Towards Automated Team Analysis: A Machine Learning Approach

Taylor Raines, Milind Tambe, Stacy Marsella
Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292, USA
{Raines, Tambe, Marsella}@isi.edu

Abstract

Teamwork is becoming increasingly important in a large number of multi-agent applications. With the growing importance of teamwork, there is now an increasing need for tools for analysis and evaluation of such teams. We are developing automated techniques for analyzing agent teams. In this paper we present ISAAC, an automated assistant that uses these techniques to perform post-hoc analysis of RoboCup teams. ISAAC requires little domain knowledge, instead using data mining and inductive learning tools to produce the analysis. ISAAC has been applied to all of the teams from RoboCup'97, RoboCup'98, and Pricai'98 in a fully automated fashion. Furthermore, ISAAC is available online for use by developers of RoboCup teams.

Introduction

Teamwork is becoming increasingly important in a large number of multi-agent applications, including virtual environments for training [Tambe et al. 1995] and education [Kitano 1997], multi-robotic space missions, and software agents on the Internet [Williamson et al. 96]. With the growing importance of teamwork, there is now an increasing need for tools for analysis and evaluation of such teams. Indeed, in multi-agent domains with tens or even hundreds of agents in teams, agent interactions are often highly complex and dynamic, making it difficult for human developers to analyze agent and agent-team behaviors. The problem is further exacerbated in environments where agents are developed by different developers, where even the intended interactions are unpredictable. Automated assistants may aid human developers in analyzing agent teams in such situations, locating problematic team behaviors, diagnosing causes of such problems, and suggesting alternative courses of action.

This paper focuses on such an automated assistant for analyzing agent teams. Previous work in flexible teamwork (e.g., [Jennings 1995, Tambe 1997]) has largely focused on guiding autonomous agents in their teamwork. However, the problem of analyzing the behavior of such teams from a global perspective to aid human developers in improving team performance has been largely unaddressed. To remedy this situation, we are developing an automated assistant, itself an agent called ISAAC. ISAAC (ISI Soccer Automated Assistant Coach) analyzes agent-teamwork in the domain of RoboCup soccer [Kitano 1997]. While RoboCup is used since it poses agent-team analysis as a fundamental challenge, ISAAC's techniques are intended for applications in other domains, such as agent-teams in battlefield simulations [Tambe et al. 1995].

ISAAC performs post-hoc, off-line analysis of teams using agent-behavior traces in the domain. This analysis is performed bottom-up, using data mining and inductive learning techniques. By mining large numbers of behavior traces, ISAAC attempts to isolate the key factors determining the successes or failures of these teams. The developer provides ISAAC only with the knowledge of the features to be analyzed in a domain and the moments in time to analyze them (typically based on intermediate successes or failures), but not any information as to how these features should be used. ISAAC looks for interesting patterns based on user provided features, thus combining the analytical capabilities of the machine with the domain knowledge of the user.

ISAAC's patterns can also be parameterized in terms of tolerance for different error types. For instance, ISAAC many be parameterized to ensure it finds patterns covering every intermediate failure, although this may introduce errors in that the patterns may cover success cases also.

In addition to discovering interesting patterns of behavior that contribute to intermediate successes and failures, ISAAC also supports hypothetical reasoning via its perturbation analysis. In particular, by perturbing patterns that led to failures, developers can discover changes to agent behaviors that can avoid such failures. As shown later, perturbation analysis generates suggestions to improve team behavior without demanding a radical change in agent skill set.

ISAAC has been applied to all of the teams from RoboCup '97, RoboCup '98, and Pricai '98 in a fully automated fashion. Looking at intermediate successes and failures of these teams (goals scored by and against these teams), this analysis has revealed many interesting results including surprising weaknesses of the leading teams in both the RoboCup '97 and RoboCup '98 tournaments. ISAAC has also received a highly enthusiastic response from the RoboCup community. (ISAAC is available online at http://coach.isi.edu, see Figure 1.)

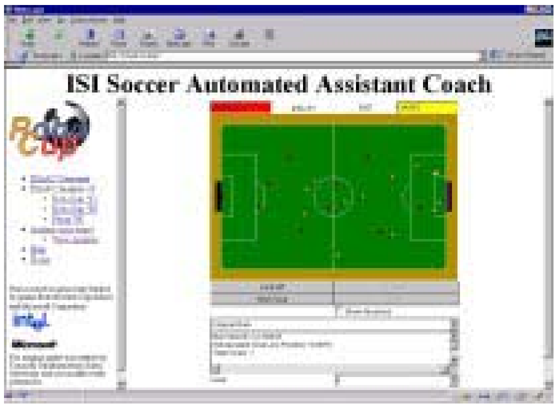


Figure 1: ISAAC exists on the web.

