Team Playing Behavior in Robot Soccer: A Case-Based Reasoning Approach

Raquel Ros[1], Ramon López de Màntaras[1], Josep Lluís Arcos[1] and Manuela Veloso[2]

[1] IIIA - Artificial Intelligence Research Institute
CSIC - Spanish Council for Scientific Research
Campus UAB, 08193 Barcelona, Spain
[2] Computer Science Department, Carnegie Mellon University
Pittsburgh, PA 15213 USA
{ros,mantaras,arcos}@iiia.csic.es, veloso@cs.cmu.edu

Abstract. This paper presents extensions and improvements of previous work, where we defined a CBR system for action selection in the robot soccer domain. We show empirical results obtained with real robots, comparing our team playing approach with an individualist approach.

1 Introduction

Action selection in robotics is a challenging task: the robot has to reason about its world believes (the state of the environment), and rationally act in consequence in order to complete a task (typically divided in subtasks). Moreover, in the case of a robot team, robots must agree on the decisions made (who and what to do to complete the subtasks), jointly execute the actions, and coordinate among them to successfully perform the task. Working with real robots has additional difficulties that must be considered while developing their reasoning system. Thus, the reasoning engine must be capable of dealing with high uncertainty in the robot's perception (incoming information of the world), and be robust in case of failure, since the outcomes of the actions performed are unpredictable. Not to mention that decision must be made on real time and in our case, with limited computational resources.

This paper presents extensions and improvements of previous work [7], where we defined a CBR system for the robot soccer domain. Given a state of the environment (snapshot of the game) the aim of the approach is to define the sequence of actions (gameplays) the robots should perform during a game. In this first attempt, we presented a preliminary model of our CBR system and we also showed initial experiments with one robot in a simulated environment.

Because of the high uncertainty in our domain, in [5] we introduced the concept of "scope" of a case. With this concept we refer to regions in the field where a case should be retrieved. Dealing with regions is much more intuitive and feasible than dealing with points in the field, since we are interested in observing if the ball or a robot is in a given region, rather than if it is in an exact position.

Once we verified the effectiveness of this new concept, we have now introduced it in the system as part of the description of a case.

In several different domains it has been proved that teamwork improves the performance of a task. Robot soccer is one of these domains. Having a single player running across the field with the ball may result in success if no problems arise during the performance. But, what if while attempting to reach the attacking goal it loses the ball? It could be a perfect opportunity for an opponent to take the ball. Having teammates that can help during the task is essential to increase robustness in case of failure. Therefore, in [6] we presented our first multi-robot case-based approach, where a fixed robot was in charge of the reasoning process (retrieving cases) and coordinating the execution of the case with the rest of the teammates. In order to increase the robustness of our multi-robot approach, we now present an additional mechanism, where the best candidate among the available robots is selected as the coordinator for each cycle of the CBR process.

Based on the successful results obtained in our previous work, we have extended our system including the opponents. Due to the incorporation of new cases with teammates and opponents, the complexity of the case base has also increased. Thus, in this paper we also present a new representation of the case base to facilitate the access during retrieval. Furthermore, since now we are working with teammates, we have emphasized the cooperative behavior of our system, i.e. we prefer to retrieve a case with teammates, than a case with a single robot. To this end, we define a retrieval process that prioritizes cases with multiple robots, rather than cases with a single robot.

In this paper we present the current version of our system and the empirical results obtained with real robots. We compare our approach with an individualist approach in two scenarios with a team of two robots. The aim of the experiments is to prove that the performance of the robots using the extended system is more cooperative, and hence, more robust in case of failure since more than one robot participates as much as possible during the execution of the task.

We focus our work on the the Four-Legged League (RoboCup). Teams consist of four Sony AIBO robots. The robots operate fully autonomously and they can communicate with each other by wireless. The field dimensions are 6m long and 4m wide and it represents a Cartesian plane. There are two goals (cyan and yellow) and four colored markers the robots use to localize themselves in the field. There are two teams in a game: a red team and a blue team. Figure 1 shows a snapshot of the simulated field. A game consists of two parts of 10 minutes each. At any point of the game, if the score difference is greater than 10 points the game ends. For more details on the official rules of the game refer the RoboCup Four-Legged League Rule Book.

The paper is organized as follows. Section 2 reviews the related work. In Section 3 we define the case structure and related details and the retrieval process is described in Section 4. We present the multi-robot approach and the case reuse in Section 5 and we show empirical results in Section 6. Finally, we conclude the paper in Section 7.

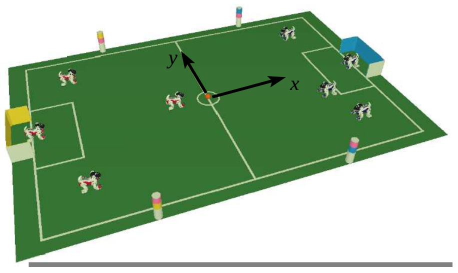

Fig. 1. Snapshot of the field (image extracted from the Official Rule Book).

