On being a teammate: Experiences acquired in the design of RoboCup teams

Stacy Marsella, Jafar Adibi, Yaser Al-Onaizan,
Gal A. Kaminka, Ion Muslea, Milind Tambe
Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way
Marina del Rey,CA 90292
robocup-sim@isi.edu

Abstract

Increasingly, multi-agent systems are being designed for a variety of complex, dynamic domains. Effective agent interactions in such domains raise some of the most fundamental research challenges for agent-based systems, in teamwork, multi-agent learning and agent modelling. The RoboCup research initiative, particularly the simulation league, has been proposed to pursue such multi-agent research challenges, using the common testbed of simulation soccer. Despite the significant popularity of RoboCup within the research community, general lessons have not often been extracted from participation in RoboCup. This is what we attempt to do here. We have fielded two teams, ISIS97 and ISIS98, in RoboCup competitions. These teams have been in the top four teams in these competitions. We compare the teams, and attempt to analyze and generalize the lessons learned. This analysis reveals several surprises, pointing out lessons for teamwork and for multi-agent learning.

1 Introduction

Increasingly, multi-agent systems are being designed for a variety of complex, dynamic domains. Effective agent interactions in such domains raise some of most fundamental research challenges for agent-based systems. An agent in such domains must model other agents' behaviors, learn/adapt from its interactions, form teams and act effectively in a team, negotiate with other agents, and so on. For each of these research problems, the uncertainty and the presence of multiple cooperative and non-cooperative agents, only conspires to exacerbate the difficulty.

Consider for instance the challenge of multi-agent teamwork, which has become a critical requirement across a wide range of multi-agent domains[14, 9, 15]. Here, an agent team must address the challenge of designing roles for individuals (i.e., dividing up team responsibilities based on individuals' capabilities), doing so with fairness, and reorganizing roles based on new information. Furthermore, agents must also flexibly coordinate and communicate, so as to perform robustly despite individual members' incomplete and inconsistent view of the environment, and despite unexpected individual failures. Learning in a team context also remains a difficult challenge — indeed, the precise challenges and possible benefits of such learning remain unclear.

To pursue research challenges such as these and stimulate research in multi-agents in general, the RoboCup research initiative has proposed simulation and robotic soccer as a common, unified testbed for multi-agent research[5] (*www.robocup.org*). The RoboCup initiative has proved extremely popular with researchers, with annual competitions in several different leagues. Of particular interest in this paper is the simulation league, which has attracted the largest number of participants. The stated research goals of the simulation league are to investigate the areas of multi-agent teamwork, agent modelling, and multi-agent learning[6].

Yet, the lessons learned by researchers participating in RoboCup, particularly the simulation league, have largely not been reported in a form that would be accessible to the research community at large. There are just a few notable exceptions[11]. However, extracting such general lessons in areas of teamwork, agent modelling and multi-agent learning is a critical task for several reasons: (i) to meet the stated research goals of the RoboCup effort (at least the simulation league); (ii) to establish the utility of RoboCup and possibly other common testbeds for conducting such research; (iii) to enable future participants to evaluate some of the types of research results to be expected from RoboCup.

This paper attempts to remedy the above situation by extracting the general lessons learned from our experiences with RoboCup. We have fielded two different teams in RoboCup simulation league competitions, ISIS97 and ISIS98, which competed in RoboCup97 and RoboCup98, respectively. ISIS97 won the third place prize in over 30 teams in RoboCup97 (and was also the top US team), while ISIS98 came in fourth in over 35 teams in RoboCup98. As one of the top teams, there is indeed an increased responsibility to report on the general lessons extracted.

Our focus in this paper is not on any one specific research topic, but rather on all aspects of agent and team design relevant to the RoboCup research challenges. Our methodology is one of building the system first, and then attempting to analyze and generalize why it does or does not work. Fortunately, the presence of two RoboCup teams, ISIS97 and ISIS98, often with contrasting design decisions, aids in this analysis. ISIS97 is an earlier and much simpler team compared to ISIS98, but is often able to compensate for its weaknesses in novel ways.