While this paper describes the current state of ISAAC, by the time of the RoboCup'99 competitions, we plan to have several enhancements available, including capabilities to analyze long sequences of team behaviors and evaluations by teams that used ISAAC in preparing for RoboCup'99.

Overview of Automated Analysis

Our approach to the team analysis problem is based on a bottom-up analysis of the team's behavior traces. Patterns of successful or unsuccessful behavior are first derived from the team's behavior traces via a data-driven inductive learning step. Based on those patterns, the analysis suggests possible improvements to the team's behavior via a process of perturbing the patterns. Again this phase is data-driven. Each suggested improvement is matched against the team's existing behavior traces to test the likely consequences of the improvement.

A preliminary issue for this approach is how to focus the analysis on those parts of team's behavior history relevant to the team's success. There may be many critical events along the path to the team's eventual success or failure that are widely separated in time, only loosely coupled to each other, but nevertheless critical to the team's success. For instance, in a team sport, there are many distinct scoring opportunities embedded in a larger history which may contain much that is irrelevant to the team's success.

This issue of focus is addressed by characterizing points in the history where there are intermediate successes or failures. When something is occurring that can directly influence the eventual success or failure of the team's performance outcome, this is considered to be an intermediate success or failure case. At present, we assume the identification of these intermediate points is part of the domain specific knowledge available to the automated analysis technique. Automated analysis is focused on these event subsequences or points in the history.

Having isolated cases of intermediate success or failure, there remains the question of how to analyze them. We approach the analysis as a problem of classification, of forming classes over the cases of success and failure based on a set of potentially relevant features in these cases. These features, along with the decision on what are the cases of intermediate success, are the only background information or bias given to our analysis agent. The features chosen must have the breadth to cover all information necessary for the analysis, but should also be independent of each other if at all possible. Another possibility here would be to provide all available attributes and then use an attribute selection method to prune out less helpful attributes [Caruana et al. 1994].

Currently, an inductive learning mechanism is used to form the classes of success and failure. Each class is a similar kind of success or failure, based on its feature description. The inductive learning mechanism isolates the specific features that are most important in this determination. These classes and the cases they govern are displayed to the user who is free to make the final determination about the validity of the analysis. By themselves, the features that describe a class provide implicit advice for improving the team.

More explicit exploration of this advice is performed using an automated *perturbation analysis*. Particular features are modified and the cases in the history that are consistent with this derived class are identified, allowing them to be explored. This makes it possible to assess which changes would have the greatest impact on the team.

ISAAC Analysis as applied to RoboCup Teams

In applying the approach to a domain, the domain specific information has to be identified that would be used by ISAAC as bias in its analysis. In the RoboCup domain, success means outscoring the opponent. Shots on goal are therefore key points of intermediate success or failure as these are situations that can directly affect the outcome of the game. Thus, the current focus of ISAAC's analysis in RoboCup is shots on a team's goal as well as shots by the team on an opponent's goal. In the future ISAAC's analysis will be extended to cover other critical situations.

Having defined shots on goal as a revealing portion of the overall history to analyze, we need to determine which features might be useful in classifying the success or failure of a shot on goal. After an initial set of experiments with a relatively large feature set, ISAAC currently relies on the following feature set to characterize successes and failures:

- Ball Velocity
- Distance to Goal (of shooter)
- Angle from Center Line (of shooter)
- Distance of Closest Defender (closest to shooter)
- Angle of Closest Defender from Center of Goal
- Extrapolated Goal Line Position (where the ball would pass the goal line based on initial trajectory, measured as distance from goal center)
- Number of Defenders (between shooter and goal)

In this set of features, only 2 attributes are used to characterize the shooter, ball velocity and extrapolated goal line position, since the shooter can only choose where to shoot and how hard to shoot. Thus, the analysis deliberately abstracts over player identity, player motion and facing direction.

