In *Proc. of the AAAI Workshop on Auction Mechanisms for Robot Coordination (AuctionBots)*
pages 51-57, Boston, Massachusetts, July 17, 2006

Commbots: Distributed control of mobile communication relays

Brian P. Gerkey **Roger Mailler** **Benoit Morisset**
gerkey@ai.sri.com **mailler@ai.sri.com** **morisset@ai.sri.com**

Artificial Intelligence Center
SRI International
Menlo Park, California, USA

Abstract

Digital wireless communication networks are vital in modern life. However, these networks rely on centralized infrastructure that is extremely vulnerable to disaster or attack. We seek to improve the communication environment in post-disaster scenarios by developing groups of mobile communication relay robots, or *commbots*, that can be deployed to replace damaged infrastructure. The job of a commbot is to position itself so as to maximize overall network performance, which presents a rich and complex distributed learning and control problem. We have implemented three distributed algorithms and have tested them in simulation on a variety of instances of the commbots problem. We present results from these simulations and compare and contrast the different approaches.

Introduction

Digital wireless communication networks are an increasingly vital tool in modern life. We rely on GSM, GPRS, and WiFi networks every day for voice, email, text messaging, Web access, and other digital interactions. While under nominal conditions such networks generally function efficiently and reliably, the centralized infrastructure on which they depend is extremely vulnerable to disaster or attack. We need look no further than the aftermath of September 11, 2001 or hurricane Katrina in 2005 to see proof of the fragility of digital network infrastructure and the unfortunate consequences of a municipal communication breakdown. When mobile phones stop functioning, victims cannot call for help, rescue workers cannot coordinate with each other, and government officials cannot obtain the situational awareness that they need.

We seek to improve the communication environment in such post-disaster scenarios by developing self-organizing communication networks that can be deployed on demand to replace damaged infrastructure. One solution in this direction is for the rescue teams to carry their own digital repeaters, placing them in the environment where needed. This approach has two major disadvantages. First, the humans must know enough about radio characteristics to decide where relays are needed. Given the complexity of RF propagation patterns in indoor environments, it is unrealistic to assume that paramedics, for example, would be able to decide where to drop relays as they explore a building. Given the high frequencies (and thus short wavelengths) typically used in digital radios, a small move (e.g., < 1 meter) can have a large impact on signal quality. Second, the resulting physical arrangement of relays is fixed. The static nature of the arrangement severely limits the ability of the network to adapt to changes in its environment, whether from the mobility of users or the loss of one or more robotic teammates (because of limited battery life, malfunction, etc.).

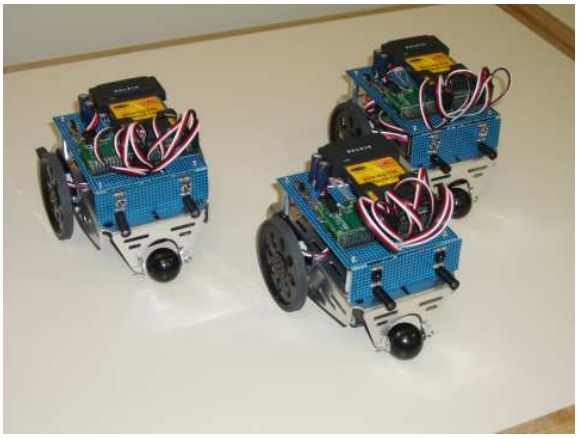

Figure 1: *Commbot research prototypes. The robots are based on an off-the-shelf hobbyist kit, augmented with a low-power ARM-based Linux computer equipped with an 802.11b WiFi adapter. Robot hardware is accessed through the Player device interface (Vaughan, Gerkey, & Howard 2003).*

We propose to address these issues by endowing the relays with physical mobility. We envision small, inexpensive, mobile robots that act as digital radio relays (Figure 1). With appropriate coordination and control, a system of such communication robots, or *commbots*, could form a mobile *ad hoc* network (MANET) to support rescue operations and logistics. Such a network could carry whatever digital traffic is necessary for the task, including text, audio, and video. The same approach could be used to create a digital radio network where no previous infrastructure existed, such as in biological field research, or planetary exploration.

