Two Fielded Teams and Two Experts: A RoboCup Challenge Response from the Trenches

Milind Tambe, Gal A. Kaminka, Stacy Marsella, Ion Muslea, Taylor Raines
Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292, USA
{tambe,galk,marsella,muslea,raines}@isi.edu

Abstract

The RoboCup (robot world-cup soccer) effort, initiated to stimulate research in multi-agents and robotics, has blossomed into a significant effort of international proportions. RoboCup is simultaneously a fundamental research effort and a set of competitions for testing research ideas. At IJCAI'97, a broad research challenge was issued for the RoboCup synthetic agents, covering areas of multi-agent learning, teamwork and agent modeling. This paper outlines our attack on the entire breadth of the RoboCup research challenge, on all of its categories, in the form of two fielded, contrasting RoboCup teams, and two off-line soccer analysis agents. We compare the teams and the agents to generalize the lessons learned in learning, teamwork and agent modeling.

1 Introduction

Increasingly, multi-agent systems are being designed for a variety of complex, dynamic domains. To stimulate and pursue research towards such multi-agent systems, the *RoboCup* initiative has proposed simulation and robotic soccer as a common, unified domain for multi-agent research[Kitano *et al.*, 1997]. RoboCup has now blossomed into a significant effort of international proportions.

At IJCAI'97, a broad research challenge was issued for the RoboCup synthetic agents[Kitano *et al.*, 1997]. This paper responds to this challenge, in the form of research lessons drawn from several systems we have constructed for RoboCup. In particular, we fielded the ISIS97 and ISIS98 teams, which won third place and fourth place at RoboCup97 and RoboCup98 respectively (out of 30 to 35 participating teams). We have also constructed two experts, ISAAC and TEAMORE, for off-line review of RoboCup. Our response draws from these multiple systems for two reasons. First, the RoboCup challenge covers a broad spectrum of multi-agent research, and requires teams and off-line experts to be built. Indeed, it proposes three separate challenge areas, *learning, teamwork* and *agent modeling*. Second, these challenge areas often do not have just one right answer, rather, they point to tradeoffs, which we explore via multiple systems.

Our challenge response also attempts to extract general lessons from RoboCup. Indeed, despite the RoboCup aim to stimulate general multi-agent research, few RoboCup researchers have extracted domain-independent research lessons (there are a few notable exceptions[Stone and Veloso, 1998b]). This paper attempts to remedy this situation.

2 Background: Domain and Agents

The RoboCup simulation league uses a complex, dynamic, noisy soccer simulation, called the *soccerserver*, which simulates the players' (22) bodies, the ball and the soccer field with goals and flags. Software agents (11 agents per team) provide the "brains" for the simulated bodies. 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 an agent's actions (one action per 100ms) and sensory updates (one perceptual update every 150-300ms). The players also have limited stamina.

The software agents we constructed to control the player bodies are based on a two-tier architecture. The lower-level, developed in C, processes input received from the simulator, and together with recommendations of an intercept microplan and possible kicking directions, sends the information up to the higher-level. The higher-level is implemented in the Soar integrated architecture[Newell, 1990]. Soar uses the information it receives to reach a decision about the next action and communicates its decision to the lower-level, which then forwards the relevant action to the simulator. Soar's operation involves dynamically executing an operator (reactive plan) hierarchy. The operator hierarchy shown in Figure 1 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. Team operators constitute activities that the agent takes on as part of a team or subteam, shown in [] (e.g., [Play]). In contrast, the "normal" individual operators are ones that players execute as individuals (e.g., Intercept). The implication of this distinction will be clarified later.