Characterizing the rest of the context is more difficult since there is no easy manner in which to characterize abstract positional information such as the other players' locations. In the above feature set, information is provided about how many defenders are between the shooter and the goal, but additional positional information is only provided for the closest defender. In general, more features may produce a more accurate analysis if those features are important in the decision making process but there is an additional cost in processing more features. Also, as confirmed by our initial set of experiments, irrelevant features may actually decrease accuracy of the rules produced by the inductive learning mechanism [Caruana et al. 1994].

Having determined which features to use in the analysis and the points in the history (the cases) to examine, the task is transformed to mining the data by feeding it into an inductive learning algorithm. A decision tree induction algorithm [Russell, Norvig 1994] is used for this analysis. Currently, we use C5.0, an updated version of the C4.5 system [Quinlan 1994]. From the resulting decision tree, C5.0 forms rules representing distinct paths in the tree from the root of the tree to a leaf classification of success (goal-score) or failure (goal not scored). As a whole, the set of rules describes classes of similar successes and classes of similar failures.

Figure 2 shows an example success rule, describing a pattern where shots taken on the Windmill Wanderer team will fail to score (No Goal). It is important to keep the distinction between a shot failure and a team failure, since when we analyze shots on a team, a shot failure is a team success. This rule states that when the closest defender is sufficiently far away and sufficiently close to the shooter's path to the center of the goal, and the shooter is towards the edges of the field, this shot on the Windmill Wanderer team will fail to score. When viewed using ISAAC, the user can see that the defender is far enough away to have sufficient time to adjust and intercept the ball in most of these cases. Thus the user is able to validate ISAAC's analysis.

```
Distance of Closest Defender > 218
Angle of Closest Defender
        wrt Center of Goal <= 8.981711
Angle from Center of Field > 40.77474
        ➔    class No Goal
```

Figure 2: Sample Rule from shots on Windmill Wanderer team of RoboCup'98

The application of a decision tree induction algorithm to this analysis problem must address some special concerns. The goal-shot data has many more failure cases (failed goal shots) than success cases (goals scored). However, analyzing such data using a traditional decision tree induction algorithm such as C4.5 gives equal weight to the cost of misclassifying successes and failures. This usually yields more misclassified success cases than misclassified failure cases. For example, in our analysis of Andhill from the RoboCup'97 tournament, our original analysis misclassified 3 of 306 'no goal' cases, but misclassified 18 of 68 'goal' cases. Since an even smaller portion of the success cases is correctly classified, this produces perhaps overly specific rules that govern success cases. To compensate for this lopsided data set, the ability of C5.0 to weight the cost of misclassification is used. Specifically, the cost of misclassifying a success case is set to be greater than the cost of misclassifying a failure case [Ting 1998].

More generally, differential weighting of misclassification cost provides a mechanism for tailoring the classes and therefore the advice ISAAC implicitly provides. Consider shots on goal against a team. If a very high cost is assigned to misclassifying a successful shot on goal, the rules produced will cover almost all cases of successful shots, and likely quite a few misclassified failure cases. In this case, ISAAC's analysis will broaden the conditions under which a shot is considered to be potentially successful and as such is implicitly advising to make the team very defensive. On the other hand, if a low cost is assigned, the rules may not cover all of the successful cases. Therefore, ISAAC would only give "advice" relevant to stopping the majority of shots on goal. This may not be appropriate if we consider any goal to be a serious failure. Therefore, we give the user a choice as to how to set the level of defensiveness or aggressiveness of ISAAC's analysis through the weight on success case misclassifications.

This is all of the background information that ISAAC needs for the RoboCup domain. With this information, ISAAC can be applied to logs to form failure and success rules for both shots by the team on the opponent's goal and shots on the team's goal by other opponents.

Perturbation Analysis – recommending changes

After ISAAC has produced rules determining which circumstances govern success and failure classifications, ISAAC uses a perturbation analysis to determine which changes would produce the most benefit. 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 perturbations of a rule are mined from the data and examined to determine which conditions have the most effect in changing the outcome of the original rule, turning a failure into a success.

An interesting point about the perturbations is that they will always have some cases, although some perturbations will have many more than others. Take for instance a rule that governs failure cases and has 3 conditions for the rule to be valid. Reversing each one of these conditions will produce at least one success case. Why? Consider a condition that, when reversed, did not produce a success case. In this case, our inductive learning algorithm would have pruned this condition, determining it to be unnecessary, and we would have had a rule with 2 conditions [Quinlan 1994].

