Impact of Human Advice on Agent Teams: A Preliminary Report

Nathan Schurr
University of Southern California
Henry Salvatori Computer Center 237
Los Angeles, CA 90089-0781
schurr@usc.edu

Paul Scerri
University of Southern California
Henry Salvatori Computer Center 237
Los Angeles, CA 90089-0781
scerri@isi.edu

Milind Tambe
University of Southern California
Henry Salvatori Computer Center 232
Los Angeles, CA 90089-0781
tambe@usc.edu

ABSTRACT

Given a large-scale team composed of intelligent members that are following derived heuristics, the performance of the team is often suboptimal. Providing more accurate heuristics may be infeasible or even impossible. We propose an approach that incoporates a human advisor interacting with a multiagent team in which the advice that the human gives is a component of the heuristic that each agent already uses. This approach allows us to have a clear way of adjusting the level of the team's autonomy, addresses the issue of who on the team will be affected by the advice, and also factors in advice immediately (while the team is still performing).

We looked back at data from previous experiments with an advisor (fire chief) in a disaster rescue simulation. Then we applied our approach to a domain where robots maintain a room full of sensors. We studied different advisors and varied how much the team members listened.

Our initial results show that the problem of how to have a team of multiagents interpret advice from a human turned out to be much harder than we first thought. Our hypothesis is that human advice will be of assistance when the human can provide strategic advice that was previously unknown to the agents. Yet the the strategic advice must still be given in a manner in which the agents can understand through their limited methods of communication.

1. INTRODUCTION

Teams of intelligent members are emerging as an exciting paradigm for achieving complex tasks in dynamic, distributed environments. These teams may be comprised of agents, robots or even people, yet such teams must coordinate despite uncertainty, spatial distribution and limited communication bandwidth. These challenges are evident in a scenario in which close to 100 robots are tasked to perform surveillance and tracking in an unknown environment. In order to overcome these challenges, the team of robots is guided by a Team Oriented Program (TOP) [4, 7] that describes their plan and overarching goals.

The challenge of performing effective coordination [3] requires that the team use heuristic algorithms of lower complexity, which may turn out to be suboptimal. This lack of optimality occurs because heuristics are commonly derived by an interpreter from a pre-defined, generalized TOP that does not take into account the dynamics of the environment. Unfortunately, due to its inability to observe the entire state of the whole world, an agent team will often not be able to detect when its approximations are not working effectively and its behavior is unnecessarily poor. Even if a team becomes aware of its poor performance, it might not be aware of how remedy its position. One approach to improving a team's functioning is to have a human watch the execution of the team and provide advice when required to improve the performance. The idea of advice works best when the human has a reasonable overview of team performance.

The goal of our research is to develop general methods for building advice-taking teams and this paper presents our framework and some initial experiments. Various techniques have been applied to the general problem of allowing human reasoning to improve agent behavior. This idea of human advice has been explored in the field of reinforcement learning (RATLE) [1] and single agent planning [2]. However, previous techniques have been mostly applied to the case of advising a single agent. There has been some work on advising teams, such as the work at CMU where a software agent coach advises a simulated soccer team [5]. Our focus is on a human advisor to a team of agents. Two limitations with this previous work become apparent when considering the case of advising a team of agents. First, these prior methods have worked with advice that is only given offline. This does not suit the dynamic and unpredictable environments that we are focusing on. Myers' Advisable Planning Systems require advice prior to the formation of the plans [2]. Even in RATLE, execution of the reinforcement learning is stopped, advice is factored in and then execution can resume. Second, previous approaches do not address how to have a single piece of advice apply to all of the relevant members of a team. We expect that as multiagent teams grow larger, it will become infeasible for a human to provide advice to each individual agent on a team.

In this paper, we propose an approach where the human provides general, high-level advice that is incorporated into agent decision making as an additional factor to consider when choosing between

AAMAS'03 Workshop: Humans and Multi-Agent Systems, July 14/15, 2003, Melbourne, Australia.

