A Case-Based Approach for Coordinated Action Selection in Robot Soccer

Raquel Ros [1], Josep Lluís Arcos, Ramon Lopez de Mantaras

IIIA - Artificial Intelligence Research Institute
CSIC - Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Spain

Manuela Veloso

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Abstract

Designing coordinated robot behaviors in uncertain, dynamic, real-time, adversarial environments, such as in robot soccer, is very challenging. In this work we present a case-based reasoning approach for cooperative action selection, which relies on the storage, retrieval, and adaptation of example cases. We focus on cases of coordinated attacking passes between robots in the presence of the defending opponent robots. We present the case representation explicitly distinguishing between controllable and uncontrollable indexing features, corresponding to the positions of the team members and opponent robots, respectively. We use the symmetric properties of the domain to automatically augment the case library. We introduce a retrieval technique that weights the similarity of a situation in terms of the continuous ball positional features, the uncontrollable features, and the cost of moving the robots from the current situation to match the case controllable features. The case adaptation includes a best match between the positions of the robots in the past case and in the new situation. The robots are assigned an adapted position to which they move to maximize the match to the retrieved case. Case retrieval and reuse are achieved within the distributed team of robots through communication and sharing of own internal states and actions. We evaluate our approach, both in simulation and with real robots, in laboratory scenarios with two attacking robots versus two defending robots as well as versus a defender and a goalie. We show that we achieve the desired coordinated passing behavior, and also outperform a reactive action selection approach.

Key words: case-based reasoning, action selection, robot soccer, cooperative task execution

1 Introduction

In order for a robot to perform an apparently simple task, such as actuating a ball towards a goal point, the robot needs multiple capabilities, including object detection, perception of the environment, building of an internal world model, making decisions when planning the task, navigation while avoiding obstacles, execution of planned actions, and recovering from failure. The complexity of each individual ability, and therefore the overall robot's behavior design, is related to the complexity of the environment where the robot carries out the task: the higher the complexity of the environment, the more challenging the robot's behavior design. Robot soccer is a particularly complex environment, particularly due to its dynamic nature resulting from the presence of multiple teammate and opponent robots.

In general in multi-robot domains, and robot soccer in particular, collaboration is desired so that the group of robots work together to achieve a common goal. It is not only important to have the agents collaborate, but also to do it in a coordinated manner so that the task can be organized to obtain effective results. A wide variety of methods has been investigated to address multi-agent coordination, including task division with role assignment (Stone and Veloso, 1999; Vail and Veloso, 2003; Gerkey and Mataric, 2004), and establishment of mutual beliefs of the intentions to act (Grosz and Kraus, 1996). Communication among agents underlies these approaches for collaboration.

In this work, we are particularly interested in the action selection and coordination for joint multi-robot tasks, motivated by a prototype environment of robot soccer. We have successfully applied Case-Based Reasoning (CBR) techniques to model the action selection of a team of robots within the robot soccer domain (Ros et al., 2006; Ros and Veloso, 2007; Ros et al., 2007). However, our previous approach did not address the dynamic intentional aspect of the environment, in particular, in robot soccer, the presence of adversaries. Many efforts aim at modeling the opponents (Wendler and Bach, 2004; Ahmadi et al., 2003; Steffens, 2004; Huang et al., 2003; Miene et al., 2004; Marling et al., 2003), in particular when the perception is centralized (Riley and Veloso, 2002). Instead, we address here a robot soccer framework in which the robots are fully distributed, without global perception nor global control, and can communicate.

Email addresses: `ros@iiia.csic.es` (Raquel Ros), `arcos@iiia.csic.es` (Josep Lluís Arcos), `mantaras@iiia.csic.es` (Ramon Lopez de Mantaras), `veloso@cs.cmu.edu` (Manuela Veloso).

[1] Raquel Ros holds a scholarship from the Generalitat de Catalunya Government. This work has been partially funded by the Spanish Ministry of Education and Science project MID-CBR (TIN2006-15140-C03-01) and by the Generalitat de Catalunya under the grant 2005-SGR-00093.

We follow a CBR approach where cases are recorded and model the state of the world at a given time and prescribe a successful action (Kolodner, 1993; Lopez de Mantaras et al., 2006; Riesbeck and Schank, 1989; Veloso, 1994). A case can be seen as a recipe that describes the state of the environment (problem description) and the actions to perform in that state (solution description). Given a new situation, the most similar past case is retrieved and its solution is reused after some adaptation process to match the current situation. We model the case solution as a set of sequences of actions, which indicate what actions each robot should perform (Veloso, 1994). Our case-based approach is novel in the sense that our cases represent a multi-robot situation where the robots are distributed in terms of perception, reasoning, and action, and can communicate. Our case-based retrieval and reuse phases are therefore based on messages exchanged among the robots about their internal states, in terms of beliefs and intentions.

Our case representation ensures that the solution description in the cases indicates the actions the robots should perform; that the retrieval process allocates robots to actions; and finally, with the coordination mechanism, that the robots share their individual intentions to act. Our approach allows for the representation of challenging rich multi-robot actions, such as passes in our robot soccer domain, which require well synchronized positioning and actions.

Previously, our retrieval process consisted of obtaining the most similar case based on two measures: similarity between the problem to solve and a given case, and the cost of adapting the problem to that case. Opponents were static, and therefore, simply modeled as obstacles the robots had to avoid (Ros et al., 2007). We extend the retrieval algorithm to include an applicability measure that determines if the reuse of a given case is feasible or not, as a function of the opponent and despite its static similarity degree. Furthermore we view the position of the opponents and the ball in a case description as uncontrollable features, while the positions of the teammate robots are viewed as controllable features, in the sense that robots in the current situation can move to match the positions in the past cases.

In terms of the reuse step, before the robots start executing the assigned actions according to the case, they pre-position themselves in the initial positions of the retrieved case. Instead of following an independent positioning strategy (Ros et al., 2007), we propose an alternative positioning strategy that reduces the time spent on satisfying the initial conditions. We show empirical results that confirm the effectiveness of our collaborative approach compared to an individualistic approach. The scenarios include two moving opponents and two attacking robots.

The article is organized as follows. Section 2 presents our particular robot soccer domain and describes the case representation. Section 3 introduces the

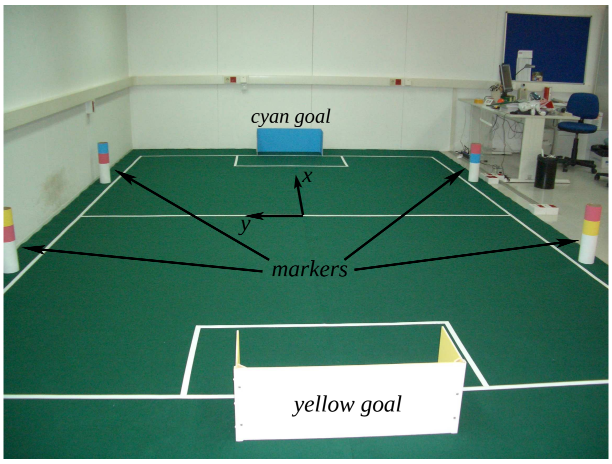


Fig. 1. Snapshot of the Four-Legged League field (image extracted from the IIIA lab).

case retrieval in terms of matching the situation. Section 4 focuses on the multi-robot architecture for case retrieval and case reuse in our distributed multi-robot system. Section 5 presents the empirical evaluation. We review related work within the robot soccer domain in Section 6. Finally, Section 7 draws the conclusions and discusses future work.

2 Case-based Representation of Robot Soccer Play

Our work is prototyped within the RoboCup robot soccer research initiative RoboCup (2008), which aims at creating a team of soccer robots capable of beating the human worldcup soccer champions by 2050 (Kitano et al., 1997). RoboCup includes several leagues to incrementally address the multiple challenges towards achieving its ambitious goal. The leagues vary depending on whether the robots are real or simulated and whether they have a centralized or distributed perception and control.

Our work focuses on the RoboCup Four-Legged Soccer League, where teams consist of four Sony AIBO robots which are fully autonomous with on-board perception (vision), reasoning, communication and action. The robots operate in a game with no external control either by humans or by computers. A playing field, as shown in Figure 1 for RoboCup 2007, consists of a green carpet with white lines, two colored goals (cyan and yellow), and four colored markers for robot localization. The field dimensions and positioning of the markers are predefined.

The teams are composed necessarily of a goalie and three other robots, which can be assigned defending and attacking roles. The rules of the game, including the detection of fouls and goals, are enforced by human referees who actuate an external computer game controller, which sends messages to the robots about the status of the game. Our work is situated within this league, as it captures the research challenges we address of fully distributed communicating and acting teams of robots in adversarial environments.

Cases relate game situations with possible gameplays. As a prescription of a successful gameplay, a case stores the context (a partial snapshot of the environment) that makes appropriate the gameplay. Thus, only the context relevant to the gameplay is stored in the case (i.e. only those robots involved are represented). The case definition is composed of three parts: the problem description, which corresponds to the state of the world; the solution description, which indicates the sequence of actions the robots should perform to solve the problem; and finally, the case scope representation, used in the retrieval step, which defines the applicability boundaries of cases. We formally define a case as a 3-tuple:

$$case = (P, A, K)$$

where P is the problem description, A, the solution description, and K, the case scope representation. We next describe each case component applied to the robot soccer domain.

2.1 Problem Description

The problem description corresponds to a set of features that describe the current world state. In the robot soccer domain we consider the following features as the most relevant for describing the state of the game:

$$P = (B, G, Tm, Opp)$$

where:

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

$$x_B \in [-2700, 2700]\text{mm},\ x_B \in \mathbb{Z} \quad y_B \in [-1800, 1800]\text{mm},\ y_B \in \mathbb{Z}$$

(2) G: defending goal

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

(3) Tm: teammates' global positions

$$Tm = \{tm_1 : (x_1, y_1), \dots, tm_n : (x_n, y_n)\}$$

$$x_i \in [-2700, 2700]\text{mm},\ x_i \in \mathbb{Z} \qquad y_i \in [-1800, 1800]\text{mm},\ y_i \in \mathbb{Z}$$

where tm_i is the robot identifier and $n \in [1, 4]$ for teams of 4 robots. We only include in the problem description those robots that are relevant for the case, since not all four robots are always involved.

(4) *Opp*: opponents' global positions.

$$Opp = \{opp_1 : (x_1, y_1), \ldots, opp_m : (x_m, y_m)\}$$

where opp_i is the opponent identifier and $m \in [1, 4]$ for teams of 4 robots. This set could be empty for cases where no opponents are included. Similarly to the teammates feature, we only consider the relevant opponent robots, and not all those that are currently on the field.

Henceforth, we will refer to a teammate robot either as "teammate" or "robot" and to an opponent robot, as "opponent".

2.2 Solution Description

The solution of a case corresponds to the sequences of actions each teammate performs. We call them *gameplays*. In this work, a gameplay must also satisfy two conditions: (*i*) at least one robot has as its first action to get the ball; and (*ii*) only one robot can control the ball at a time. Formally, we define a gameplay as:

$$A = \begin{pmatrix} tm_1 : [a_{11}, a_{12}, \ldots, a_{1p_1}], \\ \ldots \\ tm_n : [a_{n1}, a_{n2}, \ldots, a_{np_n}] \end{pmatrix}$$

where $n \in [1, 4]$ is the robot identifier, and p_i is the number of actions teammate tm_i performs (we only describe the actions for the teammates included in the problem description). Actions are either individual actions, such as "get the ball" or "kick", or joint actions, such as "pass ball to robot tm_i". Actions may have parameters that indicate additional information to execute them. For instance, in the turn action we can either specify the heading the robot should have or a point to face; in the kick action we indicate which type of kick to perform (forward, left,...), etc.

During the execution of the solution, all robots on the team start performing their sequences of actions at the same time. The duration of each action is implicitly given by the action type and its initiation depends on the action preconditions. Consider the following situation: robot tm_2 must pass the ball to robot tm_1 who then kicks it forward, and robot tm_3 has to move to a point p. Without explicitly indicating the timestep of each action, the timing of the overall performance will be: robot tm_2 starts moving towards the ball to get it, while robot tm_1 waits for robot tm_2 to executes the pass. Once robot tm_2 has done the pass, tm_1 receives the ball and kicks it forwards. In the meantime, since robot tm_3 has no preconditions, he starts moving to point p independently from the state in which the other robots are. In this example, the solution is be:

$$A = \begin{pmatrix} tm_1 : [\ wait,\ receive\_ball(tm_2),\ kick(\text{forward})], \\ tm_2 : [\ get\_ball,\ pass\_ball(tm_1)], \\ tm_3 : [\ go\_to\_point(p)] \end{pmatrix}$$

2.3 Case Scope Representation

Because of the high degree of uncertainty in the incoming information about the state of the world, the reasoning engine cannot rely on precise values of the positions of the opponents and the ball on the field to make decisions. Therefore, we model these positions as regions of the field called *scopes*. The scopes are elliptic[2] regions centered in the object's position with radius τ^x and τ^y. The case scope is defined as:

$$K = (ball : (\tau_B^x, \tau_B^y), opp_1 : (\tau_1^x, \tau_1^y), \ldots, opp_m : (\tau_m^x, \tau_m^y))$$

where τ_B^x and τ_B^y correspond to the x and y radius of the ball's scope, opp_i is the opponent identifier, and τ_i^x and τ_i^y, correspond to the radius of the scope of opponent opp_i ($i \in [1, m]$). If there are no opponents in the case, then we do not include any opponent pair, $opp : (\tau^x, \tau^y)$. Notice that we only consider opponents, and not teammates. As we will explain during the retrieval process, we define two different measures for each type of players. While one requires the use of scopes, the other does not.

We must also anticipate that the ball's scope is fundamental for the retrieval process as we will explain in Section 3. A case might be considered a potential solution only if the position of the ball described in the problem to solve is within the ball's scope of the case. Otherwise, the case is dissmissed. In (Ros and Arcos, 2007) we have presented an approach for automatically acquiring the case scope based on the robot's perception. In the present work we have manually extended the learned case base to complete it with more complex cases, i.e. including opponents.

The advantage of representing the opponents combining their positions ($opp_i : (x_i, y_i)$) and their scopes (τ_i^x, τ_i^y) is that we can easily define qualitative locations of the opponents on the field with respect to the ball. Reasoning with qualitative information is advantageous in this kind of domains, especially, as we have said, because 1) there is high uncertainty in the incoming information, and 2) qualitative information facilitates the generalization of similar

[2] We are assuming that error in perception follows a normal distribution. Thus, the projection of a 2D Gaussian distribution on the XY plane corresponds to an ellipse.