3 Response to the Learning Challenge

ISIS teams have addressed the problems of off-line skill learning and on-line adversarial learning, with results used in actual

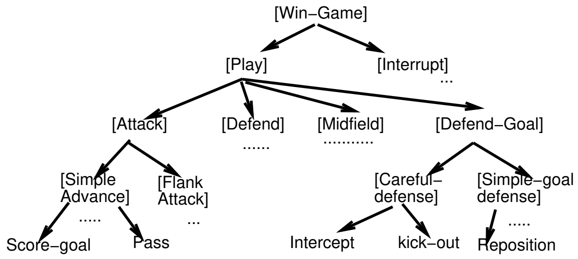

Figure 1: Operator hierarchy for player-agents.

competitions. Different learning algorithms are integrated in ISIS via a *divide-and-conquer* approach, i.e., different modules (skills) within individual agents are learned separately.

3.1 Offline Skill Learning

Shooting a ball to score a goal is clearly one of the critical skills in soccer. Yet, our initial, heuristic, hand-coded approaches (e.g., shoot to a corner of the goal) failed, because (i) small variations in shooter-position sometimes had dramatic effects on the best shooting direction and (ii) large number of heuristic rules were needed.

We addressed these problems via automated, *off-line* learning of the shooting rules. A human specialist created a set of 3000 shooting situations (each of them labeled with an optimal shooting direction: UP, DOWN, and CENTER) that were used as training examples for C4.5[Quinlan, 1993]. Each such shooting scenario was used as a training case described by 40 attributes: the recommended kicking direction, the shooter's facing direction, and the shooter's angles to the visible players, flags, lines, ball, and goal. The system was trained on 1600 randomly chosen examples, and the other 1400 examples were used for testing. We repeated this procedure 50 times, and the average accuracy of the rules on the testing sets was 70.8%. Even though the predictive power appears low, the kicking rules were quite efficient in practice. This is because learned-rules covered far more difficult shots than were actually used in practice.

While the C4.5 learned rules dramatically improved shooting skills, in the competitions, the rules sometimes appeared to take unnecessarily risky shots on the goal. This occurred because offline learning assumed the worst about the opponents' level of play, while in practice, weaker teams provided easier opportunities that did not justify such risks. Thus, while one key lesson learned here is that a divide-and-conquer learning technique may be promising for agent design, another key lesson is that off-line learning in dynamic multi-agent contexts must be sensitive to the varying capabilities of other agents.

3.2 Online Adversarial Learning

A key skill in RoboCup where adaptation to the opponent is critical is that of intercepting the ball. In particular, an opponent may kick/pass/run harder than normal, thereby requiring a player to adapt by running harder, modifying their path or forgoing interception. To enable players to adapt their intercept online to adversaries, ISIS exploits reinforcement learning.

One key difficulty in applying reinforcement learning however is rapid adaptation — in the course of a game, there are not many opportunities to intercept the ball. To address this concern, our approach employs intermediate reinforcement, rather than waiting for the end of the intercept. A player intercepts the ball by stringing together a collection of micro-plans 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 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. Typically, the actual turn is calculated from the turn increment in the following fashion:

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

Experiments: In accordance with the IJCAI challenge, experiments were performed against publicly available teams, specifically CMUnited97 (team of Stone and Veloso of Carnegie Mellon, 4th at RoboCup'97) Andhill97 (team of T. Andou of NTT labs, 2nd at RoboCup'97). In each experiment, each player started with a default value of 2.0 for their turn increment across all input conditions. The online learning in these games results in turn increment values that range from +5 down to -1, across input conditions. While these may appear small numbers, because of the multiplicative factors, and since the intercept plan is invoked repeatedly, even a small change is overall very significant.