In this paper, we present preliminary work on the commbots problem, focusing on the underlying issues of control and coordination. We have implemented three distributed algorithms and provide a comparative study of their performance in a simulation environment. We hope to motivate other researchers to apply their expertise to this interesting and important problem.

Problem Statement

The application area that motivates our work is communication support in post-disaster situations. Consider a rescue crew entering a large office building after a major earthquake. The radios carried by the rescuers become increasingly unreliable as they penetrate deeper into the building. To avoid losing communication with each other and with their command facilities outside the building, rescue personnel are able to drop commbots that act as radio relays. The relays are effectively disposable, and hundreds may be deployed in a single scenario.

Assumptions

Because of the need to make them small, light, and inexpensive, we assume that the commbots are relatively simple. A commbot can move slowly and uncertainly within some limited range of its original location, and it has no knowledge of the positions of its fellow robots. Commbots do not have a map of their environment, nor do they carry precise sensors (e.g., scanning laser range-finders) that would allow them to build maps. The robots do *not* share a common coordinate system or have any knowledge of the relative locations of their teammates. A commbot's primary sensor is its radio; it is by discovering and interacting with its neighbors, both immediate and distant, that a commbot must decide how to act.

We assume that the commbots are uniquely identifiable (e.g., by MAC address), and that they use a common MANET routing protocol, such as AODV (Perkins, Belding-Royer, & Das 2003), OLSR (Clausen & Jacquet 2003), or TBRPF (Ogier, Templin, & Lewis 2004). The routing protocol is responsible for establishing and updating routes through the network. This protocol provides the commbot with the following information:

- list of immediate (one hop) neighbors
- list of all outgoing routes, where a route comprises
  - the list of nodes through which the route passes
  - a real-valued metric quantifying the quality of the route

Additionally, the commbot can determine, with some uncertainty, its pose relative to where it was originally placed (probably through odometric integration). Given this information, the commbot must decide where to move, within some predefined radius of its initial location.

Objective function

The global goal is to optimize network performance, which can be measured in many ways. For the purposes of the experiments presented in this paper, we use an objective function that seeks to minimize the number of components in the communication graph, as well as to maximize the total quality of routes. We are given

- C: set of n commbots
- $R(i,j)$: indicator function that returns 1 if there exists a route from commbot i to commbot j, 0 otherwise
- $Q(i,j)$: real-valued function that returns the quality of the best route from commbot i to commbot j

We define the utility U of an arrangement of commbots as follows:

$$U = \sum_{i \in C} \sum_{j \in C, j \neq i} \alpha R(i,j) Q(i,j) - \beta \neg R(i,j) \qquad (1)$$

The first term is the total quality of all outgoing routes from commbot i; the second term is a penalty for any nodes that commbot i cannot reach. Our goal is to position the commbots so as to maximize U.

A key characteristic of this problem is that the route qualities are not known *a priori*, nor can they be predicted. The value of $Q(i,j)$ for some configuration of commbots can be determined only experimentally by moving the robots into that physical configuration. Furthermore, the impact on global utility of moving a single commbot is effectively unbounded. Consider a star-like configuration, in which the central commbot is the only route between the other $(n-1)$ commbots: a single move of that central commbot could change the global utility between the minimum and maximum attainable values.

In other words, despite being additive across robots, the objective function is not decomposable, monotonic, or convex. As a result, the only way (to our knowledge) to guarantee a globally optimal solution is to enumerate and evaluate all possible configurations. Since the number of configurations grows exponentially with the number of commbots, this brute force approach is clearly intractable. In addition, the brute force approach is impossible in practice to execute because it requires global knowledge of the positions of all the robots and a way to synchronously command them in order to explore the space of configurations. The goal of our work is to develop distributed algorithms that provide parsimonious tradeoffs between solution quality and overhead (e.g., running time, battery usage). We aim for techniques that efficiently explore the space of configurations and find *good*, but not necessarily *optimal*, solutions.

Related Work

The problem of controlling mobile communication relays has received some attention in the robotics community. One approach is to deploy the robots in a convoy, with robots in the rear deciding when to stop in order to maintain a communication link back to a command center (Nguyen, Farrington, & Pezeshkian 2004). This technique will build a single route from a command center to the front-most robot, but could not build a more general network to service multiple users and/or command centers. Another approach is to analyze the environment and use spatial reasoning with heuristics (e.g., try to maintain line of sight) to decide where relays should be placed (Konolige *et al.* 2004). This method

requires a map of the environment, which is not available in the problem we study in this paper.