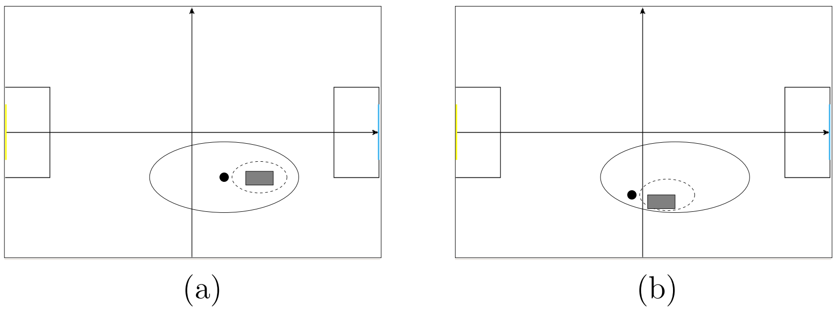


Fig. 2. (a) Example of the scope of a case. The black circle represents the ball and the gray rectangle represents the defender. The ellipses correspond to the ball's scope (solid ellipse) and the defender's scope (dashed ellipse). (b) Example of a simplified problem description. We translate the defender's scope based on the current ball's position.

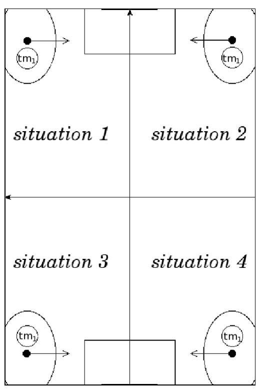
situations. For instance, it is more general to reason about a defender being in a region in front of the ball, rather than the defender being in position (x, y).

Figure 2a shows a simple example of this situation. The interpretation of this case is that if we want to consider it as a potential solution for a given problem, then the ball should be located within the ball's scope and an opponent should be positioned in front of it. Figure 2b depicts a problem example where the defender is considered to be in front of the ball because it is located within the defender's scope. Note that the defender's scope has been translated with respect to the reference point, i.e. the current position of the ball in the problem. Since the ball is also situated within the ball's scope of the case, we can state that the case in Figure 2a matches the problem in Figure 2b and the solution of the case might be useful to solve the problem.

2.4 Domain Properties

We can observe two symmetric properties of the ball's, teammates' and opponents' positions and the defending goal: one with respect to the x axis, and the other one, with respect to the y axis and the defending goal. That is, as shown in Figure 3a, a robot at point (x, y) and defending the yellow goal describes *situation 1*, which is symmetric to *situation 2* ($(x, -y)$, defending the yellow goal), *situation 3* ($(-x, y)$, defending the cyan goal) and *situation 4* ($(-x, -y)$, defending the cyan goal). Similarly, the solution of a problem has the same symmetric properties. For instance, in a situation where the solution is *kick to the right*, its symmetric solution with respect to the x axis would be *kick to the left*. Thus, for any case in the case base, we can compute its symmetric descriptions, obtaining three more cases. Figure 3b describes the case in *situation 1*.

Due to these symmetric properties, the case base could either be composed of



(a)

$$case = \left(\begin{array}{c} P = \left(\begin{array}{l} B = (2240, 1470), \\ G = \text{yellow}, \\ Tm = \{tm_1 : (1990, 1470)\}, \\ Opp = \oslash \end{array} \right), \\ S = \left(tm_1 : \begin{bmatrix} get\_ball, \\ kick(\text{right}) \end{bmatrix} \right), \\ K = \left(ball : (609, 406) \right) \end{array} \right)$$

(b)

Fig. 3. (a) Situation 1 corresponds to the original description of the case. While situations 2, 3 and 4 correspond to its symmetric descriptions. The ellipse corresponds to the ball's scope and the arrow, to the action (right or left kick). (b) Example of the case described in situation 1.

a single representation of four cases, i.e. one case would represent four cases (one per quadrant), or having all four cases explicitly included in the case base. The former option implies having a smaller case base, which is an advantage during the retrieval step since less cases have to be evaluated. However, the computational cost would significantly increase, since before trying to solve any new problem, we would have to map it to the evaluated case quadrant (or vice versa), and repeat this transformation for any case in the case base. On the contrary, explicitly including all four representations eliminates the overhead computation, reducing the computational cost (which in this domain is essential). Case indexing techniques can be then used to deal with the search space in large case bases. In this work, we use simple techniques to this end, as we will explain in next section.

3 Case Retrieval

Case retrieval is in general driven by a similarity measure between the new problem and the solved problems in the case base. We introduce a novel case retrieval method where we evaluate similarity along three important aspects: the similarity between the problem and the case, the cost of adapting the problem to the case, and the applicability of the solution of the case. Before explaining in detail these measures we need first to define two types of features describing the problem:

- *controllable* features, i.e. teammates' positions (the robots can move to more appropriate locations if needed).
- *non-controllable* features, i.e. the ball's and opponents' positions and the

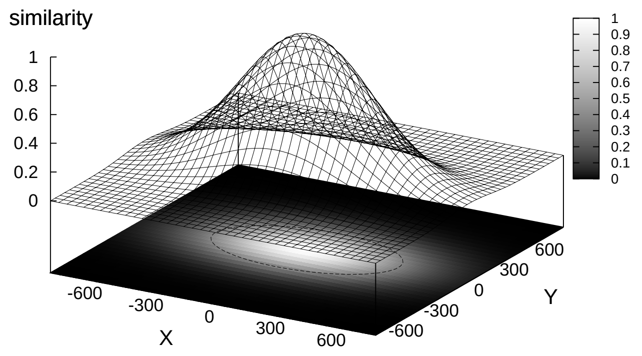


Fig. 4. 2D Gaussian centered in the origin with $\tau^x = 450$ and $\tau^y = 250$. The ellipse on the plane XY corresponds to $G(x, y) = 0.367$.

defending goal (which we cannot directly modify).

The idea of separating the features into controllable and non-controllable ones is that a case can be retrieved if we can modify part of the current problem description in order to adapt it to the description of that case. Hence, we can easily cover a larger set of problems using fewer cases. Given the domain we are dealing with, the modification of the controllable features leads to a planning process where the system has to define how to reach the positions of the robots indicated in the retrieved case in order to reuse its solution. On the contrary, non-controllable features can only be evaluated through similarity since we cannot directly alter their values.

Next we first define the three measures proposed in this work, and then we explain how to combine them to retrieve a case.

3.1 Similarity Measure

The similarity measure is based on the ball's position. We are interested in defining a continuous function that given two points in a Cartesian Plane indicates the degree of similarity based on the distance between the points. The larger the distance between two points, the lower the similarity degree between them. We propose to use a Gaussian function, which besides fulfilling these properties, it is parameterized by its variance. We can use this parameter to model the maximum distance allowed for two points to have some degree of similarity. Since we are working in a two-dimensional space, we use a 2D Gaussian function, $G(x, y)$, to compute the degree of similarity between two points.

Hence, we define the similarity function for the ball feature as:

$$\begin{aligned} sim_B(x_B^p, y_B^p, x_B^c, y_B^c) &= G(x_B^p - x_B^c, y_B^p - y_B^c) \\ &= exp\left(-\left[\left(\frac{x_B^p - x_B^c}{\tau_B^x}\right)^2 + \left(\frac{y_B^p - y_B^c}{\tau_B^y}\right)^2\right]\right) \end{aligned}$$

where (x_B^p, y_B^p) corresponds to the ball's position in problem p; (x_B^c, y_B^c), to the ball's position in case c; and τ_B^x and τ_B^y, to the ball's scope indicated in the case as defined in Section 2.3. Figure 4 draws a 2D Gaussian function and its projection on the XY plane (sequence of ellipses with increasing radius as the similarity decreases). The ellipse in the figure corresponds to the Gaussian's projection with radius τ_B^x and τ_B^y representing the scope of the ball, i.e. the region within which we consider two points to be similar enough. The value of the Gaussian function at any point of this ellipse is $G(x, y) = 0.367$. Thus, we use this value as the threshold for considering similarity between two points.

3.2 Cost Measure

This measure computes the cost of modifying the values of the controllable features (teammates' positions). Similar ideas were presented by Smyth and Keane (1998) where they compute the cost of adapting the case to the problem to solve, i.e. the cost of modifying the solution of the case. In contrast, in our work, we propose to adapt the problem to solve to the case, i.e. we modify the problem description, so the solution of the case can be directly reused. We compute the adaption cost as a function of the distances between the positions of the team robots in the problem and the adapted positions specified in the case.

We refer to the adapted positions as those locations where the robots should position themselves in order to execute the solution of the case. These locations are computed having the position of the ball in the problem as the reference point. Figure 5 illustrates a simple adaptation example with two robots. Robot tm_2 is the one that controls the ball first, while tm_1 waits to receive the pass.

In order to compute the adaptation cost we must first determine the correspondence between the robots described in the case and the ones described in the problem, i.e. which robot tm_i^c from the case description corresponds to which robot tm_j^p in the problem description. The case description may contain fewer robots than the problem description, but not more (eg. the problem may include all four robots, while the case may describe a gameplay with two players). Thus, the matched robots in the problem description are the relevant robots that will take part of the case reuse. As we explain in Section 4.2, the robots not involved in the case reuse may perform a default behavior in the meantime.

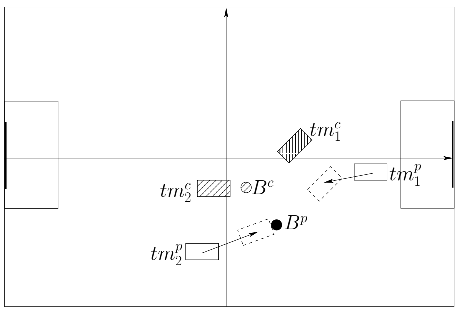
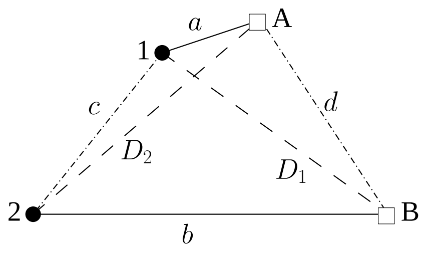


Fig. 5. Case description (B^c, tm_1^c, tm_2^c), and current problem description (B^p, tm_1^p, tm_2^p). The dashed rectangles represent the adapted positions of the robots with respect to the ball's position described in the problem to solve.

To establish the correspondence between robots we must find the best match, i.e. the one that minimizes the cost. Moreover, the cost function includes one restriction: the distances the robots have to travel must be limited by a given threshold, thr_c, because, due to the domain's dynamism, we cannot allow the robots to move from one point to another for long periods of time. Otherwise, in the meantime, the state of the world may have significantly changed and thus, the case may not be useful anymore.

In this work, since the maximum number of robots is small we can easily compute all possible matches. However, as the number of robots becomes larger, the number of combinations increases factorially. Thus, an efficient search algorithm is required, as the one we proposed in (Ros et al., 2006).

We are interested in using a cost function that not only considers the distance the robots have to travel to their adapted positions, but also verifies that the resulting paths are easily accessible for each robot (not disturbing other robots). Thus, we have studied two alternative functions to compute the cost: the sum of distances the robots have to travel and the maximum distance. The sum of distances aggregates all available distances in order to compute the outcome, while the max function is based only on one distance (the maximum), without considering the remaining ones. Therefore, we could characterize the sum as a more informed measure, where all values affect the outcome. Moreover, interestingly, the maximum distance function has a drawback when considering trapezoidal (not necessarily having two parallel sides) configurations. Consider the layout depicted in Figure 6, where we have to find the optimal match between points $\{1, 2\}$ and $\{A, B\}$. We have depicted in solid lines the distances the robots would have to travel using the sum function, and in dashed lines, the distances using the max function. As we can observe, using the latter function the robots' paths intersect. This situation will happen whenever both trapezoid diagonals, D_1 and D_2, are shorter than the trapezoid larger side, b, and the matching points correspond to the end points of the middle sides, c and d.



$$max(D_1, D_2) < max(a, b)$$
$$a + b < D_1 + D_2$$

Fig. 6. Trapezoidal layout of the matching between pairs $\{1, 2\}$ and $\{A, B\}$. The correspondence based on the sum function is represented by solid lines, while the max function is represented by the dashed ones.

Hence, in this domain we define the adaptation cost as the sum of distances the robots have to travel from their current locations to their adapted positions:

$$cost(p, c) = \sum_{i=1}^{n} dist((x_i^p, y_i^p), adaptPos_i)$$

where n is the number of robots that take part of the case solution, $dist$ is the Euclidian distance, (x_i^p, y_i^p) is the current position of robot tm_i^p and $adaptPos_i$, its adapted position.

3.3 Case Applicability Measure

This last measure considers the opponents' positions. As mentioned in Section 1, we use it to take into account the adversarial component of the domain. Defining all possible configurations of opponents during a game, i.e. opponents' positions on the field, is impossible. Hence, achieving a complete case base composed of all possible situations would not be feasible. Moreover, as previously mentioned, uncertainty about the opponents' positions is always present. For these reasons we believe that a certain degree of generalization must be included in the reasoning engine when dealing with this feature. Thus, we propose to combine the following two functions (described in detail in next subsections):

- *free path function*: the trajectory of the ball indicated in the case must be free of opponents to consider the evaluated case to be applicable.
- *opponent similarity*: a case includes information about opponents when these are relevant for the described situation, i.e. they represent a significant threat for the robots to fulfill the task, such as an opponent blocking the ball or an opponent located near enough to get the ball first. We are interested in increasing the relevance of these cases upon others when the current problem includes significant opponents that may lead the attacking team to fail. Thus, the more opponents locations described in the problem match with the opponents locations described in the case, the higher the

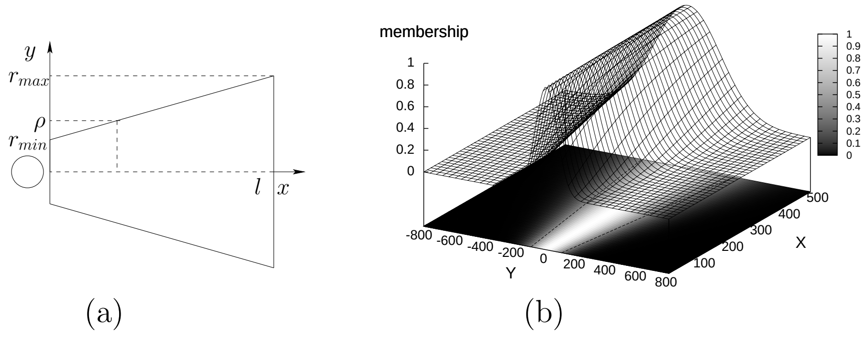


Fig. 7. (a) Ball's trajectory represented by an isosceles trapezoid defined by the minimum and maximum radius, and the trajectory length. (b) Membership function μ corresponding to the fuzzy trajectory with $r_{min} = 100$, $r_{max} = 300$ and $l = 500$. The lines on the plane XY correspond to $\mu(x, y) = 0.367$

similarity between the problem and the case.