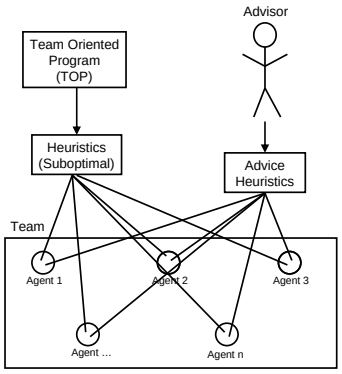

Figure 1: Overall framework.

options (see Figure 1). We believe that effective advice will be given online and in a team member independent manner, with individual team members considering both the context and advice when making coordination decisions. The human gives advice by sharing their perceived importance of tasks to be accomplished by the team. This will result in a change of the team's behavior. We model decisions made by team members as being a function of various environmental and team factors. Each team member has an additional advice factor that has the potential to be altered by the team-level advice. This framework has three key differences from previous work. First, we apply a weighting to the human advice in order to help have a clear way of adjusting the level of the team's autonomy. Second, we address the issue of team-level advice by having each agent factor in the advice given to the team directly into their own advice component mentioned above. They look to the behavior changes that the human has suggested and find which of the changes are associated with behaviors that they are involved in. Third, our approach also takes advantage of the dynamism in the advisor and the domain by allowing for the immediate factoring of advice. Having instant feedback allows the advisor to make gradual adjustments to the behavior change advice in order to have overall team performance achieve nearer to optimal behavior.

The problem of how to have a team of multiagents interpret advice from a human turned out to we much harder than we first thought. Our results from these initial experiments show that in most cases the human hurts the overall performance of the team. This is so surprising because our previous work that dealt with an emergency rescue domain suggested that human advisors could play a major role in benefiting a multiagent team [6]. Nonetheless our preliminary study shows the potential for high-level advice being critical to increasing multiagent team performance.

Our hypothesis is that human advice will be of assistance when the human can provide strategic advice that was previously unknown to the agents. Yet the the strategic advice must still be given in a manner in which the agents can understand through their limited methods of communication.

2. DOMAIN

Here we will define the domain which we would like to focus on. We take a large-scale team of agents that are following a Team Oriented Program (TOP) [4, 7] and making decisions based on a heuristic. We assume complete communication between agents and

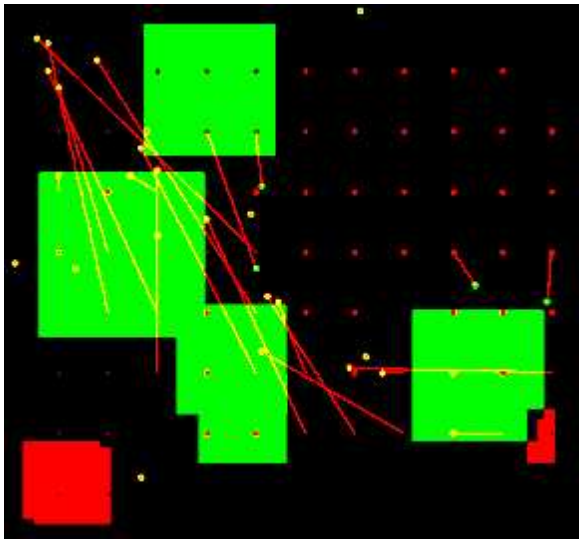

Figure 2: Screenshot of Simulator.

the advisor in order to concentrate on the issue of how to interpret advice. However there are limits to the amount of bandwidth and knowledge that the agents have. All agents are assumed to be collaborative and commit actions based on a heuristic that is aimed at benefitting the whole team, but need not. Although advice can conceivably stem from a vast array of sources, we focus on how to interpret advice that comes from humans. For our preliminary study, we will assume that the human advisor has complete, accurate information about the state of the team. The human is not necessarily an expert of the domain and needs no knowledge of the heuristic model or TOP that the team is following. Instead, the advisor has a general knowledge of what the team's abstract goals are and what is used as a measure of the team's performance.

We have been referring to a human advisor, yet there is the need here to distinguish this advisor from simply a user of the team of agents. In the case of a human user, commands that encompass certain tasks are given to the team. These tasks are then carried out by the team. The advice that we mention falls under the Myers category of strategic advice [2]. Contrary to commands, this advice is above the level of tasks and provides general guidelines on which types of roles should be prioritized. Another key difference is that a piece of advice depends entirely on the reactions of the agent that is receiving the advice. In a user-tool relationship, the tool will do whatever it is told to do by the user. These commands are indisputable and must be carried out. Advice differs greatly because advice may be carried out, but has the potential to be considered only lightly or even ignored. Even with regards to the advice being ignored, it can be ignored completely as if no advice was ever received or it can be temporarily ignored until higher priority tasks are finished and the advice has more of an effect on the relative situation.