The analysis does reveal several general lessons in the areas of teamwork and multi-agent learning. With respect to teamwork, in the past, we have reported on our ability to reuse STEAM, a general model of teamwork, in RoboCup[13]. This paper takes a step further, evaluating the effectiveness of STEAM in RoboCup, to improve our understanding of the utility of general teamwork models. It also provides an analysis of techniques for the division of team responsibilities among individuals. For instance, compared to ISIS98, ISIS97 agents had relatively little preplanned division of responsibility. Yet, it turns out that via a technique we call *competition within collaboration*, ISIS97 agents compensate for this weakness. A similar situation arises in team monitoring. Compared to ISIS98, ISIS97 agents have a significantly limited capability for maintaining situational awareness or monitoring surroundings. However, ISIS97 agents illustrate that this weakness can be overcome via relying on distributed monitoring. The techniques discovered in ISIS97 were unexpected, and they only became clear when compared with ISIS98. However, they provide an insight into design techniques more suitable for simpler agent teams.

With respect to multi-agent learning, we focused on a *divide-and-conquer* learning approach in designing agents. With this approach, different modules (skills) within individual agents were learned separately, using different learning techniques. In particular, one of the skills, to pick a direction to shoot into the opponents' goal while avoiding opponents, was learned *off-line* using C4.5[8]. Another skill, to intercept the ball, used a mix of *off-line* and *on-line* learning. One of the key surprises here was the degree to which individual agents specialized in their individual roles. Thus, sharing experiences of individuals in different roles or equivalently training individuals by letting them execute different roles would appear to be significantly *detrimental* to team performance. Indeed, this lesson runs contrary to techniques of cooperative learning where experiences are shared among agents.

2 Background: Simulation League

The RoboCup simulation league domain is driven by a public-domain *server* which simulates the players' bodies, the ball and the environment (e.g., the soccer field, flags, etc). Software agents provide the "brains" for the simulated bodies. Thus, 22 agents, who do not share memory, are needed for a full game. Visual and audio information as "sensed" by the player's body are sent to the player agent ("brain"), which can then send action commands to control the simulated body (e.g., kick, dash, turn, say, etc.). The server constrains the actions an agent can take and the sensory information it receives. For instance, with the server used in the 1997 competition, a player could only send one action every 100 milliseconds and received perceptual updates only every 300 milliseconds. The server also simulates stamina: If a player has been running too hard, it gets "tired", and can no longer dash as effectively. Both actions and sensors contain a noise factor, and so are not perfectly reliable. The quality of perceptual information depends on several factors, such as distance, view angle, and view mode (approximating visual focus). All communication between players are done via the server, and are subject to limitations such as bandwidth, range and latencies. Figure 1 shows a snapshot of the soccer server with two competing teams: CMUnited97 [11] versus our *ISIS* team.

In RoboCup97, ISIS97 won the third place prize (out of 32 teams). It won five soccer games in the process, and lost one. In RoboCup98, ISIS98 came in fourth (out of 37 teams). It won or tied seven soccer games in the process, and

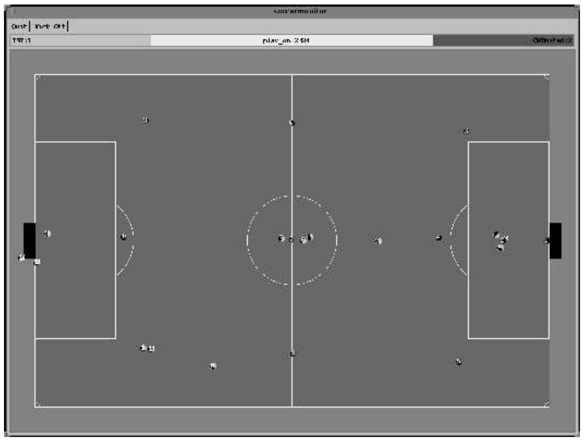

Figure 1: The Robocup synthetic soccer domain.

lost two. Some interesting observations in the tournaments have been that ISIS has never lost a close game. That is, ISIS's wins are either by large goal margins or sometimes by narrow, nail-biting margins (in overtime). However, the three games that ISIS has lost in competitions have been by large margins. Another key observation has been that individual ISIS97 or ISIS98 players have often been lacking in critical skills, even when compared to opponents where ISIS97 or ISIS98 won. For instance, ISIS98 players had no offside skills (a particular soccer skill), yet it won against teams that did check for offside. Thus, teamwork in ISIS appears to have compensated for its lacking skills.

3 The ISIS Architecture

An ISIS agent uses a two-tier architecture. The lower-level, developed in C, processes input received from the simulator, and together with its own recommendations on turning and kicking directions, sends the information up to the higher level. For instance, the lower level computes a direction to shoot the ball into the opponents' goal, and a micro-plan, consisting of turn or dash actions, to intercept the ball.