3.3.1 Free Path Function

Given a case, the *free_path* function indicates whether the trajectories the ball follows during the execution of the case solution are free of opponents or not.

Because of the ball's movement imprecision after a kick (either due to the robot's motion or the field's unevenness), the ball could end in different locations. Hence, we represent a trajectory by means of a fuzzy set whose membership function μ indicates the degree of membership of a point in the trajectory such that the closer the point to the center of the trajectory, the higher the membership value. More precisely, this function is defined as a sequence of Gaussians, where the width of each Gaussian increases from a minimum radius (for $x = 0$) to a maximum one (for $x = l$) along the trajectory. We formally define the membership function for a trajectory t_j depicted in Figure7b as:

$$\mu_{t_j}(x, y) = exp\left(- \left[\frac{y}{\rho(x, r_{min}, r_{max}, l)} \right]^2 \right)$$

where r_{min}, r_{max} correspond to the minimum and maximum radius respectively, and l, corresponds to the length of trajectory t_j . Finally, ρ is a linear function that indicates the radius of the Gaussian as a function of x. The projection of the μ function on the XY plane results in a sequence of trapezoids. The trapezoid defined by r_{min}, r_{max} and l (Figure7a) corresponds to the projection of $\mu_{t_j}(x, y) = 0.367$, which covers the area of the field where the ball could most likely go through according to the experimentation we have performed, i.e. the ball's trajectory. We use this value as the threshold, thr_t, to consider whether a point belongs to the trajectory or not.

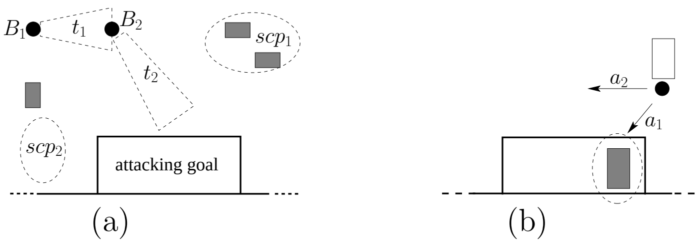


Fig. 8. (a) Example of the ball's path performed by a pass between two players (robots are not depicted for simplicity). The dashed ellipses represent the opponents scopes described in the case, and the gray rectangles, the opponents described in the problem to solve. (b) Opponent similarity as a justification of the action to perform. Action a_1 represents kicking towards the goal, while action a_2, kicking towards the robot's right side.

We call *ball path* the sequence of trajectories the ball travels through in the solution of case c. Hence, we must verify that there are no opponents in the current state of the game (problem p to solve) located within any of the trajectories of the ball path. Figure 8a depicts an example. The initial position of the ball corresponds to B_1. After the first trajectory, t_1, the ball stops at B_2 and continues the second trajectory, t_2. Each trajectory results from a robot's kick, and it is computed on-line during retrieval based on the case description (teammates' positions and their associated actions). Formally, we define the free path function as:

$$free\_path(p, c) = 1 - \max_{t_j \in T}(\phi_{t_j}(Opp))$$

where,

$$\phi_{t_j}(Opp) = \begin{cases} 1, & \exists opp_i \in Opp \ (\mu_{t_j}(x_i, y_i) > thr_t) \\ 0, & \text{otherwise} \end{cases}$$

and T is the sequence of fuzzy trajectories (t_1 and t_2 in Figure 8a) described in case c, Opp is the set of opponents in problem p, (x_i, y_i) corresponds to the position of opponent opp_i, and $\mu_{t_j} \in [0, 1]$ is the membership function. From the above expressions, we see that a point (x, y) is within a trajectory t_j if $\mu_{t_j}(x, y) > thr_t$, where $thr_t = 0.367$. The free path function could indicate the degree of path freedom using μ directly, instead of ϕ. In other words, we could define it as a fuzzy function as well.

3.3.2 Opponent's Similarity

Due to the uncertainty and imprecision properties of the domain, opponents on the field are modeled by means of scopes (elliptic regions defined in Section 2.3), instead of only considering (x, y) positions. The opponent's similarity measure indicates the number of scopes that are occupied by at least one opponent in the problem to solve. We call them *conditions*. The more conditions are satisfied, the more similar will be the current state of the game and the case description. Figure 8a shows an example where only one condition is satisfied, since only one scope (scp_1) is occupied by at least one opponent. We define the opponent similarity function between a problem p and a case c as:

$$sim_{opp}(p,c) = |\{scp_j \mid scp_j \in Scp,\ \exists opp_i \in Opp\ (\Omega_{scp_j}(x_i^p, y_i^p) > thr_{opp})\}|$$

where,

$$\Omega_{scp_j}(x_i^p, y_i^p) = G(x_i^p - x_j^c, y_i^p - y_j^c) = exp\left(-\left[\left(\frac{x_i^p - x_j^c}{\tau_j^x}\right)^2 + \left(\frac{y_i^p - y_j^c}{\tau_j^y}\right)^2\right]\right)$$

and Scp is the set of scopes in case c (scp_1 and scp_2 in Figure 8a), Opp is the set of opponents described in problem p, and (x_i^p, y_i^p) corresponds to the position of opponent opp_i. Each scope scp_j is defined by an ellipse with radius τ_j^x and τ_j^y centered in (x_j^c, y_j^c) (the opponent's position described in case c). We define Ω as a Gaussian function, where the projection on the XY plane for $\Omega(x,y) = 0.367$ corresponds to an elliptical region on the field with radius τ_j^x and τ_j^y (analogously to the ball similarity measure). Thus, to consider that an opponent is within a given scope we set the threshold thr_{opp} to 0.367. Once again, we could use the degree of occupation of a given scope instead of considering a boolean function.

We must notice that this measure is not crucial for the selection of a case as a candidate (as we describe in the next section). Its importance is that it allows to rank cases in order to select the best one to retrieve. While the free path function is fundamental when deciding whether a solution can be applicable or not, the opponent similarity measure can be seen as a justification of the actions defined in the case solution. Consider the example shown in Figure 8b. The robot in front of the ball can either kick towards the goal (action a_1), or kick towards its right (action a_2). The selection of one action or the other is decided by checking the existence of an opponent in between the ball and the goal. Hence, if there is no opponent, it is easy to see that the most appropriate action to achieve the robot's objective is to kick towards the goal. But if an opponent (a goalie) is right in front, it makes more sense to try to move to a better position where the robot can then try some other action. Therefore, we can view the existence of an opponent as a justification for the selected action, in this example, kick towards the right.

3.4 Case Selection

After describing the different measures, we now have to combine them to retrieve a case to solve the current state of the game (the new problem p). Because of the real time response requirements and the limited computational resources of the robots, we need to reduce as much as possible the search space. Therefore, we use a filtering mechanism for the retrieval process. Each case c is evaluated using the measures explained in the previous sections. A case is rejected as soon as one of the conditions is not fulfilled. If a case fulfills all the conditions, then it becomes a candidate case. Besides, cases are stored in a list indexed on the defending goal feature. Thus, we only evaluate those cases that have a defending goal equal to the one in the problem to solve.

The filtering mechanism is shown in Algorithm 1. We first verify the ball similarity between the problem and the evaluated case (line 1), i.e. whether the current ball position is within the ball's scope indicated in the case ($thr_b = 0.367$ as explained in section 3.1). Next, in lines 2 to 6 we check that every distance between the current robots' positions and their adapted positions (obtained after the correspondence computation as explained in Section 3.2) is below the cost threshold ($thr_c = 1500$mm, introduced in Section 3.2 as well, and obtained through empirical experimentation). Finally, if the ball's path is free of opponents (line 7) then we consider the evaluated case as a valid candidate (line 8).

Algorithm 1 IsCandidate(p, c)

```
 1: if $sim_B(B^p, B^c) > thr_b$ then
 2:    for all $(tm^p, tm^c) \in correspondence(p, c)$ do
 3:       if $dist(tm^p, tm^c) > thr_c$ then
 4:          return False
 5:       end if
 6:    end for
 7:    if $free\_path(p, c)$ then
 8:       return True
 9:    else
10:       return False
11:    end if
12: else
13:    return False
14: end if
```

From the set of candidate cases that passed the filtering process, we select a single case using a ranking mechanism based on the following three criteria:

- *number of teammates that take part in the solution of the case*: in this work we are interested in using cases involving more than one robot in the solution to favor a cooperative behavior instead of an individualistic one. Therefore, the more teammates implied in the gameplay the better.
- *number of fulfilled conditions in opponent's similarity*: as explained in Section 3.3.2, the more conditions are satisfied, the more similar will be the current state of the game and the case representation.
- *trade-off between adaptation cost (as explained in Section 3.2) and similarity*

(as explained in Section 3.1): among those cases whose similarities to the problem to solve are within a given rank, the one with lower cost is preferred. Having a case with high similarity is as important as having cases with low cost. Therefore, when the similarity of a case is very high, but its cost is also high, it is better to select a somewhat less similar case but with lower cost. Thus, we classify the candidate cases into four lists: int_H, int_h, int_l and int_L, where H, h, l and L correspond to the following similarity intervals:

$$H = [0.8, 1.00] \quad h = [0.6, 0.8) \quad l = [0.4, 0.6) \quad L = (0.0, 0.4)$$

Each one of the four lists contains the candidate cases belonging to the same similarity interval ordered by their cost. For example, if cases c_3, c_7 and c_{10} have a similarity to the problem to solve within 0.8 and 1.00, and $cost(c_{10}) \leq cost(c_3) \leq cost(c_7)$, then $int_H = [c_{10}, c_3, c_7]$.

The last open issue is to decide in which order to apply these three criteria to sort the cases. To this end, we have performed several experiments in simulation evaluating different ordering functions based on two quantitative measures (time invested to achieve the goal and number of cases used to this end) and a qualitative measure (how rational was the robots performance from an external point of view). We concluded that the most suitable ordering corresponds to: 1) number of fulfilled conditions, 2) number of teammates and 3) trade-off between cost and similarity. This way we ensure that: 1) the case is as similar as possible with respect to the opponents positions; 2) the solution involves as much teammates as possible to enhance collaboration among robots; and 3) the trade-off between cost and similarity is taken into account.

Finally, the retrieved case will be the first one of the list sorted according to the above three criteria. The overall retrieval process is presented in Algorithm 2.

Algorithm 2 Retrieve(p, CB)

```
1: for $c$ in $CB$ do
2:     if $IsCandidate(p, c)$ then
3:         $candidates \leftarrow$ append($c, candidates$)
4:     end if
5: end for
6: $ordered\_list \leftarrow$ sort($candidates$)
7: $ret\_case \leftarrow$ first($ordered\_list$)
8: return $ret\_case$
```

Note that the particular ordering presented above has been chosen for the experiments we present in this work. However, the behavior of the system can be easily modified by adding new criteria, removing the existing ones and/or changing the order in which they are applied. Thus, the retrieved case could be different and therefore, the behavior of the robots could vary as well.

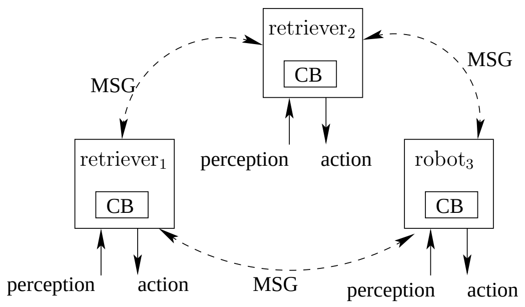


Fig. 9. Multi-robot architecture for $n = 3$ robots and $k = 2$ *retrievers*. Each robot has a library of cases (CB).

4 Retrieval and Reuse in a Multi-Robot Architecture

In the previous section we have described how to retrieve a case given a new problem to solve. Thus, the following step consists in reusing the retrieved case. However, unlike conventional CBR systems, it is not a single agent who queries the system but a team of robots. In this section, we introduce the multi-robot architecture, next, we describe how the team decides which robot retrieves a case, and finally, how they reuse the retrieved case.

4.1 Multi-Robot System

The multi-robot system is composed of n robots. All robots interact with the environment and with each other, i.e. they perceive the world, they perform actions and they send messages (MSG) to each other to coordinate (eg. retrieved case, match, abort execution,...), to exchange information about their internal state (eg. retrieving, adapting, executing,...) and their internal beliefs about the world state (own position, ball position and distance to it, etc).

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 9 shows the described architecture. This architecture is useful in heterogeneous teams where robots may have different computational power and capabilities. Thus, while some robots may reason (retrievers), the others would only execute the actions.

The first step of each cycle is to decide which of the k *retriever* robots is going to actually retrieve a case to solve the new problem (since only one case can be executed at a time). 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 low uncertainty as possible. The remaining features are either common to all the robots, or given by an external system. Therefore, we propose that the robot retrieving the case should be the closest 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 beliefs about the state of the world are constantly updated and sent to the other robots. As a consequence, we cannot ensure that all robots agree who is the one closest 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 each robot has a unique fixed Id). Once this latter robot selects the coordinator, it sends a message to the rest indicating the Id of the coordinator, so all robots (including the coordinator) know which robot is in charge of retrieving the next case to reuse.

After the coordinator is selected, he retrieves a case according to the process described in Section 3 and informs the rest of the team which case to reuse. He also informs about the correspondences between the robots in the current problem and the robots in the retrieved case (so they know what actions to execute accessing their case bases). In Section 3.2 we described how these correspondences are determined. At this point it is worth noticing that even if the current problem situation involves several robots, the retrieved case may only include a single robot (i.e. a individualistic gameplay) if the overall similarity (which includes the adaptation cost) selects an individualistic case as the most adequate for the current problem. If no case is retrieved, the robots perform a default behavior (the designer should decide the most appropriate one based on the domain requirements).