The simulator we use replicates a situation in which a sensor network is to be maintained over a period of time. Throughout this time a set of robots make sure that the sensor network stays while the batteries of the sensors are continually being depleted and some sensors even break down and fail. In order to keep sensors from running completely out of battery, this team of robots can each take

a single sensor at a time to a charging station to replenish the battery. Then the sensor must be placed back into the physical network. The robots that move the sensor around can be thought of as shepherds moving around the sensors (sheep). For example, when a sheep is out of battery, a shepherd will move the sheep to the recharging station and back again to its original location (home). At any one time, a particular shepherd robot will be assigned to the role of recharging a single sheep sensor. After the shepherd's sheep has been recharged and returned to its home, that shepherd may then switch to the role of recharging any available sheep. The shepherds adhere to a TOP that they all possess. This TOP explains to the group how it will complete the recharging process on a particular sheep as well as the method that each shepherd will implement when choosing which sheep to begin recharging. Figure 2 shows a screenshot of this simulator, in which the shepherds (bright dots) trying to pick up the sheep that are arranged in a grid (dark dots). Once a shepherd has been assigned to a pick up a sheep, there is a line drawn from the shepherd's current location to the sheep's home. The brighter colored rectangules are regions of advice and will be explained further in section 4.

3. MULTIAGENT ASPECTS

Critics might argue that there is nothing different that comes up when considering a large team of agents, and the same issues would come up when giving advice to a single agent. Actually, giving advice to these large teams brings up new isssues that do not arise when dealing with giving advice to a single agent primarily because of some key traits that exist when considering a team with a large number of agents.

Coordination of multiple agents is the primary concept that comes into play when dealing with large teams and complex tasks. Advice for coordination has the potential to arise even in a scenario such as the shepherd and sheep scenario mentioned in section 2. Though not currently implemented in this way, imagine that the task of picking up a sheep is a coordinated task that requires two shepherds. These two shepherds are needed in order for each one to hold an opposite end and pick up the sheep. Here lies an example of where coordination advice can be a suggestion of how many shepherds should pick up a sheep at the same time. In this case, unless two shepherds pick up a particular sheep in a coordinated manner, the efforts of the shepherds are futile. Furthermore, sending too many shepherds at once would be a waste of both resources and time. This complex problem of having a decentralized team of agents work together in order to act at coinciding times is an aspect that is unique to advice taking teams.

This coordination of large-scale teams has three main implications. One implication is that the advice about coordination is different from other kinds of advice. This is because coordination advice inherently involves a concept that is dependant on more than one agent. Consequently, the team must agree on the validity and implications of the advice. Thus, team members may need to negotiate in order to operationalize the advice in real-time, as no single agent could operationalize it alone. Otherwise the advice could have a great potential for negatively impacting the team. The second implication is that if we were to advise a team on coordinated action by advising individuals, it would be more tedious. An advisor can either advise the team together or give advice to each individual on what it should do. In large-scale teams, addressing the whole team is easier, while addressing individual members may require too much micro-management. Providing advice at the level of the team when doing coordinated action is somewhat easier for the human than providing direct advice to individual agents. Lastly, the third implication of team-level advice is that while it is useful, it means that the advice is not tailored to each individual. Yet the advice must still must have effects on an individual agent level. Thus, given team advice, each individual has to interpret that advice in its own context and act in the best interest of the team. For instance, if the team is advised to make a certain task top priority, if everyone in the team rushes to do the same task, that would be highly problematic. At the same time, some members of the team, while not performing the task, should know that it has been made a priority because of their current task's effect on the prioritized task.

This suggests that the advice at the team level may need to be high-level advice and it may need to be more abstract than at the single agent level. Within this new coordination advice type, lies the ability to affect not only what a team does, but also how a team works together. Furthermore, it allows the human advisor to give compact advice, without unnecessarily dealing with the intricacies of the team's actions.

4. APPROACH

In our approach to having the multiagent team take advice, we leverage the heuristic that already exists in each of the agents. These heuristics that are derived from the team's TOP can be viewed as the method that each agent takes in order to arrive at the decision of what that agent should do next. These guided actions are designed to be the actions that would best help the team give the current observable conditions. This heuristic can be represented by having the agent choose a free task such that a*A + b*B + c*C + ... is a maximum for that task, with A,B,C,... being the factors of the heuristic and a,b,c,... being the respective weights of those factors. For example, in our sheep and shepherds example, when deciding on which sheep to pick up next, the shepherds factor in the distance the shepherd is from the sheep, the distance that the sheep is away from the charging station, and the battery level of that sheep. Though this initial framework is fairly simple, it is straightforward to generalize it to other problems/domains.

