CBR for Dynamic Situation Assessment in an Agent-Oriented Setting

Jan Wendler and **Mario Lenz***

Abstract

In this paper, we describe an approach of using case-based reasoning in an agent-oriented setting. CBR is used in a real-time environment to select actions of soccer players based on previously collected experiences encoded as cases.
Keywords: Case-based reasoning, artificial soccer, agent-oriented programming.

Introduction

In recent years, agent-oriented technologies have caught much attention both in research and commercial areas. To provide a testbed for developing, evaluating, and testing various agent architectures, RoboCup Federation started the Robot World Cup Initiative (RoboCup) (Kitano *et al.* 1997), which is an attempt to foster AI and intelligent robotics research by providing a somewhat standardized problem where wide range of technologies from AI and robotics can be integrated and examined.

In particular, during the annual RoboCup championships different teams utilizing different models and methods compete with each other in the domain of soccer playing. Key issues in this domain are that, firstly, a true *multi agent* behavior is required in which each player corresponds to an autonomous agent which, however, has to cooperate with the other agents/players. Secondly, the agents have to deal with an uncertain environment, primarily because of the unpredictable behavior of the opponent team.

Besides the various leagues for robots of different sizes, a *Simulation League* is also part of RoboCup. As the name indicates, soccer players are simulated in this league by means of software programs rather than using *real* robots. For this, RoboCup Federation provides a *SoccerServer*, which simulates all the events that occur on the pitch (namely, movements of players and the ball), and a *SoccerMonitor* which is used to visualize the game (Noda 1995). The programs of the competitors implement the brains of the virtual players.

Dept. of Computer Science, Humboldt University, D-10099 Berlin, {wendler, lenz}@informatik.hu-berlin.de

In this paper, we will restrict ourselves to the *Simulation League*; that is, we will consider software agents only. On the one hand, we thus ignore some of the major problems of robotics, such as analyzing sensor input or moving in an uncertain environment. On the other hand, we can concentrate on the problems of intelligent agent behavior which is particularly difficult as a player has to come up with its decisions and actions in real time (the player can produce a command describing an action every 100 msec). In particular, we will address the issue of how agents can learn during the game and thus adapt to the opponent's behavior. For this CBR techniques are applied; in terms of the specified categories this approach clearly belongs to a *Slave-Master* configuration, that is, CBR is used to support other reasoning mechanisms internal to the agents.
In the next section, we give a short introduction into the *SoccerServer* and in the current implementation. After having described the problem, we will outline the CBR approach in detail. Last but not least, we will present some preliminary results.

The Domain of Artificial Soccer

The Simulation Environment

The simulator program of the soccer simulation consists of two subsystems, the *SoccerServer* and the *SoccerMonitor*. The SoccerServer is the simulator's engine, whereas the SoccerMonitor visualizes the game.

The SoccerServer runs as a process and waits for the login of the 22 different player–agents (clients). The communication between the server and his clients runs via UDP (User Datagram Protocol); thus real distribution of the client-programs over several computers is possible. The player programs are only allowed to communicate with each other through the server.

When all the player–agents have logged in properly, the human observer can start the match by the SoccerMonitor, and the SoccerServer starts the simulation. Two interfaces are used in the communication between server and clients: The server sends character string, which encode visual and audible information, in predefined intervals to the player programs. The players react by transmitting commands from a certain skill library.

The "visual" information of a player depends on its facing direction and the distance of observable objects. It can observe only objects in a region of 45, 90 or 180 degree. Beyond a certain distance, first the player number and then even the team identity is omitted by the server. The player gets information about the distance of objects and their deviation from the facing direction. For very close objects it also gets expected position changes. All information is subject to certain noise.

Players do not get information about their absolute position on the field. They only receive their relative coordinates with respect to some landmarks like goals, lines, and flags, if these objects are within their field of view.

Every agent can send commands to the SoccerServer in constant time intervals. Available commands are *Kick*, *Dash*, *Turn* for acting on the playground, and *Say* for "broadcasted" messages. The necessary parameters of actions, like power and direction, are relative to the agent, too.

Auditory information from the referee is available to all players regardless of their distance to the referee. The built-in referee of the SoccerServer judges goals, outside, corner kick and offside. Auditory information from other players (command *Say*) is available for all players in a distance of less than 50 m, and it is restricted to only one message per 200 msec.

With the special command *ChangeView* the agent can adjust its view angle and quality. This property influences the update rate of visual information. We use mainly a 180 degree view angle and high quality information, which implies a 300 msec update interval. Commands are accepted every 100 msec by the server. This clocked process with tight time slices results in a real-time behavior of the whole system.