2 Related Work

Researchers have focused their work on different techniques to model the agents' behaviors in the action selection domain. In the CBR field, Wendler et al. [8] describe an approach in the Simulation League to select actions based on previously collected experiences encoded as cases. Thus, many parameters they take into account are not considered in our domain, and also they do not have to deal with the major problems involved when working with real robots. Marling et al. [3] introduce three CBR prototypes in their robots team (RoboCats, in the Small Size League): the first prototype focused on positioning the goalie; the second one, on selecting team formations; and the third one, on recognizing game states. All three systems are mainly based on taking snapshots of the game and extracting features from the positions of the robots during the game.

Lam et al. [1] focus their research on learning from observation. The aim of this technique is to model agents that learn from observing other agents and imitating their behavior. As in CBR, the learning agent selects the most similar past observed situation with respect to the current problem and then reproduces the solution performed at that time. The main difference between these approaches is that the learning agent is not able to improve the observed agent since there is no feedback in the model. Although our work does not include yet the revise step, the main differences with this work are: the number of agents implied in the scenes (we include teammates which interact among them); the solution of the problem (we deal with a sequence of actions for each teammate instead of a single action in [1]); and the objects locations (robots and ball are within fixed regions of field in [1], whereas we deal with variable regions).

In the Simulation League, Riedmiller et al. [4] focus their work on Reinforcement Learning applied to two different levels: moving level and tactical level. The former refers to learning a specific move (learning to kick). While the latter refers to which move should be applied at a certain point (*pass the ball*). Lattner et al. [2] present an approach that creates patterns based on the qualitative information of the environment. The result of learning is a set of prediction rules that give information about what (future) actions or situations might occur with some probability if certain preconditions satisfy. Patterns can be generalized, as well as specialized.

3 Case Definition

A case represents a snapshot of the environment at a given time from a single robot point of view. We call this robot the *reference* robot, since the information in the case is based on its perception and internal state (its believes). The case definition is composed of three parts: the problem description, which corresponds to the state of the game; the knowledge description, which contains additional information used to retrieve the case; and finally, the solution description, which indicates the sequence of actions the robots should perform to solve the problem. We formally define a case as a 3-tuple:

$$case = ((R, B, G, Tm, Opp, t, S), K, A)$$

where:

1. R: robot's relative position (x_R, y_R) with respect to the ball and heading θ.

$$x_R \in [-2700..2700]\text{mm} \quad y_R \in [-1800..1800]\text{mm} \quad \theta \in [0..360)\text{degrees}$$

2. B: ball's global position (x_B, y_B)

$$x_B \in [-2700..2700]\text{mm} \quad y_B \in [-1800..1800]\text{mm}$$

3. G: defending goal

$$G \in \{\text{cyan}, \text{yellow}\}$$

4. Tm: teammates' relative positions with respect to the ball.

$$Tm = \{tm_1, tm_2 ..., tm_n\}$$

where tm_i is a point (x, y) and $n = 1..3$ for teams of 4 robots. This set could be empty for cases where no teammates are implied in the case solution.

5. Opp: opponents' relative positions with respect to the ball.

$$Opp = \{opp_1, opp_2, ..., opp_m\}$$

where opp_i is a point (x, y) and $m = 1..4$ for teams of 4 robots. This set could be empty for cases where no opponents are described in the case.

6. t: timing of the match. Two halves parts of 10 min.

$$t \in [0..20]\text{min}, t \in \mathbb{N}$$

7. S: difference between the goals scored by our team and the opponent's team. The maximum difference allowed is 10. The sign indicates if the team is losing (negative) or winning (positive).

$$S \in [-10..10]$$

8. K: scope of the case. We define the scope as the regions of the field within which the ball and the opponents should be positioned in order to retrieve that case. We represent the scope as ellipses centered on the ball's and opponents' positions indicated in the problem description.

$$K = (\tau_x^B, \tau_y^B, \tau_x^1, \tau_y^1, \ldots, \tau_x^m, \tau_y^m)$$

   where τ_x^B and τ_y^B correspond to the x and y radius of the ball's scope, τ_x^i and τ_y^i, to the radius of opponent i's scope ($i = 1..m$). If there is no opponent in the case, then we do not include the τ values for opponents.
9. A: sequence of actions (gameplays) each robot performs.

$$A = \{tm_0 : [a_{01}, a_{02}, \ldots, a_{0p_0}], \ldots, tm_n : [a_{n1}, a_{n2}, \ldots, a_{np_n}]\}$$

   where $n = 0..3$ is the Id of the robot, and p_i the number of actions teammate tm_i performs (tm_0 corresponds to the *reference* robot). The actions are either individual actions, such as "get the ball and kick", or joint actions, such as "get the ball and pass it to robot tm_i".

