I Know What You're Doing: A Case Study on Case-Based Opponent Modeling and Low-Level Action Prediction

Thomas Gabel and Eicke Godehardt

Faculty of Computer Science and Engineering
Frankfurt University of Applied Sciences
60318 Frankfurt am Main, Germany
{tgabel|godehardt}@fb2.fra-uas.de

Abstract. This paper focuses on an investigation of case-based opponent player modeling in the domain of simulated robotic soccer. While in previous and related work it has frequently been claimed that the prediction of low-level actions of an opponent agent in this application domain is infeasible, we show that – at least in certain settings – an online prediction of the opponent's actions can be made with high accuracy. We also stress why the ability to know the opponent's next low-level move can be of enormous utility to one's own playing strategy.

1 Introduction

Recognizing and predicting agent behavior is of crucial importance specifically in adversary domains. The case study presented in this paper is concerned with the prediction of the low-level behavior of agents in the highly dynamic, heterogeneous, and competitive domain of robotic soccer simulation (RoboCup). Case-based reasoning represents one of the potentially useful methodologies for accomplishing the analysis of the behavior of a single or a team of agents. In this sense, the basic idea of our approach is to make a case-based agent observe its opponent and, in an online fashion, i.e. during real game play, build up a case base to be used for predicting the opponent's future actions.

In Section 2, we introduce the opponent modeling problem, point to related work, and argue why knowing an opponent's next low-level actions can be beneficial. The remainder of the paper then outlines our case-based methodology (Section 3), reviews the experimental results we obtained (Section 4), and summarizes and discusses our findings (Section 5).

2 Opponent Modeling in Robotic Soccer Simulation

RoboCup [12] is an international research initiative intending to expedite artificial intelligence and intelligent robotics research by defining a set of standard problems where various technologies can and ought to be combined solving them. Annually, there are championship tournaments in several leagues – ranging from rescue tasks over real soccer-playing robots to simulated ones.

2.1 Robotic Soccer Simulation

The focus of the paper at hand is laid upon RoboCup's 2D Simulation League, where two teams of simulated soccer-playing agents compete against one another using the Soccer Server [10], a real-time soccer simulation system.

The Soccer Server allows autonomous software agents written in an arbitrary programming language to play soccer in a client/server-based style: The server simulates the playing field, communication, the environment and its dynamics, while the clients – eleven autonomous agents per team – connect to the server and are permitted to send their intended actions (e.g. a parameterized kick or dash command) once per simulation cycle to the server via UDP. Then, the server takes all agents' actions into account, computes the subsequent world state and provides all agents with (partial) information about their environment via appropriate messages over UDP.

So, decision making must be performed in real-time or, more precisely, in discrete time steps: Every 100ms the agents can execute a low-level action and the world-state will change based on the individual actions of all players. Speaking about low-level actions, we should make clear that the actions themselves are "parameterized basic actions" and the agent can execute only one of them per time step:

– $dash(x, \alpha)$ – lets the agent accelerate along its current body orientation by relative power $x \in [0, 100]$ (if it does not accelerate, then its velocity decays) into direction $\alpha \in (-180, 180]$ relative to its body orientation
– $turn(\alpha)$ – makes the agent turn its body by $\alpha \in (-180, 180]$ where, however, the Soccer Server reduces α depending on the player's current velocity in order to simulate an inertia moment
– $kick(x, \alpha)$ – has an effect only, if the ball is within the player's kick range (1.085m around the player) and yields a kick of the ball by relative power $x \in [0, 100]$ into direction $\alpha \in (-180, 180]$
– There exist a few further actions (like tackling[1], playing foul, or, for the goal keeper, catching the ball) whose exact description is beyond scope.

Given this short description of the most important low-level actions that can be employed by the agent, it is clear that these basic actions must be combined cleverly in consecutive time steps in order to create "higher-level actions" like intercepting balls, playing passes, doing dribblings, or marking players. We will call those higher-level actions *skills* in the remainder of this paper.

Robotic Soccer represents an excellent testbed for machine learning, including approaches that involve case-based reasoning. For example, several research groups have dealt with the task of learning parts of a soccer-playing agent's behavior autonomously (for instance [9, 8, 3]). In [6], as an other example, we specifically addressed the issue of using CBR for the development of a player agent skill for intercepting balls.

[1] To tackle for the ball with a low-level action $tackle(\alpha)$ means to straddle for the ball and thus changing its velocity, even if it is not in the player's immediate kick range; such an action succeeds only with limited probability which decreases the farther the ball is away from the agent.

