructed by the articulation point is as shown in Fig. 9. The nodes move towards each other and get connected forming one biconnected graph(Fig. 9).

F. Effect of angle error

Fig. 5 shows the error in angle calculated using signal strength as illustrated in sectionIII-C. We study the effect of angle error in our algorithm to achieve biconnectivity and give ways to mitigate this effect. Fig. 10 shows the potential worst case in angle estimation.

As shown in Fig. 5, the average error in angle estimate is about 20 degrees. Fig. 10(a) shows the ground truth configuration of two nodes A and B that are trying to estimate the bearing of each other. For simplicity, they are facing along the same axis but in opposite directions as indicated by the arrows. Their relative bearing is 30 degrees as shown. Node A's angle estimate might be twenty degrees off from this on average. This might be either toward or

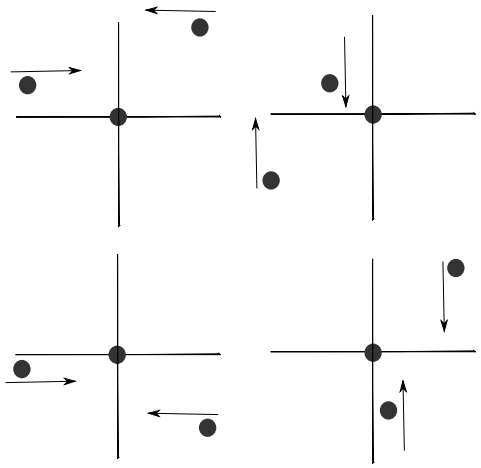

Fig. 6. Motion commands at articulation point (Neighbors in diff quadrants)

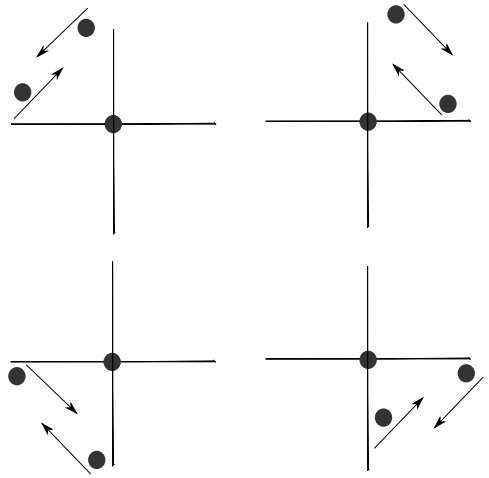

Fig. 7. Motion commands at articulation point (Neighbors in same quadrant)

away from the edge. Hence, half the time the angle estimate is greater than the actual angle (as shown in Fig. 10(b)). Similarly, node B's estimate is greater than the actual angle half the time. The angle error gets exacerbated about one fourth the time as shown in Fig. 10(c).

Consider another node C that would have a similar error of around 40 degrees one fourth of the time. The worst case scenario is shown in Fig. 11. One-eighth of the time, the nodes might move roughly at ninty degrees from what was commanded and this leads to undesired results (as in figure).

Our current solution for this problem is to perform recalculation of the relative bearing after some distance. That way, every new measurement of the bearing results in halving the probability of this worst case scenario for each of the edges. The total error thereby is rapidly decreased in cases where nodes are further apart. We are working on methods to better this angle estimate as well.

G. Future work

This is preliminary work in our study of networking of robots. Our algorithm is a heuristic - the next step is to analyze its performance in detail both in simulation and analytically which will hopefully lead us to better solutions. Also the larger problem of connecting disconnected groups of robots is not addressed here. Lastly, connectivity and coverage are always at opposite ends of this optimization problem. The final goal is to jointly optimize connectivity and coverage at the same time.

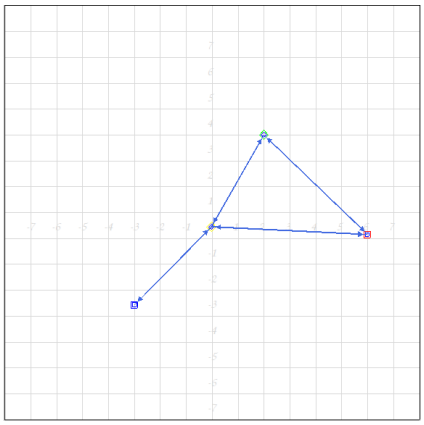

Fig. 8. Initial configuration with four nodes

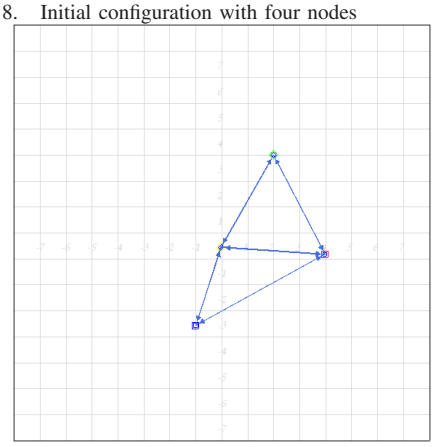

Fig. 9. Final configuration (graph biconnected)