Case Description Details

Problem description. Each robot constantly reports its position to the rest of the robots in the same team. Thus, the *reference* robot can update Tm at every time step. Regarding the opponents' positions, the robots may include a vision processing system to detect them. However, in this work we do not use this system because it is not robust enough. The purpose of this research is to study the performance of the CBR approach, and not to improve robustness to the perception system. Therefore, to test our system independently from vision issues, the robots from the opponent team also report their positions to all the robots in the field. Since we are only interested on the opponents near the ball (an opponent far from the ball does not take part in the immediate gameplay) the *reference* robot only considers the existence of an opponent (active opponent) if it is within a given radius from the ball's position.

Knowledge description. We are more interested in defining qualitative positions of the ball and opponents rather than using precise positions. Hence, describing the scope of cases based on ellipses is beneficial in two aspects: first, because of the high degree of uncertainty in our domain, dealing with exact positions is not feasible; and second, we can easily describe the opponents' positions with respect to the ball by means of qualitative positions. Figure 2a shows in a section of the field an example of the ball's and opponent's scope of a given case. Given a new problem, if the ball and the opponent are within the scopes, i.e. the ball is within the solid ellipse, and the opponent in front of it, the case would be considered as a potential solution. Note that the scope of the opponent is computed with respect to the actual ball's position. Figure 2b illustrates an example of a new problem to solve, where the ball is within the ball's scope, and the scope of the

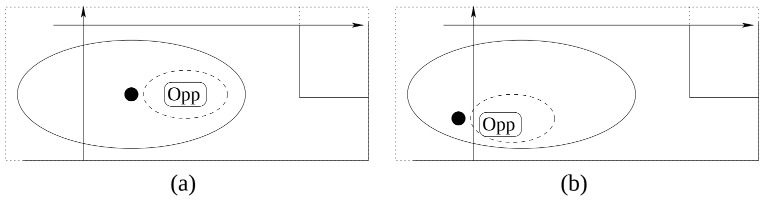

Fig. 2. (a) Example of the scope of a case. The black circle represents the ball and the rectangle represents the opponent. The ellipses correspond to the ball's scope (solid ellipse) and the opponent's scope (dashed ellipse). (b) Example of a problem. The scope of the opponent is translated with respect to the ball.

opponent is located with respect to this position. To solve this problem, the case shown in Figure 2a would be retrieved.

The initial scopes of the cases (values of τ_x and τ_y for the ball and opponents) are initially given by hand when creating the cases and then automatically adjusted by means of a learning mechanism presented in [5].

Solution description. Although actions have different durations, through the execution of joint actions there is no need of explicit action synchronization between robots, nor to specify timings to actions. This is so, because each action corresponds to low level behaviors which are triggered when a set of preconditions are fulfilled. For instance, a pass between two robots corresponds to two sequences: "pass the ball" and "wait for ball". The first robot (the one that initiates the pass) gets the ball and then kicks it towards the second robot. Meanwhile, the robot receiving the ball remains in its position until the ball is close enough to it. Once the ball approaches the second robot, it will catch the ball and continue with whatever action is indicated in the gameplay (such as "kick to goal").

4 Case Retrieval

A case can be retrieved if we can modify part of the current problem description in order to adapt it to the description of the case. We separate the features of the problem description in two sets: *controllable* indices and *non-controllable* indices. The former ones refer to the *reference* robot and teammates (since they can move to more appropriate positions), while the latter refers to the ball, opponents, defending goal, time and score difference (which we cannot directly modify). The modification of the controllable features leads to a planning process where the system has to decide how to reach the positions of the robots indicated in the retrieved case in order to reuse its solution.

We compute two different measures to retrieve the most similar case when solving a new problem: the *similarity* for the *non-controllable* features and the *cost* for the *controllable* features. We briefly describe them next.

Similarity function This measure indicates how similar the non-controllable features are between the problem and the case. We define different functions for each domain of features and we then compute the overall similarity using the harmonic mean of the individual similarities.

The similarity sim_B for the ball and the similarities sim_{Opp_i} for the opponents are computed using a 2D Gaussian function:

$$sim(x_1, y_1, x_2, y_2) = e^{-\left(\frac{(x_1-x_2)^2}{2\tau_x^2} + \frac{(y_1-y_2)^2}{2\tau_y^2}\right)}$$

where the point (x_1, y_1) refers to either the robots' or the ball's position in the problem and (x_2, y_2) refers to the positions in the case. τ_x and τ_y correspond to the scopes (K) of either the ball or the opponents described in the case.

To compute the opponents' similarity we first must determine the correspondence between the opponents of the problem and the case, i.e. which opponent opp_i from the problem description corresponds to which opponent opp_j in the case description. Once we obtain the correspondences, we compute the similarity for each pair using the Gaussian function defined above.