2.2 Related Work on Opponent Modeling

Opponent modeling is an important factor that can contribute substantially to a player's capabilities in a game, since it enables the prediction of future actions of the opponent. In doing so, it also allows for adapting one's own behavior accordingly. Case-based reasoning has been frequently used as a technique for opponent modeling in multi-agent games [4], including the domain of robotic soccer [13, 1].

Using CBR, in [13] the authors make their simulated soccer agents recognize currently executed higher-lever behaviors of the currently ball leading opponent player. These include passing, dribbling, goal-kicking and clearing. These higher-level behaviors correspond to what we refer to as skills, i.e. action sequences that are executed over a dozen or more time steps. This longer time horizon allows the agent to take appropriate counter measures.

The authors of [11] also deal with the case-based recognition of skills (higher-level behaviors, to be exact the shoot-on-goal skill) executed by an opponent soccer player, focusing on the appropriate adjustment of the similarity measure employed. While we do also think opponent modeling is useful for counteracting adversary agents, we, however, disagree with these authors claiming that "in a complex domain such as RoboCup it is infeasible to predict an agent's behavior in terms of primitive actions". Instead we will show empirically that such a low-level action prediction can be achieved during an on-going play using case-based methods. To this end, the work presented in this paper is also related to the work by Floyd et al. [5] whose goal is to mimic the overall behavior of entire soccer simulation teams, be it for the purpose of analysis or for rapid prototyping when developing one's own team, without putting too much emphasis on whether the imitators yield competitive behavior.

2.3 Related Previous Work

What is the use of knowing exactly whether an opponent is going to execute a $kick(40, 30°)$ or a $dash(80, 0°)$ low-level action next? This piece of information certainly does not reveal whether this opponent's intention is to play a pass (and to which teammate) in the near future or to dribble along. Clearly, for answering questions like that the approaches listed in the previous section are potentially more useful. But knowing the opposing agent's next low-level actions is extremely useful, when knowing the *next state on the field* is essential (cf. Figure 1 for an illustration).

In [7], we considered a soccer simulation defense scenario of crucial importance: We focused on situations where one of our players had to interfere and disturb an opponent ball leading player in order to scotch the opponent team's attack at an early stage and, even better, to eventually conquer the ball initiating a counter attack. We employed a reinforcement learning (RL) methodology that enabled our agents to autonomously acquire such an aggressive duel behavior, and we successfully embedded it into our soccer simulation team's defensive strategy. So, the goal was to learn a so-called "duelling skill" (i.e. a higher-level behavior which in the end yields a sequence of low-level actions) which made our agent conquer the ball from the ball-leading opponent.

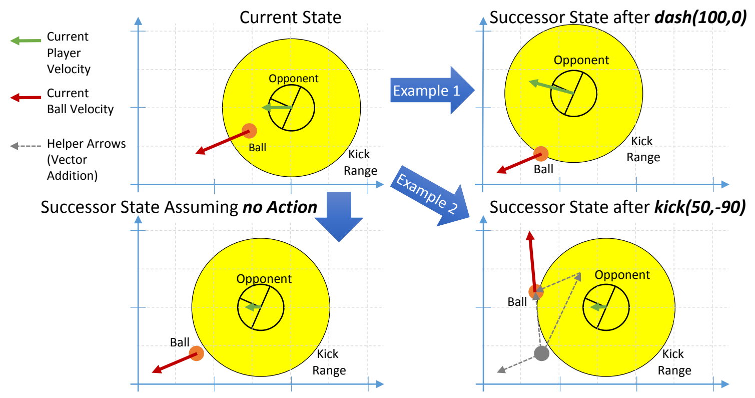
Current Player Velocity
Current Ball Velocity
Helper Arrows (Vector Addition)
Current State
Opponent
Ball
Kick Range
Example 1
Example 2
Successor State after *dash(100,0)*
Successor State Assuming *no Action*
Successor State after *kick(50,-90)*

Fig. 1. In the top-left we see the current state of an opponent agent in ball possession. If we assume, this agent does not take any low-level action, then the resulting successor state looks like the one in the bottom left figure: Player and ball have moved according to their recent velocities while the magnitude of the velocity vectors have decayed according to the rules of the simulation. How different the successor state may look, if the opponent, however, does take an action (which is most likely), is shown in the right figures. In example 1 (top) the agent accelerates full power along its current body orientation, while the ball is not affected. In example 2, the player kicks the ball with 50% power into -90° relative to its current body orientation which yields a resulting ball velocity vector as shown in the bottom right.