The lower-level does not make any decisions. Instead, all decision-making rests with the higher level, implemented in the Soar integrated AI architecture[14]. Once the Soar-based higher-level reaches a decision, it communicates with the lower-level, which then sends the relevant action information to the simulator. Soar's operation involves dynamically executing an operator (reactive plan) hierarchy. The operator hierarchy shown in Figure 2 illustrates a portion of the operator hierarchy for ISIS player-agents. Only one path through this hierarchy is typically active at a time in a player agent. The hierarchy has two types of operators: Individual operators represent goals/plans that the player makes and executes as an individual. Team operators constitute activities that the agent takes on jointly as part of a team or subteam and are shown in [].

ISIS97 and ISIS98 share the same general-purpose framework for teamwork modelling, STEAM[13]. STEAM models team members' responsibilities and joint commitments[3] in a domain-independent fashion. As a result, it enables team members to autonomously reason about coordination and communication, improving teamwork flexibility. The [Defend-Goal] team operator demonstrates part of STEAM.[1]

[1]Another part of STEAM deals with team reorganization, which

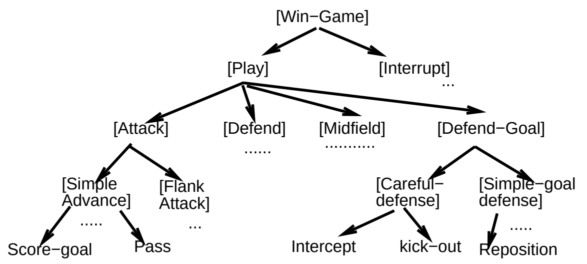

Figure 2: A portion of the operator hierarchy for player-agents in RoboCup soccer simulation. Bracketed operators are team operators, others are individual operators.

It is executed by the goalie subteam. In service of [Defend-Goal], players in this subteam normally execute the [Simple-goal-defense] team operator to position themselves properly on the field and to try to be aware of the ball position. Of course, each player can only see within its limited cone of vision, and can be unaware at times of the approaching ball. If any one of these players sees the ball as being close, it declares the [Simple-goal-defense] team operator to be irrelevant. Its teammates now focus on defending the goal in a coordinated manner via the [Careful-defense] team operator. Specifically this includes intercepting the ball (the Intercept Operator) and then clearing it (the Kick-Out operator). Should any one player in the goalie subteam see the ball move sufficiently far away, it again alerts its team mates (that [Careful-defense] is achieved). The subteam players once again execute [Simple-goal-defense] to attempt to position themselves close to the goal. In this way, agents coordinate their defense of the goal. All the communication decisions are handled automatically by STEAM.

4 Analysis of Teamwork

4.1 Lessons in (Re)using a Teamwork Model

In past work, we have focused on STEAM's reuse in our ISIS teams[13], illustrating that a significant portion (35-45% when measured in terms of the rules) was reused, and that it enabled reduced development time. The use of the teamwork model is a shared similarity between ISIS97 and ISIS98. However, a key unresolved issue is measuring the contribution of STEAM to ISIS's performance. This issue goes to the heart of understanding if general teamwork models can actually be effective.

To measure the performance improvement due to STEAM, we experimented with two different settings of communication cost in STEAM. At "low" communication cost, ISIS agents communicate "normally". At "high" communication cost, ISIS agents communicate no messages. Since the portion of the teamwork model in use in ISIS is effective only with communication, a "high" setting of communication cost essentially nullifies the effect of the teamwork model in execution.

Tables below shows the results of games for the two settings of communication cost, illustrating the usefulness of STEAM. Table 1 compares the performance of the two settings against Andhill97. It shows the goal difference between ISIS97 and Andhill97 was -52 at 15 games (mean -3.46 per game) for "low" cost, and the goal difference was -66 (mean -4.40) for "high" cost. Table 2 shows that the goal difference between ISIS97 and CMUnited97 for "low" was 49 at 15 games (mean 3.26), and it was 26 (mean 1.73) for "high". In both cases, the difference in the means of the goal-differences were seen to be significant under a T-test. That is, STEAM's communication (low cost) helped to significantly improve ISIS's performance in both cases. Thus, general teamwork models like STEAM can not only reduce development overhead, but can contribute to team performance.

is not used in ISIS.

Comm cost	Total games	Total score in goals	Total Goal difference	Mean goal difference
Low	15	4-56	-52	-3.46
High	15	0-66	-66	-4.40

Table 1: STEAM in ISIS97 against Andhill97.

