A Case-Based Approach for Action Selection and Coordination in Robot Soccer Gameplays

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 a robot's behavior in imprecise, uncertain, dynamic, unpredictable and real-time response domains is very challenging, and even more if an adversarial component is also present. An example of such domains is the robot soccer. In this work we propose the use of Case-Based Reasoning techniques to handle all these features in the action selection problem of a team of robots. Moreover, we are also interested in obtaining a cooperative behavior among robots to successfully perform joint tasks. Thus, we include explicit passes between robots, not only to enhance collaboration, but also to face the adversarial component of the domain, i.e. the opponents. We evaluate our approach with respect to a reactive approach in 2 vs. 2 scenarios, where two attackers play against two moving opponents, i.e. a defender and a goalie. We successfully show that our approach not only achieves the expected team behavior, but also outperforms in general the compared approach.

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

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.

1 Introduction

Designing a robot's behavior is a challenging task that researchers have addressed for a long time. A simple task such as moving an object from one point to another implies the intervention of a set of abilities that a robot must be provided with in order to successfully achieve its goals. Some of these are: ability to perceive its environment and build its internal world model, ability to make decisions when planning the task, ability to navigate through the environment and to avoid obstacles during the navigation, ability to execute the planned actions, and ability to recover from failure, among others. Clearly, 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 results the robot's behavior design. Besides, the robot features also play an important role in the robot's performance of the task. To easily understand the difficulties that the environment complexity and the robot's structure involves (such as imprecision, uncertainty, predictability, or real-time response), let us compare an industrial robot with a rescuer robot.

Usually in a factory, the environment of a moving robot is specially designed so that it can easily travel from/to a set of predefined locations through well-marked fixed paths (easily visible[2] from the robot's point of view). Besides, the access or use of these paths are prohibited for other users to avoid obstacles that could interfere with the robot's path. The tasks to perform are pre-programmed consisting of fixed sequences of actions. The robot's hardware (its physical structure) is specially designed for the task to fulfill attaching manipulating tools to the robot with high precision in their movements. In case of failure, the robot usually stops performing the task and alerts a human operator to solve the problem.

A rescuer robot instead, deals within a completely unstructured environment. In this type of environments no predefined paths exist. On the contrary, the robot should find a "safe" and "easy" path to travel through. No explicit marks indicate the robot's position, and therefore, the perception module should make use of different types of landmarks chosen at execution time so the robot can localize itself in the environment. Obstacles can suddenly appear due to sliding objects, recalling the need of including an obstacle avoidance mechanism during navigation. The future set of actions the robot should perform depend on the current state of the world, and thus, decisions have to be made on-line and in real time. Although the robot may be provided with ad-

[2] The concept of visibility is not restricted to the use of cameras. Robots can include other types of sensors, such as sonars or infrareds. Instead, the visibility concept is related to the perception capacity of the robot.

ditional tools to facilitate its work, not all the necessary specialized tools can be attached, basically because not all possible situations can be considered in advance. Hence, the robot's performance accuracy usually decreases and an unpredictability component arises since the outcome of the execution of a given action in repeated occasions does not always results in the same way. Finally, the robot must be provided with a recovery mechanism to overcome unexpected situations and to autonomously abort or restart the task, when possible.

Another fundamental aspect to consider when designing the robot's behavior, and not yet mentioned, is the cooperative component. Is the robot going to interact and work with other robots? Collaboration is desired in several domains where a group of robots (also seen as agents) 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 the task can be organized to obtain effective results. Providing the agents with capacities to collaborate and to coordinate is complex, as it is not just a matter of dividing the tasks and assigning roles to each agent. Instead, it is also a matter of beliefs and commitments of all robots to fulfill a common task. Grosz and Kraus (1996) argue that collaborating agents must (*i*) establish mutual beliefs on what actions they will perform to complete the task, (*ii*) agree on who does what, and (*iii*) establish mutual beliefs of their individual intentions to act. Communication among agents is essential to achieve these requirements.

Finally, the computational resources of the robots also drive, at some extent, the choice of the techniques and approaches used in the robot's behavior design. Thus, if we are dealing with limited resources (memory capacity, CPU power, etc.) computationally complex algorithms (in terms of resource consuming) must be discarded.