We model the strategy of the game based on the time and the score difference. As time passes and depending on the score of the game, we expect a more offensive or defensive behavior. We define the strategy function as:

$$strat(t, S) = \begin{cases} \frac{t}{20(S-1)} & \text{if } S < 0 \\ \frac{t}{20} & \text{if } S = 0 \\ \frac{t}{20(S+1)} & \text{if } S > 0 \end{cases}$$

where $strat(t, S) \in [-1..1]$, with -1 meaning a very offensive strategy and 1 meaning a very defensive strategy. The similarity function for the strategies is:

$$sim_{tS}(t_1, S_1, t_2, S_2) = 1 - |strat(t_1, S_1) - strat(t_2, S_2)|$$

where t_1 and S_1 corresponds to the time and scoring features in the problem and t_2 and S_2, the features in the case.

Finally, the overall similarity is defined as:

$$sim = f(sim_B, sim_{tS}, sim_{Opp_1}, \ldots, sim_{Opp_m})$$

where f is the harmonic mean, m is the number of opponents in the case, and each argument of f corresponds to the similarity value obtained for each feature. For more details regarding the similarity functions refer to [7].

Cost function This measure computes the cost of modifying the controllable features, i.e. the cost of adapting the problem to the case. It is computed as the sum of the distances between the positions of the robots in the problem and the adapted positions specified in the case (after obtaining their correspondences).

The adapted positions correspond to the global locations where the robots should position in order to execute the solution of the case. To compute them, we transform the robots' relative coordinates to global coordinates, having the position of the ball in the problem as the reference point. Figure 3 illustrates a simple adaptation example with one robot.

Retrieving a case

Since we are working in a real time domain and because of computational limitations in the robots, it is essential to minimize the time invested during the retrieval process. To speed up the search we use an indexed list to store the cases. Thus, when a new problem enters the system we can easily access the subset of cases (CB^s) we are interested in by indexing the case base using the value of the defending goal (yellow or cyan) and the number of opponents involved in each case. Searching in the rest of the case base is useless since those cases will not match the current problem at all. In Section 6 we will show some examples of cases.

After computing the similarities between the problem and the cases in the subset CB^s, we obtain a list of potential cases. From this list, we compute the cost for each case and select those cases that have a cost lower than a given threshold. From this list of potential cases (PC) we must select one case for the reuse step.

We consider a compromise between the similarity degree between the problem and the case and the cost of adapting the problem to the case. Moreover, since we are working in a multi-robot domain (teams of robots), we are also interested in stimulating cooperation between them as much as possible. Thus, given two candidate cases, one described with a single robot, and the other, with two robots that cooperate during the execution of the solution, the system would select the second case as the retrieved case (although it might have a lower similarity).

Therefore, given the list of potential cases (PC), we first classify the cases based on the number of robots described in the case (number of teammates, n, plus one -the *reference* robot-). Each subset is further classified into four lists based on different similarity intervals: $H = [0.8, 1.0]$, $h = [0.6, 0.8)$, $l = [0.4, 0.6)$ and $L = (0.0, 0.4)$. Finally, each list is sorted based on the cost, where the first case of the list corresponds to the case with lower cost. Formally, we define the list of potential cases PC as:

$$PC = [[sim_H^{n+1}, sim_h^{n+1}, sim_l^{n+1}, sim_L^{n+1}], \ldots, [sim_H^1, sim_h^1, sim_l^1, sim_L^1]]$$

where $sim_s^i = [c_{s1}^i, c_{s2}^i, \ldots]$ is an ordered list of cases based on their cost (i.e. $cost(c_{s1}^i) < cost(c_{s2}^i)$); $s = \{H, h, l, L\}$ stands for the similarity interval; and $i = 1..n+1$ is the number of players in the case. The retrieved case corresponds to the first element of the flatten[3] list PC: $ret\_case = \text{first}(\text{flat}(PC))$.

In summary, when a new problem enters the system, the system retrieves a case maximizing both, the number of players implied in the solution and the similarity, while minimizing the cost.

5 Multi-Robot Architecture and Case Execution

Next we describe the architecture for our multi-robot system integrating the retrieval and reuse steps of the CBR approach. The multi-robot system is composed of n robots. All robots interact with the environment and among them, i.e.

[3] We define a flatten list as a list with one single level, i.e. no nested lists.

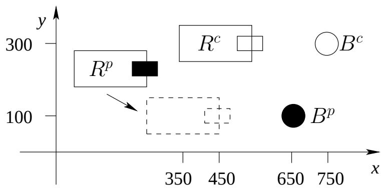

Fig. 3. Case description (R^c, B^c), and current problem description (R^p, B^p). The robot in dashed lines represents the adapted position of the robot with respect to the ball's position described in the problem.