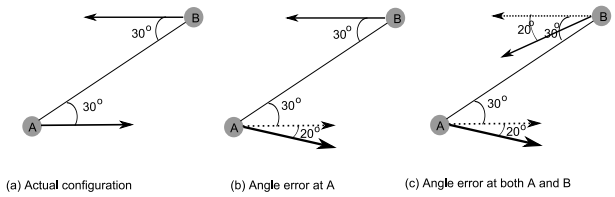

Fig. 10. Angle error propagation

IV. Probabilistic Approach

A. Multi-robot Markov Random Fields

An MRF [6] is a graphical model that factors a system into a finite set of observed and hidden, or latent, variables with pairwise interactions between them. In a multi-robot MRF, the robots are represented as nodes in the graph, with edges between pairs of robots that are in direct communication

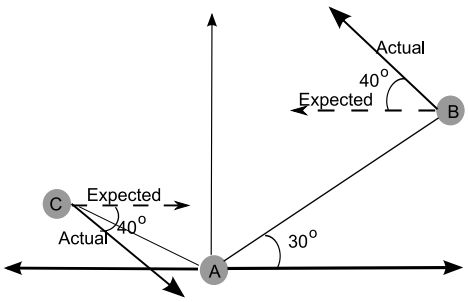

Fig. 11. Direction of motion - expected (dotted) and actual (filled)

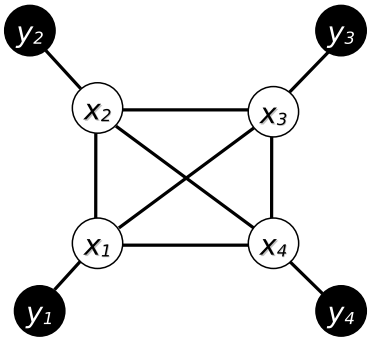

Fig. 12. An MRF with edges between the actions (x_i) of robots in communication and with edges between each robot's actions and its locally sensed information (y_i).

range. Each robot i maintains two random variables: y_i, an observed variable representing a robot's own perception; and x_i, a hidden variable representing the action that it should take (Figure 12). In this case the action space is made of all the possible directions in which the robot can move. Although these variables can either be discrete or continuous, we will break x_i into a set of the $n-1$ discrete directions and one stationary action for ease of implementation. We will then use this MRF to find the probability that a particular action is optimal (Figure 13).

Given these variables, a pairwise MRF factors all possible collective team actions x into two functions: **pairwise compatibility** $\psi_{j,i}(x_j, x_i)$ between each robot pair (ij) and **local evidence** $\phi_i(x_i, y_i)$. The joint probability distribution

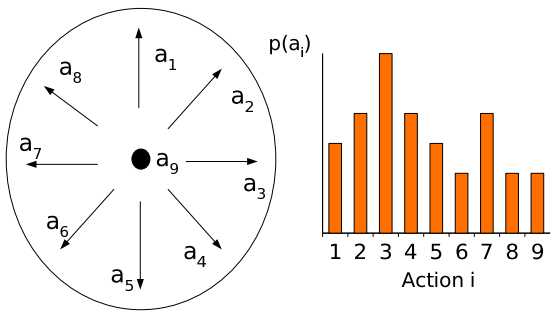

Fig. 13. A picture of the action space for each robot and an example of an associated marginal probability.

can then be stated as follows:

$$Pr(x) = \frac{1}{Z} \prod_{(ij)} \psi_{j,i}(x_j, x_i) \prod_i \phi_i(x_i, y_i) \tag{1}$$

The normalization constant Z ensures that the distribution sums to 1. The formulation in (1) has two key benefits: we factor the global coordination and local computation into distinct terms; and we can express a spectrum of multi-robot action selection methods by modifying these terms.