We have tailored our approach to the implications of team-level advice that are mentioned in Section 3. Rather than trying to give advice on an agent-by-agent basis, which is tedious with large-scale teams, the advisor is allowed to give advice on which tasks to complete and with what level of urgency. Once the team has this task associated advice, it can be translated into the human's perceived utility (H) for each task. Each member of the team has access to this human advice, H, and adds it to the agent's calculated utility; its calculation becomes a*A + b*B + c*C + ... + h*H. Now the agent must maximize this augmented heuristic that takes into account the human's advice. Although there is the potential for each agent to maintain its own weight to the advice (h), we have decided to keep the weight constant for all team members. This in order to create a team listening level. Thus h can be thought of as how much a team listens to advice, with a higher h causing H to influence the heuristic more resulting in a team that listens better. This approach also takes into account the problem of having all agents listen to advice in the same manner (see Section 3). The value of H can be a function of the agent that is computing the heuristic. The human advice (H), in our solution, factors into it the current state of the agent and the capability of the agent to perform the advised task.

Because the human advice is only a single factor (h*H) of the heuristic, the human advisor is not providing a model for how the team decides which tasks to complete and in which order. Instead,

the human is merely adding his advice, which may be positive or negative, to the current model that each agent has. The agents immediately take into account the adjusted factor once the advice has been given. The human advisor has the ability to view the entire team's positions and actions, after which the human can decide if the advice given should be altered in some way. If so, the human can again give advice and the cycle continues. This is designed to allow the human to gradually fine tune the advice that is offered to the team in order to have the team's overall performance reach closer to optimal.

This approach also makes use of the human advisor's advantages. By having the advice be an augmentation to an existing task utility function, the human does not have to worry about modeling the entire heuristic that a team must follow. This eliminates the need for the human to stoop down and think like an agent. Alternatively, the human can focus on its ability to oversee and quickly estimate the team's performance with respect to the world it is interacting with. In our domain, task based advice has the advantage of being strongly associated with geographical positions. This allows for easy abstraction of advice. Figure 2 shows the advisor's view, with the brighter colored rectangles signifying regions that should be considered as having a higher priority. For example, this method could be used in a situation where tasks in the lower left hand corner of the map need tending to, advice becomes easy to state if you can just advise the team to give higher priority to all tasks that are performed within that general area. Also the human can focus on task-based advice while letting the team handle the decision of which agents to assign to the prioritized tasks. In the context of large-scale teams, the human would be overwhelmed with the decisions of which agent should be handling which tasks. Instead the human has the opportunity to just isolate task regions to favor or ignore and the team will allocate its members accordingly.

5. EXPERIMENTS

In this section, we will provide results on two sets of experiments. The first experiments that we show below are the source of our ideas of how to build our framework. These previous experiments involve an emergency rescue domain that we have prior experience with [6]. This lead us to develop more experiments that are performed in the shepherd and sheep domain that this paper has proposed. Results in the shepherd and sheep domain unexpectedly showed that the human's advice often did not help.

5.1 Previous Experiments

In previous experiments, we focus on an emergency rescue scenario in which a human fire chief interacts with a team of fire brigades in order put out fires that are popping up all over the city. The fire chief interface consists of two frames. One frame shows a map of the city, displaying labeled markers for all of the fires that have been found, the positions of each fire brigade, and the location of the role each fire brigade is assigned to. The fire chief does not have direct access to the simulation state through the simulator itself, but is instead updated according to only the messages received by the fire chief's proxy. Therefore, the fire chief may be viewing a delayed picture of the simulation's progress. The other frame displays a list of all of the role allocation tasks that have been allocated to the fire chief. By clicking on a task, the relevant capability information about each fire brigade is shown. The right-side window lists the fire brigades' distances to the fire, their water levels, and the roles they are currently performing. The fire chief can then view this data and find an appropriate agent to fulfill the role.

# Brigades	*maxAsked*= 0%	*maxAsked*= 100%	*maxAsked*= ∞
3	58(3.56)	73(16.97)	74(0.71)
10	52(19.09)	42(14.00)	73(4.24)

Table 1: Domain-level performance scores.

# Brigs.	*max Asked*	Domain Roles	Fire Chief Roles	Tasks Performed	% Tasks Performed
3	0%	116 (7.12)	401 (51.81)	27 (6.55)	23.29 (6.51)
	100%	146 (33.94)	407 (54.45)	24 (6.36)	16.02 (0.63)
10	0%	103 (38.18)	864 (79.90)	67 (2.83)	14.49 (2.13)
	100%	98 (42.40)	563 (182.95)	41 (8.38)	48.06 (19.32)

Table 2: Role and fire-chief task metrics.