An important feature of the soccer simulation domain is that the *model* of the environment is known. This means given, for example, the current position and velocity of the ball, it is possible for any agent to calculate the position of the ball in the next time step (because the implementation of the physical simulation by the Soccer Server is open source[2]). As a second example, when knowing one's own current position, velocity and body angle, and issuing a $turn(68°)$ low-level action, the agent can precalculate the position, velocity and body orientation it will have in the next step. Or, finally, when the agent knows the position and velocity of the ball, it can precalculate the ball's position and velocity in the next step, for any $kick(x, \alpha)$ command that it might issue.

Knowing the model of the environment (formally, the transition function $p : S \times A \times S \to \mathbb{R}$ where $p(s, a, s')$ tells the probability to end up in the next state s' when executing action a in the current state s), is extremely advantageous in reinforcement learning, since then model-based instead of model-free

[2] In practice, the Soccer Server adds some noise to all low-level actions executed, but this is of minor importance to our concerns.

learning algorithms can be applied which typically comes along with a pleasant simplification of the learning task.

So, in soccer simulation the transition function p (model of the environment) is given since the way the Soccer Server simulates a soccer match is known. In the above-mentioned "duelling task", however, the situation is aggravated: Here, we have to consider the influence of an opponent whose next actions cannot be controlled. In [7], we stated that the opponent's next (low-level) actions can "hardly be predicted [which] makes it impossible to accurately anticipate the successor state", knowing which is, as pointed out, extremely useful in RL. In the paper at hand, we will show that predicting the opponent's next low-level action might be easier than expected. As a consequence,

- in [7] we had to rely on a rough approximation of p, that merely takes into account that part of the state that can be influenced directly by the learning agent and which ignored the part of the future state which is under direct control of the ball-leading opponent (e.g. the position of the ball in the next state). This corresponded to the unrealistic assumption of an opponent that never takes any action (cf. Figure 1, bottom left).
- in future work we can employ a much more accurate version of p based on the case-based prediction of the opponent's low-level actions described in the next section.

3 Case-Based Prediction of Low-Level Actions

In what follows, we differentiate between an opponent (OPP) agent whose next low-level actions are to be predicted as well as (our) case-based agent (CBA) that essentially observes the opponent and that is going to build up a case base to be used for the prediction of OPP's actions.

When approaching the opponent modeling problem as a case-based reasoning problem, the goal of the case-based agent is to correctly predict the next action of its opponent given a characterization of the current situation. Stated differently, the current state of the system (including the case-based agent itself, its opponent as well as all other relevant objects) represents a new query q. CBA's case base $\mathcal{C}$ is made up of cases $c = (p, s)$ whose problem parts p correspond to other, older situations and corresponding solutions s which describe the action OPP has taken in situation p. Next, the case-based agent will search its case base for that case $\hat{c} = (\hat{p}, \hat{s}) \in \mathcal{C}$ (or for a set of k such cases) whose problem part features the highest similarity to the current problem q and employ its solution $\hat{s}$ as the current prediction of the opponent's next action.

3.1 Problem Modeling

In the context of this case study we focus on dribbling opponents, i.e. the opponent has the ball in its kick range and moves along while keeping the ball within its kick range all the time. Stated differently, we focus on situations where OPP behaves according to some "dribble skill" (a higher-level dribbling behavior). Consequently, OPP executes in each time step one of the three actions $kick(x,\alpha)$, $dash(x,\alpha)$, or $turn(\alpha)$. The standard rules of the simulation allow x to be from $[0,100]$ and α from $(-180°,180°]$ for kicks and turns. For dashes, α is allowed to take one out of eight values (multiples of 45°). In almost all cases occurring during normal play, however, a dribbling player is heading more or less towards his opponent's goal which is why the execution of low-level turn actions represents an exceptional case. Therefore, for the time being, we leave turn actions aside and focus on the correct prediction of dashes and kicks including their parameters x and α.