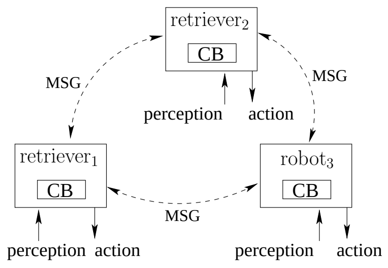

Fig. 4. Case-based multi-robot architecture for $n = 3$ robots and $k = 2$ *retrievers*.

they perceive the world, they perform actions and they send messages (MSG) to each other to coordinate and to exchange information about their internal states.

We distinguish a subset of k $(1 \leq k \leq n)$ robots, called *retrievers*. These robots are capable of retrieving cases as new problems arise. All robots have a copy of the same case base so they can gather the information needed during the case reuse. Figure 4 shows the architecture described. Given a new problem to solve, the first step of the process is to decide which of the *retriever* robots is going to actually retrieve a case to solve it (since only one case can be executed at a time). We believe that the most appropriate robot to perform this task should be the one that has the most accurate information about the environment. From the set of features described in a case, the only feature that might have different values from one robot to another, is the ball's position. Moreover, this is the most important feature in order to retrieve the correct case and we must ensure as less uncertainty as possible. The remaining features are either shared among the robots, or given by an external system, i.e. defending goal, the score and time of the game. Therefore, we propose that the robot retrieving the case should be the one closer to the ball, since its information will be the most accurate (the further a robot is from an object, the higher the uncertainty about the object's information). From now on, we will refer to this robot as the *coordinator*.

Since we are working with a distributed system, the robots may have different information about each other at a given time. Their believes about the state of the world are constantly updated. They are also constantly sending messages about their current internal states (position, ball's position, etc.) to the rest of the robots. As a consequence, we cannot ensure that all robots agree on who is the one closer to the ball at a given time. To solve this issue, only one robot is responsible for selecting the *coordinator*. In order to have a robust system (robots may crash, or be removed due to a penalty), the robot performing this task is always the one with lower Id among those present in the game (since the robots always have the same Id). Once it selects the *coordinator*, it sends a message to all the robots indicating the Id of the new *coordinator*.

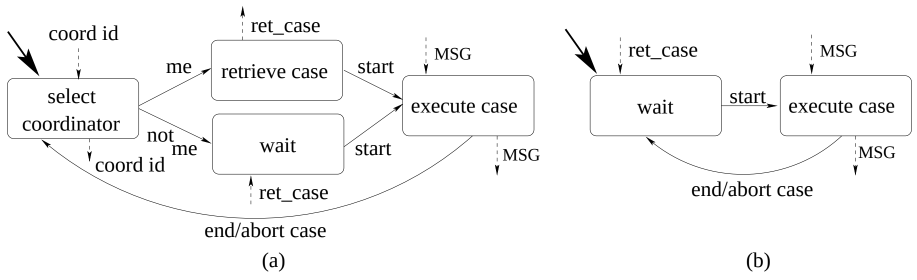

Fig. 5. Finite state machine for (a) the *retriever*'s behavior (b) rest of the robots.

After the *coordinator* is selected, it retrieves a case according to the process described in Section 4 and informs the rest of the team which case to reuse. It also informs the correspondences between the robots in the current problem and the robots in the retrieved case (so they can know which actions to execute accessing their case bases).

At this point the case execution begins. Firstly, all robots that take part of the solution of the case (including the *coordinator*) move to their adapted positions (computed as showed in Section 4). Once they reach them, they send a message to the *coordinator* in order to synchronize the beginning of the gameplays execution with the rest of the robots. Next, they all execute their actions until ending their sequences. Finally, they report the *coordinator* that they finished the execution and wait for the rest of the robots to end. When the *coordinator* receives all messages, it informs the robots so they all go back to the initial state of the process, i.e. selecting a new *coordinator*, retrieving a case and executing its solution. Figure 5 depicts the finite state machines for (a) the *retrievers*' behavior and (b), for the rest of the robots.

The execution of a case may be aborted at any moment if any of the robots either detects that the retrieved case is not applicable anymore or an expected message does not arrive. In either case, the robot sends an aborting message to the rest of the robots so they all stop executing their actions. They once again go back to the initial state in order to restart the process. For more details on the behaviors presented refer to [6].

6 Evaluation

To evaluate the approach presented in this paper, we have compared it with a behavior-based approach. We briefly introduce the basic ideas of this approach.

