Affective Recruitment of Distributed Heterogeneous Agents

Aaron Gage and **Robin R. Murphy**
{agage, murphy}@csee.usf.edu
Center for Robot-Assisted Search and Rescue
University of South Florida
4202 E. Fowler Ave. ENB 118 Rm. 342
Tampa, FL 33620

Abstract

Members of multi-robot teams may need to collaborate to accomplish a task due to differences in capabilities. This paper describes an extension of the ALLIANCE architecture that enables agent recruitment within a decentralized UAV-UGV robot team without task preemption but 1) uses a formal model of emotions and 2) handles heterogeneity. Affective computing allows recruitment to be robust under loss of communication between agents and minimizes the number of messages passed. Data from 66 simulations show that the *affective* strategy succeeds with a random message loss rate up to 25% and requires 19.1% fewer messages to be sent compared to *greedy* and *random*, and that of these, *affective* scales best with team size. Comparisons of broadcast to unicast messaging are also made in simulation.

Introduction

Collaboration among members of a heterogeneous multi-robot team is motivated by a need to complete tasks that require more capabilities than a single robot can provide. This paper contributes an *affective* recruitment protocol that enables robots to request and receive assistance from other members of the team using an emotional model. Simulation results show that this approach is robust in terms of communication losses, requires less communication than other obvious methods, and that it scales well with the number of robots in the team. Thus, although the task domain for this work is landmine detection, other areas (such as low-power systems, swarms, sensor networks, and stealth applications) may also benefit.

The recruitment protocol described in this paper borrows the *standards-based* emotion SHAME from Ortony's formal model of emotions (Ortony 2002). Emotions are useful in robots, as they provide a mechanism for self-regulation, such that a change in a robot's state or behavior can be induced (Murphy *et al.* 2002) if the robot is making no progress on a task. Emotions have been shown to produce emergent social interactions and improve task performance of a team of heterogeneous robots (Murphy *et al.* 2002). In this work, emotions determine when a robot decides to help another. The protocol is designed for a fully decentralized robot team, without task preemption, and to be tolerant of unreliable communications. It is assumed that the robots have a common coordinate frame for the purposes of estimating distances and for locating each other.

This paper presents a formal description of the protocol and emotional model. This work is motivated by a landmine detection task using a robot team with a single unmanned aerial vehicle (UAV) and multiple unmanned ground vehicles (UGVs). The UAV is tasked with performing a raster scan over a minefield, identifying possible mines. When a possible mine is detected, the UAV recruits a UGV, using the *affective recruitment* strategy, to investigate further. Results from 72 simulations testing scalability, robustness under communication failures, and the impact of unicast versus broadcast messaging are presented and discussed. Four additional simulations of illustrative use-cases demonstrate where *greedy* recruitment does not produce optimal results, but *affective* does.

Related Work

(Cao *et al.* 1995) provides a survey of the issues inherent in multi-robot cooperation. The work in this paper is an extension of the ALLIANCE architecture (Parker 1998) for coordinating distributed multi-robots. ALLIANCE's *impatience* and *acquiescence* produce a behavior that closely resembles recruitment: when a robot is having difficulty completing a task, another robot will come over and relieve it. However, there are four important differences between this work and ALLIANCE. First, ALLIANCE assumes that the robots in the team are all on the same task, and that any of them can substitute for another. In our approach, teams of heterogeneous robots on different tasks can be used. Second, ALLIANCE implicitly assumes that the communication among all members of the robot team is reliable, whereas in our approach, it is expected that communications (or even entire robots) may fail at any time. Further, the recruitment protocol will compensate for temporary communication failures. Third, ALLIANCE assumes that a robot can be preempted from its task, whereas in this approach, there is no preemption. The final difference is that this approach is based on a formal theory of emotions by Ortony (Ortony 2002), where SHAME is a *standards-based* emotion.