Case Structure The state of the dribbling opponent (OPP) can be characterized by the x and y position of the ball within its kick range ($pos_{b,x}$ and $pos_{b,y}$) relative to the center of OPP as well as the x and y components of the ball's velocity ($vel_{b,x}$ and $vel_{b,y}$; of course, these values are also relative to OPP's body orientation). Moreover, OPP's x and y velocities ($vel_{p,x}$ and $vel_{p,y}$) are of relevance, making six features in total. The seventh relevant feature, OPP's current body orientation θ_p can be skipped due to the arguments mentioned in the preceding paragraph. Furthermore, the y component of OPP's velocity vector $vel_{p,y}$ is, in general, zero since a dribbling player almost always dribbles along its current body orientation. While this allows us to also skip the sixth feature, we remove a redundancy in the remaining features (and thus arrive at only four of them) by changing to a relative state description that incorporates some background knowledge[3] from the simulation. Hence, the problem part p of a case $c=(p,s)$ is a four-tuple $p=(pos_{bnx},pos_{bny},vel_{bnx},vel_{bny})$ with

$$\begin{aligned}
pos_{bnx} &= pos_{b,x} + 0.94 \cdot vel_{b,x} - 0.4 \cdot vel_{p,x}\\
pos_{bny} &= pos_{b,y} + 0.94 \cdot vel_{b,y} - 0.4 \cdot vel_{p,y}\\
vel_{pnx} &= 0.94 \cdot vel_{b,x} - 0.4 \cdot vel_{p,x}\\
vel_{pny} &= 0.94 \cdot vel_{b,y} - 0.4 \cdot vel_{p,y}
\end{aligned}$$

where all components characterize the next state as it would arise, if the agent would not take any action (cf. Figure 1).

The solution s of a case $c=(p,s)$ consists of a class label l ("dash" or "kick") as well as two accompanying real-valued attributes for the power x and angle α parameters of the respective action. Thus, the solution is a triple $s=(l,x,\alpha)$.

3.2 Implementing the CBR Cycle

The case-based agent CBA observes his opponent OPP and, in doing so, builds up its case base. Note that all agents in soccer simulation act on incomplete and uncertain information. Their visual input consists of noisy information about objects in their limited field of vision. However, if the observed opponents are

[3] Knowledge about how the Soccer Server decays objects.

near and constantly focused at, CBA is provided with sufficiently accurate visual state information. In order to fill the contents of the cases' solution parts, however, CBA must apply inverse dynamics of the soccer simulation. If CBA, for example, observes that the velocity vector of the ball has been changed at time $t+1$ as in the bottom right part of Figure 1, then it can conclude that OPP has executed a $kick(50, -90°)$ action at time t and can use that information to complete the case it created at time step t.

With ongoing observation of dribbling opponent players, CBA's case base $\mathcal{C}$ grows and becomes more and more competent. Therefore, after $|\mathcal{C}|$ exceeds some threshold, CBA can utilize its case base and query it to find a prediction of the action that OPP is going to take in the current time step.

Retrieval and Similarity Measures We model the problem similarity using the local-global principle [2] with identical local similarity measures for all problem attributes, $sim_i(q_i, c_i) = (\frac{q_i - c_i}{max_i - min_i})^2$, where min_i and max_i denote the minimum and maximum value of the domain of the ith feature. The global similarity is formed as a weighted average according to

$$Sim(q, c) = \frac{\sum_{i=1}^{n} w_i \cdot sim_i(q_i, c_i)}{\sum_{i=1}^{n} w_i}$$

where attributes pos_{bnx} and pos_{bny} are weighted twice as much as vel_{pnx} and vel_{pny}.

We perform standard k-nearest neighbor retrieval using a value of $k = 3$ in our experiments. When predicting the class of the solution, i.e. the type of the low-level action (dash or kick), we apply a majority voting, and for the prediction of the action parameters (x and α) we calculate the average over all cases among the k nearest neighbors whose class label matches the majority class.

4 Experimental Results

To evaluate our approach we selected a set of contemporary soccer simulation team binaries (top teams from recent years) and made one of their agents (OPP) dribble for up to 2000 simulated time steps[4]. Our case-based agent CBA was allowed meanwhile to observe OPP and build up its case base. We evaluated CBA's performance in predicting OPP's low-level actions for increasing case base sizes.

Figure 2 visualizes the learning progress against an opponent agent from team WrightEagle. As can be seen, compelling accuracies can be achieved for both, the correctness of the type of the action (dash or kick) as well as for the belonging action parameters. Interestingly, even the relative power / angle of kicks can be predicted quite reliably with a remaining absolute error of less than ten percent / ten degrees.

[4] Duration of a regular match is 6000 time steps.