The behavior-based approach consists in defining high level behaviors (state-based behaviors) the robot executes based on the state of the environment. For example, a robot defending its goal should get the ball and clear it from the defense region. It could be seen as a general rule-based approach (although it is not defined as such). The robots have different assigned roles. Based on these roles they coordinate to prevent from going towards the ball at the same time and to collide between them as they move with the ball. When a robot decides

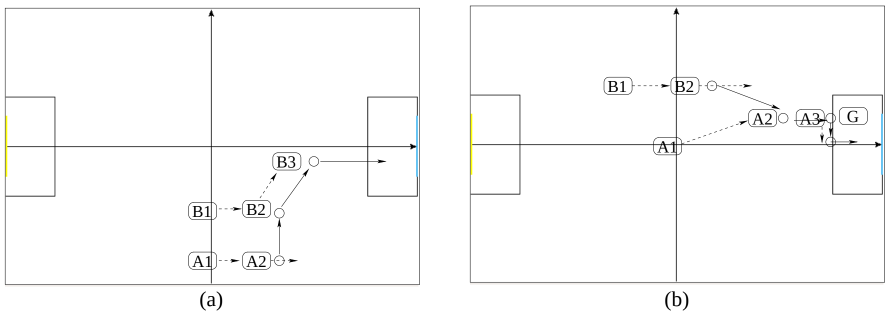

Fig. 6. The letters correspond to the robot (A, B and G), the numbers to the time step of the execution (1, 2 and 3) and the arrows represent the ball's and robots' movement, solid and dashed respectively. (a) Scenario 1 using the case-based approach: "multiple right side" case followed by "single right middle" case. (b) Scenario 2 using the case-based approach: "multiple left middle" case followed by "single goalie front" case.

to go after the ball it informs its teammates so they try to move away from its trajectory and eventually, the possible trajectory of the ball to avoid intercepting it. The roles are also used to maintain the robot's positions within certain regions of the field. Therefore, the robots can be organized in different layouts on the whole field as needed.

There are four main differences with our approach: (*i*) behaviors (rules) are applicable only if all preconditions are fully satisfied (true or false); (*ii*) there are few behaviors (rules), and therefore, they are very general; (*iii*) the approach has an implicit coordination mechanism, where coordination results as an emerging property (robots actually play always individually and passes are unintentional); and, (*iv*) the approach does not have a representation model. Thus, modifying the behaviors results in a very tedious task if the user is not familiar with it.

The goal of our experimentation is to prove that the resulting behavior of the robot team using our approach is more cooperative than a robot team using the behavior-based approach. In other words, our approach results in a collective or "team playing" behavior (participation of more than one robot of the same team during the execution of a task), as opposed to individual behavior (only one robot executing the task).

A trial consists on positioning the robots and the ball on the field and the robots' task is to move the ball until reaching the penalty area (rectangular box in front the attacking goal). Two sets of experiments where performed. Each set is composed of 15 trials. Figure 6a illustrates the first scenario, where two robots (A and B) initiate the task in the right side of the field (negative y). While in the second scenario, Figure 6b, two robots are positioned in the left middle side of the field, and an opponent is also included (the goalie) in the left side of the attacking goal (cyan goal). Both approaches (case-based and behavior-based) were tested in the two scenarios. Next, we describe the results obtained for each approach.

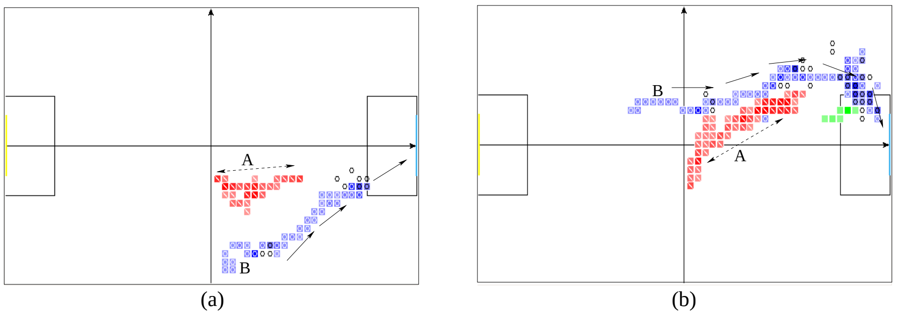

Fig. 7. Behavior-based approach. (a) Scenario 1. (b) Scenario 2.

6.1 Behavior-Based approach

During the experiments with the behavior-based approach, we observed that due to its individualistic nature, in general only one robot was implied in the execution of the task. From the 30 trials (15 for each scenario), 4 times the ball went out of field (failing the experiment). Although the remaining trials were fulfilled, a single robot was always after the ball while the second robot remained behind it to avoid intercepting either the first robot or the ball. Hence, for an external observer, the performance was lacking of teamwork or cooperation (although the robot was actually avoiding to cross the path of the first robot as part of its teamwork mechanism).

Figure 7 shows examples of the robots' paths in both scenarios. We can observe how robot B is the robot that goes after the ball constantly, while robot A remains behind it, moving back and forth to avoid robot B. This behavior may be reasonable when there are no opponents, but is not effective when there are opponents around.

6.2 Case-Based approach

Given the symmetric properties of the features of the field, for each manually created case in any of the four quadrants of the field, we can easily generate three more cases using symmetric transformations. Our case base is composed of 56 cases, even though only 14 cases were manually created.