The work in this paper uses a contract-net protocol (CNP)

(Smith 1980) with a first-price auction (Monderer & Tennenholtz 1998). MURDOCH (Gerkey & Mataric 2002) is an instantaneous greedy task scheduler that also uses CNP with a first-price auction; its communications impact increases linearly with team size (Gerkey & Mataric 2003), whereas the approach in this paper scales more slowly. Other auction-based approaches include LEMMING (Ohko, Hiraki, & Anzai 1996) that reduces communications by applying the outcome of one auction to successive auctions but assumes that robots will stay in contact, and CEBOT (Cai, Fukuda, & Arai 1997) that uses a World Model, which is not appropriate for a decentralized approach.

Approach

In this approach, a fixed recruitment protocol begins with a robot (*requester*) broadcasting a request for assistance (in the form of a *HELP* message), and ends when another robot (*responder*) has arrived and begun performing a task on behalf of the *requester*. Our approach uses one primary *standards-based* emotion (SHAME) to modulate responses to *HELP* messages and determine when a robot will allow itself to be recruited. As a robot refuses to help its teammates (by ignoring *HELP* messages), its SHAME increases. When its level of SHAME passes a threshold, the robot will respond. Once the robot decides to respond, its SHAME will be reset to zero (just as motivations in ALLIANCE are reset when they cross a threshold (Parker 1998)). SHAME will also decay over time.

Three communication issues guide the design of this protocol. First, the recruitment algorithm should use a minimal amount of bandwidth. Applications that require low-power or stealthy behavior benefit from prevention of unnecessary transmissions, and in any case, the communication requirement should scale well with the number of agents. Second, the delivery method for messages is broadcast. In our test domain, the real robots will be using a wireless network to communicate. Wireless Ethernet channels are a shared medium, so any transmissions are automatically broadcasts, and received packets that are not intended for a particular robot are simply ignored. As a beneficial side-effect, the amount of network traffic scales slowly with the size of the robot team as seen in Results. The third design issue is that the protocol must be robust in terms of network failure. In a fully distributed system, it is assumed that anything can fail at any time, and that no member of the team should wait forever for a failed robot to respond. Therefore, the recruitment protocol is based on a 3-way TCP/IP handshake and recovers gracefully from lost messages or failed robots.

Implementation

The recruitment protocol uses a set of six messages. Each message contains the ID number of the sender, the ID number of the recipient, if any, whether the message is broadcast or unicast, and a message type. There are six message types in the recruitment protocol: *HELP*, *ACCEPT*, *RESPONDER*, *ARRIVAL*, *AGREE*, and *ACKACK*.

The protocol is shown graphically in Figure 1. It begins when the *requester* robot broadcasts a *HELP* message with its location and a percept that a robot must have to be a *responder*. If another robot decides to assist, then it responds with an *ACCEPT* message that contains an estimate of the time needed to reach the *requester* based on the location provided in the *HELP* message and the robot's rate of travel. The process by which a robot decides whether to assist is described after the protocol messages.

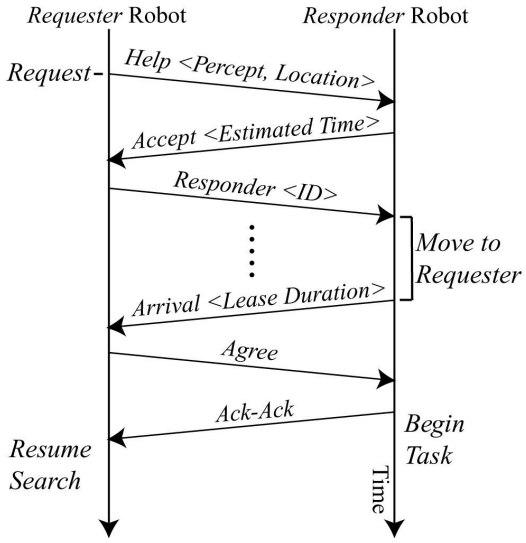

Figure 1: Recruitment protocol in terms of the messages sent between robots.

When the *requester* robot receives an *ACCEPT* message, it broadcasts a *RESPONDER* message to all robots with the ID of the *responder*. For the *responder* robot, this serves as confirmation that its offer to help was accepted, and it will begin moving to assist. For all other robots, this message is an explicit notification that their help is not needed.