We conducted tests with three different fire chiefs. Each completed several practice runs with the simulation prior to experiments in order to minimize any learning effects. Each scenario was run for 100 time steps, with each step taking 30 seconds. The total data presented here represents 20 hours of run-time with a human in the loop.

Table 1 shows the team's domain-level performance across each experimental configuration. The scoring function measures how much of the city was destroyed by fire, with higher scores representing worse performance. The table shows the mean scores achieved, with the standard deviations in parentheses. Examining our two dimensions of interest, we can first compare the two rows to examine the effect of increasing the complexity of the coordination problem. In this case, increasing the number of fire brigades improves performance, as one might expect when adding resources while keeping the number of initial tasks fixed.

However, we can dig a little deeper and examine the effect of increasing complexity on the fire chief's performance. In the simpler configuration, asking the fire chief earlier (i.e., *maxAsked*= 0) improves performance, as the team gets a head start on exploiting the person's capabilities. On the other hand, in the more complex configuration, asking the fire chief earlier has the opposite effect. To better understand the effect of varying the point at which we assign roles to people, Table 2 presents some of the other statistics we gathered from these runs (mean values, with standard deviations in parentheses). With 3 brigades, if we count the mean number of roles taken on by the fire chief, we see that it stays roughly the same (401 vs. 407) across the two *maxAsked* settings. In this case, asking the fire chief sooner, allows the team to exploit the person's capabilities earlier, without much increase in his/her workload. On the other hand, with 10 brigades, the fire chief's mean role count increases from 563 to 716, so although the proxies ask the fire chief sooner, they are imposing a significant increase in the person's workload. Judging by the decreased average score in the bottom row of Table 1, the increased workload more than offsets the earlier exploitation of the person's capabilities. Thus, our experiments provide some evidence that increasing domain-level scale has significant consequences on the appropriate style of interaction with human team members.

This experiment encouraged us to pursue this method of giving advice. The next experiments aim to reinforce our understanding.

5.2 Shepherd Experiments

In order to follow up on these and leverage human advice, we set out to find out how the human advisor could help the team. For

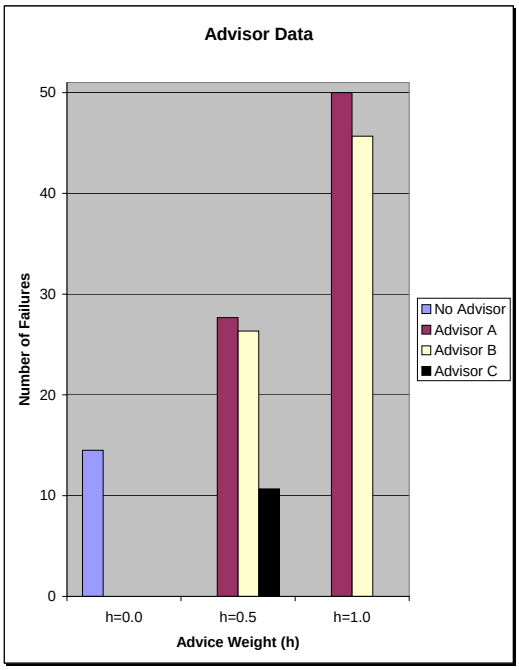

Figure 3: Advisor Performance (number of failures).

these initial experiments, we simulated the sheep and shepherd scenario with a 30 shepherd team responsible for 80 sheep placed in a 100 by 100 square room. Each battery starts out with a value of 1.0 and at every time step the each sheep's battery value is decreased by a value of .00125. Once a sheeps battery level reaches zero, the sheep is considered to be dead and unrecoverable (unable to charge again). Thus a sheep could die in 800 steps. A shepherd is allowed to move in one of the 4 primary directions a distance of 1 unit in the room each time step. Experiments were run over a 2500 time step period. The primary metric that we used to measure the team's overall performance was how many of the 80 sheep ended up dying by the end of the scenario's duration.

The human has the ability to give advice to the team by increasing or decreasing the priority of the sheep in a selected region. A region of unlimited size can be selected by an advisor using the mouse, pressing to start expanding region, and releasing to stop. We performed three tests each with the human advice weight (h) set to both 1.0 and 0.5. We ran the same configuration 30 times without any advice (human advice weight (h) set to 0).