4.2 Case Reuse

The case reuse begins when the selected coordinator informs the team about the retrieved case. Figure 10 describes the finite state machine (FSM) for the case reuse process. First, all robots involved in the solution of the case start moving to their adapted positions, ADAPT state, computed as shown in Section 3.2. The robots that are not involved in the solution remain in the WAIT END state (either waiting at their positions or performing an alternative de-

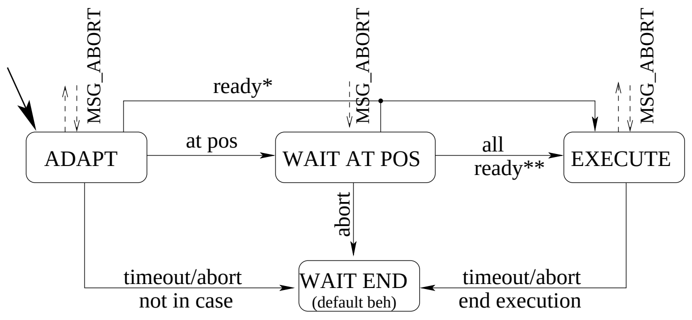


Fig. 10. Finite state machine for the case execution (*independent positioning strategy, **dependent positioning strategy).

fault behavior) until the execution ends.

Due to the dynamic and adversarial properties of the domain, we are interested in preventing the robots from waiting for long periods of time, and instead, having the robots executing their actions as soon as possible. Otherwise, while they remain inactive the state of the world may significantly change, or even worse, the opponents may take the ball. As we described in Section 2.2, there is always a robot that goes first to get the ball. All robots know who this robot is and also know in which state all robots are (adapting, reusing, waiting, etc.). When this robot arrives to its adapted position, i.e. next to the ball, all robots start executing their sequences of actions immediately (state EXECUTE), even if they have not reached their adapted positions yet. We call this strategy *dependent positioning*, since the robots' positioning depends on a single robot. An alternative strategy, *independent positioning* introduced in (Ros and Veloso, 2007), would be to have the robots waiting for all robots to reach their adapted positions and then start executing their actions. However, the risk of losing the ball with this latter approach increases, since while the robots wait for their teammates, an opponent can easily steal the ball. We believe that an independent strategy is more convenient to use in other domains where the initial positions of all robots are crucial for the successful fulfillment of the task.

The execution of the case continues until all robots finish their sequences of actions. Finally, they report to the coordinator that they finished the execution and wait for the rest of the robots to end (WAIT END state) while performing a default behavior (stop, move close to the ball, etc.). When the coordinator receives all messages, it informs the robots so they all start a new cycle, i.e. selecting a new coordinator, retrieving a case and executing its solution.

The execution of a case may be aborted at any moment if any of the robots involved in the case reuse either detects that the retrieved case is not applicable anymore or a timeout occurs. In either case, the robot sends an aborting message to the rest of the robots so they all stop executing their actions.

Then, once again they go back to the initial state in order to restart the process.

5 Experimental Evaluation

The goal of the experimentation is to empirically demonstrate that with our approach the performance of the robots results in a cooperative behavior where the team works together to achieve a common goal, a desired property in this kind of domain. The approach allows the robots to apply a more deliberative strategy, where they can reason about the state of the game in a more global way, as well as to take into account the opponents. Thus, they try to avoid the opponents by passing the ball to teammates, which should increase the possession of the ball, and therefore, the team should have more chances to reach the attacking goal.

We compare our approach with the approach presented by the Carnegie Mellon's CMDash'06 team. In their approach they have an implicit coordination mechanism to avoid having two robots "fighting" for the ball at the same time. The robot in possession of the ball notifies the rest of the team, and then the other robots move towards different directions to avoid collisions. The robots also have roles which force them to remain within certain regions of the field (for instance, defender, striker, etc.). The resulting behavior of this approach is more individualistic and reactive in the sense that the robots always try to go after the ball as fast as possible and move alone towards the attacking goal. Although they try to avoid opponents (turning before kicking, or dribbling), they do not perform explicit passes between teammates and in general they move with the ball individually. Passes only occur by chance and are not previously planned. Henceforward we will refer to this approach as the *reactive approach.*

5.1 Experimental Setup

Two types of experiments were performed: simulated experiments and real robots experiments in two vs. two scenarios. We next roughly describe the case base used during the experimentation, the behaviors of the robots, and the evaluation measures.

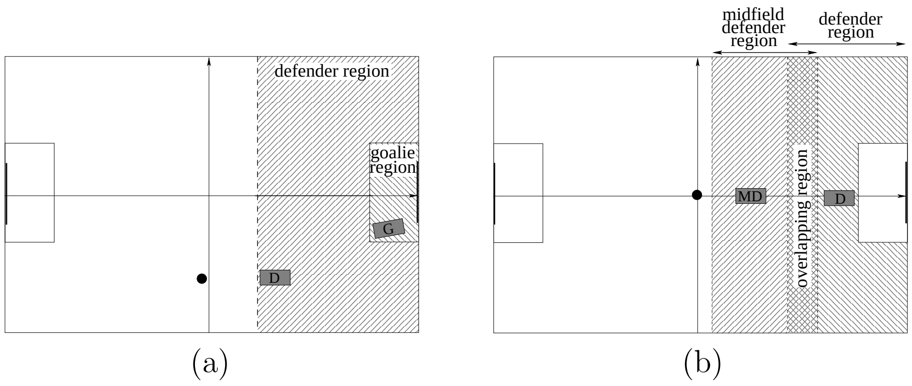


Fig. 11. Opponents' home region configuration: (a) DG configuration, i.e. defender (D) and goalie (G); and (b) 2D configuration, i.e. midfield defender (MD) and defender (D).

5.1.1 Case Base

The case base is composed of 136 cases. From this set, 34 cases are hand-coded, while the remaining ones are automatically generated using spatial transformations exploiting the symmetries of the soccer field as described in 2.4. We can classify the cases used in the evaluation along the following components: (*i*) strategic component, region of the field a case covers (in this work: back, center, front, corner, side and front-side); (*ii*) teamwork component, number of teammates included in the problem description (individualistic -only one robot, or cooperative -two robots in this work); and (*iii*) adversarial component, number of opponents described in the case (none or more -two in this work). We do not include defensive cases since the proposed scenarios are oriented towards attacking situations only.

Moreover, thanks to the flexible case representation by means of scopes, cases may cover general situations (a wide area of the field) or specific situations (particular configurations). Our case base is composed of these two types of cases guaranteeing that most of the time, at least one case will be retrieved. This wide coverage is also achieved by the adaptation step (as mentioned in Section 3), which permits having a low number of cases to cover a large number of situations.

5.1.2 Robot's Behaviors

The attackers are the players to be evaluated, i.e. they use either the CBR approach or the reactive approach. As mentioned in Section 4, the robots using the CBR approach perform a default behavior when no case is found. In these experiments, the default behavior corresponds to the reactive approach.

We have implemented a simple behavior for the opponents (defender, midfield defender and goalie). Each robot has a home region and he cannot go beyond that region. If the ball is within his home region, then the robot moves towards the ball and clears it. Otherwise, the robot remains in the boundary of his home region, facing the ball to maintain it in his field of view. The experiments consist of two vs. two games. Thus, we have defined two possible configurations for the opponents (see Figure 11): a defender and a goalie (called the *DG* configuration), and two defenders, i.e. a midfield defender and defender (referred to as the *2D* configuration). Defenders are not allowed to enter the penalty area, which is reserved for the goalie. In the 2D configuration each defender has its own home region with an overlapping area. This strategy (assigning regions to players) is commonly used in robot soccer teams to ensure that all the regions on the field are covered by at least one player and to avoid all robots chasing the ball at the same time (which could be advantageous for the other team).

5.1.3 The Scenarios

We have defined four basic scenarios for the experimentation stage. Each scenario is used with either configuration (DG or 2D). In scenario 1 (Figure 12a) the ball (small circle) and the attackers (A and B) are positioned in the middle-back of the field, while in scenario 2 (Figure 12b), they are located in the left side of the field. The opponents (defender, D, and goalie, G, in these figures) remain within their home region (when playing against two defenders, the goalie is replaced with a defender). These scenarios correspond to general situations where the attackers are coming from the back of the field towards the attacking goal, while the opponents are waiting at their positions.

In scenarios 3 and 4 (Figures 12c and 12d), the ball and attackers are located in the middle-front of the field. These scenarios are more interesting from a strategic point of view, since the first decision (action) the attackers make (execute) is critical in their aim to reach the goal avoiding the defender(s) whose main task is to intercept or to steal the ball.

These two sets of scenarios are general enough to represent the most important and qualitatively different situations the robots can encounter in a game. Initiating the trials on the left or right side of the field does not make much difference wrt the actions the robots might perform in any of the two evaluated approaches, since they would perform the corresponding symmetric actions instead. We have neither defined any scenario with the ball near the attacking goal because the defenders would not be able to do much since they cannot enter the penalty area. Instead, we are interested in the defenders being active opponents complicating the attackers' task.

Finally, regarding the corners, although they are also interesting areas to evaluate, we have not included any specific scenario with this initial layout because

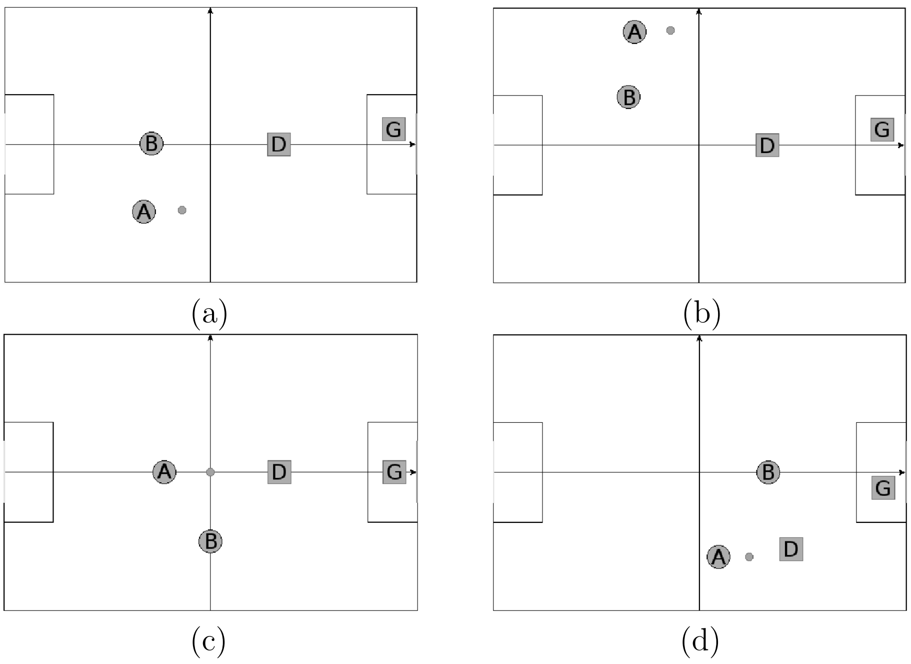


Fig. 12. Scenarios used during the experimentation with the DG configuration. Teammates are represented with circles, while opponents, with squares. (a) Scenario 1, (b) scenario 2, (c) scenario 3 and (d) scenario 4.

the big challenge within the corners is not really focused on the strategy to use, but on improving the localization of the robots. Computing the position of the robot with a minimum degree of accuracy when it is located in a corner is a very difficult localization task. The visible objects the robot can detect from that position are not enough to ensure a robust localization. Hence, we preferred to omit these initial situations because there are high chances that both approaches perform poorly. Nevertheless, during the experiments the ball may end up in a corner situation, and the approaches must somehow overcome these situations for the robots to achieve their goal.

We must remark that although the starting layout is fixed for any experimentation trial, the development of the trials varies (sometimes widely) from one to another due to the uncertainty in perception and the imprecision of the actions the robots perform. Thus, the range of situations that may occur during a trial is not known in advance.

5.1.4 Evaluation Measures

We defined two main measures to assess the performance of the approaches. The first one is based on the final outcome of a trial, while the second one is based on the opponents' ball possession during the trial.

A trial ends when either the ball goes out of the field, enters the goal, or the goalie blocks it (in the DG configuration). In order to evaluate each trial we classify the possible outcomes as:

- *goal*: the ball enters the goal.
- *close*: the ball goes out of the field but passes near one of the goalposts.
- *block*: the goalie stops or kicks the ball.
- *out*: the ball goes out the field without being a goal or close to goal.

We also consider the *to goal* balls, which correspond to balls that are either *goals* or *close* to goal. This measure indicates the degree of goal intention of the kicks. Thus, although the balls might not enter the goal, at least they were intended to do so.

Regarding the defenders' ball possession, for any trial we count the number of times that the defender touched or kicked the ball away. This measure shows the effectiveness of a cooperative behavior. We can intuitively state that having a pass when a defender is in front reduces the chances of losing the ball, if the pass does not fail. Therefore, the likelihood of successfully completing the task increases.

5.2 Simulation Experiments

The simulator used for this part of the experiments is an extended version of *PuppySim 2*, created by the CMDash team. We implemented some additional features for our experiments, such as managing team messages, robots walking while grabbing the ball, etc. The final version of the simulator is a simplified version of the real world. The robots' perception is noiseless, i.e. the ball's position and the location of all robots on the field is accurate. However the outcome of the actions the robots perform have a certain degree of randomness. The kicks are not perfect and the ball can end in different points within its trajectory (defined in Section 3.3.1). In addition, when the robot tries to get the ball, it does not always succeed, simulating a "grabbing" failure (a very common situation with the real robots). The ball's movement is modeled taking into account the friction with the field, starting with a high speed and decreasing through time and gradually ceasing (if no one intercepts it before).

We have performed two sets of experiments, one for each opponent configuration. We next proceed to analyze the experimentation results.