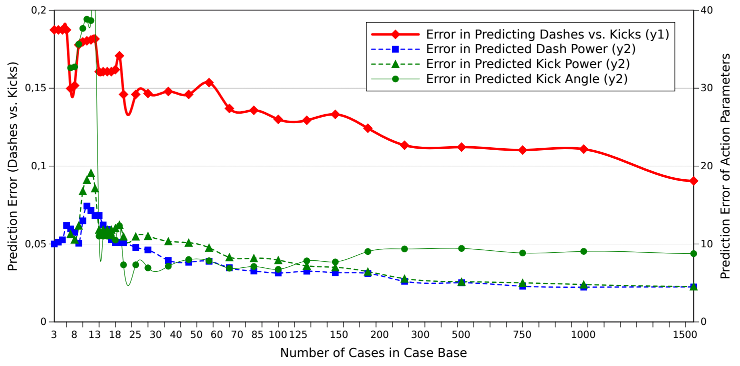

Fig. 2. Progress of CBA's competence in predicting the next low-level actions of a dribbling opponent agent from team WrightEagle. A case base of about 1500 cases was created during the course of 2000 simulated time steps.

Figure 3 focuses on different opponent agents and highlights the fact that a substantial improvement in action type prediction accuracy can be obtained with as little as 100 collected cases. Baseline to all these classification experiments is the error of the "trivial" classifier (black) that predicts each action type to be of the majority class. The right part of Figure 3 presents the recall of both, dash and kick actions. Apparently, dashes are somewhat easier to predict than kicks where, however, the recall of the latter is still above 65% for each of the opponent agents considered.

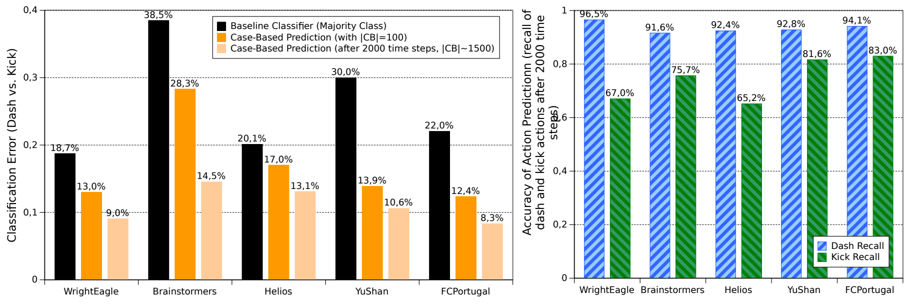

Fig. 3. Left: Case-based prediction of the type (dash or kick) of the next low-level action for opponents from different teams. Right: Recall, i.e. share of dashes that were correctly predicted as dashes and kicks that were correctly predicted as kicks.

In Figure 4, we present aggregate numbers (averages over all opponents) that emphasize how accurately the parameters of an action were predicted, given that the type of the action could be identified correctly. To this end, dash angles α are disregarded since more than 99.2% of all dash actions performed used

$\alpha = 0$, i.e. yielded a dash forward. Here, we compare (a) an "early" case base with only 10 cases, (b) an intermediate one[5] with $|\mathcal{C}| = 100$ as well as (c) one that has resulted from 2000 simulated time steps and contains circa 1500 cases. Interestingly, even in (a) comparatively low errors can be obtained. In (b) and (c), however, the resulting average absolute prediction errors become really competitive (± 2.9 for dash powers x with $x \in [0, 100]$, ± 6.3 for kick powers x with $x \in [0, 100]$, and $\pm 19.7°$ for kick angles α with $\alpha \in [0°, 360°]$).

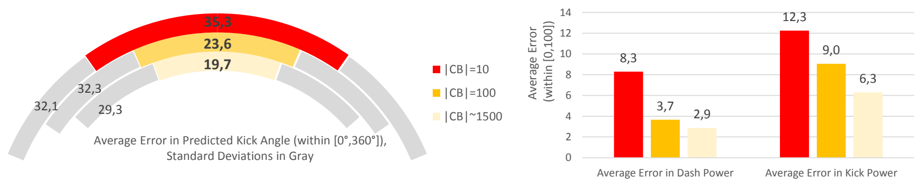

Fig. 4. Exactness of the prediction of the action parameters for different stages during ongoing learning (10, 100, and ≈1500 cases in the case base). Left: Average error of the predicted angle of a kick action. Right: Average error of the predicted relative power of a kick action and dash action (averages over agents from all opponent teams considered).

5 Discussion and Conclusion

Clearly, dribbling opponents are very likely to behave differently when they are disturbed, tackled, or attacked by a nearby opponent. Therefore, the approach presented needs to be extended to "duelling situations" as they frequently arise in real matches. For example, in scenarios like that the dribbler will presumably not just dribble straight ahead, but also frequently execute turn actions (e.g. in order to dribble around its disturber). This represents an aggravation of the action type prediction problem since then three instead of two classes of actions must be considered (dask, kick, turn).