The second stage of the protocol begins when the *responder* robot arrives near the *requester* robot and provides an *ARRIVAL* message, which contains the duration of a lease. The purpose of the lease is to compensate for a complete failure of either robot or a loss of communications (as the recruitment ends when the lease expires). If the *requester* robot agrees to the lease, then it will respond with an *AGREE* message. Finally, the *responder* robot will send an *ACKACK* message and take over the investigation of the possible mine. Once the analysis is complete, the agreed lease duration will expire and the recruitment ends.

The *affective recruitment* strategy uses an emotional model to determine under what conditions a robot will respond to a *HELP* message, assuming that it is otherwise available (not on task, able to provide the required percept). At the heart of this model is the affective variable SHAME. The notation for this model will be presented first, followed by details on how to choose the parameters.

Given a team of n robots, $\{r_1, \ldots, r_n\}$, each robot r_i in the team maintains a level of SHAME, s, such that $0 \leq s \leq 1$, and s is initialized to zero. Each r_i also has a threshold, t, where $t \leq 1$. Three additional parameters affect the behav-

ior of *affective recruitment*: c is a constant that is added to s each time r_i ignores a *HELP* message. $d()$ is a function of the distance D between the *requester* and r_i, and is also used to increase s. $k()$ is a decay function in terms of elapsed time ΔT since the previous *HELP* message.

When a *HELP* message arrives, each robot r_i will first account for the decay of its SHAME since the previous request: s is updated as $s = s - k(\Delta T)$. Next, if $s > t$, then robot r_i sends an *ACCEPT* message to the *requester*. Otherwise, if $s \leq t$, then r_i ignores the request and s is updated as $s = s + c + d(D)$.

The parameters were chosen based on experience as follows. t determines how resistant r_i is to being recruited. If $t < 0$, then *affective recruitment* will become *greedy*, in which r_i immediately offers its help when a request is made. The more t approaches 1, the more r_i will ignore requests before responding. In this work, the value $t = 0.75$ was used. The terms c and $d()$ are related to t. c controls how quickly a distant robot will respond to a request. If r_i and the *requester* are infinitely far apart, then t/c is approximately the number of requests r_i will ignore before responding (considering that s will decay between requests). In this work, the value $c = 0.2$ was used, so r_i would tend to respond after approximately 4 requests. $d()$ penalizes r_i for ignoring a request from a close neighbor, and should increase as the distance D decreases. In this work, $d(D) = 0.5/D$ so that r_i responds within two requests if it is within 1 unit of the *requester*, and $d()$ has little effect beyond 10 units. The decay function $k()$ determines how quickly r_i will lose its SHAME after ignoring a request. The function $k(\Delta T) = 0.005 \times \Delta T$ was used so that r_i would lose SHAME acquired from a single request in about 40 seconds, and would require 200 seconds to go from $s = 1$ to $s = 0$. This relatively low rate of decay keeps r_i responsive to the needs of the team; if the decay were faster, then periodic requests would tend to be ignored. If the decay were negative, then r_i would tend to "want" to help more over time until it was recruited and s was subsequently reset to zero.

The *requester* robot will continue to send *HELP* messages periodically until it receives an *ACCEPT* message in reply. If the *requester* receives more than one *ACCEPT* message in response to a single *HELP* broadcast, then it will examine the *ACCEPT* messages and choose the sender that specified the least time needed to arrive.

Experiments

Three recruitment strategies (*greedy, random,* and *affective*) were tested in simulation for a mine detection task. The purpose of these experiments was to measure the performance of the recruitment strategies according to two metrics: the number of messages sent among robots in the team, and the total amount of time that a robot had to wait for assistance. The size of the robot team, rate of random message loss, and messaging type (unicast or broadcast) were varied to test the impact on the recruitment process for each strategy.