Because of the non-deterministic nature of the real world we are working on, although the initial layout of robots and ball is always the same, different outcomes can occur after executing the same actions several times. For instance, the ball's trajectory is not exactly the same, a robot may lose the ball when attempting to grab it, the kick strength can be stronger or weaker, etc. Therefore, in both scenarios, even if the first case that the robots retrieve is the same, the next retrieved case may not be the same because it will depend on the outcome of the actions performed during the execution of the first case (the final positions of the robots and the ball).

After studying the results obtained from the experiments, we can classify the trials in different groups based on the sequence of retrieved cases during the performance. For space reasons we only describe the second scenario which is

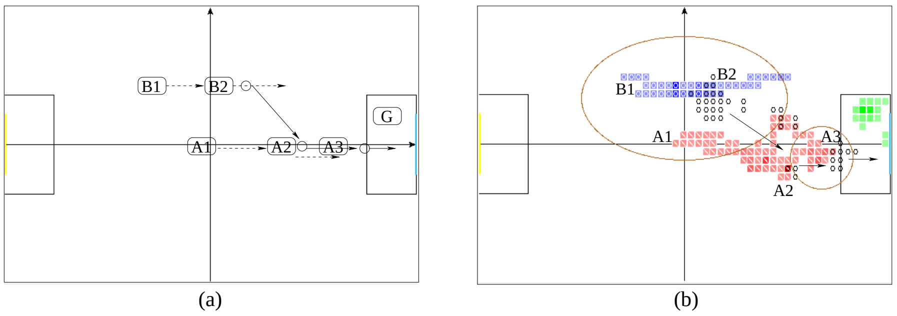

Fig. 8. Scenario 2 using the case-based approach: (a) "multiple left middle" case followed by "single goalie diagonal" case. (b) robots' real paths performed during the execution of both cases.

more interesting since an opponent is included. However, the ideas discussed are based on both sets of experiments.

Figure 6b shows the execution scheme for the second scenario: robot B passes the ball to robot A, who then kicks it near the penalty area ("multiple left middle" case). The interesting situation occurs next, when the ball is near the penalty area but there is an opponent between the player and the goal. Hence, the retrieved cases now must include an opponent in their descriptions. We classified the 15 trials in two groups based on the retrieved cases for this second part of the execution:

Opponent: the goalie is either in between the ball and the goal (Figure 6b) or diagonally located with respect to the ball, i.e. not obstructing the trajectory of the ball to the goal (Figure 8). In the first case, the attacking robot must grab it under its chin, move sideways to the right with the ball to avoid the goalie and kick forward. In the second case, since the goalie is diagonally located, there is no need to avoid it, and the robot can directly kick the ball.

No opponent: the task is achieved with a single case (Figure 9a and 9b)), or the opponent was not considered during the next retrieval because the opponent is not within the radius of the ball (not a valid opponent as mentioned in Section 3). Therefore, a case with no opponents is retrieved (Figure 9c and 9d).

In both scenarios the first retrieved cases are always the same since the initial positions are fixed. From that point on, depending on the events occurred during the execution, the next case may vary. In any case, the robots always made a good decision and performed the task successfully in a cooperative way.

After discussing the qualitative results of the experimentation, we now show the most significant data obtained from the experiments with the case-based approach. Figure 10 shows a table with the number of retrieved cases in both scenarios. As we can see, from a total of 65 cases, 57 were correctly retrieved and successfully executed. The 8 remaining where aborted during execution because the robots realized that the cases were not applicable anymore. Figure 11 depicts the regions of the field covered during the performance of both experiments. From the case base, a total of 11 different cases were retrieved in both scenarios.