5.2.1 Defended and Goalie (DG) Configuration

We performed 500 trials for each approach and each scenario, i.e. a total of 4000 trials. Table 1 summarizes the ball outcome classification obtained for all

scn	app	ball classification (%)					defender's ball possession		time
		goal	close	block	out	to goal	avg	stdev	
1	cbr	25	9	38	28	34	1.34	1.37	24.39
	reactive	25	3	35	37	28	1.91	1.39	20.38
2	cbr	26	8	38	28	34	1.38	1.29	28.74
	reactive	25	6	28	41	31	2.13	1.82	27.17
3	cbr	25	6	29	40	31	1.35	1.23	27.03
	reactive	13	4	24	59	17	2.20	1.33	17.11
4	cbr	36	8	45	11	44	0.43	0.94	15.90
	reactive	22	4	49	25	26	0.85	1.42	18.43

Table 1
Ball outcome classification, defender's ball possession and time (DG configuration).

four scenarios (results in percentage) and the defender's performance during the experimention. It shows the average and the standard deviation of the number of times the defender either touched the ball or kicked it per trial. We also indicate the average execution time (in seconds) per trial.

As we can see the percentage of balls *to goal* with the CBR approach is higher in all four scenarios compared to the reactive approach. Moreover, the percentage of balls *out* are lower when using the CBR, indicating that the defender had less opportunities to take the ball and kick it out of the field. The differences are especially significant in scenarios 3 and 4, where the initial position of the defender is right in front of the ball. In these situations, it is difficult for a robot to move with the ball without losing it, which is what the reactive approach would do. Thus, the chances for the opponent to steal the ball increase. On the contrary, performing a pass between teammates is more useful, since the team keeps the possession of the ball, decreasing the opportunities for the defender to take it. Moreover, we believe that using passes in situations where it is not clear which is the best strategy (i.e. perform a pass or act individually) does not degrade the overall performance either. This is the intended strategy using the CBR approach.

Regarding the defender's performance, we can see that in general the defender playing against the reactive approach had more chances for reaching or taking the ball than when playing against the CBR approach. This is reinforced by the fact that the average time per trial is lower for the reactive approach compared to the average time for the CBR approach (except for the last scenario). Thus, even over shorter periods of time, the defender still has more chances to clear the ball. The higher average values of ball possesion for both approaches correspond to the first three scenarios. This is obvious because in

	scenario 1		scenario 2		scenario 3		scenario 4	
robot	A	B	A	B	A	B	A	B
AVG	3.3	16.0	7.2	26.2	2.8	18.6	16.5	21.6
%	17	83	21	79	13	87	43	57

Table 2
Average and percentage of backing up times per robot (A or B) and scenario for the reactive approach.

	scenario 1		scenario 2		scenario 3		scenario 4	
case	indiv	coop	indiv	coop	indiv	coop	indiv	coop
AVG	3.9	5.0	4.1	6.0	3.6	5.3	2.4	2.5
%	44	56	41	59	40	60	49	51

Table 3
Average and percentage of individualistic and cooperative cases used by the CBR approach.

these scenarios the ball is located further from the goal compared to the fourth scenario. Hence, the chances for the defender to steal the ball are higher since the distance the attacking robots have to travel to reach the goal is longer.

In order to show the degree of collaboration among robots we computed two more measures in this experimentation set. As we previously described, the reactive approach provides the robots with a simple coordination mechanism: while a robot possesses the ball, the second robot performs a default behavior to avoid interfering with the first one. Thus, in general, during a trial the robot starting the first action (e.g. get the ball and kick it) moves with the ball while the second one is backing up. Once the action ends, both robots will try to get near to the ball, but the chances for the second robot to arrive first are lower since it had previously moved away from the ball. The first robot instead, has more chances to get the ball first, while the second robot will have to move away from the ball again. For each trial, we counted the number of times each robot backed up as shown in Table 2 (average and percentage of times the robots backed up per trial). As we can see, except for the last scenario, the percentage of times that robot A backs up is significantly lower compared to robot B. Hence, we can conclude that in general, because of the strategy used, robot A (the robot that first gets the ball) acts individually without involving robot B in the task.

Since the reactive and CBR approaches are very different, we cannot apply the “number of backing ups” measure to the latter one. Therefore, to demonstrate collaborative behavior with the CBR approach, we compared the number of reused cases that implied a single robot in the solution, individualistic cases, with respect to cases with more than one robot, cooperative cases (in this

scn	app	**ball classification (%)**			scn	app	**ball classification (%)**		
		to goal	out	timeout			to goal	out	timeout
1	cbr	49.5	10.0	40.5	3	cbr	26.0	12.0	62.0
	react	30.0	22.5	47.5		react	14.5	35.0	50.5
2	cbr	32.0	12.0	56.0	4	cbr	59.5	5.0	35.5
	react	20.0	27.0	53.0		react	29.5	19.0	51.5

Table 4
Ball outcome classification (2D configuration). All trials included.

work two robots). The percentage of the type of cases used and the average per trial is detailed in Table 3. As we can observe, in general, half of the time (or even slightly more) the robots retrieve cooperative cases, i.e. cases where an explicit pass between two robots takes place. This is due to the fact that the robots start their performance in the middle of the field and with a defender in between them and the goal. In this situation a cooperative strategy is more useful since the robots can work together to get closer to the goal. Once they get near the penalty area, the best strategy is to try to score individually, and not to have passes between teammates.

5.2.2 Midfield defender and Defender (2D) Configuration

The aim of these experiments is to evaluate the performance of the approaches when playing against two more active opponents. In the previous configuration the goalie remains within the penalty region, and therefore, it could be stated that the attackers had an advantage while moving towards the attacking goal since only one opponent is actually active. Thus, in this configuration, having two defenders distributed on the field makes the attackers' task harder.

We have performed 200 trials per scenario and per approach, i.e. a total of 1600 trials. We have also slightly modified the ball classification outcome for these experiments compared to the previous ones. We observed that the time required to end a trial was too long since the defenders were really good at preventing the attackers from reaching the goal. Therefore we opted to introduce a timeout to end the trials, if no other ending situation occurred. Based on the average time of the DG experiments we set the timeout to 60 seconds (twice the highest DG experiments average time). Besides, since there is no goalie in this configuration, we eliminated the *goal* classification, and instead, considered that a trial ends when the ball enters the penalty area (*to goal* outcome).

Table 4 summarizes the results for the ball classification in percentage. We can easily observe that the attackers playing with the CBR approach always

scn	app	**midfield ball poss**		**defender ball poss**		time (sec)
		avg	stdev	avg	stdev	
1	cbr	1.14	1.30	0.51	0.74	22.29
	react	3.15	1.57	0.71	0.88	30.03
2	cbr	1.59	1.30	0.38	0.65	27.51
	react	3.59	1.72	0.55	0.85	33.07
3	cbr	2.08	1.41	0.54	0.73	28.00
	react	3.54	1.97	0.53	0.77	26.87
4	cbr	0.94	1.31	0.45	0.74	18.15
	react	2.92	1.74	1.00	0.89	24.87

Table 5
Midfield defender's and defender's ball possession and time (2D configuration). Timeout trials not included.

achieved a significantly higher percentage of *to goal* trials and a significantly lower number of *out* balls. The latter indicates that the defenders had less chances to steal the ball, and therefore, clear it out of the field. Regarding the *timeout* trials, they occur because the attackers were not able to beat the defense. Comparing both approaches, only in the last scenario there is a noticeable timeout difference, where the CBR approach clearly shows its efficiency for breaking the opponents' defensive strategy by achieving a higher percentage of *to goal* balls and a lower percentage of *out* balls.

Regarding ball possession and average time per trial, Table 5 summarizes the results obtained considering only the *to goal* balls and *out* balls to end each trial. We did not take into account the timeouts because they would "penalize" the performance of both approaches (the longer the time of a trial, the more chances for the opponents to possess the ball). As the table shows, the midfield defender had much more chances to steal the ball when playing against the attackers using the reactive approach than when playing against the CBR attackers (based on the Student t-test the differences for all four scenarios are extremely significant). This clearly shows that, in order to avoid the defender, the strategy used by the CBR approach is much more efficient than the reactive one. While the former one encourages cooperation through passes, the latter one is mainly based on individualistic plays (passes only occur by chance).

With respect to the defender's average ball possession, we can observe that is only slightly higher in the reactive trials (except for the last scenario where it doubles the average). However, we must also have into account that the chances for this defender to take the ball are lower in these experiments, since the midfield defender blocked the attackers' game more times. Thus, although the differences of ball possession is not significant, we can still conclude that the defender playing against the reactive approach had more opportunities to take the ball.

In general, similarly to the results obtained in the DG configuration, the CBR behavior once again outperforms the reactive one. Not only the duration of the trials is shorter, but also the average number of times the defenders stole the ball is lower. Also notice that the difference between the midfield defender ball possession when playing against the CBR and the reactive approach is much greater in this configuration compared to the previous DG configuration. Since the midfield defender's region is closer to the center line of the field, it has more opportunities to approach the ball and steal it when the attackers start moving individually towards the goal. However, when the players use the CBR approach the use of passes allows them to easily avoid the defense, reducing the defenders' ball possession. Based on these results, we may conclude that the CBR approach is better at (1) avoiding the opponents and (2) moving the ball quicker towards the attacking goal, when possible.

From a qualitative point of view, we may conclude that in general the defenders were really good at blocking the attackers game. In fact we noticed (since we did not know it in advanced) that the best attacking strategy was to first avoid the midfield defender, and then immediately shoot a hard kick from a long distance (around the defenders' overlapping region) instead of moving with the ball towards the penalty area avoiding the next defender. Since the CBR approach includes passes, it was easier for the attackers using this approach to apply this strategy. The attacker closest to the ball passed it to the second attacker, that could then try the hard kick, if the retrieved case indicated so. On the contrary, with the reactive approach, the first attacker moved with the ball using short kicks to avoid the midfield defender. Although it could succeed, it was difficult to repeat this same tactic with the second defender without failing. We must remark that the case base has not been specifically designed to achieve the "pass-and-shoot-to-goal" behavior. It is the retrieval process that selects the most appropriate cases from the set of individualistic and cooperative cases based on the current situation to solve.

5.3 Real Robot Experiments

The robots we have used in this experimentation are four Sony AIBO ERS-7 robots. The AIBO robot is a four-legged dog robot. A camera is located in its nose with a field of view of 56.9° wide and 45.2° high. Its internal CPU is a 576MHz processor with 64MB RAM. As we can see, it is a very limited processor and therefore we need to implement fast algorithms with low computational complexity. The robots communicate through a standard wireless network card.

The low level behaviors (such as walk or kick) are equal for both approaches, i.e. we use the CMDash'06 team code. The difference is with respect to the reasoning engine used in each approach (either CBR or reactive). Regarding the opponents' positions, the robots were not able to detect opponents effectively because the vision system is not robust enough. Therefore, to evaluate the approaches independently from vision issues, the robots from the opponent team also report their positions to all the robots in the field. Finally, during the experimentation with the CBR approach, after every cycle (i.e. retrieving and executing the case) all robots stop for 5 seconds in order to update their localization in the field with higher certainty and thus, increase the accuracy of the case retrieval.

Since working with real robots is harder than simulation (it is unfeasible to reproduce with the real robots the volume of experimentation done in simulation), for this second part of the evaluation we only used the third and fourth scenarios. As mentioned before, we believe these are more interesting scenarios than the first two scenarios because the defender is located in front of the ball, blocking the first movement the attacker could perform. With respect to opponents, we selected the DG configuration since it allows the attackers to have more chances for achieving the final goal, i.e. actually scoring, and also because, the trials take a shorter time.

We performed 30 trials per approach and per scenario, 120 trials in total. Next we evaluate both scenarios separately discussing for both approaches: first, the approach performance; second, the ball classification outcome; and finally, the defender's performance.

5.3.1 Experiments with Scenario 3

CBR approach performance. After analyzing the results of the 30 trials performed by the robots, we sketch the general behavior of the CBR approach in Figure 13. As we can observe, given the initial positions of the robots, the first action is to perform a pass to avoid the defender (Figure 13a). Hence, robot A moves towards the ball to start the pass, while robot B moves towards the front to receive the ball. Meanwhile, the defender (robot D) remains in its home region facing the ball. As the pass takes place, the defender moves to a better position to continue blocking the ball. Since robot A has ended its sequence of actions (gameplay) it performs the default behavior, maintaining a close distance to the ball, but without going after it. When robot B receives the ball, it performs a kick towards the middle line (Figure 13b) and the first case execution ends. The

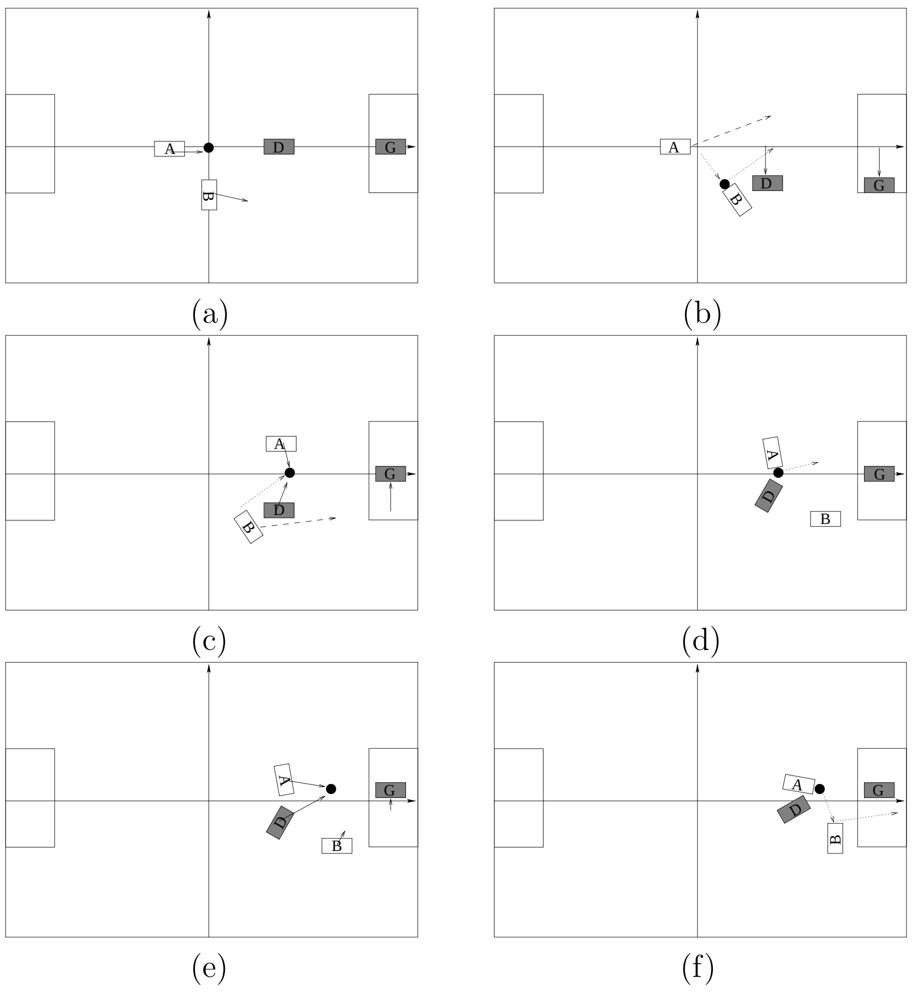


Fig. 13. Sketch performance of the CBR approach in scenario 3. Solid arrows represent the robots movements, dashed arrows, the default behaviors (moving next to the ball), and pointed arrows, the ball's movement.

next retrieved case consists of moving the ball forward in order to place it closer to the attacking goal. Hence, since robot A is closer to the ball, it is in charge of reusing alone the second case, while robot B moves next to the ball towards a better position executing the default behavior. Meanwhile the defender tries to reach the ball as well (Figures 13c and 13d). Finally, the last case is retrieved, which once again consists in a pass between robots A and B to avoid the goalie (robot G). Therefore, robot A moves to take the ball, while robot B waits for the pass (Figure 13e). After receiving the ball, kicks it towards the goal (Figure 13e).