Comm cost	Total games	Total score in goals	Total Goal difference	Mean goal difference
Low	15	50-1	+49	3.26
High	15	31-5	+26	1.73

Table 2: STEAM in ISIS97 against CMUnited97.

4.2 Lessons in Team Monitoring

In designing individual ISIS98 players, we provided them detailed capabilities to locate their own x,y positions on the RoboCup field, as well as the x,y position of the ball. This was an improvement in design over ISIS97, where individuals did not even know their own or the ball's x,y location. That is, ISIS97 players estimated all of these positions heuristically, and often inaccurately. Thus, for instance, individual ISIS97 players on the defender subteam may not know if the ball is far or near the goal. Thus, ISIS98 players were individually more situationally aware of their surroundings. The expectation was that this would lead to a significant improvement in their performance over ISIS97, particularly in those behaviors where situational awareness is important.

The surprise in actual games however was that in behaviors which appeared to require good situational awareness, ISIS97 players appeared to be just as effective as ISIS98 players! A detailed analysis revealed an interesting phenomena: ISIS97 players were compensating for their lack of individual monitoring (and situational awareness) by relying on their teammates for monitoring. In particular, while individuals in ISIS97 were unaware of their x,y locations, their teammates acted as reference points for them, and provided them the necessary information.

Consider for instance the [Careful-defense] team operator discussed earlier. This operator is terminated if the ball is sufficiently far away. As a team operator, it also requires that the defenders inform each other if the ball is sufficiently far away. In ISIS98, players were easily able to monitor own x,y location and ball x,y location, so that they could usually quickly recognize the termination of this operator. In ISIS97, individually recognizing such termination was difficult. However, one of the players in the subteam would just happen to stay at a fixed known location (e.g., the goal). When it recognized that the ball was far away, it would inform the teammates, given its joint commitments in the team operator. Thus, other individuals, who were not situationally well-aware, would now learn about the termination of the team operator. (This technique failed if the player at the fixed location moved for some reason.)

The key lesson to take away is that in a multi-agent system, there is a tradeoff in monitoring. One approach is to design an agent, with complex monitoring capabilities, that is situationally well-aware of its surroundings. Another is to design a much simpler monitoring agent, but rely on teammates to provide the necessary information. In the first case, agents are more independent, while in the second case, they must rely on each other, and behave responsibly towards each other.

Another key lesson is that the design of a team's joint commitments (via team operators) has a significant impact on how individual skills may be defined. For instance, given the definition of [Careful-defense], with its accompanying joint commitments, individual players need not be provided complex monitoring capabilities. Similarly, definition of individual skills should impact the design of a team's joint commitments. Thus, for instance, for ISIS98 players, given their individual situational awareness, the commitments in [Careful-defense] to inform each other when the ball is far or near may not be as useful.

4.3 Lessons in Designing Role Responsibilities

In teamwork, role responsibilities are often designed so as to achieve load balancing among individuals, and to avoid conflicts among them. With these goals in mind, when defining roles for ISIS98 players, we provided them detailed, non-overlapping regions in which they were responsible for intercepting and kicking the ball. Essentially, each player was responsible for particular regions of the field. Furthermore, these regions were flexible. The players would change regions if the team went from attack mode to defense mode, i.e., if the ball moved from the opponent's half to own half. This ISIS98 design was a significant improvement over our earlier ISIS97 design. There, players have regions of the fields that are their responsibility, but the division is very relaxed with considerable overlap. So effectively, multiple players will share the responsibility of defending a specific area of the field, and thus could conflict, for instance, by getting in each other's way.

Again, the expectation was that ISIS98 would perform significantly better than ISIS97, given that ISIS98 had a carefully laid out, but flexible, plan for division of responsibilities. This division of responsibility was intended to have an additional side-effect of an overall conservation of stamina, which was particularly desirable because stamina was a more critical issue in RoboCup98.

The surprise when we played ISIS97 and ISIS98 against a common opposing team was that ISIS98 was not outperforming ISIS97 as expected. The analysis revealed that ISIS97 managed to attain a reasonable division of responsibilities, by hitting upon a style of play that can be characterized as *competition within collaboration*. Essentially, multiple players in ISIS97 may chase after the ball, competing for opportunities to intercept the ball. Players that were out of stamina (tired), players that got stuck, lost sight of the ball etc., would all fall behind. Thus, the ISIS97 player that was best able to compete (i.e., get close to the ball first), would get to kick the ball. This technique in some cases attained a more dynamic load-balancing, when compared to the planned division of responsibilities in ISIS98. For instance, in ISIS98, a player, even if very tired, would still have to continue to assume responsibility for its region. In ISIS97, that player would be unable to get to the ball, and another one with more stamina would take control.