The results show some surprising differences in what is learned. For instance, the same player may learn very different turn increments against different teams. Figure 2 compares the mean results for Player 1, a forward, in games against CMUnited97 with games against Andhill97. The mean for all players is also shown. The x-axis plots the clock ticks (continued until 15000) and the y-axis plots the turn increment. This data is for the input condition of balls moving across the player's field of vision, a middling-to-close distance away. 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 considerably larger increment (difference in means is significant using a Welch two-sided t-test, p-value=.0447). Figure 3 shows that different players against the same team do learn different increments. It plots mean turn-increments for Player 1 and Player 10 for the the same input condition as above, against CMUnited97. The difference in the means is significant (using a Welch two-sided t-test, p-value = 6.36e-06)

Lessons learned: Player 1 distinctly tailors its intercept to its role and particular opponents. This occurs because CMUnited97's defenders often clear the ball with a strong sideways kick, which player 1 continuously faces. Player 1's adaptation not only illustrates the benefits of on-line learning, but also a general point: it shows a high specialization of (intercept) skills according to the role and situations faced. Thus, sharing experiences of individuals in different roles or training individual across roles would appear to be detrimental, i.e., *there are key limits to social learning*. Of course, it does not rule

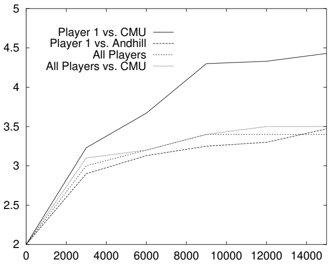

Figure 2: Player 1 against CMUnited97 & Andhill97.

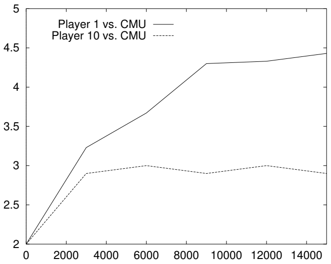

Figure 3: Players 1 & 10 against CMUnited97.

out social learning. Indeed, in the results above, the trends of the changes were shared across players, so that some social learning can be carried out. Thus, our goalee agents, which tend not to get as many intercept opportunities during the game, rely on mean intercept values from the other players.

4 Teamwork Challenge Response

The teamwork challenge covers team planning, plan decomposition, execution, etc. There is not necessarily one best answer to this challenge, and our two RoboCup teams present at least some of the tradeoffs.

4.1 Team Plan Execution

Our response to the team plan execution challenge is a key distinction of our ISIS teams — the use of a general-purpose teamwork model called STEAM[Tambe, 1997]. Based on the notion of joint commitments[Cohen and Levesque, 1991], STEAM enables team members to autonomously reason about coherence during team plan execution. It also enables team reorganization upon disablement of a team member and selective communication. STEAM's reasoning is instigated by a (sub)team's execution of a team operator. For an example of STEAM in operation, consider the SIMPLE-DEFENSE team operator executed by the goalee subteam to position themselves on the field and watch out for the ball (Fig 1). Each player sees only 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-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 and then clearing it. Should any one player in the goalee subteam see the ball move sufficiently far away, it again alerts its team mates (that CAREFUL-DEFENSE is achieved). The subteam players once again revert to SIMPLE-DEFENSE. All the communication decisions for the subteam coordination here are handled automatically by STEAM.

For teamwork, one evaluation criteria in [Kitano *et al.*, 1997] is generality, i.e., reuse of the teamwork capability across applications. STEAM was originally used in battlefield simulations[Tambe, 1997], and its generality is illustrated in its reuse in RoboCup. We may measure this reuse in terms of the number of STEAM rules reused. STEAM originally had 283 rules, of which 35%-45% are used in ISIS. Without STEAM reuse, communication in ISIS would have required dozens of domain-specific coordination plans.

A second evaluation criteria is general performance. To this end, we measure impact of STEAM on ISIS97, by experimenting with different settings of communication cost in STEAM. In particular, at "low" communication cost, ISIS97 agents communicate a significant number of messages, while at "high" communication cost, ISIS agents communicate no messages. Since the portion of STEAM in use in ISIS is effective only with communication, a "high" communication cost essentially nullifies the effect of STEAM. Table 1 below shows the results of games for the two settings of communication cost, illustrating the usefulness of STEAM. It compares the performance of the two settings against Andhill97 and CMUnited97 in approximately 60 games. It shows that the mean goal difference between ISIS97 and Andhill97 was -3.38 per game for "low" cost, and was -4.36 per game for "high" cost. This difference in the means is significant using a t-test (null hypothesis p=0.032). It also shows a similar comparison for 30 games between ISIS97 and CMUnited97. It shows that the mean goal difference between ISIS97 and CMUnited97 for "low" was 3.27, and was 1.73 for "high" (again, using a t-test, p=0.022). Thus in both cases, STEAM's communication (low cost) helped to significantly improve ISIS's performance.