The sequence detailed above is an ideal execution, where the attackers manage to score and the trial ends. Unfortunately, because of the high imprecision of the actions executed, the performances of the trials varied from one to another retrieving different cases (thus, executing different actions) to overcome the altered sequence. The critical points where a modification of the ideal execution occurs are:

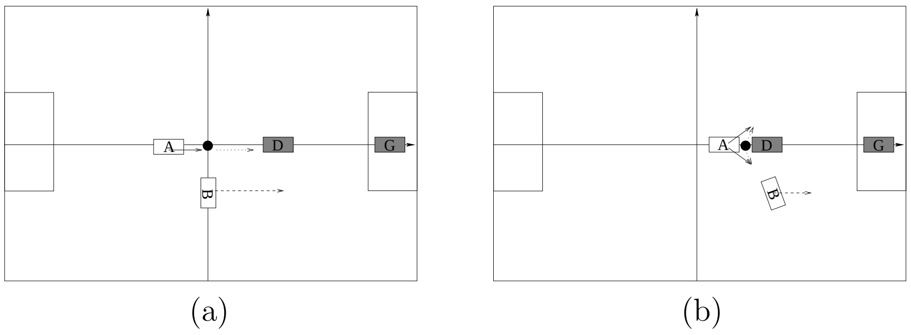


Fig. 14. Sketch performance of the reactive approach in scenario 3.

- during a pass (Figures 13b and 13f): the pass could fail because (*i*) the ball is sent in the wrong direction (usually due to incorrect localization of the robots), (*ii*) the receiver does not succeed in grabbing the ball, or (*iii*) the defender intercepts the pass.
- during the adaptation of the case (Figures 13c and 13e): while the robot is moving towards the ball, the defender may reach the ball first, clearing the ball or kicking it out of the field.

Reactive approach performance. This approach only considers the opponent's position when the opponent is very close to the ball (approximately 40 cm away at most), blocking a forward kick. Hence, in the initial trial layout, the defender is too far from the ball to be considered and therefore, robot A first performs a forward kick (Figure 14a). In the next timestep, the ball is close enough to the defender and thus, the reactive approach includes him as an obstacle that must be avoided. Since explicit passes are not modeled in this approach, the only chance for avoiding the opponent is to dodge it, moving in diagonal (either to the right or to the left) while grabbing the ball as shown in Figure 14b. The opponent, in this case the defender, also moves towards the ball and both robots collide fighting for the ball. The outcome is either a success for the attacker, getting rid of the defender and kicking the ball forward, or a success for the defender, stealing the ball and clearing it.

The overall performance of the reactive approach is generally the same: trying to move the ball close to the attacking goal, and dodging the opponent when it gets near the ball approaching from the front. At some point, the attacker reaches the attacking goal and tries to score avoiding the goalie by either turning or dodging side to side.

Ball classification The CBR approach outperforms the reactive approach. As summarized in Table 6 the percentage of balls *to goal* is higher in the CBR approach (30%) than in the reactive one (17%). The same is true about the percentage of *blocked* balls, i.e. 43% for the CBR approach and 30% for the reactive approach. Hence, the chances for scoring with the CBR approach are higher because the attackers reached the attacking goal more times, ending the trial by either scoring or trying to score. This fact is also derived from the percentage of balls *out*, where we can observe that the

scn	app	**ball classification (%)**					**num of *out* balls**		**defender's ball poss.**	
		goal	close	block	out	to goal	def	att	avg	stdev
3	cbr	20	10	43	27	30	6	2	1.40	1.16
	react	10	7	30	53	17	11	5	2.27	1.93
4	cbr	20	3	60	17	23	2	3	0.60	0.72
	react	30	7	33	30	37	5	4	1.07	0.87

Table 6
Ball outcome classification, number of *out* balls by the defender and the attackers, and defender's ball possession (real robots).

percentage for the reactive approach (53%) almost doubles the percentage for the CBR approach (27%). More precisely, the number of balls *out* due to the defender's actions is higher for the reactive approach (11) than for the CBR approach (6).

Defender's ball possession The chances for the defender to steal the ball are higher when the attackers use the reactive approach. Table 6 lists the average and standard deviation of the number of times, per trial, the defender possessed the ball, i.e. either touched or kicked the ball. This average is 2.27 when playing against the reactive attackers, and 1.40 when playing against the CBR attackers. This means that in average, at least twice per trial the defender had the opportunity to either block the ball or even worse, to clear the ball from its defending zone. Thus, we can state that the teamwork playing strategy in the CBR approach—more precisely the passes between teammates—is indispensable for reducing the opponent's chances to intercept the ball. This fact is also confirmed by the number of balls *out* mentioned above, where the defender kicked the ball out of the field more times when playing against the reactive approach.

5.3.2 Experiments with Scenario 4

CBR approach performance Similarly to the previous scenario, the first action the attackers perform is a pass between them to avoid the defender (Figure 15a and Figure 15b). After the first case execution, the ball ends close to the penalty area, where the goalie is expecting it as shown in Figure 15c. Since the goalie is on the right side of its penalty area, it is not only blocking the ball from a front kick, but also incapacitating robot A from scoring. Hence, the only option for robot B is to individually try to score dodging the goalie (Figure 15d), while the defender comes from the back trying to take the ball. Once again, failures during the execution can occur due to the reasons already mentioned in the previous scenario (errors during passes or defender reaching the ball first).

Reactive approach performance In contrast to the third scenario, now

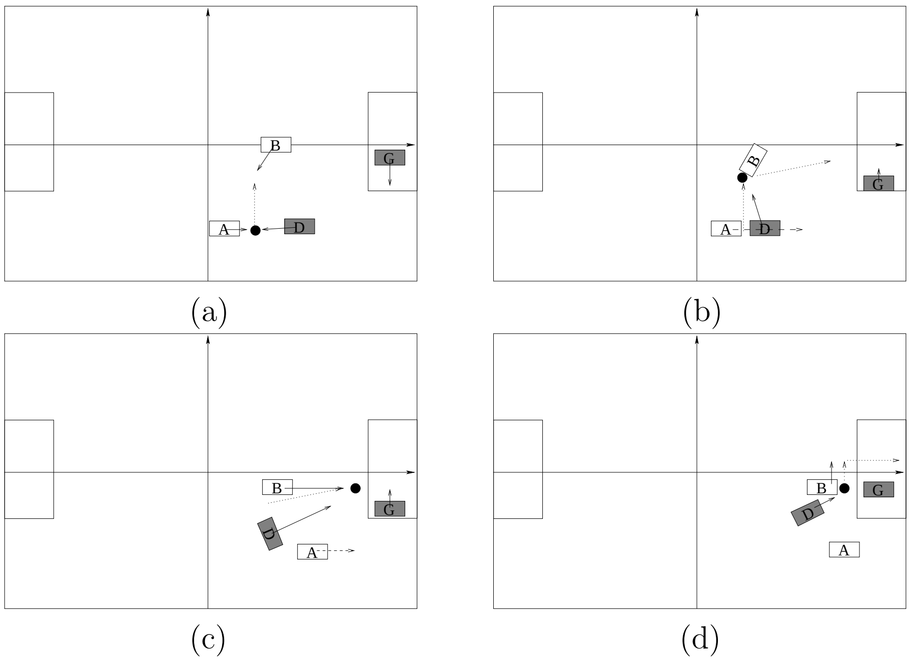
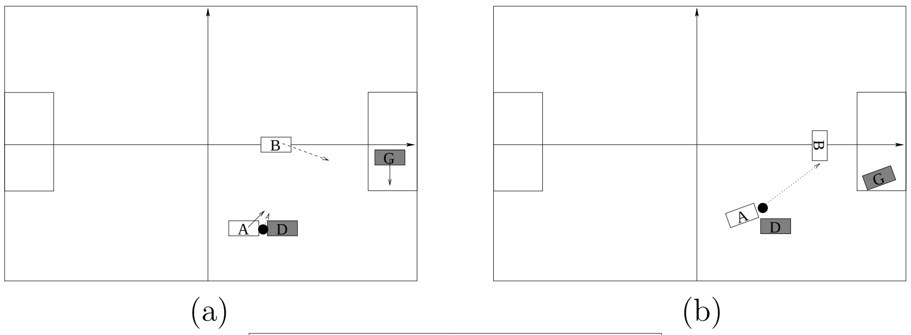


Fig. 15. Sketch performance of the CBR approach in scenario 4.

the defender is positioned close enough to the ball so that the attacker can detect him. Hence, using the dodging tactic robot A tries to avoid the defender, moving diagonally towards the left and kicking the ball forward (Figure 16a). Meanwhile, robot B moves towards the attacking goal, avoiding to intercept the ball. Once robot A has kicked the ball, robot B can immediately go after it (Figure 16b). This action could be interpreted as a pass, although it was not really meant to be. Next, robot B is close enough to the attacking goal and alone with the goalie, and therefore, tries to score (Figure 16c).

We must once again recall that the scenario described above corresponds to an ideal execution. As the results we have obtained show, most of the time the defender prevented the attackers from reaching the goal or at least, greatly complicated their task.

Ball classification In this scenario the CBR approach is not as efficient as the reactive approach. As we can observe in Table 6 the percentage of balls *to goal* using the reactive approach (37%) is higher than using the CBR approach (23%). However, we must also emphasize the fact that the percentage of *blocked* balls by the goalie is much higher for the CBR approach (60%, doubling the reactive approach). Therefore, we confirm that altough the attackers with the CBR approach did not manage to score as many goals as the attackers with the reactive approach, most of the time they reached the attacking goal and aimed at scoring. This is also supported by the fact that the percentage of *out* balls in the reactive approach (30%) almost doubles the percentage in the CBR one (17%). Moreover, the defender playing against the reactive robots had more opportunities to kick the ball out of the field (5 times vs. 2 against the CBR approach), preventing the



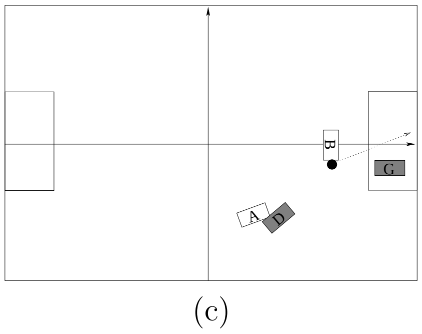


Fig. 16. Sketch performance of the reactive approach in scenario 4.

attackers from reaching the attacking goal.

Defender's ball possession Similarly to scenario 3, as Table 6 summarizes, the average number of times the defender intercepted the ball when playing against the reactive attackers (1.07) is higher than when playing against the CBR attackers (0.60). As mentioned in the discussion of the reactive approach performance at the beginning of this section, the first attacker's action is to dodge the defender moving forward with the ball, instead of performing a pass, as the CBR approach does. Hence, although the attacker might try to avoid the defender, most of the time, the defender manages to block his movements, forcing the attacker to lose the ball. Therefore, on average, at least once per trial the defender blocks the ball, complicating the task of the attacker, whereas with the CBR approach, it happened on average about once every two trials.

5.4 Discussion

In general, the results obtained both in simulation and with the real robots, confirm that the Case-Based Reasoning approach is better than the reactive approach, not only on placing a higher percentage of balls close to the opponent's goal, but also achieving a lower percentage of *out* balls. More precisely, the results obtained in the third scenario with the real robots confirms the simulation results. In the fourth scenario, once again the average of balls *out* is higher for the reactive approach, which confirms that the defense had more chances to steal the ball.

However, in the last scenario the reactive approach achieved a higher percentage of balls *to goal* compared to the CBR approach. We must point out that comparing the ideal executions of both approaches (Figures 15 and 16) we can easily observe that the reactive approach is more efficient in moving faster the ball towards the attacking goal, i.e. with two kicks the robots can score. On the contrary, the attackers with the CBR approach need at least three kicks to complete the same task. Hence, the chances for the goalie to move towards a better position to block the ball also increase, as confirmed in the percentage of *blocked* balls by the goalie (60% for the CBR approach vs. 33% for the reactive approach). These results also support the finding that the CBR approach had more chances to get closer to the attacking goal, i.e. succeeded in avoiding the defender in the first step (Figure 15b). When playing with the reactive approach, the attacker's first action was blocked most of the times (Figure 16a). Otherwise, as the ideal sequences shows, the attackers using the reactive approach would have had even more opportunities to score, considerably increasing the percentage of *to goal* balls.

We must analyze one last aspect. Comparing the results obtained for the fourth scenario in simulation and with the real robots, we can see that in simulation, the CBR approach outperformed the reactive approach. However, we must always have in mind that the high uncertainty in the perception for the robots is not considered in the simulated environment. If we take a look again at Figure 15c, we can observe that after reusing the first case, the ball stops close to the penalty area. Although the goalie is not allowed to leave its home region (the penalty area), he may believe that the ball is within its home region due to the high uncertainty in the robot's world model and, therefore, may try to grab it anyway. In the simulated environment this situation occurs, unless the ball is actually within the penalty area. Hence, while in the real world the trial could end with the goalie clearing the ball, i.e. *block* ball in the statistics, in simulation the attacker would probably try to score, i.e. a *to goal* ball.