The local evidence $\phi(x_i, y_i)$ expresses the likelihood of robot i choosing each of its actions x_i, given its observations y_i. This function is analogous to likelihood models as they are used in Bayes filters [7] for localization. In this case, the likelihood function takes care of optimizing the locally observable network. That is it pushes the robot towards optimizing the connections it already has. This function would prefer actions that provide "good enough," but not maximal, signal quality between robots. If the robots were to simply maximize the signal strengths, they would tend to cluster very close together, leading to isolated network components, and providing poor spatial coverage. We counteract these tendencies by designing the function to prefer signal levels that are high enough to provide the necessary quality of service, but not any higher. Also the local evidence function will have a term which encourages robots to stay connected to terminals. For the non-stationary actions, combining these functions gives:

$$\phi_i(x_i, y_i) = \alpha \sum_{j \epsilon N(i)} S_j(y_i)\theta_j(x_i, y_i) + \beta S_t(y_i)\theta_t(x_i, y_i) \tag{2}$$

The function S_j is the squared difference between the optimum signal strength and the actual signal strength from robot j (or terminal t for S_t). θ_j is inversely proportional to the squared difference between the angle of x_i and the optimal angle of movement for correcting the signal strength of robot j. This angle will be calculated by keeping a record of signal strength observations as the robot moves and estimating the signal strength gradient. α and β are constants to appropriately balance the weights of the two terms. The stationary action will have a local evidence function proportional to the inverse of the squared difference between the desired signal strengths and the current signal strengths:

$$\phi_i(x_n, y_i) = \left(\alpha \sum_{j \epsilon N(i)} S_j(y_i) + \beta S_t(y_i) \right)^{-1} \tag{3}$$

The pairwise compatibility $\psi_{j,i}(x_j = a_s, x_i = a_t)$ encodes the likelihood of robots i and j selecting actions $a_s \in x_i$ and $a_t \in x_j$, respectively, from their combined action space $x_j \times x_i$. We want this function to make robots move towards the areas where robots have less connections, but we want to ensure that the robots do not move towards isolated robots that are not part of a useful path. To accomplish this we will use a compatibility function that makes a robot encourage other robots to move toward the direction where

it senses fewer robots, but only if it believes that direction points toward a terminal. We use the following equation:

$$\psi(x_i, x_j) = R(x_i, x_j)T(x_i, x_j) \tag{4}$$

R is proportional to the number of robots robot i can detect in direction j and T is inversely proportional to the difference between angle x_j and the direction robot i believes the terminal to be from robot j.

B. Belief Propagation

To perform action selection with a multi-robot MRF, each robot must compute the probability of its actions being optimal conditioned on all the other robots' action selections (1). That is, for each robot i, we want to compute the marginal probability $p_i(x_i)$, which expresses the likelihood of robot i taking an action, given both its own observations and knowledge of the other robots' actions. Naively, this inference procedure would require communicating all robots' observations to a centralized decision-maker. Instead, we will exploit the factored structure of the MRF to apply to BP algorithm [6], which performs inference in a distributed manner.

BP operates by passing "advice messages" between robots, and using these messages, in combination with local observations, to maintain a belief $b_i(x_i)$ for each robot i. When BP converges, the belief $b_i(x_i)$ is exactly equal to the marginal probability $p_i(x_i)$ that we need for coordinated action selection. Robot i exchanges messages only with its neighbors $N(i)$, ensuring that the algorithm can scale to large teams. Robot i's belief $b_i(x_i)$ is given by the following product, with normalization constant Z:

$$b_i(x_i) = Z\phi_i(x_i, y_i) \prod_{j \in N(i)} m_{j,i}(x_i) \tag{5}$$

The term $m_{j,i}(x_i)$ is an advice message from robot j to robot i suggesting how robot i should act, given what robot j knows about the world. Robot j computes its message to a neighboring robot i as a product of robot j's local evidence, the pairwise compatibility between their actions, and incoming messages from robot j's neighborhood (except for those coming from robot i), summed over all possible actions x_j:

$$m_{j,i}(x_i) = \sum_{x_j} \phi_j(x_j, y_j)\psi_{j,i}(x_j, x_i) \prod_{k \in N(j) \setminus i} m_{k,j}(x_j) \tag{6}$$

The essence of inference with belief propagation lies in these pairwise messages $m_{j,i}(x_i)$, which will be sent wirelessly from robot j to robot i at each iteration of the robot control loop. These messages are continually updated according to (6), and they, in turn, update each robot's beliefs. When all the robots' beliefs converge, they are exactly equal to the marginal probabilities for each robot's state conditioned on the information available to all the robots. Each robot listens only to its neighbors, but since the neighbors' beliefs are conditioned on the information from their neighbors, the final result are beliefs that are conditioned on the information available to the whole team.