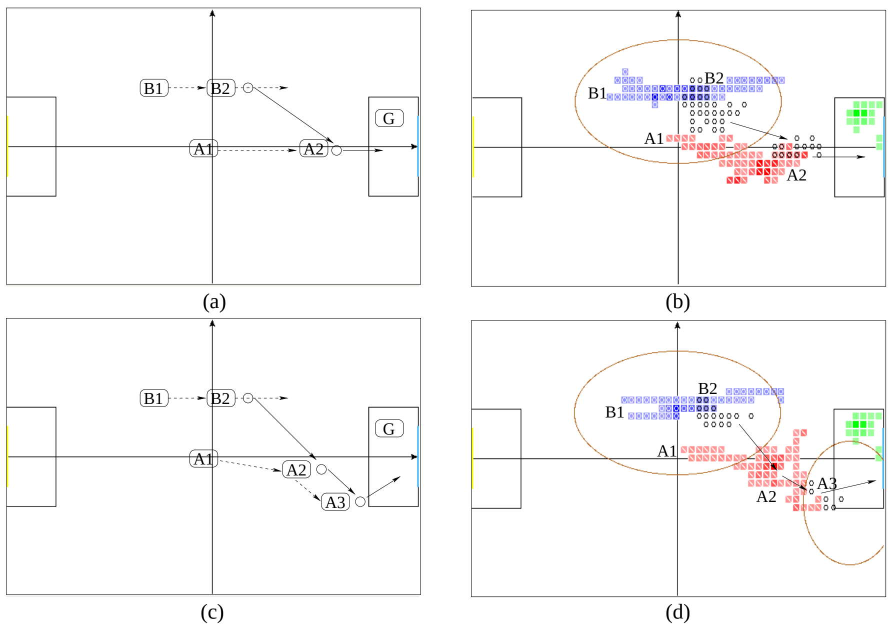

Fig. 9. Scenario 2 using the case-based approach: (a) “multiple left middle” case with no more cases; (c) “multiple left middle” case followed by “single right front” case; and (b) and (d) robots’ real paths performed during the execution of both cases.

From an observer point of view, these 8 cases were incorrectly retrieved since the states of the environment were not similar to the cases descriptions. However, from the robots’ point of view, the cases indeed matched the states of the environment at the retrieving stage, but due to localization errors, the robots’ believes were wrong. From the moment they correctly relocalized (and therefore, the position of the ball), they realized that the cases did not match the state of the environment and aborted the execution.

We also noticed that during the experiments with the behavior-based approach, the robots collided between them 8 times and 4 times the ball went out of the field. While with our approach, the robots never collided, neither kicked the ball out of the field.

Finally, we must also point out that during a game our approach results in a more controlled strategy, rather than an aggressive one where the robots are constantly trying to individually get the ball and score. This is because we include a reasoning module which takes care of higher level decisions. Although the chances of scoring increases with an aggressive strategy, it also increases the number of lost balls, which may allow the opponent team to score more goals.

7 Conclusions and Future Work

We have presented an extension of our CBR system for action selection in the robot soccer domain. Since we have increased the complexity of the system

	retrieved cases	lost cases	completed execution
Scenario 1	36	6	30
Scenario 2	29	2	27
Total	65	8	57

Fig. 10. Case results in both scenarios.

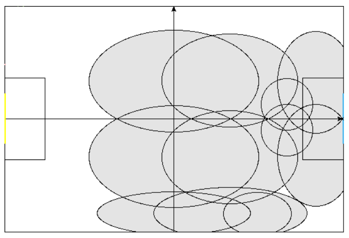

Fig. 11. Case execution coverage.

(including teammates and opponents), we have developed a new representation of the case base (an indexed list) as well as a retrieval process that prioritizes the participation of more than one robot. Hence, as shown in the evaluation, the robots behavior results in a "real" team playing performance (more cooperative) and instead of an individualistic performance, which we believe is more adequate for robot soccer. To this end, we also presented a mechanism to select the robot in charge of the retrieval process and the coordination of the team.

As future work we propose to have the different robots retrieving cases, and therefore, the need of an agreement mechanism will arise. For instance, it would be interesting to integrate a negotiation mechanism based on appropriate information (e.g. how well localized a robot is) to allow the robots to decide the most appropriate case in a better informed manner. We also plan to further extend the system by a more complete coverage of the possible situations that arise during a game after introducing more teammates and opponents.

References

1. K. Lam, B. Esfandiari, and D. Tudino. A Scene-based Imitation Framework for RoboCup Clients. In *MOO - Modeling Other Agents from Observations*, 2006.
2. A. Lattner, A. Miene, U. Visser, and O.Herzog. Sequential Pattern Mining for Situation and Behavior Prediction in Simulated Robotic Soccer. In *9th RoboCup International Symposium*, 2005.
3. C. Marling, M. Tomko, M. Gillen, D. Alexander, and D. Chelberg. Case-based reasoning for planning and world modeling in the robocup small size league. In *IJCAI Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments*, 2003.
4. M. Riedmiller, A. Merke, D. Meier, A. Hoffmann, A. Sinner, O. Thate, and R. Ehrmann. Karlsruhe brainstormers — A reinforcement learning approach to robotic soccer. *Lecture Notes in Computer Science*, 2019, 2001.
5. R. Ros and J.L. Arcos. Acquiring a Robust Case Base for the Robot Soccer Domain. In *Proc. 20th Int. Joint Conf. on Artificial Intelligence*, 2007.
6. R. Ros and M. Veloso. Executing Multi-Robot Cases through a Single Coordinator. In *Proc. of Autonomous Agents and Multiagent Systems*, 2007. To appear.
7. R. Ros, M. Veloso, R. López de Màntaras, C. Sierra, and J.L. Arcos. Retrieving and Reusing Game Plays for Robot Soccer. In *Advances in Case-Based Reasoning*, volume 4106, 2006.
8. J. Wendler and M. Lenz. CBR for Dynamic Situation Assessment in an Agent-Oriented Setting. In *Proc. AAAI-98 Workshop on CBR Integrations*, 1998.