This competition in ISIS97 arises because the responsibility for intercepting the ball is not explicitly modeled as a team operator and each individual thereby makes the decision to intercept the ball on their own. The price of this competition is that more individual agents may waste their resources chasing after the same opportunity. Another important item is that in ISIS98, the agents followed the roles designed by the human. In ISIS97, the agents' behavior was more unpredictable, as they were not following particular role definitions.

One key lesson learned is the contrasts among the techniques for role responsibility design, which bring forth some novel tradeoffs. In particular, for simpler teams, the technique of *competition within collaboration*, would appear to be a reasonable compromise that does not require significant planning of division of responsibilities.

5 Analysis of Learning

We focused on a *divide-and-conquer* learning approach in designing agents. With this approach, different modules (skills) within individual agents were learned separately, using different learning techniques. To date, learning has been applied to (i) learning of goal shots, to shoot when attempting to score a goal (using C4.5) and (ii) selection of a plan to intercept an incoming ball (using reinforcement learning).

5.1 Offline Learning of Goal Shots

Scoring goals is a critical soccer skill. However, our initial hand-coded, approaches to determining a good direction to kick the ball, based on heuristics such as "shoot at the center of the goal", or "shoot to a corner of the goal", failed drastically. In part, this was because heuristics were often foiled by the fact that small variations in the configuration of players around the opponent's goal or a small variation in the shooter's position may have dramatic effects on the right shooting direction.

To address these problems, we decided to rely on automated, *offline* learning of the shooting rules. A human expert created a set of shooting situations, and selected the optimal shooting direction for each such situation. The learning system trained on these shooting scenarios. C4.5[8] was used as the learning system, in part because it has the appropriate expressive power to express game situations and can handle both missing attributes and a large number of training cases.

In our representation, each C4.5 training case has 39 attributes, such as the recommended kicking direction, the shooter's facing direction, and the shooter's angles to the other visible players, the 12 flags, the 4 lines, the ball, and the opponent's goal. The system was trained on over roughly 1400 training cases, labeled by our expert with one of UP, DOWN, and CENTER (region of the goal) kicking directions. The decision was based on actually having the ball be kicked in each direction with a fixed velocity and judging which shot was best, factoring in the other players' fixed location and the randomizations associated with kicking (e.g., wind).

The result was that given a game situation characterized by the 39 attributes, the decision tree selected the best of the three shooting directions. The resulting decision tree provided a 70.8%-consistent set of shooting rules. These learned rules for selecting a shooting direction were used successfully in RoboCup'97.

The C4.5 rules were a dramatic improvement over our original hand-coded efforts. However, there were still cases under actual playing conditions where the shooting direction calculated by these rules seemed inappropriate. Particularly, in some cases, the C4.5 rules would attempt very risky shots on the goal, when a more clear shot seemed easily possible. The reason this occurred was that offline learning was done using the human expert's labeling which was based on assumptions about opponents' level of play in RoboCup matches – the expert tended to assume the worse. However, in practice, especially against weaker teams, easy opportunities appeared to have been thrown away by taking some unnecessary risks.

Thus, one key lesson learned here is was that it may be possible to approach the agent design problem via a divide-and-conquer learning technique. Another key lesson is that off-line learning in dynamic multi-agent contexts must be sensitive to the varying capabilities of other agents.

5.2 Online Learning of Intercepts

Intercepting the ball is another critical basic skill. However, it is not a simple, static skill. Whether in human soccer or RoboCup, there are many external and internal playing conditions that can impact a player's intercept. The opposing side may kick/pass/run harder than normal, thereby requiring a player to run harder, modify the path they take or forgo interception. Properties of the ball's motion or visibility can also dramatically impact play. Human players, at least, fluidly adapt to these conditions. However, unlike real soccer players, our ISIS players' intercept skills were not adapting very well to differing internal and external factors.