Given this experimental setup, without any advice, an average of around 15 sheep died by the end of the experiment (see Figure 3). However, as seen in the graph, the human advisors on average didn't help the number of sheep deaths decrease. In fact, after being advised the team of shepherds often ended up doing worse. Yet, this is not to say that augmenting the team's optimized heuristic in the manner that we provided was unable to improve performance at all. Advisor C actually managed to find a strategy during his experiments that allowed him to average of about 10 sheep dying. No one else was able to find a useful strategy.

Advice Level (h)	Advisor A	Advisor B	Advisor C
.5	243	92	127
1.0	357	591	N/A

Table 3: Average number of pieces of advice offered.

6. DISCUSSION

In accordance with the findings of previous work, we expected the human advisor to be much more adept at giving advice. These preliminary results show that it is a very difficult task to have the human advisor give the right kind of advice and also have that advice be interpreted correctly in order to improve performance of the team. Throughout the experiments, the fundamental structure for the team to interpret advice was not altered, yet Advisor C was the only advisor that didn't hurt, but rather helped the team. Consequently, we believe that the reason for this difference lies in the type of advice that is given. It is our belief that advice at the team-level enables a degree of abstraction that is not possible with single agent advice. Taking advantage of these team-specific advice types becomes necessary to allow an advisor to more effectively advise a team. As teams grow in size, team-level advice looks to be more useful and more tractable for the humans advising them. We maintain that this approach to advice gave Advisor C the advantage in the experiments. Specifically the coordination team advice is the type that arose during our advice taking experiments with the shepherd and sheep scenario. Advisors A and B concentrated on lower-level methods of advice that led them to far greater number of clicks in most cases. When studying both the log files and the way that Advisors A and B tended to dispense advice, mainly low-level advice was being given. A big percentage of the items of advice were single role priority changes that were given to basically task out role by role, which one should be done next. Though still complex, given the current domain's configuration, the shepherds already knew the same amount of information as the advisors and were able to do make more optimal decisions for this domain.

Advisor C, on the other hand, found a way to use the task advice available. From the table above (see Table 3) Advisor C does not give much more advice than the other advisors, so there must be some other explanation for this difference. We hypothesize that the offering of high-level advice gave Advisor C the advantage in performance results. By using the interface to give more general regions of advice on what large areas to concentrate on and stay away from, Advisor C was able to convey a more abstract level of advice. With this method Advisor C was able to suggest to the agents to let some of the sheep die and concentrate fully on the other sheep, thus getting better performance from the team. The fact that only C was able to figure out how to modify the advice available would suggest a need for different kinds of advice to be available to advisors. This kind of high-level advice is precisely what we consider to be helpful advice to intelligent multiagent teams that lets both the team and human advisor leverage their strengths.

In the future, there is a lot work to still be done on both sides of the advice process: advice-taking teams and advice-giving humans. Our primary direction will be investigating the validity of our hypothesis that an abstract level of advice is the best way for humans to advise intelligent, large-scale multiagent teams. An additional topic to explore is the area of how then to take these abstract levels of advice and have a multiagent team correctly interpret and apply

such advice. We believe that improving this link between human advisor and multiagent team can serve as both a method to greatly improve the performance of the team and make it easier for the advisor to understand the team's performance.

Acknowledgments

We would like to thank Praveen Paruchuri, Alessandro Farinelli, Rajat Bhattacharjee, Dasarathi Sampath and Hyuckchul Jung for help testing.

7. REFERENCES

[1] Richard Maclin and Jude W. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22(1-3):251–281, 1996.

[2] K. Myers. Advisable planning systems. In *Advanced Planning Technology*, 1996.

[3] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[4] David V. Pynadath, Milind Tambe, Nicolas Chauvat, and Lawrence Cavedon. Toward team-oriented programming. In *Proceedings of Agent Theories, Architectures, and Languages Workshop*, pages 233–247, 1999.

[5] Patrick Riley and Manuela Veloso. Coaching a simulated soccer team by opponent model recognition. In *Proceedings of the Fifth International Conference on Autonomous Agents (Agents-2001)*, 2001. (extended abstract).

[6] P. Scerri, D. V. Pynadath, L. Johnson, Rosenbloom P., N. Schurr, M Si, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. In *The Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003 (to appear).

[7] G. Tidhar. Team oriented programming: Preliminary report. In *Technical Report 41, Australian Artificial Intelligence Institute, Melbourne, Australia*, 1993.