Comm cost	Mean goal difference against Andhill97	Mean goal difference against CMUnited97
Low	-3.38	3.27
High	-4.36	1.73
p(null hypo)	0.032	0.022

Table 1: ISIS97: Mean goal difference with/without STEAM.

4.2 Team Monitoring Challenge

In response to the team monitoring challenge (part of the team plan execution challenge), we contrast ISIS98 with ISIS97. Our individual ISIS98 players very precisely monitored their own and the ball's x,y positions on the RoboCup field. In contrast, ISIS97 players only approximately (and often inaccurately) estimated their own or the ball position (without x,y). Thus, ISIS98 players were individually more situationally aware, and were expected to outperform ISIS97 players.

The surprise: In actual games (e.g., against CMUnited97) however, ISIS97 players appeared to be as effective as ISIS98 players. Our analysis revealed that ISIS97 players were compensating for their lack of individual monitoring by relying on their teammates. Consider for instance the CAREFUL-DEFENSE team operator discussed earlier. This operator is terminated if the ball is sufficiently far away. In ISIS97, without x,y locations, 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), acting as a reference. When it recognized that the ball was far away, it would inform the teammates, as per its joint commitments in the team operator. Thus, other players, who were not situationally well-aware, would now know the ball is far away. In contrast, ISIS98 players, with x,y computations, would individually quickly recognize the termination of this operator.

Table 2 shows the means of goal differences for ISIS98 with differing communication costs and different opponents (over 170 games against CMUnited97, 60 against Andhill97). STEAM's communication ("low" communication cost) does not provide a statistically significant improvement over no-communication (using a two-tailed t-test). This indicates decreased reliance on communication among teammates, and contrasts with results for ISIS97 from Table 1.

Comm cost	Mean goal difference against Andhill97	Mean goal difference against CMUnited97
Low	-1.53	4.04
High	-2.13	3.91
p(null hypo)	0.58	0.13

Table 2: Impact of STEAM in ISIS98.

Thus, the response to the team monitoring challenge is the discovery of a general tradeoff: one monitoring approach provides individual agents with complex monitoring capabilities, making them situationally well-aware and hence independent of others (for monitoring). Another approach provides simpler monitoring capabilities to agents, but they must now rely on teammates to compensate for the lack of own capabilities.

4.3 Plan Decomposition Challenge

The RoboCup challenge of team plan decomposition focuses on designing *roles* for individual agents in a team. Ideally, roles should divide the team responsibilities fairly, avoid conflicts, and conserve resources by avoiding redunducies. Indeed, in ISIS98, these factors led to players' roles being defined in terms of non-overlapping regions of the soccer field, in which they were responsible for intercepting and kicking the ball. These regions were flexibly changed, if the team went from attack to defense mode. In contrast, in ISIS97, players' roles (also defined in terms of regions), had a significant overlap, possibly wasting stamina. Thus, the role non-overlap plan decomposition of ISIS98 was expected to be significantly superior to the *role overlap* style of ISIS97.

The surprise: When we played ISIS97 and ISIS98 against CMUnited97, however, ISIS97 was not outperformed as expected. In particular, ISIS97 managed to attain a reasonable division of responsibilities, via *competition within collaboration*. Essentially, multiple players in ISIS97 would chase the ball, competing for opportunities to intercept the ball. Players that were out of stamina, or those that lost sight of the ball etc., would all fall behind, and the player best able to compete (i.e., get close to the ball first) would get to kick the ball.