One could address this problem by a precise re-engineering approach, by using all of the parameters available from the server, and then trying to precisely handcode the intercept. We have taken a different approach, driven by the question: what would happen if players in ISIS98 are allowed to learn plans themselves, and what would that learning tell us? In particular, would there be differences in what is learned across different players? Would there be differences across different opponents? We therefore pursued a reinforcement learning approach [12, 4] to enable players to adapt their intercept online, under actual playing conditions using just the perceptual information provided by the server to the player: the ball's current direction, change in direction, distance.

Although our concern is more on what is learned online as opposed to how it is learned, any approach to the online learning of intercept must deal with several difficulties. One key difficultly revealed in applying reinforcement learning is that in the course of a game, there are not many opportunities to intercept the ball. Furthermore, even within those opportunities, an agent is often unable to carry through the full intercept, since other players may happen to kick the ball, or the ball may simply go outside the field, etc. Whereas that suggests the need for rapid adaptation, it is also the case that inappropriate adaptations can have dire consequences.

To address these concerns, it was important to design intermediate reinforcement, occurring as an intercept plan was in progress and not just when the plan completed. Specifically, ISIS98 uses the same simple intercept micro-plan structure used in ISIS97. A player intercepts the ball by stringing together a collection of micro-plans, where each micro-plan consists of a turn followed by one or two dashes. For every step in a micro-plan, ISIS98 has an expectation as to what any new information from the server should inform it as to the ball's location. Failure to meet that expectation results in a learning opportunity. To allow transfer to similar states, the input conditions are clustered. Repeated failures lead to changes in the micro-plan assigned to an input condition. In particular, the turn increment specific to that input condition is adjusted either up or down upon repeated failure. For most input conditions, the actual turn is calculated from the turn increment in the following fashion:

$$Turn = BallDir + (TurnIncrement * ChangeBallDir)$$

5.2.1 Online Learning Experiments

We have performed several preliminary experiments in which ISIS made online adjustments to the turn increment factor. These experiments involved six extended length games between ISIS and two other teams, CMUnited97 (team of Stone and Veloso of Carnegie Mellon) and Andhill97 (team of T. Andou of NTT labs). They have illustrated several interesting trends. In each experiment, each player started with a default value of 2.0 for their turn increment across all input conditions. The learning could result in turn increment values that range from +5 down to -1, across input conditions. While these may appear to be small numbers, because of the multiplicative factors, and because the intercept plan is invoked repeatedly, these changes are overall very significant.

The results we observed were that for any particular input condition, the trend in the learning seems uniform in its direction across teams and positions. Against these two teams, if one player playing Andhill97 increased the turn increment under a certain input condition then all the players with sufficient training examples would tend to show an increase whether in games with Andhill97 or CMUnited97. There however can be striking differences in magnitude. Below, we consider two illustrative examples.

Test 1: Same player against different Teams

In particular, lets consider the case of what a player in a particular position learns while playing a game against CMUnited97 as opposed to what the same player learns playing against Andhill97. In Figure 3, the mean results for Player 1, a forward, are graphed against the mean for all players, at 3000 ticks of the game clock until the end of the game at 15000 (RoboCup games normally run to 6000, here the time has been lengthened to simplify data collection). This particular data is for the input condition of balls moving across the player's field of vision, a middling-to-close distance away. Figure 3 shows that against Andhill97, the player is learning a turn increment similar to the mean across all players for this input condition. However, against CMUnited97, the player is learning a much larger increment. The difference between the means for CMUnited97 and Andhill97 at 15000 ticks is significant (under a t-test).

Test 2: Different Players Against Same Team

It is also the case that different players against the same team do learn different increments. Consider Figure 4. It plots mean turn-increments for Player 1 and Player 10 (a fullback) for the the same input condition as above, against CMUnited97. The differences in the means are significant.

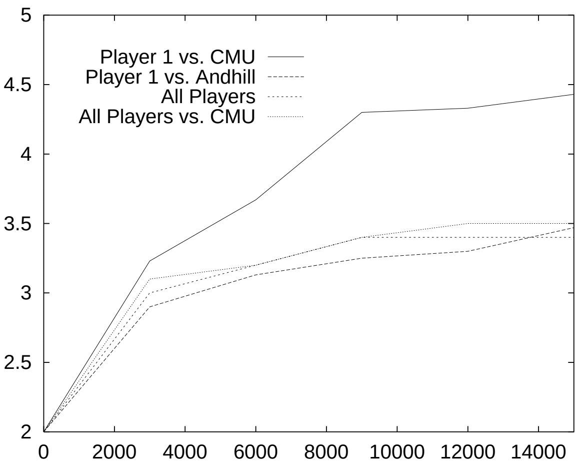

Figure 3: Contrast of Player 1 versus CMUnited97 with Player 1 versus Andhill97 under same input condition. Means across all players provided for comparison. Player 1 is a forward.