As shown using a detailed example in the following section, perturbations of a failure rule enable users to see what minimal modifications could be made to agent behaviors to convert the failures into success. Perturbations of success rules are also useful however. There are two reasons for this. The first reason is that some changes to the rule will take a team further from success than another. For example, a team may succeed 95% of the time when all conditions are met. The percentage of success may drop to 50% if the first condition is changed and down to 5% if the second condition is changed. In this case, the developer may decide that making the attempt even if the first condition is not met is still the correct course of action while making the attempt if the second condition is not met is a bad decision.

The second reason we want to look at perturbations of a successful rule is due to ISAAC's graphical nature. Since we allow the user to see how the team fails, more insight can be drawn as to why these conditions are important. Human oversight is important at this juncture to determine if the reasons ISAAC comes up with are truly the reasons the team is succeeding or failing. Looking at the cases makes it easy for a user to validate or refute the analysis.

ISAAC in practice

In its current online implementation, ISAAC can take log files of RoboCup games provided by any user with WWW and FTP access. ISAAC mines the data from the logs, looking for shots on goal and gathering the relevant features from each shot. This data is provided to our inductive learning algorithm for rule generation. ISAAC then takes these rules and again mines the data for those cases matching the rules and their perturbations. As a final step, ISAAC dynamically produces the web pages for the analysis to be viewed online, highlighting some of the key features for better understandability by the user.

Figure 3 shows ISAAC highlighting some key portions of a rule. The current rule only applies when the shooter is in the highlighted area of the field between the two lines, and is shooting to the highlighted portion of the goal. Showing the features as highlights on the field allows the user to easily delineate the portions of the field that the rule is describing. 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 taking shots on goal, the Andhill97 team

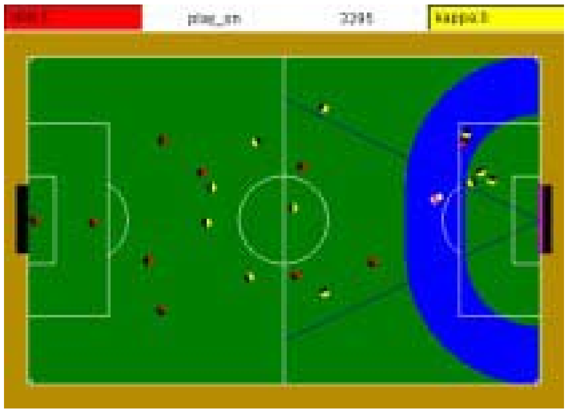
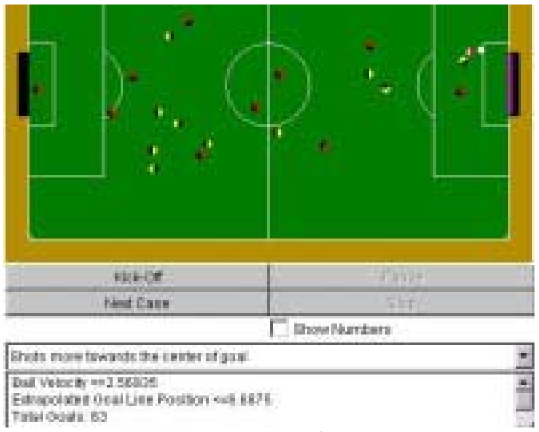


Figure 3: ISAAC highlighting key features.

often fails to score when (i) ball velocity is less than 2.37 meters per time step and (ii) the shot is aimed at greater than 6.7 meters from the center of goal (which is barely inside the goal). ISAAC reveals that shots governed by this fail to score 66 times without a successful attempt. 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 (See Figure 4).



Shots more towards the center of goal

Ball Velocity <=2.36925
Extrapolated Goal Line Position <=6.6875
Total Goals: 63

Figure 4: Analysis of Andhill97

Now consider the perturbations of this rule. In cases where the rule is perturbed such that ball velocity is greater than 2.37 m/t and the shot aim is still greater than 6.7m, Andhill scores twice and fails to score 7 times. In another perturbation, where ball velocity is again less than 2.37 m/t but now shot aim is equal to or less than 6.7m, Andhill is now scoring 51 times and failing to score 96 times. These perturbations suggest that improving Andhill97's shot aiming capabilities can significantly improve performance.