Auction- and market-based techniques have been used by many researchers to solve multirobot coordination problems (Dias *et al.* 2005). The algorithms include iterated first-price auctions of single-robot tasks (Gerkey & Matarić 2002) and combinatorial auctions of multirobot tasks (Dias 2004). A persistent question in such synthetic markets is how to determine bidding rules when individual robots cannot directly evaluate the global objective function (Tovey *et al.* 2005). The problem that we address differs from those studied in previous work primarily in that the global objective function is not separable or otherwise decomposable across robots, which significantly complicates the process of deciding on local utility values.

Another field of research that has investigated optimization in distributed environments is the work from the distributed constraint optimization (DCOP) community (Modi *et al.* 2003; Mailler & Lesser 2004). This work focuses on solving problems that are naturally distributed and need to be optimized, but make three key assumptions that make them difficult to adapt to the commbots problem. First, this line of work assumes that the robots know their relationships with one another. As was mentioned in the problem description, commbots are not aware of their impact on the overall utility until they actually sample a particular configuration. The second assumption is that the global utility function is monotonic. This assumption is also not valid because the goal is to optimize route quality, which cannot be estimated without considering the route as a whole. In fact, the overall utility of the route might be very negative until the final critical robot is placed into the correct position. The third invalid assumption is that communication between the agents is perfect. This is certainly not true because the motion of the commbots may cause the communication network to vary from being completely connected to completely disconnected, making communication impossible. In addition to not addressing many of the difficulties associated with this problem, DCOP algorithms are designed to produce optimal results and because of that do not scale very well. Scalability is a key requirement of the commbots problem, so approximate algorithms that are highly scalable are preferred to optimal solutions.

Algorithms

We have implemented two centralized and three distributed algorithms. The centralized algorithms are enumerative brute force and simulated annealing. Because of the exponentially large search space, the brute force approach is viable only for small populations of fewer than 10 commbots. Simulated annealing, on the other hand, is very efficient and on small populations where we have the brute force solution against which to compare, annealing always finds an optimal solution. Because of the lack of scalability of the brute force approach, in the experimental results we present the annealing solution as a surrogate optimum. These global algorithms, which are not realizable on any physical commbot system, are used only to establish the relative performance of the distributed algorithms.

The distributed algorithms, explained in detail below, are a local form of simulated annealing, a modified version of distributed breakout (Yokoo & Hirayama 1996), and auction-based team formation. Because a single commbot does not have access to global state information, it cannot directly evaluate the objective function given in Equation 1. Instead it uses the following local (unbounded) approximation, in which the outer summation over all commbots is removed:

$$U_l(i) = \sum_{j \in C, j \neq i} \alpha R(i,j) Q(i,j) - \beta \neg R(i,j) \qquad (2)$$

The three distributed algorithms use $U_l(i)$ to estimate the utility of a configuration from the perspective of commbot i.

Local annealing

The first approach we applied to this problem was a localized version of the simulated annealing algorithm (Kirkpatrick, Gelatt, & Vecchi 1983). Simulated annealing works by selecting a random next state and if that next state has a better utility than the current one, the state is changed. If, however, the next state is worse than or equal in value to the current utility, the move is taken with probability $e^{\frac{\alpha \Delta U}{t}}$ where for the sake of this paper, $\alpha = 0.5$.

There are several reasons to choose local simulated annealing as an approach to solving the commbots problem. First and foremost, one of the key characteristics of the commbots problem is that the robots do not know the utility of a state without first sampling it. This leads to a common problem seen in distributed learning where robots have to trade off getting up-to-date utility estimation through exploring new states or exploiting current knowledge to obtain a good solution. Simulated annealing is a good method to use, because even when the utility of moving to another position is perceived to be suboptimal, robots will still move there with some probability, thereby updating their estimate. In this implementation, the estimation of the value of a position is a running average of the last five samples taken at that position. This estimation does not explicitly take into account the position of any of the other robots in the environment, which eliminates the need for the robots to explicitly communicate. For the most part, this technique works because each robot is trying to move to its best position which makes the system as a whole climb the utility gradient. To bootstrap the estimation process, at startup, each robot does one sample from each of its potential positions before starting the annealing process.