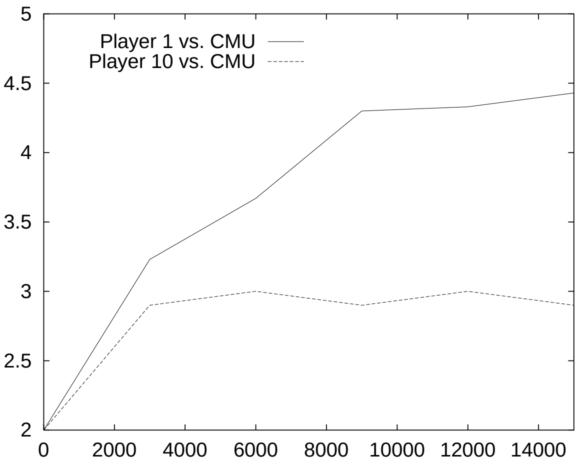

Figure 4: Contrast of Player 1 versus CMUnited97 with Player 10 versus both CMUnited97, under the same input condition. Player 1 is a forward and Player 10 is a goal defender.

5.2.2 Lessons Learned

The key here is that Player 1 distinctly tailors its intercept to its role and its particular opponents. There is a domain level analysis, which clarified why player 1 had specialized its role so significantly — CMUnited97's defenders often cleared the ball with a strong sideways kick, which player 1 continuously faced. However, there is a larger point. These results argue for the specialization of skills according to both role and the specific conditions under which the skill is exhibited. Thus, sharing experiences of individuals in different roles or equivalently training individuals by letting them execute different roles would appear to be *detrimental* to an agent's performance.

While the magnitude differed significantly, the trends of the changes were shared across players. This suggests there is still benefit to social learning, or cross-agent communication of learning experiences. In particular, in the case of goalees, which do not get as many chances typically to intercept the ball during the game, we have found it particularly useful to transfer mean values from the other players. However, the key is to recognize that there are interesting limits to such social learning. Hence, learn socially, but with real caution!

6 Related Work

As outlined earlier, we as RoboCup researchers, particularly in the simulation league, have not communicated our research in general terms to the multi-agent or agent community at large (there are some notable exceptions).

Within RoboCup-based investigations, ISIS stands alone with respect to investigation of a general, domain-independent teamwork model to guide agents communication and coordination in teamwork. Some researchers investigating teamwork in RoboCup have used explicit team plans and roles, but they have relied on *domain-dependent* communication and coordination. A typical example includes work by Chng and Padgham[2]. They present an elaborate analysis of roles in motivating teamwork and team plans. In this scheme, agents dynamically adopt and abandon roles in pre-defined tactics. The responsibilities and actions of each agent are determined by its current role in the current plan. Unlike ISIS agents, whose team-related responsibilities are part of the general domain-independent STEAM model, Chang and Padghams roles include both team- level responsibilities as well as personal responsibilities — and so there is no separation of the domain-dependent from the domain-independent responsibilities. A similar scheme is used by Stone and Veloso [10]. They offer an approach to managing flexible formations and roles within those formations, allowing agents to switch roles and formations dynamically in a domain-dependent manner. Their agents synchronize their individual beliefs periodically in a fixed manner, in contrast with ISIS's STEAM in which communications are issued dynamically and can be parameterized based on the domain of deployment. Another RoboCup effort focusing on explicit team plans and roles is [1]. They define levels of teamwork, starting at basic roles (which are static throughout the game), and building on top of those with formations and team plans for carrying out more complex tactics. These teamwork levels determine the agents coordination responsibilities and prioritize its actions. Other investigations of teamwork in RoboCup have used implicit or emergent coordination. A typical example is Yokota et al.[16].

Our application learning in ISIS agents is similar to some of the other investigations of learning in RoboCup agents. For instance, Luke et al.[7] use genetic programming to build agents that learn to use their basic individual skills in coordination. Stone and Veloso[11] present a related approach, in which the agents learn a decision tree which enables them to select a recipient for a pass.

7 Lessons Learned from RoboCup

Challenges of teamwork and multi-agent learning are critical in the design of multi-agent systems, and these are two of the critical research challenges of the RoboCup simulation league. As participants in the RoboCup competitions, it is critical that researchers extract general lessons learned, so as to meet the goals of the RoboCup research initiative. This is what we have attempted in this paper.