Experimental Observation

Tables 1 and 2 show some results from ISAAC's analysis of the top four teams from both RoboCup'97 and RoboCup'98. Results show that on average there are 10-15 rules formed by ISAAC's analysis per team for analysis of offense, and 5-10 rules formed per team for analysis of defense.

Analysis of Shots by this team			
RoboCup '97	Cases	Success	Rules
Andhill	269	84	19
AT_Humboldt	336	91	15
CMUnited	213	58	9
ISIS	190	31	16
Average	252	66	14.75
Analysis of Shots on this team			
Andhill	131	16	12
AT_Humboldt	117	13	3
CMUnited	193	12	10
ISIS	123	19	6
Average	141	15	7.75

Table 1: Analysis of RoboCup '97

Analysis of Shots by this team			
RoboCup '98	Cases	Success	Rules
AT_Hum98	367	95	8
CMUnited	171	64	8
WW	127	38	11
ISIS	173	37	12
Average	209	58.5	9.75
Analysis of Shots on this team			
	Cases	Success	Rules
AT_Hum98	151	6	4
CMUnited	Not Available		
WW	133	3	5
ISIS	189	30	8
Average	157.7	13	5.7

Table 2: Analysis of RoboCup '98

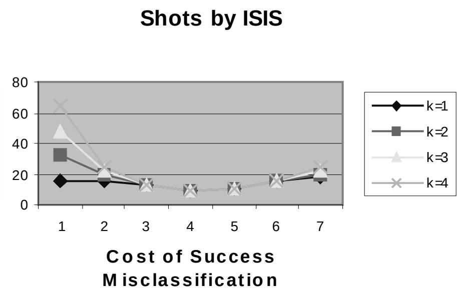
Furthermore, for analysis of offense (shots on opponents' goals) the average number of rules for RoboCup'97 for top the 4 teams is 14.75, while it is 9.75 for RoboCup'98 teams. For defense, the average is 7.75 for RoboCup'97, and it goes down to 5.66 in RoboCup'98. Thus, the average number of rules formed has decreased from RoboCup'97 to RoboCup'98 for the top four teams. This decrease is most starkly observed in the case of AT_Humboldt. ISAAC derives 15 rules from 336 cases in RoboCup'97 and 8 rules from 367 cases in RoboCup'98 for its offense. While we are currently investigating the reason for this decrease in rules, and the significance of such a decrease, the key here is that ISAAC opens up the possibility of such a global tournament-wide analysis of agent-team behaviors. Note that no analysis was possible of the CMUnited defense in RoboCup'98 because no goals were scored against them in the tournament.

We also performed experiments to determine the optimal weight for success misclassification for RoboCup teams. When we originally used ISAAC to analyze the RoboCup '97 tournament, ISAAC returned a rather poor result for the analysis of many of the teams. Specifically, ISAAC was not identifying many of the success cases. For example, the CMUnited team had 12 goals scored against them in the tournament, but using the inductive learning algorithm with no weight for success misclassification yielded rules that only correctly identified 4 of these goals. We therefore increased the misclassification cost until more of the goals were correctly identified.

Increasing the cost of success misclassifications yielded an increase in the number of success cases correctly classified, but also increased the number of failure cases misclassified. Therefore the issue arose as to how many failure misclassifications were acceptable to correctly identify one extra success case.

To explore this issue, the top 4 teams from RoboCup '97 were analyzed and weights were used from the original 1:1 ratio to a misclassification cost of 7:1 for failure misclassifications to success misclassifications. The optimal cost was determined in terms of lowest total error. Total error cost was considered to be (Total Number of Failure Cases Misclassified) + k*(Total Number of Success Cases Misclassified) with values of k from one to four.

Total error as stated above was plotted against different success misclassification costs for different values of k. The results showed that the optimal cost tended to be the same regardless of the value of k. For instance, Figure 5 shows that



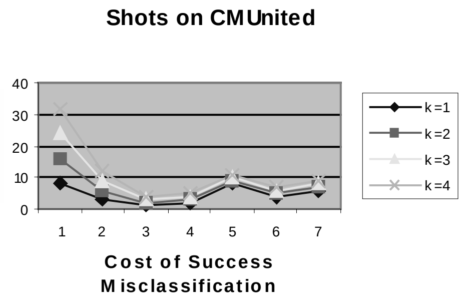


Figure 5: Total Error for Differing k Values For Shots by ISIS and Shots on CMUnited

the optimal cost for misclassified success cases for shots by ISIS was 4, and the optimal cost for shots on CMUnited was 3, both regardless of the value of k. And for the different values of k, none of the teams varied in their optimal values by more than 1. Using this optimal cost, ISAAC now produces rules to govern 11 of the 12 success cases for the shots on CMUnited. The current version of ISAAC uses a weight of 3 which was found to be optimal or near optimal in all cases, but future versions may run this experiment to tailor the optimal weight to each specific team, while the current system allows users to change this weight manually.