Thus, a key lesson is tradeoff in role design: a flexible, role no-overlap design reduces conflicts, conserves resources, but requires careful off-line role planning. It can also fail in dynamic load balancing, e.g., an ISIS98 player, even if very tired, is still solely responsible for its region. In contrast, ISIS97's role overlap can exploit *competition within collaboration* to more autonomously plan its role division, and attain more dynamic load balancing, e.g., if a player is tired, a teammate with more stamina will get to the ball quicker. However, role overlap may waste resources, due to redundant actions.

5 Agent Modeling Response

The agent modeling area provides a key difficult RoboCup challenge: *off-line review* by an expert to analyze teams. We have constructed two contrasting agents in response to this challenge. Both agents use a domain-independent approach that avoids the encoding of extensive domain knowledge and rely instead on extensive data-mining. These agents are thus collaborative *assistants*, relying on the "knowledge-rich" human observer to complete the analysis. Since both agents rely on data-mining, they excel at uncovering unexpected phenomena. Within the complexity of the RoboCup environment, these off-line review agents appear capable of capturing novel regularities that escape unaided human observers.

5.1 Off-line Expert Agent

ISAAC is a web-based off-line soccer expert (Fig. 4, http://coach.isi.edu). It is focused on automated analysis to aid in improving a team's behavior. ISAAC approaches the problem by investigating actions that did not produce the desired result and then classifying the contexts in which failures occured. Based on that classification, ISAAC recommends changes in behavior to avoid the failures: either the team should perform a different action in that context or should perform the action in a modified context. More specifically, ISAAC's analysis starts with logs of a particular team's games. From the logs, ISAAC extracts interesting behaviors and the outcomes of those behaviors. For instance, shots on own or opponent goals are interesting behaviors, so ISAAC gathers data on such shots, and whether they succeed. ISAAC then classifies these successes and failures into subclasses with similar contexts. For instance, a subclass might be all goal shots with a near-by opponent, that fail. Currently, C4.5 is used to induce these subclasses by generating rules that classify the successes and failures of the shooting team.

ISAAC's next step is to formulate suggestions that may improve the team's performance, once again, using a knowledge-lean approach. To that end, ISAAC formulates and analyzes perturbations of the rules. Each rule consists of a number of conditions that must be satisfied for the rule to be valid. We define a perturbation to be the rule that results from reversing one condition. Thus a rule with N conditions will have N perturbations. The successes and failures governed by the

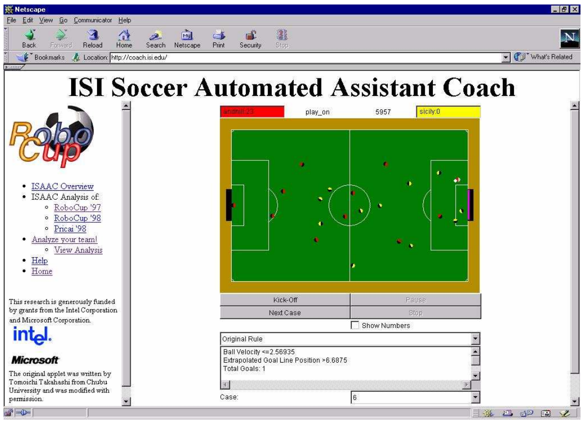

Figure 4: ISAAC exists on the web.

perturbations of a rule are examined to determine which conditions have the most effect in changing the outcome of the original rule, turning a failure into a success.