Regarding the defenders' performance, we have confirmed that using the CBR approach is a better strategy to avoid the defenders stealing the ball because of the explicit passes between teammates. The reactive strategy almost doubles the chances for the defender to steal the ball compared to the CBR approach, even playing shorter periods of time.

With respect to the attackers, when using the CBR approach the number of times that the default (reactive) behavior was triggered was not significant and, therefore, we do not include these measures in the results. These situations only occurred when the costs of all the available cases exceeded the cost threshold, but not because there were no cases defined. However, when the attackers moved towards the ball using the reactive approach, they reduced their distances to the ball and, consequently, some of the available cases that had previously been filtered out became applicable.

Finally, and to conclude, the fourth scenario verifies that in this kind of domains (high uncertainty, highly dynamic and real time response requirements) it is useful to have a reactive strategy rather than a deliberative strategy to solve critical situations. When the players are in front of the goal, there is no need of reasoning about interesting plays. Instead, trying to score as fast as possible is the best strategy. Thus, acting fast is crucial. However, a deliberative strategy that includes collaboration outperforms a reactive one for the remaining situations, i.e. in the middle, back, sides and corners of the field where the opponents have more opportunities to steal the ball.

6 Related Work

We first briefly describe the different RoboCup Soccer leagues because they are subsequently referred to in this section that reviews related work. Then, we focus on CBR techniques that have been applied to the RoboCup domain and mostly related to the action selection problem. Next, we take a look at works where other approaches besides CBR have been previously used in the same context. Finally, we situate our work within the reviewed literature summarizing the main features of our CBR approach.

6.1 RoboCup Soccer leagues

The RoboCup Soccer competition involves several leagues. The Simulation League is completely based on software simulation with no physical robots. Thus, it is mainly focused on the decision-making of the players, multi-agent systems and artificial intelligence without the constraints of the robotic hardware. Moreover, all the data gathered by the simulator during a game can be used afterwards for learning, opponent modeling, etc.

In the Small Size League, teams consist of real robots and a host computer which receives the images sent by an overhead camera. Teams are free to design the robot's hardware while fulfilling the size requirements. All the information, as well as the decision making, is processed in the host computer. This league focuses on the problem of intelligent multi-agent cooperation and control with a hybrid centralized/distributed system.

The Middle Size League deals with teams of real autonomous robots (there is no central computer). Any hardware configuration is allowed, while fulfilling the size boundaries. This league also focuses on multi-robot systems and control, but with a completely distributed system.

Finally, in the Four-Legged League (currently replaced by the Standard Platform League) teams consist of four Sony AIBO robots which operate fully autonomously, i.e. there is no external control, neither by humans nor by computers. Robots of the same team are allowed to communicate among them (as human players do during a soccer game through speech, gestures and/or signals) through a wireless network. The field dimensions are 6m long and 4m wide.

A game consists of two 10 minute halves. The teams change the defended goal during the half-time break. A game may end if the score difference is greater than 10 points. The main difference with the Middle Size League is that teams are restricted to use the same hardware (i.e. identical robots), therefore the teams concentrate on software development. It is important to mention that it is the league that has to deal with highest perception uncertainty due to limited hardware resources.

6.2 CBR applied to RoboCup

Very close to our work, Karol et al. (2004) presented an initial attempt to include a CBR system in the action selection process for a team of robots in the Four-Legged League. The problem description includes the robots' positions, the degree of ball possession, as well as meta-level features to guide the retrieval process. As within our work, the solution corresponds to the gameplay. They proposed three possible similarity measures, all based on comparing the robots positions on the field. Since the work was only a first introduction of the model, no further evaluation of the system was repported.

Three CBR reasoners prototypes were presented by Marling et al. (2003): the first one focused on positioning the goalie, the second one on selecting team formations, and the last one on recognizing game states. The prototypes were used in the Small-Size League, although the experiments were only validated in simulation. For all prototypes, the case description represents a snapshot of the field. The features in the problem description and the solution of the case differ from one prototype to another based on the task to learn. Compared to our work, their approaches are more focused on strategic decisions (related to coaching) rather than explicitly deciding the players' actions.

The following works have been all applied to the Simulation League. We must recall that within this league, players are simulated agents, and therefore, harder problems (such as opponent modeling) can be addressed.

In the work presented by Chen and Liu (2002) they introduce a hybrid architecture for soccer players. The deliberative layer corresponds to the CBR system and the reactive layer corresponds to fuzzy behaviors (motor schemas introduced by Arkin (1989)). Cases are obtained from expert knowledge and are represented with fuzzy features. Similar to our work, they introduce the concept of *escape conditions*: a new case is retrieved only if the escape conditions of the previous retrieved case are satisfied. This way, the deliberative system monitors the current execution and the retrieval process only takes place when necessary. The solution of a case describes the behavior for a single player, and not the behavior for more players as in our approach. Regarding the evaluation, it is not clear how efficient the system is since they do not include an analysis of the team performance.

Recent work by Berger and Lämmel (2007) proposes the use of a CBR system to decide whether a "wall-pass" should be performed or not (our approach indicates different sequences of actions instead). A "wall-pass" consists on passing the ball to a teammate, to immediately receive a pass again from the latter. The idea is to distract the opponent so the first player can move to a better position. Similar to our work, a case represents the positions of the most relevant players on both teams in a given situation.

Wendler and Lenz (1998) proposed to learn about the opponents and, based on the observations, the system indicates the positions where the players should move. Our approach not only indicates the robots' positions, but also the actions to be performed. Continuing with the ideas of studying the opponent team, in (Wendler and Bach, 2004) they addressed the behavior recognition and prediction problem based on external observation. The CBR system models the function that maps the situations of a game and the behaviors of the players. The results obtained through experimentation show that although the system performs quite well, the prediction model is team-dependent, i.e. it is specific for each team. Therefore, when switching opponent teams, behavior predictions are degraded.

Focusing on prediction and opponent modeling Ahmadi et al. (2003) and Steffens (2004) coincide on arguing that the similarity measure should be adapted to the situation and role of the agent whose action is to be predicted. While Ahmadi modifies the weights of the positions of players taken into account, Steffens proposes a similarity measure that considers more or fewer features when comparing the current problem with the cases. The experimentation is focused on evaluating the system's prediction efficiency, and does not tackle the problem of how the team should behave afterwards to overcome the predicted behavior.

Learning from Observation aims at modeling 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. Lam et al. (2006) and Floyd et al. (2008) focus their research on action selection based on scene recognition in the Simulation League. Similar to our work, a matching process is defined in order to map the objects in the current scene with the ones indicated in previous scenes or cases. A k-nearest neighbor algorithm is used to obtain the most similar scenes. If more than one scene is retrieved, the most common action (majority voted) is selected. This work is closely related to ours. However, the main differences are: the number of agents implied in the case solution (we include teammates which interact among them, while they only include one teamate); modeling uncertainty (in our work we include fuzzy functions to this end); the solution description of a case (we deal with a sequence of actions for each teammate instead of a single action); and finally, adaptation (we include an adaptation stage, where the robots move towards the adapted positions, while they only reproduce the action indicated in the retrieved case).

Reinforcement Learning (RL) (Sutton and Barto, 1998) is a classical machine learning technique that has been frequently used in the RoboCup domain. Riedmiller et al. (2001) and Kleiner et al. (2003) focused their work on learning on two different levels: skill level (low level) and tactical level (high level). While the former works propose learning the levels sequentially, i.e. first learn the skills and then how to use them, the latter work opts for a simultaneous learning approach. Riedmiller et al. (2001) apply their work to the Simulation League and Kleiner et al. (2003), to the Middle-Size League. Ahmadi and Stone (2008) introduce an MDP for action selection between two types of kicks in the Four-Legged League (instead, we include a larger variety of actions, including different types of kicks). To handle the adversarial component, they recalculate the learned policy on-line when opponents appear in scene. (Park et al., 2001) propose the use of different learning modules for each player within the Small Size League. Each learning module decides between two actions: either continue with the default navigation behavior, or to switch to the *shoot* action (again, in our work different actions can be selected). Combined approaches such as the one presented by Duan et al. (2007) have also been introduced. In their work they propose a hierarchical learning module that combines fuzzy neural networks (FNN) and reinforcement learning (RL). The learning task includes dynamic role assignment, action selection and action implementation (low level skills).

Pattern Recognition has been used to solve the opponent modeling problem in the Simulation League. Huang et al. (2003) presented a mechanism to recognize and retrieve teams' plans. Lattner et al. (2006) and Miene et al. (2004) proposed an approach that applies unsupervised symbolic off-line learning to a qualitative abstraction in order to create frequent patterns in dynamic scenes.

Fuzzy Theory is another selected approach since it naturally handles imprecision and uncertainty, which are highly present in the real world (Saffiotti, 1997). A fuzzy logic based strategy for dynamic role assignment was introduced by Sng et al. (2002). Their work was applied to the Small-Size League, where a centralized coordinator assigns roles to players (but does not indicate explicit actions they should perform, as we do). Lee et al. (2005) presented similar work where a fuzzy logic system is used as a mediator to handle situations where more than one robot may be responsible for an area. Their work is applied to the Middle-Size League. Wu and Lee (2004) focused their research on the selection of five action categories within the Small-Size League. Given a set of input variables, the output of the fuzzy rules indicate the action to perform by the robot. Although this work is more related to ours, in the sense that explicit actions are chosen, the approach only considers a single player and therefore, no cooperation can be considered.

Neural Networks (NN) (Rumelhart and McClelland, 1986) have been proved to efficiently perform in many domains, including robot control. Initial work was presented by Kim et al. (1997). They proposed an action selection mechanism (ASM) based on the role of the player. In order to handle the adversarial component, the ASM also calculates the level of disturbance of opponents, i.e. how the existence of an opponent interferes in the current situation. This latter contribution is similar to the work presented by Ahmadi and Stone (2008). In order to compute the level of disturbance a multi-layer perceptron is proposed. Jolly et al. (2007) present a more complete work, where a two-stage approach using NN for action selection in the Small-Size League is proposed. More precisely, their work is focused on deciding which of the two robots near the ball must go after it while the other remains as a supporter.

Evolutionary Algorithms (EA) have been proposed in several occasions within the RoboCup domain. Nakashima et al. (2006) proposed a method for acquiring team strategies in the Simulation League. Park et al. (2006) used evolutionary algorithms to determine the appropriate fuzzy control rules for the path planning problem in robot soccer. Luke et al. (1998) studied the feasibility of evolving a fair team through genetic programming in the Simulation League.

Konur et al. (2004) focused their work on learning *decision trees* for action selection for a whole team (defenders, attackers and midfields) in the Simulated League. *Bayesian* classification methods are very common to use when dealing with uncertainty because they are based on probability theory. In the Simulation League 3D, Bustamante et al. (2007) use a fuzzy extension of Naive Bayesian Networks to determine which player is the most appropriate to receive a pass. Fraser and Wotawa (2005) proposed a framework based on traditional STRIPS (Fikes and Nilsson, 1971). They extend the classical planning problem definition by plan invariants. A plan invariant is a manually defined logical statement that has to remain satisfied in the initial and all subsequent states (similar to our work and the escape conditions proposed by Lin et al. (2002)). They also introduce a mechanism for achieving cooperative planning through role allocation.

6.4 Summary

We can classify the work reviewed based on (*i*) the problem the work is addressed and (*ii*) the league where it is applied. The problems we have focused our attention on are: *action selection*, *opponent modeling* or *state recognition* (team formation recognition), *role assignment*, *positioning* (locations on the field where the robots should move), and *skills* (learning low level actions such as kick the ball, walk, etc.). The available leagues are: Simulation League, Small-Size League, Middle-Size League and Four-Legged League.

Thus, within this classification, our work covers the action selection, role assignment, and positioning problem domains within the Four-Legged League. As we describe through this paper, the CBR system retrieves and adapts a case specifying the actions the robots must perform (action selection problem). Moreover, it indicates which actions each robot should perform and their initial positions. Hence, the role assignment and positioning problems are solved through the case reuse. Although two works have been presented in the action selection domain within this league (Karol et al., 2004; Ahmadi and Stone, 2008), we introduce a complete framework addressing the decision-making of a multi-robot system, where a set of actions for each robot is selected (not only two possible actions as in (Ahmadi and Stone, 2008)), and the subsequent execution of these actions (the work presented in (Karol et al., 2004) is preliminary and only refers to the first stage, the decision-making). Furthermore, we include the cooperative aspect in the task execution through explicit passes between robots, not only to enhance collaboration among robots, but also to face the adversarial component of the domain. This latter component has only been explicitly addressed in (Ahmadi and Stone, 2008) and (Kim et al., 1997).

Finally, to evaluate our approach, we have performed experiments consisting of two vs. two scenarios both in simulation and with real robots. Two attackers collaboratively play against two non-random opponents and making use of explicit passes, when possible. It is important to remark that experimenting with real robots with more than one robot per team is not common due to the complexity of the task. Indeed, the experiments in the reviewed literature dealing with real robots were performed with the following limitations: (*i*) with a single robot (Kleiner et al., 2003; Park et al., 2006); (*ii*) without opponents (Lee et al., 2005); (*iii*) in 1 vs. 1 scenarios (Wu and Lee, 2004; Kim et al., 1997); (*iv*) against one static opponent (Kleiner et al., 2003); or (*v*) against two random opponents (opponents just moved from one position to another randomly) (Ahmadi and Stone, 2008).

7 Conclusions and Future Work

In this work we have presented a Case-Based Reasoning approach for action selection and coordination in joint multi-robot tasks in domains where imprecision, uncertainty, unpredictability and real-time response are present. Robot soccer is an example of such domains, and it is the one in which this work is applied. Thus, we must also consider one more ingredient when attacking the problem: the adversarial component.

We have successfully extended our previous work (Ros et al., 2007) with new mechanisms to handle this latter component. More precisely, we have introduced the applicability measure, which is composed of two functions: the *free path* function, to guarantee that there are no opponents intercepting the ball's path; and the *opponent similarity*, to evaluate the relevance of the existing opponents on the field. Given a new problem to solve, the retrieval process first evaluates all cases filtering those that are not useful to solve the current problem. Next, the candidate cases are ranked based on a set of criteria. After a case is retrieved, the adaptation phase takes place. The robots move to a more appropriate position to directly reuse the case solution using a new positioning strategy to prevent the robots from being inactive too much time. Finally, the robots reuse its solution executing the sequences of actions specified in the case. To this end, a multi-robot architecture and a coordination mechanism have been presented.