Related Work

Knowledge Base Refinement

SEEK, and its progeny, SEEK2, is an approach to knowledge base refinement, an important aspect of knowledge acquisition [Ginsberg et al. 1985]. Knowledge base refinement is characterized by the addition, deletion, and alteration of rule-components in an existing knowledge base, in an attempt to improve an expert system's performance. While the alteration of a rule may seem comparable to ISAAC's perturbation analysis, the goals are varied. The refinement done by systems such as SEEK are used to increase performance of the system to correctly classify future cases. ISAAC's goal is not that of increased performance in terms of cases classified but that of increased understandability of the rules produced. By looking at changes to the automatically produced rules, the user gains insight as to the effects of each component of a rule.

Decision Tree Confidence Factors for Multiagent Control

Peter Stone and Manuela Veloso have incorporated a previously trained decision tree into a full multiagent behavior that is capable of controlling agents throughout an entire game, also using RoboCup as their domain [Stone et al. 1998]. In their work, they created artificial situations to train their decision tree, and then used the learned behaviors from this tree in game situations. In contrast, ISAAC looks at game situations to produce its decision tree and subsequent rules, and then makes suggestions based on actual game play. However, ISAAC need not reason about execution time, a rather large sub-problem stemming from their work.

Inductive Verification and Validation

The designers of the KULRoT RoboCup team used inductive logic programming to validate their multi-agent system [Driessens et al. 1998]. They induced rules intended to verify the programming that their agents were supposed to follow. The major difference between this and our work is that since they were attempting to validate their programming, they knew what they were looking for in their agents and could therefore incorporate all of the background knowledge that the agents were using. We use little of this background knowledge but generalize our work such that it is capable of analyzing any RoboCup team.

Comprehensible Knowledge Discovery

Pazzani uses an inductive learning algorithm called FOCL [Pazzani 1997], an extension to FOIL [Quinlan 1990], that uses expert rules in combination with inductive learning techniques. FOCL adds predicates to rules using either the expert rules or an inductive method, depending on which is more informative. FOCL also uses monotonicity constraints to produce rules that are more intuitive with human observations.

ISAAC currently does not use expert knowledge of this type in the analysis, but there is the possibility to do so in the future if this knowledge is available, perhaps even using the FOCL algorithm. ISAAC does, however, allow the use of biased weighting of classes and performs further analysis on the rules through its perturbation analysis.

Summary

Multi-agent teamwork is a critical capability in a large number of applications including training, education, entertainment, design, and robotics. The complex interactions of agents in a team with their teammates as well as with other agents, and with other dynamic events in the environment, makes it extremely difficult for human developers to understand and analyze agent-team behavior. It is thus increasingly critical to build automated assistants to aid human developers in analyzing agent team behaviors. While previous research has begun to address the problem of guiding agent behavior for flexible teamwork, the problem of automated analysts to aid developers is largely unaddressed.

We have taken a step towards these automated analysts by building an agent called ISAAC for post-hoc, off-line agent-team analysis. ISAAC uses inductive learning techniques to discover patterns (in the form of rules) in agent team behavior that contribute to successes or failures of a team, and presents these to a user. It also allows a user to examine the domain situations that led to the generation of the rules. ISAAC also supports perturbations of rules, to enable users to engage in "what-if" reasoning. ISAAC has currently been applied in the context of the RoboCup soccer simulation. ISAAC is available on the web, and it enables the RoboCup community to see the analysis of their teams that participated in RoboCup'97, RoboCup'98, and Pricai'98 competitions. Users can also analyze the teams they are developing towards RoboCup'99 competitions.

So far, the feedback from the user community has been highly enthusiastic. We have also received feedback for improvements in ISAAC. The suggestions range from details of units used in ISAAC's measurements to suggestions about the next step to go beyond analysis of individual intermediate successes and failures (to sequences of behaviors). We continue to encourage user participation in ISAAC and feedback and hope it will contribute to an improvement of team performance in RoboCup'99.