Players have some restricted resources, the most important being stamina and time. Their stamina is decreased according to the power of dash commands. It is exhausted after a full speed run over about half of the playground, after that it is restored only slowly. The agents also have a limited time for deliberation and planning, if they want to react quickly to new sensor information.

Current Implementation

The team AT Humboldt (Burkhard, Hannebauer, & Wendler 1997), which became World Champions at the RoboCup 1997 in Nagoya, is based on the *Belief–Desire–Intention* (BDI) architecture well known in the agent–oriented community (Rao & Georgeff 1995). In the BDI–approach, which is based on the philosophical work of Bratman (Bratman 1987), agents maintain a model of their world which is called *belief* (because it might not be true knowledge). The way from belief to actions is guided by the *desires* of the agent. Bratman has argued that *intentions* are neither desires nor beliefs, but an additional independent mental category. Intentions are considered as partial plans for the achievement of goals by appropriate actions.

All components of a BDI–architecture can be identified in our planning process. The belief component models the environment of the agent, according to sensor information. It simulates the movement of objects outside the agent's field of view. Beliefs about future development are realized through a simulation element. It enables the agent to evaluate the utility of possible actions of speculative future developments for possible actions.

The desire and intention component is modelled by the deliberator of the agent. The deliberator is started with a set of options, which are possible desires, and a set of possible constraints, each time new sensor information is available. Possible desires are e.g. to perform a goal-kick, to dribble or to run free. A possible constraint is to conserve stamina.

The deliberator asks all options and all constraints about their expected utility, to collect the desires which seems to be feasible. The best of this desires, which is possible in the current situation, is choosen. At this selection, the previous desire (the desire of the last deliberation process) will be taken into account. At this phase of the deliberation the best posibility to reach the choosen goal is computed and fixed as an intention. This corresponds to a long-term plan with some parameters which can be also seen as a partial plan. The plans are based on the "capabilities" of the agents. This means that plans are in a close correspondence to the skills of the agents, which are split the long-term partial plans into short-term plans. These short-term are nothing more then the available commands of the SoccerServer.

The part of the deliberation process, which will be described in this article is the computation of the long-term plan of the desire to *run free.*

The Problem of Uncertainty

A key problem that the players have to deal with in the RoboCup is the uncertainty in the soccer world which arises mainly because of two reasons:

- Firstly, the information provided to the players by the above mentioned *SoccerServer* is insecure and, in some sense, unreliable. In particular, a kind of *fuzziness* is included in the information given. For example, data about other players on the pitch is less precise the further these are away.
- Secondly, the behavior of the opponent team is, of course, unpredictable.

While uncertainty of the first kind is inherent to the entire system, there are certain ways to overcome the second point. In particular, our approach is to *learn* about the opponent's behavior and to adapt the actions of the soccer players accordingly.

As an example consider the situation depicted in Figure 1: Knowing the position and movement of the ball, the position of another teammate P_B, and positions of players of the opponent team, player P_S has to decide

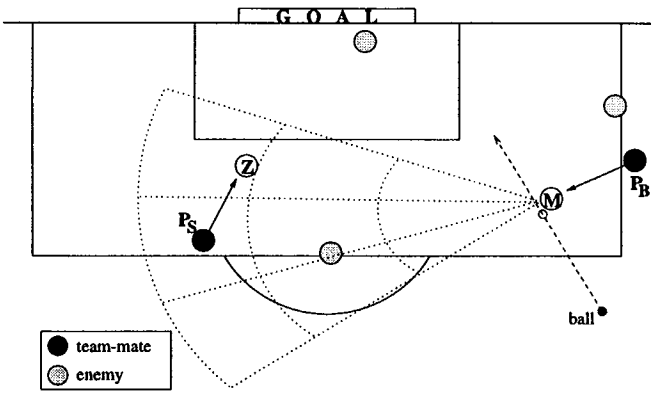

Figure 1: Finding a good position for the expected movements

about where to move in order to be in good position for the expected actions on the pitch. We will refer to this position as the *preference position.*

There is certain information which allows to actually *calculate* precisely what the situation might look like after 2, 3, or 4 time steps. This information includes in particular the speed and direction of all the players as well as the ball. However, there are certain drawbacks when attempting to *calculate* to preference position:

1. As mentioned above, the information provided by the *SoccerServer* is insecure and imprecise.
2. Calculation of preference positions has to take a number of parameters into account and requires a complete modeling of the involved players. This may be too computationally expensive for players having to react in real time.
3. Even worse, calculation would require a complete domain modeling at compile time; i.e., the *programmer* of the system would have to consider all situations that might possibly occur during the game. This seems infeasible.
4. Last but not least, the behavior of other players is, of course, unpredictable. Due to the multi-agent environment where each player is implemented by means of an autonomous program, this is true for both opponent players as well as teammates.