Our research in RoboCup began with the foundation of a general model of teamwork, STEAM. Using STEAM, ISIS can operate flexibly in the highly dynamic environment of

RoboCup. The fact that STEAM has served the research well has been demonstrated both empirically and in the pressure of the RoboCup97 and RoboCup98 competitions. Here are some of the key lessons learned via our analysis of ISIS97 and ISIS98:

- Reuse of general teamwork models can lead to improved performance.
- Interesting tradeoffs exist in individual and team situational awareness (monitoring) in multi-agent systems. In particular, responsible team behavior enables the design of simpler situational awareness (monitoring) capabilities for individuals.
- *Competition within collaboration* can provide a simple but powerful technique for designing role responsibilities for individuals.
- Divide-and-conquer learning can be used to enable different learning techniques to co-exist and learn different skills in designing individual agents. This can reduce the complexity of the learning problem.
- Some multi-agent environments can lead to a significant role specialization of individuals. Thus, sharing experiences of individuals in different roles or equivalently training individuals by letting them execute different roles can sometimes be significantly *detrimental* to team performance. That is, there has to be a check on social learning.
- For the human designer, outside of the multi-agent environment, it is often very difficult to comprehend the agents' experiences inside the environment and therefore difficult to design for those experiences.
- RoboCup simulations are capable of providing a surprise.

Acknowledgment

This research is supported in part by NSF grant IRI-9711665, and in part by a generous gift from the Intel Corporation.

References

[1] T. F. Bersano-Begey, P. G. Kenny, and E. H. Durfee. Agent teamwork, adaptive learning, and adversarial planning in robocup using a prs architecture. In *RoboCup-97: The first robot world cup soccer games and conferences*. Springer-Verlag, Heidelberg, Germany, 1998.

[2] S. Ch'ng and L. Padgham. Team description: Royal merlbourne knights. In *RoboCup-97: The first robot world cup soccer games and conferences*. Springer-Verlag, Heidelberg, Germany, 1998.

[3] P. R. Cohen and H. J. Levesque. Teamwork. *Nous*, 35, 1991.

[4] T. Dean, K. Basye, and J. Skewchuk. Reinforcement learning for planning and control. In *Machine Learning Methods for Planning*, pages 67–92. Morgan Kaufman, San Francisco, 1993.

[5] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international conference on autonomous agents*, 1997.

[6] H. Kitano, M. Tambe, P. Stone, S. Coradesci, H. Matsubara, M. Veloso, I. Noda, E. Osawa, and M. Asada. The robocup synthetic agents' challenge. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, August 1997.

[7] S. Luke, Hohn C., J. Farris, G. Jackson, and J. Hendler. Co-evolving soccer softbot team coordination with genetic programming. In *RoboCup-97: The first robot world cup soccer games and conferences*. Springer-Verlag, Heidelberg, Germany, 1998.

[8] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[9] A. S. Rao, A. Lucas, D. Morley, M. Selvestrel, and G. Murray. Agent-oriented architecture for air-combat simulation. Technical Report Technical Note 42, The Australian Artificial Intelligence Institute, 1993.

[10] P. Stone and M. Veloso. Task decomposition and dynamic role assignment for real-time strategic teamwork. In *Proceedings of the international workshop on Agent theories, Architectures and Languages*, 1998.

[11] P. Stone and M. Veloso. Using decision tree confidence factors for multiagent control. In *RoboCup-97: The first robot world cup soccer games and conferences*. Springer-Verlag, Heidelberg, Germany, 1998.

[12] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[13] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research (JAIR)*, 7:83–124, 1997.

[14] M. Tambe, W. L. Johnson, R. Jones, F. Koss, J. E. Laird, P. S. Rosenbloom, and K. Schwamb. Intelligent agents for interactive simulation environments. *AI Magazine*, 16(1), Spring 1995.

[15] M. Williamson, K. Sycara, and K. Decker. Executing decision-theoretic plans in multi-agent environments. In *Proceedings of the AAAI Fall Symposium on Plan Execution: Problems and Issues*, November 1996.

[16] K. Yokota, K. Ozako, Matsumoto A., T. Fujii, Asama H., and I. Endo. Cooperation towards team play. In *RoboCup-97: The first robot world cup soccer games and conferences*. Springer-Verlag, Heidelberg, Germany, 1998.