Future Work

We plan to extend our idea of intermediate successes or failures into a "points of flux" analysis where we consider any time in a dynamic domain where an event is occurring that could affect the outcome. While this type of analysis would not have rigidly defined success and failure cases, we may be able to overcome this problem by using gradations of success, whereby we include some extra domain knowledge to determine how well the agents are performing at these points.

More recently, ISAAC has been extended to analyze sequences of behaviors (again using C5.0), such as sequences of actions (e.g., passes) that lead up to successes or failures [Lesh et al. 1998]. In the RoboCup scenario, we consider each time a player successfully kicks the ball to be a point of flux. We hope to use this research as an extension to our shots on goal analysis to then analyze decisions about passing or shooting. This preliminary 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 passing. In analyzing agent behavior in complex multi-agent dynamic environments, the approach of using knowledge poor data-driven analysis techniques combined with human oversight has shown considerable promise.

Acknowledgements

This research is supported by a gift from the Intel Corporation and Microsoft Corporation.

References

[Atkins et al. 1997] Atkins, E., Durfee, E., Shin, K. Detecting and Reacting to Unplanned-for World States. *In Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, 1997.

[Caruana et al. 1994] Caruana, R., Freitag, D. Greedy Attribute Selection. *In 11th Proceedings of the 11th International conference on Machine Learning (ICML)*, 1994.

[Driessens et al. 1998] Driessens, K., Jacobs, N., Cossement, N., Monsieurs, P., DeRaedt, L., Inductive Verification and Validation of the KULRoT RoboCup Team. *In Proceedings of the Second RoboCup Workshop*, 1998.

[Ginsberg et al. 1985] Ginsberg, A., Weiss, S., Politakis, P., SEEK2: A Generalized Approach to Automatic Knowledge Base Refinement. *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1985

[Jennings 1995] Jennings, N. Controlling Cooperative Problem Solving in Industrial Multi-agent System Using Joint Intentions. *In Artificial Intelligence, Vol. 75*, 1995.

[Kaminka, 1998] Kaminka, G. A., and Tambe, M. What's Wrong with Us? Improving Robustness through Social Diagnosis. *In Proceedings of the Fifteenth National conference on Artificial Intelligence (AAAI-98)* Wisconsin, July 26-30, 1998.

[Kitano et al. 1997] Kitano, H., Tambe, M., Stone, P., Veloso, M., Noda, I., Osawa, E. & Asada, M. The RoboCup synthetic agent's challenge. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1997.

[Lesh et al. 1998] Lesh, N., Martin, N., Allen, J. Improving Big Plans *In Proceedings of the National Conference of Artificial Intelligence (AAAI)*, 1998.

[Pazzani 1997] Pazzani, M. Comprehensible Knowledge Discovery: Gaining Insight from Data. *In First Federal Data Mining Conference and Exposition*, 1997.

[Quinlan 1990] Quinlan, J. Learning Logical Definitions from Relations. *In Machine Learning, 5*, 1990.

[Quinlan 1994] Quinlan, J. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1994.

[Russell, Norvig 1995] Russell, S., Norvig, P. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, Inc., 1995.

[Stone et al. 1998] Stone, P., Veloso, M. Using Decision Tree Confidence Factors for Multiagent Control. *In Proceedings of the International Conference on Autonomous Agents*, 1998.

[Tambe et al. 1995] Tambe, M. Johnson, W. L., Jones, R., Koss, F., Laird, J. E., Rosenbloom, P.S., Schwamb, K. Intelligent Agents for Interactive Simulation Environments. *In AI Magazine, 16(1) (Spring)*, 1995.

[Tambe 1996] Tambe, M. Tracking Dynamic Team Activity. *In Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1996.

[Tambe 1997] Tambe, M. Towards Flexible Teamwork. *In Journal of Artificial Intelligence Research, Vol. 7*, 1997.

[Ting 1998] Ting, K. Inducing Cost-Sensitive Trees via Instance Weighting. *In Principles of Data Mining and Knowledge Discovery (PKDD 98)*, 1998.

[Williamson et al. 1996] Williamson, M. and Sycara, K. and Decker, K. Executing Decision-theoretic Plans in Multi-agent Environments. *In Proceedings of the AAAI Fall Symposium on Plan Execution: Problems and Issues*, 1996.