An alternative to calculating the preference position is to use a case-based reasoning approach in which *snapshots* of the game are considered as cases and similar situations are recalled and evaluated for determining the preference position. This will be described in the following section.

CBR for Dynamic Situation Assessment

In order to apply CBR, one has to come up with a number of decision about the actual design of the CBR systems. These include questions, such as:

- What is a case? How are cases represented?
- How is the similarity measure defined?
- How is the structure of the case memory?
- Where do cases come from?

We will address these questions in the following.

Case Structure

Based on the information provided by the *SoccerServer*, there are certain parameters which obviously should be included in the case representation (see also Figure 1). These are

State of the pitch: The position of other players should, of course, be taken into account. For this, we represent the considered area by means of segments which are either occupied by some player or not. We do not distinguish here between teammates and players of the opponent team as for selecting the preference position this is of no interest[1].

Time steps until P_B controls the ball: Based on the information provided by the *SoccerServer*, one may calculate the expected time until P_B will control the ball and thus is able to pass the ball to other teammates. Note, however, that P_S does not know precisely what P_B will do because of the players being autonomous agents. An assumption here is that P_B will behave somewhat optimal with respect to the performance of the entire team.

Preference direction of P_B: As P_B is a member of the same team, P_S may know about the preference movements of P_B which mainly depends on the starting position of the player; i.e., whether he plays on defending or forward position.

Available power resources: As power resources are limited, player P_S has to take into account which actions can still be performed within a certain time frame. If the player still has sufficient power, he can easily reach positions further away. In contrast, if power is nearly consumed, he might only reach neighboring segments.

Distance to the ball at position M: In order to save power resources, it is reasonable to let the ball move instead of running themselves. Consequently, movements should be selected carefully depending on the distance to the point where the ball will be reached by player P_B. Where to small distances are inefficient, to large distances include the risk that the ball leave the right path to much.

Based on this, a case is represented by a feature vector where each value encodes a certain value of the above parameters. While for the last four aspects encoding is straight-forward, this is more difficult for the first parameter. As sketched in Figure 1, we segmented the pitch in such a way that the position M where P_B will likely control the ball is in the middle of a circle. In the real world, each segment would correspond to an area of about 8 meters length and an arc of 20 degrees.

[1] It does not make any sense to move into a place which is occupied by a player of the same team.

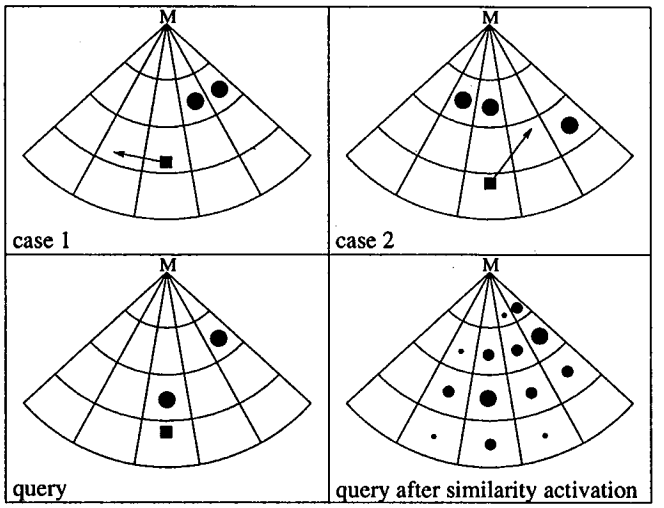

Figure 2: Finding a good position for the expected movements

Similarity Measure

Given the above described case representation, the definition of a similarity measure is straight-forward: For the parameter State of the pitch, neighboring segments are considered similar; all other parameters are encoded by means of a numeric value thus allowing for a simple similarity assessment. Given these, we can define a composite similarity measure which simply computes the weighted sum over all local similarities where all features have the same relevance.

Case Memory and Retrieval

A crucial point in the RoboCup domain is that all players have to react in real time. Consequently, there is an urgent need for a highly efficient memory organization and retrieval procedure. For this reason, we applied the *Case Retrieval Net* (CRN) model (Lenz 1996; Lenz & Burkhard 1996).

Figure 2 briefly sketches how the spreading activation mechanism of CRNs is used to determine the similarity of cases describing previously observed situations. The upper two pictures (A and B) show cases in the case base. Circles denote segments occupied by some other player; the position of the player itself is marked by the square. In both cases, the arrow denotes the solution, i.e., the preference position taken in this situation.