Let us consider a simple example of ISAAC's analysis of Andhill97 against other teams in the RoboCup '97 tournament. One of ISAAC's learned rules states that when taking shots on goal, the Andhill97 team often fails to score when (i) ball velocity is less than 2.57 meters per time tick and (ii) the shot is aimed at greater than 6.7 meters from the center of goal (which is just inside the goalpost). ISAAC reveals that shots governed by this rule score once, and fail to score 70 times. That Andhill97, the 2nd place winner of 97 had so many goal-shot failures, and that poor aim was at least a factor was a surprising revelation to the human observers. The user can review the "video" of these shots on goal in ISAAC's log monitor to better appreciate what is occurring in these cases. Perturbations of this rule can also be considered. In cases where the rule is perturbed so that ball velocity is greater than 2.57m/t and the shot aim is still greater than 6.7m, Andhill scores once and fails to score 3 times. In another perturbation, where ball velocity is again less than 2.57m/t, but now shot aim is equal to or less than 6.7m (see Figure 5), Andhill is now scoring 63 times and failing to score 106 times. These perturbations suggest that improving Andhill97's shot aiming capabilities can significantly improve performance.

More recently, ISAAC has been extended to analyze sequences of behaviors (again using C4.5), such as sequences of actions (e.g., assists) that lead up to successes or failures. This analysis has revealed that out of the top four teams of RoboCup'97, ISIS is at one extreme with little or no emergent pattern of assists, while CMUnited shows deeper patterns of assists and secondary assists.

5.2 Teamwork Review Agent

TEAMORE (or TEAmwork MOnitoring REview) is an agent that performs off-line review via a contrast of the behaviors of the agents in a team. TEAMORE is based in *socially-attentive monitoring*, originally applied in battlefield simulations[Kaminka and Tambe, 1998], underscoring its generality. In RoboCup, TEAMORE complements the use of goals scored as a measure of teamwork.

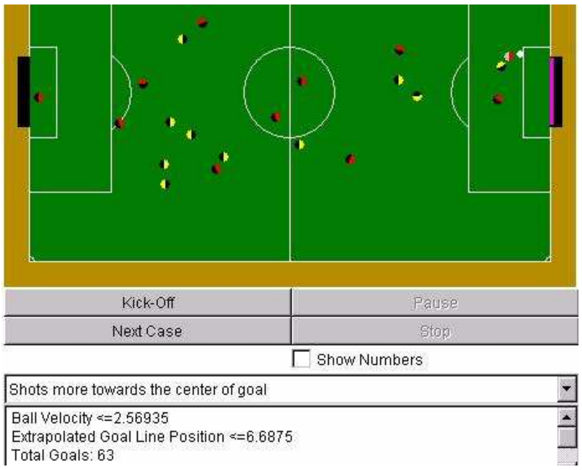

Figure 5: ISAAC's rule perturbations for Andhill'97.

TEAMORE currently assumes that it has access to plans or behaviors executed by agents during a game (via agents' execution traces). TEAMORE then compares the plans being executed by different team members, identifying *discrepant* situations, where team-members failed to agree on the joint team-plan that they should have been executing together. For instance, suppose the forwards are to execute a team tactic together, but one forward fails to execute it — this is a team discrepancy. TEAMORE uses this discrepancy information over time as the basis for a quantitative measure of teamwork.

TEAMORE uses several measures of discrepancy, but one particularly useful is the average time that it takes for a sub-team (e.g., forwards) to switch from one team plan to another. In perfect teamwork situations, all team-members switch team plans (tactics) together, moving in unison from one agreed upon team plan to another. Thus, the perfect team-plan switch time is exactly one time unit. The worst possible switch is if one team member never makes the switch and the sub-team never establishes agreement.

Earlier in this paper, ISIS97 was shown (Table 1) to have significantly different score-differences with and without communications, while even a large number of games (over 230) of ISIS98 resulted in no statistically significant difference (Table 2). Table 3 presents the results of TEAMORE's analysis for the ISIS98-Andhill97 and ISIS97-Anghill97 games. The average time per switch for two sub-teams (defenders and goalees) is shown for the two settings of the communication cost (approximately half of the games were played in each setting). These results show that communications (when cost is low) do reduce the average time per switch for each of the sub-teams. This reduction is statistically significant (two tailed t-test values are shown in the table), hinting at an improvement in the quality of teamwork with communication.