Distributed breakout

The second algorithm implemented was a modified version of the distributed breakout algorithm (DBA). DBA is a hill-climbing protocol that is a distributed adaptation of the centralized Breakout algorithm (Morris 1993). DBA works by alternating between two modes. During the first mode, the *wait_ok?* mode, robots collect from their neighbors *ok?* messages containing current state information. This information is then used by each robot to calculate its locally optimal state and the improvement in its local utility if it

were to switch to this new state. Each robot then sends to its neighbors *improve?* messages containing its improvement value and then changes to the *wait_improve?* mode.

In the *wait_improve* mode, the robots collect *improve?* messages from their neighbors. The purpose of the improve message is to find the neighbor with the highest local improvement and to elect it to change its value. If a robot receives an improve message with a higher improvement than its own, it is not the leader and does not change its value. This method provides a relatively stable upward climb in utility, but can be slower than other techniques because of its controlled nature. DBA allows at most only half of the robots to change their value at any given time and takes two steps to elect a leader.

Once the robots have elected a leader, the leader changes its value. The robots send *ok?* messages to one another, and switch to the *wait_ok?* mode.

The original DBA protocol was designed to solve distributed constraint satisfaction problems (DCSPs) where the relationship between the variables was known a priori. For the protocol to work in the commbots domain, several modifications were needed. The first of these was to incorporate a method for estimating the value of links between each of the robots. Unlike the distributed annealing approach, the DBA implementation uses pair-wise estimators where the utility of a particular position is estimated based on the current state of all the robot's neighbors. This allows for a more accurate estimate of a position's true contribution to the global utility, but further slows the protocol's convergence as the robots simultaneously explore their environment and hill-climb.

One of the other key differences between the local annealing approach and the DBA approach is in how the robots choose to explore their environment instead of exploiting prior knowledge. The DBA algorithm presented in this paper calculates the explicit cost and value of the information in choosing when to explore an unknown state. The cost is calculated as a percentage of loss of utility from the optimal solution to the unexplored state. This cost is compared to the value of information that is measured as the percentage of robot/state pairs that would be sampled by moving to this new position. If the overall gain is nonnegative and the robot's current improve value is zero, then the move is made. The overall behavior exhibited by this technique is that the robots perform quite a few exploration moves early in the search, but as their knowledge and local utility increases, they become less inclined to make a costly exploratory move.

Auction-based team formation

The third algorithm employs local auctions for team formation. The intuition behind this approach is that it can be profitable to explore configurations that are similar to a known high-value configuration. The algorithm is distributed, with each commbot executing the same control code. A commbot has three modes of operation: *RANDOM*, *LEADER*, and *FOLLOWER*. After a commbot moves, it sends to all its neighbors a *location* message containing the robot's current position in its local frame.

Each commbot begins in the *RANDOM* mode, randomly exploring its local area. Using *location* messages received from neighbors and information from the routing table, the robot constructs a hash table of tuples (*state*, *utility*). The *state* is a set of positions (in their respective local frames) for some subset of the commbots, and the *utility* is locally calculated according to Equation 2 and averaged over all times that the robot has observed this *state*. This state / utility table is a more detailed memory of past experience than the memories maintained by DBA or local annealing.

Each robot keeps track of the maximum utility it has seen so far. After some period of time has elapsed without this maximum changing, a commbot transitions to the *LEADER* mode in order to explore particularly promising configurations. The robot returns to the position where it saw the maximum utility and sends *auction* messages to the neighbors that it had in that state. Each *auction* message contains a desired position for the recipient, along with an estimated utility for moving to that position. The goal of the leader is to reconstitute as much as possible of the previous best configuration by offering its neighbors an opportunity to improve their utility. Because its previous neighbors are not necessarily within communication range, some or all of the *auction* messages may not be delivered.