To evaluate the performance of our approach, we have compared it with a reactive approach, both in simulation and with real robots. We have tested four scenarios where two attackers (either using the CBR or the reactive approach) played against two moving opponents, an important step forward with respect to our previous work. In simulation we have considered two different defending configurations: a defender and a goalie (DG), and two defenders (2D). We have selected the DG configuration in the real robots experiments mainly because the trials take a much shorter time. The results have shown that generally the CBR approach outperforms the reactive one both in simulation and with real robots and against both defending configurations, not only having more opportunities to score, but also reducing the number of balls out of the field, avoiding the opponents and moving the ball quicker towards the attacking goal. Moreover, because of the use of passes between teammates, the defenders playing against the CBR attackers had less opportunities to steal or touch the ball than when playing against the reactive attackers. This clearly shows that cooperation (having passes in the soccer domain) is a good strategy in this type of adversarial domains. To achieve cooperation it is essential to know who does what. Case-based reasoning allows us to easily include this information in the case description, so all agents are aware of the actions they have to perform. Jointly with a coordination mechanism, the multi-agent system achieves a cooperative behavior.

As future work, we are interested in combining both deliberative and reactive approaches since as we could observe from the experiments, in some situations, acting reactively fast is also important. Hence, by combining both approaches, we can benefit from their advantages.

With respect to the CBR system, we believe that the time and score of the game should be taken into account in the decision making process. We could make use of these features to model the team strategy, i.e. more conservative or more aggressive. Moreover, we could combine them with the fuzzy representation of the applicability functions. Thus, the trajectories and opponents' scopes could be dynamically enlarged or shrunk based on the strategy the team should play with. The team strategy could also be modified during a game by changing the sorting criteria of the retrieval process.

We could also consider including a more sophisticated negotiation process in the multi-robot architecture for deciding which case to reuse. Instead of having a single robot selecting the retrieved case, all retrievers could propose their solutions, and by means of negotiation techniques, agree on which is the most suitable one given the state of the game.

Finally, including automatic case acquisition and revision in the current approach is fundamental if we expect the system to automatically improve its performance as well as to adapt and to learn new situations encountered through time. However, because of the complexity of the domain, the automatic acquisition of cases is very challenging. To this end at least the following questions should be analyzed: (*i*) when to consider that a potential case has *started*?, i.e. identify the initial situation; (*ii*) when does it *finish*?, i.e. which are the actions to store as part of the solution of a case; and, (*iii*) how to evaluate the *outcome* of the performed actions to determine if the case is useful or not for achieving the robot's goals (i.e. positive or negative feedback) ? This is known as the *credit assignment problem.*

References

Ahmadi, M., Lamjiri, A. K., Nevisi, M. M., Habibi, J., Badie, K., 2003. Using a Two-Layered Case-Based Reasoning for Prediction in Soccer Coach. In: Arabnia, H. R., Kozerenko, E. B. (Eds.), International Conference on Machine Learning; Models, Technologies and Applications. CSREA Press, USA, pp. 181–185.

Ahmadi, M., Stone, P., 2008. Instance-Based Action Models for Fast Action Planning. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (Eds.), RoboCup 2007: Robot Soccer World Cup XI. LNCS. Springer Verlag, pp. 1–16.

Arkin, R. C., 1989. Motor schema-based mobile robot navigation. International Journal of Robotics Research 8 (4), 92–112.

Berger, R., Lämmel, G., 2007. Exploiting Past Experience - Case-Based Decision Support for Soccer Agents. In: KI 2007: Advances in Artificial Intelligence (German National AI Conference, KI). Vol. 4667 of LNCS. Springer, pp. 440–443.

Bustamante, C., Garrido, L., Soto, R., 2007. Fuzzy Naive Bayesian Classification in RoboSoccer 3D: A Hybrid Approach to Decision Making. In: RoboCup 2006: Robot Soccer World Cup X. Vol. 4434 of LNCS. Springer, pp. 507–515.

Chen, K.-Y., Liu, A., 2002. A Design Method for Incorporating Multidisciplinary Requirements for Developing a Robot Soccer Player. In: IEEE International Symposium on Multimedia Software Engineering. IEEE Computer Society, pp. 25–32.

Duan, Y., Liu, Q., Xu, X., 2007. Application of reinforcement learning in robot soccer. Engineering Applications of Artificial Intelligence 20 (7), 936–950.

Fikes, R., Nilsson, N., 1971. STRIPS: a new approach to the application of theorem proving to problem solving. Artificial Intelligence 2, 189–208.

Floyd, M. W., Esfandiari, B., Lam, K., 2008. A Case-based Reasoning Approach to Imitating RoboCup Players. In: International Florida Artificial Intelligence Research Society Conference. AAAI Press, pp. 251–256.

Fraser, G., Wotawa, F., 2005. Cooperative Planning and Plan Execution in Partially Observable Dynamic Domains. In: RoboCup 2004: Robot Soccer World Cup VIII. Vol. 3276 of LNCS. Springer, pp. 524–531.

Gerkey, B. P., Mataric, M. J., 2004. A formal analysis and taxonomy of task allocation in multi-robot systems. Intl. Journal of Robotics Research 23 (9), 939–954.

Grosz, B. J., Kraus, S., 1996. Collaborative Plans for Complex Group Action. Artificial Intelligence 86 (2), 269–357.

Huang, Z., Yang, Y., Chen, X., 2003. An Approach to Plan Recognition and Retrieval for Multi-agent Systems. In: Workshop on Adaptability in Multi-Agent Systems (AORC 2003).

Jolly, K. G., Ravindran, K. P., Vijayakumar, R., Kumar, R. S., 2007. Intelligent decision making in multi-agent robot soccer system through compounded artificial neural networks. Robotics and Autonomous Systems 55 (7), 589–596.

Karol, A., Nebel, B., Stanton, C., Williams, M.-A., 2004. Case Based Game Play in the RoboCup Four-Legged League Part I The Theoretical Model. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (Eds.), RoboCup 2003: Robot Soccer World Cup VII. Vol. 3020 of LNCS. Springer, pp. 739–747.

Kim, H.-S., Shim, H.-S., Jung, M.-J., Kim, J.-H., 1997. Action Selection Mechanism for Soccer Robot. In: IEEE International Symposium on Computational Intelligence in Robotics and Automation. IEEE Computer Society, Washington, DC, USA, pp. 390–398.

Kitano, H., Kuniyoshi, Y., Noda, I., Asada, M., Matsubara, H., ichi Osawa, E., 1997. RoboCup: A challenge problem for AI. AI Magazine 18 (1), 73–85.

Kleiner, A., Dietl, M., Nebel, B., 2003. Towards a Life-Long Learning Soccer Agent. In: Kaminka, G. A., Lima, P. U., Rojas, R. (Eds.), RoboCup 2002: Robot Soccer World Cup VI. Vol. 2752 of LNCS. Springer, pp. 126–134.

Kolodner, J., 1993. Case-Based Reasoning. Morgan Kaufmann, San Mateo, CA, USA.

Konur, S., Ferrein, A., Lakemeyer, G., 2004. Learning decision trees for action selection in soccer agents. In: Workshop on Agents in dynamic and real-time environments (ECAI).

Lam, K., Esfandiari, B., Tudino, D., 2006. A Scene-based Imitation Framework for RoboCup Clients. In: Workshop on Modeling Others from Observations (AAAI).

Lattner, A. D., Miene, A., Visser, U., Herzog, O., 2006. Sequential Pattern Mining for Situation and Behavior Prediction in Simulated Robotic Soccer. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (Eds.), RoboCup 2005: Robot Soccer World Cup IX. Vol. 4020 of LNCS. Springer, pp. 118–129.

Lee, J., Ji, D., Lee, W., Kang, G., Joo, M. G., 2005. A Tactics for Robot Soccer with Fuzzy Logic Mediator. In: Computational Intelligence and Security. Vol. 3801 of LNCS. Springer, pp. 127–132.

Lin, Y.-S., Liu, A., Chen, K.-Y., 2002. A Hybrid Architecture of Case-Based Reasoning and Fuzzy Behavioral Control Applied to Robot Soccer. In: Workshop on Artificial Intelligence (ICS).

Lopez de Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M. L., Cox, M. T., Forbus, K., Keane, M., Aamodt, A., Watson, I., 2006. Retrieval, reuse, revision and retention in case-based reasoning. Knowledge Engineering Review 20 (3), 215–240.

Luke, S., Hohn, C., Farris, J., Jackson, G., Hendler, J., 1998. Co-evolving Soccer Softbot team coordination with genetic programming. In: RoboCup-97: Robot Soccer World Cup I. Vol. 1395 of LNCS. Springer, pp. 398–411.

Marling, C., Tomko, M., Gillen, M., Alexander, D., Chelberg, D., 2003. Case-Based Reasoning for Planning and World Modeling in the RoboCup Small Size League. In: Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments: World Modeling, Planning, Learning, and Communicating (IJCAI).

Miene, A., Visser, U., Herzog, O., 2004. Recognition and Prediction of Motion Situations Based on a Qualitative Motion Description. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (Eds.), RoboCup 2003: Robot Soccer World Cup VII. Vol. 3020 of LNCS. Springer, pp. 77–88.

Nakashima, T., Takatani, M., Udo, M., Ishibuchi, H., Nii, M., 2006. Performance Evaluation of an Evolutionary Method for RoboCup Soccer Strategies. In: RoboCup 2005: Robot Soccer World Cup IX. Vol. 4020 of LNCS. Springer, pp. 616–623.

Park, J.-H., Stonier, D., Kim, J.-H., Ahn, B.-H., Jeon, M.-G., 2006. Recombinant Rule Selection in Evolutionary Algorithm for Fuzzy Path Planner of Robot Soccer. In: Freksa, C., Kohlhase, M., Schill, K. (Eds.), KI 2006: Advances in Artificial Intelligence (German National AI Conference, KI). Vol. 4314 of LNCS. Springer, pp. 317–330.

Park, K.-H., Kim, Y.-J., Kim, J.-H., 2001. Modular Q-learning based multiagent cooperation for robot soccer. Robotics and Autonomous Systems 35 (2), 109–122.

Riedmiller, M. A., Merke, A., Meier, D., Hoffmann, A., Sinner, A., Thate, O., Ehrmann, R., 2001. Karlsruhe Brainstormers - A Reinforcement Learning Approach to Robotic Soccer. In: Stone, P., Balch, T. R., Kraetzschmar, G. K. (Eds.), RoboCup 2000: Robot Soccer World Cup IV. Vol. 2019 of LNCS. Springer, pp. 367–372.

Riesbeck, C., Schank, R., 1989. Inside Case-Based Reasoning. Lawrence Erlbaum.

Riley, P., Veloso, M., 2002. Recognizing probabilistic opponent movement models. In: Birk, A., Coradeschi, S., Tadokoro, S. (Eds.), RoboCup-2001: The Fifth RoboCup Competitions and Conferences. Springer Verlag, Berlin, pp. 205–245.

RoboCup, 2008. Robocup web site. http://www.robocup.org/.

Ros, R., Arcos, J. L., 2007. Acquiring a Robust Case Base for the Robot Soccer Domain. In: Veloso, M. (Ed.), International Joint Conference on Artificial Intelligence. AAAI Press, pp. 1029–1034, India.

Ros, R., de Mantaras, R. L., Arcos, J. L., Veloso, M., 2007. Team Playing Behavior in Robot Soccer: A Case-Based Approach. In: International Conference on Case-Based Reasoning. Vol. 4626 of LNAI. Springer, pp. 46–60, Northern Ireland.

Ros, R., Veloso, M., 2007. Executing Multi-Robot Cases through a Single Coordinator. In: International Conference on Autonomous Agents and Multiagent Systems. pp. 1264–1266, USA.

Ros, R., Veloso, M., de Mantaras, R. L., Sierra, C., Arcos, J. L., 2006. Retrieving and Reusing Game Plays for Robot Soccer. In: European Conference on Case-Based Reasoning. Vol. 4106 of LNAI. Springer, pp. 47–61, Turkey.

Rumelhart, D. E., McClelland, J. L., 1986. Parallel distributed processing: explorations in the microstructure of cognition. MIT Press, Cambridge, MA, USA.

Saffiotti, A., 1997. The uses of fuzzy logic in autonomous robot navigation.

Soft Computing 1 (4), 180–197.

Smyth, B., Keane, M. T., 1998. Adaptation-guided retrieval: questioning the similarity assumption in reasoning. Artificial Intelligence 102 (2), 249–293.

Sng, H. L., Gupta, G. S., Messom, C. H., 2002. Strategy for Collaboration in Robot Soccer. In: IEEE International Workshop on Electronic Design, Test and Applications. IEEE Computer Society, pp. 347–354.

Steffens, T., 2004. Adapting similarity-measures to agenttypes in opponent-modelling. In: Workshop on Modeling Other Agents from Observations (AAMAS).

Stone, P., Veloso, M. M., 1999. Task decomposition and dynamic role assignment for real-time strategic teamwork. Artificial Intelligence 110 (2), 241–273.

Sutton, R. S., Barto, A. G., 1998. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA.

Vail, D., Veloso, M., 2003. Dynamic multi-robot coordination. In: Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II. Kluwer Academic Publishers, pp. 87–100.

Veloso, M. M., 1994. Planning and Learning by Analogical Reasoning. Vol. 886 of LNCS. Springer Verlag.

Wendler, J., Bach, J., 2004. Recognizing and Predicting Agent Behavior with Case Based Reasoning. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (Eds.), RoboCup 2003: Robot Soccer World Cup VII. Vol. 3020 of LNCS. Springer, pp. 729–738.

Wendler, J., Lenz, M., 1998. CBR for Dynamic Situation Assessment in an Agent-Oriented Setting. In: Aha, D., Daniels, J. J. (Eds.), Workshop on Case-Based Reasoning Integrations (AAAI).

Wu, C.-J., Lee, T.-L., 2004. A Fuzzy Mechanism for Action Selection of Soccer Robots. Journal of Intelligent Robotics Systems 39 (1), 57–70.