The first two columns of table 3 show that while ISIS98 had no statistically significant difference in the goal-difference for these games, TEAMORE is able to confirm that in fact STEAM is still making a statisically significant impact on the quality of teamwork, allowing validation of design objectives. Moreover, as the last two columns (presenting TEAMORE's analysis of the ISIS97-Andhill97 games) show, the difference in quality of teamwork that STEAM makes for ISIS97 is much greater than it is for ISIS98, supporting our hypothesis that ISIS98's superior monitoring capabilities reduced its dependency on teamwork. In fact, the average-time-per-switch values for ISIS97 using communications lie in between the values for ISIS98 with and without communications. However, the values for ISIS97 without communications are much greater then those for ISIS98, explaining the much bigger impact communication had on ISIS97.

Comm cost	ISIS98 Goalees	ISIS98 Defenders	ISIS97 Goalees	ISIS97 defenders
High	13.28	12.99	32.80	57.5
Low	3.65	3.98	5.79	6.81
p(null-hypo)	9.26E-16	7.13E-5	7.13E-13	1.45E-10

Table 3: Ave time per switch in games against Andhill97.

6 Lessons Learned

Our research in RoboCup has been fueled by the IJCAI'97 challenge. We have responded to the challenge in all three categories of learning, teamwork and agent modeling. Few other RoboCup teams have attacked the IJCAI'97 challenge in this much breadth. One possible exception is [Stone and Veloso, 1998b], who have focused on layered learning for agent design, and an approach to teamwork based on *locker-room arrangements*[Stone and Veloso, 1998a]. In their teamwork approach, agents synchronize their individual beliefs periodically in a fixed manner, in contrast with ISIS's STEAM in which communications are issued dynamically. Indeed, in comparison with [Stone and Veloso, 1998a] and other teams, ISIS stands alone in its use of a domain-independent teamwork model, with its demonstrated reuse. Other RoboCup researchers have investigated individual research areas. For instance, [Luke *et al.*, 1998] have investigated an innovative approach to learning in RoboCup, but they have yet to address the agent modeling and teamwork challenge of RoboCup. Others have investigated teamwork via explicit team plans and roles[Ch'ng and Padgham, 1998], but not learning or agent modeling, and they fail the *basic performance* requirement of the RoboCup challenge, i.e., the team must be able to play reasonably well. ISIS passes this test, given its third- and fourth-place in RoboCup'97 and RoboCup'98.

In conclusion, in responding to the IJCAI challenge, we have been able to extract the following general lessons in multi-agent learning, teamwork, and agent modeling:

- Some multi-agent environments require a significant role specialization of individuals. Thus, sharing experiences of individuals in different roles can sometimes be significantly *detrimental* to team performance, placing key limits on social learning.
- Divide-and-conquer learning can be used to enable different learning techniques to co-exist, reducing the complexity of the learning problem.
- Reuse of general teamwork models can improve performance and reduce development time.
- Tradeoffs exist in individual and team monitoring, e.g., responsible team behavior enables the design of simpler monitoring capabilities for individuals.
- *Competition within collaboration* can provide a simple but powerful technique for designing role responsibilities for individuals.
- In analyzing agent behavior in complex multi-agent environments, data-driven analysis combined with human oversight appears promising.
- Comparison of the behavior of team members can provide a useful teamwork monitoring tool.

Acknowledgements

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

References

[Ch'ng and Padgham, 1998] 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.

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

[Kaminka and Tambe, 1998] G. Kaminka and M. Tambe. What is wrong with us? improving robustness through social diagnosis. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, August 1998.

[Kitano *et al.*, 1997] 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.

[Luke *et al.*, 1998] 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.

[Newell, 1990] A. Newell. *Unified Theories of Cognition*. Harvard Univ. Press, Cambridge, Mass., 1990.

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

[Stone and Veloso, 1998a] 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.

[Stone and Veloso, 1998b] 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.

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