Upon receipt of an *auction* message, a robot that is currently in the *RANDOM* or *FOLLOWER* modes will compare the estimated utility to its current utility. If the estimated utility in the message is greater, the robot will enter the *FOLLOWER* mode (if not already in that mode) and move to the desired position. Because multiple leaders may be sending *auction* messages simultaneously, they are competing for the attention of their neighbors, possibly causing them to defect from other teams.

Having constructed a team of followers that are in the desired positions, the leader systematically explores its local area, recording state and utility information. At the end of this systematic exploration, the team is dissolved and both leader and followers transition back to the *RANDOM* mode.

Termination occurs when a robot has not seen a change in the maximum utility for a sufficiently long period of time. At this point, if the robot is in the *LEADER* or *RANDOM* mode, it will return to the position where it saw the maximum utility. A robot in the *FOLLOWER* mode will remain in its current position.

Simulation

To evaluate the relative strengths and weaknesses of each of the algorithms discussed in the previous section, we constructed an abstract version of the commbots problem in a simulator. Figure 2 shows a screenshot of the simulator with 24 robots. Within the simulation, each robot is implemented as a separate thread that executes one cycle at a time. During each cycle, the robot can receive its messages, process those messages, and move only once. The robots actually move between cycles and the movement occurs instantly. To simplify the calculation of the optimal solution, the simulation discretizes the potential positions of the robots (Figure 2). To ensure that the robots do not get synchronized, the sim-

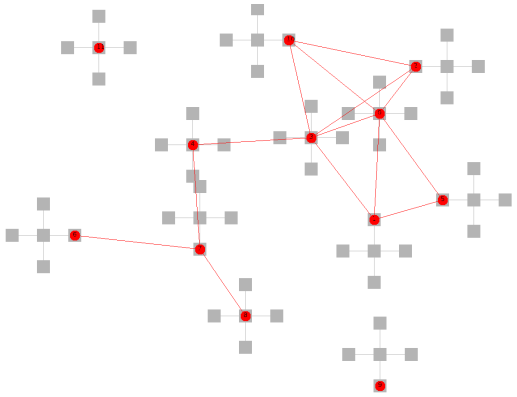

Figure 2: *Screen shot of the commbots simulator with 24 robots.*

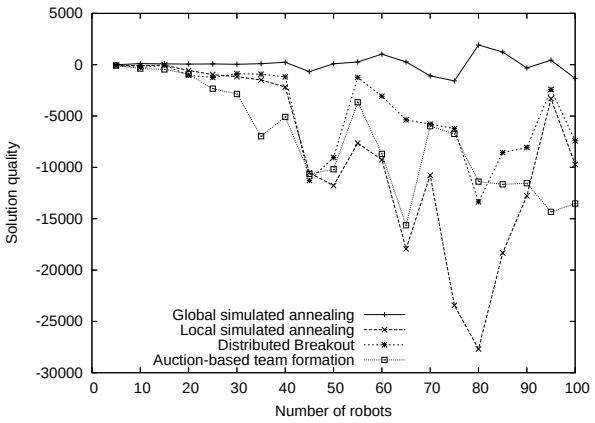

Figure 3: *Utility of various techniques for solving the commbots problem.*

ulator inserts a random duration sleep to simulate the action of movement.

Robots are able to communicate with one another only if they are in communication range. Currently, we do not allow robots to communicate over multiple hops in the communications network. This restriction creates uncertainty in knowing where other commbots are located because they are not always able to inform each other about their current state. Messages are delivered instantaneously and without loss although, in the future, we plan to relax this assumption by introducing loss according to the simulated signal qualities between the robots.

On startup, each commbot is placed in a random position within the environment and is given knowledge only of other robots within communication range. As the commbots move around and in and out of the range of other commbots, the simulator updates their respective routing tables. The commbots can check their tables at any time to get an up-to-date view of the overall quality of the network from their perspective. For the sake of the simulation, the function $Q(i, j)$ from equation 1 is computed as

$$Q(i,j) = \frac{1}{D(i,j)^2} \tag{3}$$

where $D(i, j)$ is the travel distance of the shortest path from i to j in the communication network. Although one can imagine a number of equally valid functions to optimize, this one was chosen to simulate the loss of signal quality that occurs over distance.

As the simulation progresses, the simulator measures the number of movements made by the robots, the number of messages that are transmitted, and the current global solution quality. The simulation terminates when all the robots report that they no longer wish to move.