In the lower left part of the figure (C), the current problem situation is sketched. Using this description as the query, similarity propagation within the CRN model will result in a situation as shown in the lower right part (D) of the figure. Here, the size of the circles correspond to the achieved activations of the segments. Using this information (and the information about similarity of the other parameters which are not shown here), situation A would be considered as more similar than situation B and hence the player would move to the left according to case A.

Online versus Offline Learning

Applying the above described CBR approach the dynamic situation assessment, there are two ways a player might obtain cases to reason upon:

- Firstly, during a kind of *training* the player may build up its own case base: That is, the player will learn (in a case-based manner) how to behave in certain situations which seems particularly useful for so-called standard situations, i.e., situations that often re-occur.
- Secondly, the player my acquire new cases during the game itself thus having the opportunity to adapt to the opponent's behavior.

Preliminary Results and Outlook

The agents of the team AT Humboldt have recently been extended by case-based components as described above. Every player now has the capability of acquiring new cases and maintaining its own case base representing previously observed situations.

To determine whether CRNs are efficient enough for the real-time requirements of RoboCup, we tested retrieval performance in a number of different configurations as specified by the RoboCup Federation. For example, when each player is simulated on a single Ultra Sparc-Station 1 (as will be the case for the semi-finals and finals in RoboCup 1998), then retrieval from a case base of 5,760 cases takes on average 33 msec with a maximum of 40 msec. Consequently, one can estimate that a case base of about 10,000 cases is manageable within the strict time frame of RoboCup.

Currently, investigations are being performed to determine the quality of cases and the overall usefulness of the approach. In particular, it is an open question whether the representation of the State of the pitch by means of segments has to be more fine-grained or whether the current segment structuring is sufficient.

Furthermore, we will investigate how the two modes of learning, namely offline and online learning, can be combined. For example, a question is whether and how the cases acquired online by adapting to a particular opponent team can be generalized in some way to make this knowledge available for forthcoming matches as well.

Acknowledgments

We want to thank the entire team of AT Humboldt who contributed to the success In particular, thanks go to Hans-Dieter Burkhard, Markus Hannebauer, Kay Schröter, Pascal Gugenberger, and Ralf Kühnel.

References

Bratman, M. E. 1987. *Intentions, Plans and Practical Reason.* Massachusetts: Harvard University Press.

Burkhard, H.-D.; Hannebauer, M.; and Wendler, J. BDI deliberation in artificial soccer. *AI Magazin.* to appear.

Burkhard, H.-D.; Hannebauer, M.; and Wendler, J. 1997. AT humboldt - development, practice and theory. In *Proc. First International Workshop on RoboCup*, LNCS, to appear. Springer Verlag.

Kitano, H.; Tambe, M.; Stone, P.; Veloso, M.; Coradeschi, S.; Osawa, E.; Matsubara, H.; Noda, I.; and Asada, M. 1997. The robocup synthetic agent challenge 97. In Pollack, M. E., ed., *Proceedings of the IJCAI-97*, 24–29. Morgan Kaufmann.

Lenz, M., and Burkhard, H.-D. 1996. Case Retrieval Nets: Basic ideas and extensions. In Görz, G., and Hölldobler, S., eds., *KI-96: Advances in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, 1137, 227–239. Springer Verlag.

Lenz, M. 1996. Case Retrieval Nets applied to large case bases. In Burkhard, H.-D., and Lenz, M., eds., *4th German Workshop on CBR — System Development and Evaluation —*, Informatik-Berichte, 111–118. Humboldt University.

Noda, I. 1995. Soccer server: A simulator of RoboCup. In *Proc. of AI Symposium.* Japanese Society for Artificial Intelligence.

Rao, A., and Georgeff, M. 1995. BDI agents: From theory to practice. In *Proc. of the First Int. Conf. on Multi-Agent Systems (ICMAS-95).* MIT-Press.

Appendix

1. **Integration name/category**: AT Humboldt
2. **Performance Task**: Retrieval
3. **Integration Objective**: Efficiency gains in CPU time as well as online learning
4. **Reasoning Components**: Case- and model-based reasoning
5. **Control Architecture**: CBR as a slave
6. **Reasoning Step(s) Supported by CBR**: Case-based situation assessment rather than computationally expensive and inflexible model-based computation
7. **Representations**: Cases as feature vectors corresponding to the representation in the model-based reasoning module
8. **Additional Reasoning Components**: -
9. **Integration Status**: Initial empirical evaluation
10. **Priority future work**: Detailed evaluation, competition at RoboCup-98 in France