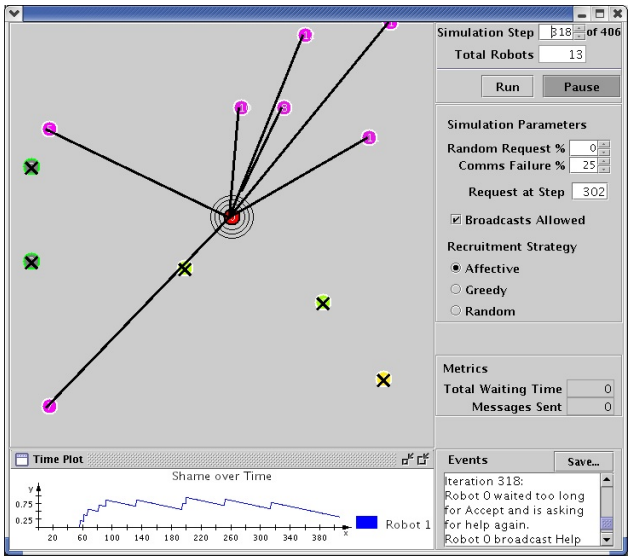

Figure 2: User interface for recruitment simulator. The UAV (center) requests assistance, and all eligible robots with sufficient SHAME respond (solid lines). Those that ignore the request are marked with an X.

Simulated Scenario

In this task, one robot was designated as an unmanned aerial vehicle (UAV) and performed a raster scan over a 100×100-unit grid at a rate of 3 units per iteration. At five locations in this scan, the UAV stopped, requested assistance, and waited for another robot to arrive. At the end of the raster scan, the UAV stopped for 20 seconds before performing the scan again in the opposite direction.

Between 3 and 22 other simulated robots, representing unmanned ground vehicles (UGVs), were placed in the 100×100-unit grid. Two of these robots were tasked with raster scans of half of the grid each, at a rate of one unit per iteration, stopping for 20 seconds (and becoming temporarily available for recruitment) at the end of the scan before restarting in the opposite direction. An additional one to twenty idle robots were distributed across the grid. The number of idle robots was varied from one to twenty to test the effect of team size on performance. These idle robots were available for recruitment by the UAV at any time. For each simulation testing the performance of a recruitment strategy, the locations and behavior of all robots were the same unless otherwise noted.

The simulator was coded in Java, using JINI, to test robot recruitment. The simulator controlled three experimental parameters: which recruitment strategy to use, the rate of random communication failures, and which messaging method (unicast or broadcast) to use. The simulator is shown in Figure 2.

Recruitment Strategies

Three recruitment strategies were tested in simulation. The first was *affective recruitment*, in which the closest idle robot

	4	8	13	23
Affective	48	51.67	61	79.7
Greedy	29	50	75	125
Random	29	50	75	125.7

Table 1: Average number of messages transmitted for each strategy for varying team size. The total number of robots in the team is shown across the top of the table.

	4	8	13	23
Affective	377.4	214.2	176.2	175.3
Greedy	260.2	120	93	92
Random	261	203.1	261.6	326

Table 2: Average UAV wait time according to team size, measured in seconds. The total number of robots in the team is shown across the top of the table.

whose SHAME was above a threshold was recruited for each request.

The second recruitment strategy was *greedy recruitment*, in which the idle robot with the minimum estimated time to arrive was recruited for each request. While *greedy recruitment* will tend to have the faster response times than *affective recruitment*, because it does not spend time building up SHAME, it requires an additional message to be sent from every idle robot so the *requester* can choose the least arrival time, and thus does not scale well in terms of communication overhead.

The third recruitment strategy was *random recruitment*, in which an idle robot was chosen at random for recruitment. When the *requester* transmitted a *HELP* message, each idle robot replied to indicate its availability, and one of these was then chosen randomly. As with *greedy recruitment*, this will lead to a faster decision than *affective recruitment*, but does not scale as well for communication. Further, *random* may choose robots that are far away, which will tend to result in longer response times.

Results

The results of three sets of simulations are provided below. In these experiments, the team size, rate of communication failures, and messaging type were varied to measure the effect on the recruitment strategies.