While the case study presented focused solely on non-attacked dribbling opponents, this approach can easily be transferred to related or similar situations where knowing the opponent's next move is crucial, too. This includes, but is not limited to the behavior of an opponent striker when trying to perform a shoot onto the goal (which typically requires a couple of time steps), the behavior of the shooter as well as the goal keeper during penalty shoot-outs, or the positioning behavior of the opponent goalie (anticipating which can be essential for the striker).

As a next step, we plan to combine the presented case-based prediction of low-level actions with the reinforcement learning-based acquisition of agent behaviors as outlined in Section 2.3. This involves, first, solving the aggravated problem of

[5] A case base of a size of about 100 to 500 cases can easily be created within the first half of a match for most players.

correctly recognizing three different classes of low-level actions mentioned at the beginning of this section and, second, a proper utilization of the thereby obtained improved model when learning a higher-level duelling skill using RL. Another interesting direction for future work is the idea to let CBA start off with some opponent model in form of a case-base acquired offline (against, for example, an older version of the team to be faced) and, using appropriate techniques for case base maintenance, to successively replace old experience by new experience gained online during the current match.

References

1. Ahmadi, M., Keighobadi-Lamjiri, A., Nevisi, M., Habibi, J., Badie, K.: Using a Two-Layered Case-Based Reasoning for Prediction in Soccer Coach. In: Proceedings of the International Conference of Machine Learning; Models, Technologies and Applications (MLMTA'03). pp. 181–185. CSREA Press (2003)
2. Bergmann, R., Richter, M., Schmitt, S., Stahl, A., Vollrath, I.: Utility-Oriented Matching: A New Research Direction for Case-Based Reasoning. In: Proceedings of the 9th German Workshop on Case-Based Reasoning. pp. 264–274 (2001)
3. Carvalho, A., Cheriton, D.: Reinforcement Learning for the Soccer Dribbling Task. In: Proceedings of IEEE Conference on Computational Intelligence and Games (CIG). pp. 95–101. Seoul, South Korea (2011)
4. Denzinger, J., Hamdan, J.: Improving Modeling of Other Agents Using Stereotypes and Compactification of Observations. In: Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 1414–1415. New York, USA (2004)
5. Floyd, M., Esfandiari, B., Lam, K.: A Case-Based Reasoning Approach to Imitating RoboCup Players. In: Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference. pp. 251–256. Coconut Grove, USA (2008)
6. Gabel, T., Riedmiller, M.: CBR for State Value Function Approximation in Reinforcement Learning. In: Proceedings of the 6th International Conference on Case-Based Reasoning (ICCBR 2005). pp. 206–221. Springer, Chicago, USA (2005)
7. Gabel, T., Riedmiller, M., Trost, F.: A Case Study on Improving Defense Behavior in Soccer Simulation 2D: The NeuroHassle Approach. In: L. Iocchi, H. Matsubara, A. Weitzenfeld, C. Zhou, editors, RoboCup 2008: Robot Soccer World Cup XII, LNCS. pp. 61–72. Springer, Suzhou, China (2008)
8. Kalyanakrishnan, S., Liu, Y., Stone, P.: Half Field Offense in RoboCup Soccer: A Multiagent Reinforcement Learning Case Study. In: RoboCup-2006: Robot Soccer World Cup X. pp. 72–85. Springer Verlag, Berlin (2007)
9. Kuhlmann, G., Stone, P.: Progress in Learning 3 vs. 2 Keepaway. In: RoboCup-2003: Robot Soccer World Cup VII. pp. 694–702. Springer, Berlin (2004)
10. Noda, I., Matsubara, H., Hiraki, K., Frank, I.: Soccer Server: A Tool for Research on Multi-Agent Systems. Applied Artificial Intelligence 12(2-3), 233–250 (1998)
11. Steffens, T.: Similarity-Based Opponent Modelling Using Imperfect Domain Theories. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG05)
12. Veloso, M., Balch, T., Stone, P.: RoboCup 2001: The Fifth Robotic Soccer World Championships. AI Magazine 1(23), 55–68 (2002)
13. Wendler, J., Bach, J.: Recognizing and Predicting Agent Behavior with Case-Based Reasoning. In: D. Polani and A. Bonarini and B. Browning (editors), RoboCup 2003: Robot Soccer World Cup VII. pp. 729–728. Padova, Italy (2004)