Test Setup and Results

To test the various protocols, we set up the simulator to vary the number of robots while keeping the ratio between communication and movement distance constant and placing the robots pseudo-randomly in the environment. The following rules were used to place the commbots:

- The first robot is placed at a random location.
- Every commbot must be within some maximum distant d_{max} to at least one other commbot.
- Every commbot must be no closer than d_{min} to all other commbots.

The communication range for the commbots was fixed at 300 units and the movement radius was 80 units. We set $d_{min} = 150$ and $d_{max} = 350$ to ensure that every commbot had at least one position from which it could communicate with another commbot and that the overlap between the commbots was kept to a minimum. For the tests we varied the number of robots in the environment from 5 to 100, testing at 5-robot intervals. We generated one scenario for each of the data points and ran each protocol 10 times to minimize the effects of variance caused by the randomized nature of the protocols.

The results of the tests can be seen in Figures 3, 4, and 5. Looking at Figure 3, one can see that none of the protocols does as well as the centralized simulated annealing approach. Although not shown in these graphs, we ran the brute force algorithm and compared the results with the centralized annealing method for a small number of commbots ($n < 10$). The global annealing approach does exactly as well as the brute force search and takes considerably less time with the brute force approach taking nearly 25 minutes to compute a solution for nine commbots on a dual-3 GHz Pentium-based desktop.

Also notable on this graph is that it appears that DBA outperforms the other methods. This is a bit deceptive as the variance (not shown because it cluttered the graph) in the samples was very large, making the difference between the protocols seemingly insignificant. Looking at Figure 5 we can see that local simulated annealing makes fewer moves before converging on a solution. In power-critical applications such as this one, this is a very desirable feature. In addition, notice that all the distributed techniques use fewer robot movements to converge on a solution than the centralized annealing approach. This is a result of the cooling

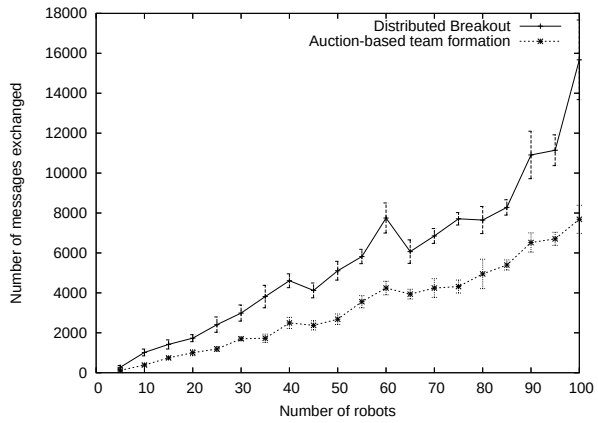

Figure 4: *Communication usage of distributed techniques for solving the commbots problem.*

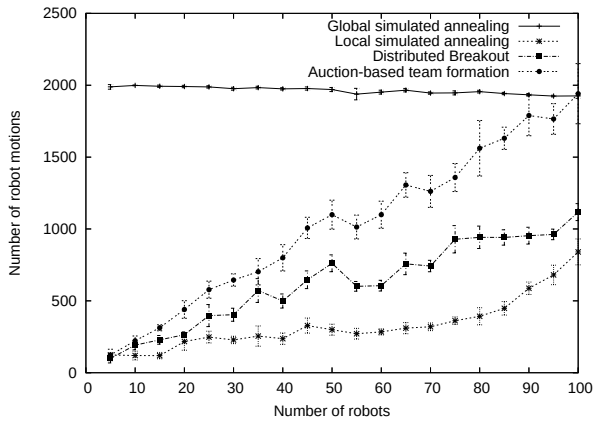

Figure 5: *Number of moves made by the commbots before convergence.*

schedule of the annealing process, which can be changed to converge faster, resulting in fewer moves.

Between the distributed protocols, the local annealing process uses the least communication because it uses no communication at all. One can see that DBA uses more messages than auction-based teams, however. This is most likely caused by the two-step nature of the DBA protocol, which causes every robot to send messages to each of its neighbors on every cycle.