Effects of Team Size

The effect of varying the size of the robot team was measured using twelve simulations: for each of the three recruitment strategies, a simulation was performed with 1, 5, 10, and 20 idle robots. The metrics for this test were the total number of messages sent among the robots, and the amount of time that passed, in seconds, from the initial UAV requests until a UGV arrived and was acknowledged. The results of these simulations are shown in Tables 1 and 2.

These results indicate that the *affective recruitment* strategy requires more messages to be sent than *greedy* or *random* when there is only one robot that can be recruited but that once the number of robots increases, *affective recruitment* requires much less communication. The reason behind this difference is that in *affective recruitment*, the UAV only needs to send out *HELP* messages and wait for a single reply to begin negotiating recruitment. As a result, the number of messages that must be sent is almost constant. The variation in the number of messages for *affective recruitment* occurs when more than one robot responds to a particular *HELP* message, or when all UGVs are far from the UAV and additional *HELP* messages must be sent to push their level of SHAME over the threshold. On the other hand, *greedy* and *random* must solicit messages from all other members of the team in order to make a choice, and the number of messages per recruitment increases linearly with the team size.

These results also show that the amount of time that the UAV spent waiting for help to arrive favors the *greedy recruitment* strategy over *affective recruitment*. This is an expected result because *affective recruitment* requires time to build up the level of SHAME in the robots before they will respond. In these simulations, the UAV waited 3 seconds between requests for help before calling again, and needed to request from five to ten times per recruitment before it received a response. If the parameters for updating SHAME were tuned or learned, this difference between *greedy* and *affective* could be reduced. Learning these parameters is a direction for future work. In general, the faster a robot's SHAME exceeds the threshold, the closer *affective recruitment* will resemble *greedy*, to the point that if the SHAME exceeds the threshold after a single request, then the two strategies are equivalent. One advantage of the *affective recruitment* strategy is that it allows for a flexible trade-off between response time and the number of messages transmitted.

Finally, these results show that *affective recruitment* outperformed *random*, because although a recruitment choice was made immediately with *random*, the closest *responder* was typically not chosen, so the time required for it to arrive was higher than for the other strategies.

Note that in this set of simulations, the locations at which the UAV made requests were fixed, and to prevent any bias from the UGV positions, three randomly generated arrangements of robots were used. The results reported above represent 36 simulations, and the averages are shown.

Effects of Communication Loss

An additional 27 simulations tested the effects of random message loss for the recruitment strategies. Each of the three strategies was tested with 5%, 10%, and 25% of the messages between robots being randomly dropped (not transmitted, but with no notification to the sender). Ten idle robots plus the two tasked UGVs and the UAV were used in each simulation, for a total of 13 robots. In these simulations, the choice of what messages to drop was made randomly by the simulator, and the impact of that choice varied. Thus, each of these tests was conducted three times to capture the typical performance of each strategy. The results from these simulations are shown in Tables 3 and 4.

These results show that the recruitment protocol continues to function despite network losses, and that the relative

	0%	5%	10%	25%
Affective	61	81.3	67.3	147.3
Greedy	75	107.3	117.7	161.7
Random	75	93	99.3	189.7

Table 3: Average number of messages transmitted according to network failure rate. The rate of random message loss is shown across the top of the table. Note that the 0% column is repeated from Table 1.

	0%	5%	10%	25%
Affective	176.2	311.7	190.1	587.4
Greedy	67	148.4	167.8	330.6
Random	261.6	385.8	340.3	897.7

Table 4: Average UAV wait time, in seconds, according to network failure rate. The rate of random message loss is shown across the top of the table. Note that the 0% column is repeated from Table 2.

performance of each of the recruitment strategies remains consistent as the rate of message loss increases. As before, *affective recruitment* requires the fewest messages to be sent, on average, followed by *greedy* and *random*. In fact, *random* performs better than *greedy*, when it was expected that they would perform equally well (since they send the same number of messages when there are no losses). By the time that losses reach 25%, the particular recruitment strategy used does not make much difference, because at that point, there is only an 18% likelihood that the six consecutive recruitment messages required by the protocol will all be sent properly. As before, *greedy recruitment* still resulted in the least time spent waiting by the UAV, followed by *affective recruitment*.