Algorithm 3 Belief Propagation algorithm for robot i

Initialize messages $m_{i,j}$ and send to all neighbors j
while 1 **do**
  Gather messages $m_{j,i}$ from all neighbors j
  Gather local evidence y_i
  Recompute beliefs $b_i(x_i)$ with new $m_{j,i}$ and $\phi(x_i, y_i)$
  Select (MAP) action from $b_i(x_i)$
  Recompute and send messages $m_{i,j}$ to each neighbor
end while

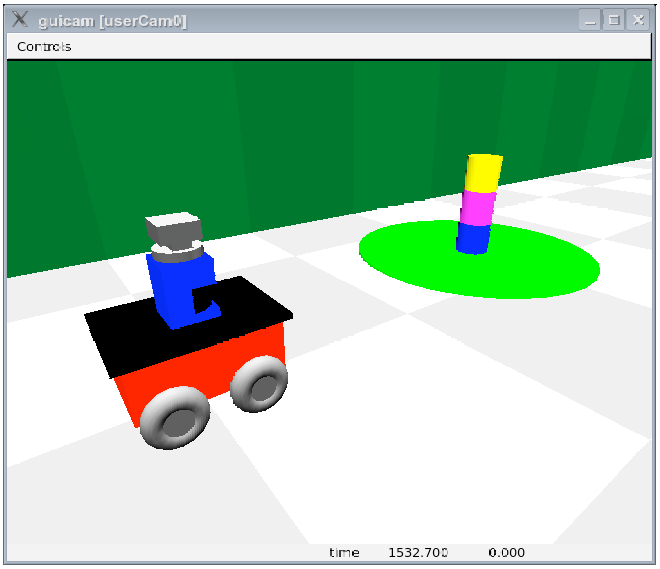

Fig. 14. An image of a Pioneer robot facing a goal beacon from our COS simulation in Player/Gazebo.

Selecting actions The result of multi-robot belief propagation will be an *action posterior*, a probability distribution over actions for each robot. Whenever an individual robot needs to make a new action decision, it computes its action posterior as a belief from local evidence and incoming messages, according to (5). We'll then use this posterior distribution to find the action with the maximum probability of being correct conditioned on the action distributions of the surrounding robots.

Our previous work in distributed action selection for multi-robot teams [8] provides some evidence that our MRF-based approach can be effective. In this work, we conducted 50 "chain-of-sight" trials in simulation (Player/Gazebo [5]) with five Pioneer robots (Figure 14). In each trial, inference using MRF produced successful action allocations, causing the team to produce a chain of sight from a start to a goal location. The 50 trials consisted of test runs on five different configurations of start, goal and robot locations, resulting in 10 trials per configuration. The number of total messages transmitted before reaching a correct arrangement ranged from 10 messages, in situations where only two robots were required for the COS, to 80 messages for complicated arrangements. The number of messages needed also varied over multiple runs on the same scenario, an indication of the randomness involved in the problem.

In 10 of these 50 trials, a strategically important robot was purposely shut down to test dynamic recovery abilities of the system. When such faults occurred, the robots properly adapted their beliefs, which resulted in reallocations that closed the gaps. Note that for obvious reasons the group can only recover from such a failure if there are enough robots left to physically span a chain. In cases where there a more robots in a group than are needed to span a minimal COS, the redundant robots do not impede normal execution and allocate themselves along the chain-of-sight path. This ability to reallocate in the case of robot failure will be especially useful in the case of providing redundant multi-hop wireless connectivity.

V. Conclusion

Both our deterministic graph-based approach and our MRF approach have shown a lot of promise in simulation. However, the real test for any robotics application is an actual embodied situation. We plan to start testing both algorithms on a common robot platform in controlled starting conditions in order to provide accurate head to head comparison.

References

[1] M. Ahmadi and P. Stone, "Keeping in touch: Maintaining biconnected structure by homogenous robots," in *National Conference on Artificial Intelligence (AAAI)*, 2006.

[2] P. Basu and J. Redi, "Movement control algorithms for realization of fault-tolerant ad hoc robot networks," *IEEE Network*, vol. 18, no. 4, pp. 36–44, 2004.

[3] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," *Discrete Math.*, vol. 86, no. 1-3, pp. 165–177, 1990.

[4] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal of Computing*, vol. 1, no. 2, pp. 146–160, 1972.

[5] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems," in *Proc. of the Intl. Conf. on Advanced Robotics (ICAR)*, Coimbra, Portugal, Jul. 2003, pp. 317–323.

[6] J. S. Yedidia, W. T. Freeman, and Y. Weiss, *Exploring Artificial Intelligence in the New Millennium*. Morgan Kaufmann, 2001, ch. Understanding Belief Propagation and Its Generalizations.

[7] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, September 2005, iSBN 0-262-20162-3.

[8] J. Schwertfeger and O. Jenkins, "Multi-robot belief propagation for distributed robot allocation," in *Proc. of the IEEE Intl. Conf. on Development and Learning*, London, England, 2007.