From the results, a number of interesting findings can be seen. The most profound finding is that the distributed protocols do not do significantly better even when they use more communication and robot movements. We suspect that the cause of this lies in the way the global utility is being calculated and the ability of the individual robots to understand their effect on it. In an effort to promote scalability, all the protocols in this paper restrict their interactions to their immediate neighbors and use only this localized information to make their decisions. However, the global utility is not an aggregation of the local utilities of the commbots, but is in fact related to the routes that are created in the network. This means that local estimations are very likely to be poor in determining the effect that any one commbot has on the overall network's quality.

The results are not entirely discouraging because they suggest that if the robots can discover information through abstraction and aggregation, they may be able to make better localized decisions. It also suggests that partial centralization-based techniques like cooperative mediation (Mailler 2004) or regional or hierarchical auctions may be very good at solving these problems effectively and efficiently.

Summary

We introduced the *commbot problem* of controlling a group of mobile communication relay robots in order to maximize network performance. We implemented three distributed algorithms and presented results from simulations in which we tested these algorithms on randomized populations of commbots. The data that we have gathered so far suggest that simple message-passing schemes in which the robots' state retention is purely local are insufficient for this problem.

We hypothesize that significant benefit can be had by focusing future algorithm development on two key areas: sharing state information more widely, and recognizing and exploiting patterns in the observed states. In addition to further algorithm design and simulation, in the near future we will deploy and test teams of physical commbots with human users. Our hope with this work is to motivate other researchers to engage with us in studying the commbots problem, which exhibits both theoretical richness and real-world importance.

References

Clausen, T., and Jacquet, P. 2003. RFC 3626: Optimized Link State Routing Protocol (OLSR).

Dias, M. B.; Zlot, R. M.; Kalra, N.; and Stentz, A. T. 2005. Market-Based Multirobot Coordination: A Survey and Analysis. Technical Report CMU-RI-TR-05-13, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Dias, M. B. 2004. *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Gerkey, B. P., and Matarić, M. J. 2002. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation* 18(5):758–768.

Kirkpatrick, S.; Gelatt, C.; and Vecchi, M. P. 1983. Optimization by simulated annealing. *Science* 220:671–680.

Konolige, K.; Ortiz, C.; Vincent, R.; Morisset, B.; Agno, A.; Eriksen, M.; Fox, D.; Limketkai, B.; Ko, J.; Stewart, B.; and Schulz, D. 2004. Centibots: Very large scale distributed robotic teams. In *Proc. of the Intl. Symp. on Experimental Robotics (ISER)*.

Mailler, R., and Lesser, V. 2004. Solving distributed constraint optimization problems using cooperative mediation. In *Proc. of the Third Intl. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2004)*.

Mailler, R. 2004. *A Mediation-Based Approach to Cooperative, Distributed Problem Solving*. Ph.D. Dissertation, University of Massachusetts, Amherst, MA.

Modi, P. J.; Shen, W.-M.; Tambe, M.; and Yokoo, M. 2003. An asynchronous complete method for distributed constraint optimization. In *Proceedings of the Second Intl. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS-03)*.

Morris, P. 1993. The breakout method for escaping local minima. In *Proceedings of the Eleventh Natl. Conf. on Artificial Intelligence*, 40–45.

Nguyen, H.; Farrington, N.; and Pezeshkian, N. 2004. Maintaining communication link for tactical ground robots. In *Proc. of the Assoc. for Unmanned Vehicle Systems (AUVSI) Unmanned Systems North America*.

Ogier, R.; Templin, F.; and Lewis, M. 2004. RFC 3684: Topology Dissemination Based on Reverse-Path Forwarding (TBRPF).

Perkins, C.; Belding-Royer, E.; and Das, S. 2003. RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing.

Tovey, C.; Lagoudakis, M.; Jain, S.; and Koenig, S. 2005. The generation of bidding rules for auction-based robot coordination. In L.E.Parker, et al., eds., *Multi-Robot Systems: From Swarms to Intelligent Automata, Volume III*. the Netherlands: Springer. 3–14.

Vaughan, R. T.; Gerkey, B. P.; and Howard, A. 2003. On device abstractions for portable, reusable robot code. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2121–2427.

Yokoo, M., and Hirayama, K. 1996. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *Intl. Conf. on Multi-Agent Systems (ICMAS)*.