A particular weakness of *greedy recruitment* is that the UAV must obtain the locations of all eligible robots before it can choose the nearest one. Assuming a decentralized team, this requires an explicit communication from all other robots to the UAV, which may be impacted by network losses. If the nearest robot to the UAV fails to receive a *HELP* message or to send a reply, then the UAV may commit to recruiting a different, more distant robot, and be forced to await its arrival. On the other hand, using *affective recruitment*, the UAV will tend to make multiple requests, which reduces the reliance on any single *HELP* message. Suppose that an eligible robot, r_1, is nearest to the UAV and fails to send a reply due to network problems. Provided that no other robots had sufficient SHAME to respond to that request (for instance, if they were far away and accrued SHAME more slowly), the UAV would quickly request again and have another chance to recruit r_1. In other words, requesting over time can find solutions that even outperform *greedy recruitment*, if the time between requests is less than the additional time a more distant robot needs to arrive.

	Broadcast	Unicast
Affective	67.3	379.3
Greedy	117.7	194
Random	99.3	183.3

Table 5: Average number of messages transmitted according to messaging type. Note that the Broadcast column is the same as the 10% column in Table 3.

Broadcast versus Unicast Messaging

Nine simulations were conducted to test the effect of using unicast (single-sender, single-receiver) transmissions instead of broadcast in the recruitment protocol. In these simulations, instead of sending a single *HELP* message to all other robots, the UAV would attempt to send *HELP* messages to all other robots individually. As with the network failure tests, 13 robots were used, of which 10 were untasked. A network failure rate of 10% was used. If there had been no losses in this test, then the number of messages would have trivially been a function of team size. The results of these simulations are shown in Table 5. Note that due to the random nature of the message losses, each strategy was tested three times and the results were averaged.

These results show that *affective recruitment* relies on broadcast messaging to minimize the total number of messages. Since *affective recruitment* sends multiple *HELP* messages before another robot has high enough SHAME to respond, these requests are multiplied by the number of idle team members, which goes well beyond the number of messages required by *greedy* or *random* (for which a single *HELP* message is sufficient). Thus, broadcast messaging is better suited for this application than unicast.

Illustrative Use Cases

Although the results above indicate that *greedy recruitment* will tend to produce the shortest wait times for the UAV, this is not universally true. There are cases in which the *affective recruitment* strategy results in shorter wait times than *greedy*. Suppose that there are three UGVs, r_1, r_2, r_3, such that r_1 is untasked, and r_2 and r_3 perform a raster scan. Let r_2 move two units per iteration, while r_1 and r_3 move one unit. Thus, r_2 and r_3 will finish their tasks at different times. Next, suppose that at time step t_0, the UAV sends a *HELP* message, and two robots, r_1 and r_3 are idle, and although r_2 can reach the UAV faster than r_1 or r_3, it is on task and cannot respond. In the *greedy* and *random* strategies, r_1 or r_3 would be chosen for recruitment immediately, whereas with *affective recruitment*, no selection would be made, but all of the UGVs would increase their levels of SHAME. If r_2 finishes its task at time step t_1, it can then be recruited by the UAV and arrive sooner than r_1 or r_3. This particular case was tested in simulation, and it was found that using *affective recruitment*, r_2 was chosen and arrived after 65.4 seconds. The *greedy* and *random* strategies selected r_1 which arrived after 95.2 seconds and 95.4 seconds, respectively.

Another simple use case demonstrates that through *affective recruitment*, the UAV will choose the nearest robot with-

out requiring that all robots reveal their locations (as with the *greedy* strategy). Suppose that, as above, three robots r_1, r_2, r_3 are idle at a particular time t_1 when the UAV makes a request, but the UAV is nearest r_2. After each ignored request, r_2's SHAME will increase faster than that of r_1 or r_3, because r_2 is closest to the UAV. As a result, r_2 will be the first to exceed its threshold for SHAME and will respond before r_1 or r_3. Thus, the UAV recruits the closest robot without requiring all robots to transmit their locations. This behavior has been verified in simulation. In this scenario, the UAV broadcast 5 *HELP* messages, at which point r_2 responded, and the recruitment completed normally. Neither r_1 or r_3 ever broadcast any messages. If this were a low-power or stealth application, r_1 and r_3 would have been spared an unnecessary transmission by using *affective recruitment* rather than *greedy*.

Robot Tests

The recruitment protocol was demonstrated on a pair of ATRV-Jr. robots, where the robots responded to requests from a simulated UAV. In these tests, the robots determined their positions via GPS, and when recruited, navigated to a specified waypoint. Further robot tests are ongoing.

Conclusions

Results from 66 simulations have shown that the *affective* strategy resulted in 19.1% lower communications overhead overall compared to *greedy* and *random*, and that *affective* scaled better (18.6% fewer messages sent with 10 idle robots, 36.3% fewer with 20 idle robots). These simulations have also shown that *greedy* required 39.4% less time to complete recruitments overall when compared to *affective*.

Note that although *affective recruitment* required more wait time to complete than *greedy*, the choice of parameters in the simulation influenced the total time required. With a careful selection or adaptation of parameters, *affective recruitment* can compromise between response time and communication requirements as needed for a particular domain.

These simulations also demonstrated that the recruitment protocol is robust in terms of communication failures such that it still functioned in an environment where 25% of all messages were lost. With communication losses, *affective recruitment* still required fewer transmissions than *greedy* and *random*, though as above, *greedy* required the least time to complete recruitments. The simulations have also provided experimental justification for using broadcasts instead of unicast messaging when performing recruitment, as *affective recruitment* required 464% more messages (and *greedy* required 65% more) using unicast than with broadcast.

These simulations assumed no preemption of robots from their tasks, and it is expected that if this restriction were lifted, *affective recruitment* would behave more intelligently than *greedy*. In particular, *affective* builds up a level of activation (SHAME) in the robots over time, whereas *greedy* makes decisions based only on the current instant in time. In terms of fairness (that is, which robots are chosen to assist), *affective* is less likely to call on the same robot twice in a row than *greedy* (because a robot's SHAME resets to zero when it is recruited). SHAME may also control other aspects of behavior, such as causing a robot to suspend or abandon its own task in order to help another, or at least make it "hurry up."

Future work will examine the impact of preemption in recruitment as well as fit-matching perceptual capabilities of robots to determine how well a given robot can perform the task it is being recruited for (expanding on the *metric evaluation* in (Gerkey & Mataric 2002)). Methods for learning the parameters used in *affective recruitment* will also be developed. Finally, the effect of non-linear parameters for updating SHAME will be examined.

Acknowledgments

Portions of this work were supported by ONR Grant N00014-03-1-0786 and DOE Grant DE-FG02-01ER45904.

References

Cai, A.; Fukuda, T.; and Arai, F. 1997. Information sharing among multiple robots for cooperation in cellular robotic system. In *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot Systems (IROS '97)*, volume 2, 1768–1774.

Cao, Y.; Fukunaga, A.; Kahng, A.; and Meng, F. 1995. Cooperative mobile robotics: Antecedents and directions. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, 226–234.

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

Gerkey, B. P., and Mataric, M. J. 2003. Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA '03)*, 3862–3868.

Monderer, D., and Tennenholtz, M. 1998. Optimal auctions revisited. In *Proceedings Fifteenth National Conference on Artificial Intelligence*, 32–37.

Murphy, R.; Lisetti, C.; Tardif, R.; Irish, L.; and Gage, A. 2002. Emotion-based control of cooperating heterogeneous mobile robots. *IEEE Transactions on Robotics and Automation*.

Ohko, T.; Hiraki, K.; and Anzai, Y. 1996. Reducing communication load on contract net by case-based reasoning — extension with directed contract and forgetting. In *Proceedings of the Second International Conference on Multi-agent Systems*, 244–251.

Ortony, A. 2002. *On Making Believable Emotional Agents Believable*. The MIT Press. chapter 6, 189–211.

Parker, L. 1998. Alliance: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation* 14(2):220–240.

Smith, R. G. 1980. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers* c-29(12).