It is easy to see that a broad range of challenges and research opportunities emerge from the above scenarios. From these challenges we focus our work on the action selection and coordination for joint multi-robot tasks. Thus, unpredictability, dynamism, uncertainty, imprecision, real-time response and coordination are some of the aspects that we must deal with. We have gradually tackled most of them in the past (Ros et al., 2006; Ros and Veloso, 2007; Ros et al., 2007), where we have successfully applied Case-Based Reasoning (CBR) techniques to model the reasoning engine of a team of robots within the robot soccer domain. However, our previous approach lacked a fundamental component: the adversarial one. This last component cannot be addressed as part of the dynamism and unpredictability features of the environment since by definition, the concept involves opposition and resistance. In other words, this component will try to prevent the robots from achieving their goals, not just as dynamic obstacles that can randomly appear or disappear unexpectedly, but in fact, as dynamic obstacles with a specific goal: lead the robots

to failure. In the domain this work is applied to, robot soccer, the adversarial component clearly corresponds to the opponent robots.

In this context, since the adversarial behavior is driven by a goal, i.e. prevent other robots fulfilling their task, the opponents actions are not random and in fact, are guided through a strategy or function. It is easy to see that a good attempt to cope with this issue would be to try to learn this strategy, so the robots could then switch their behavior with some counter-strategy. This problem has been already studied by researchers in the past (Wendler and Bach, 2004; Ahmadi et al., 2003; Steffens, 2004; Huang et al., 2003; Miene et al., 2004; Marling et al., 2003) and it corresponds to the opponent modeling problem. However, it is not always feasible to solve this problem due to the characteristics of the problem domain where the learning is addressed to. In general, a complete model of the world is required, where high precision in the information is desirable to successfully model the objects behaviors. As we review in Section 2, within the robot soccer domain, the Simulation and the Small-Size Leagues are the most suitable application domains to study this problem. In these leagues there is a central computer gathering all the information about the environment, such as ball and players positions, which then can be used as inputs to the learning mechanism. However, in other leagues, as the one this work is applied to (the Four-Legged League), each robot has its own world state, and even worst, with low accuracy, and there is no centralized system to collect and fuse all the information. Therefore, from one robot's point of view, trying to even recognize the behavior of another robot, is a very challenging problem. Due to this difficulties, alternative mechanisms must be considered to overcome the adversarial component of the domain, such as the one we propose in this work, .

The CBR approach (López de Màntaras et al., 2006) models the state of the world at a given time as a problem description, which is compared to a set of predefined situations, called cases. 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). The retrieval process consists in obtaining the most similar case, the retrieved case. Next, the solution of the retrieved case is reused after some adaptation process, as required. We model the case solution as a set of sequences of actions, which indicate what actions should each robot perform. We specify a coordination mechanism that takes place during both the retrieval and the reuse steps based on messages exchanged among the robots about their internal states (beliefs and intentions).

Hence, reviewing the requirements mentioned above by Grosz and Kraus (1996), we ensure that: the solution description indicates the actions the robots should perform (requirement *i*); the retrieval process allocates robots to actions (requirement *ii*); and finally, with the coordination mechanism, the robots share their individual intentions to act (requirement *iii*). We must re-

mark that collaboration is essential in this kind of adversarial domains. It is not enough to find a way of modeling or representing this component, but also, to determine a strategy that allows the robots to overcome this factor. In the soccer domain, it is well known that having passes between teammates is one of the most adequate strategies to avoid opponents. In contrast, using an individual strategy, where only one robot moves with the ball without taking into account its teammates, increases the chances for the opponent to block the attack. Therefore, we have successfully included the pass action in our approach, which is not common, as far as we know, in this domain (Four-Legged League).

In our previous work (Ros et al., 2007), the retrieval process consisted in 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. In this paper, we significantly extend the retrieval process with a new measure, the applicability measure. This measure makes use of the adversarial features of the environment, i.e. the opponents in this domain, to determine if the reuse of a given case is feasible or not, despite of its similarity degree and cost with respect to the problem to solve. The retrieval process consists of a ranking algorithm of filtered cases that determines the best candidate based on a set of criteria. Some criteria can be domain dependent, while others can be domain independent. It is part of the designer's task to define the most appropriate criteria to use given the domain where the approach is applied to.

Regarding the reuse step, before the robots start executing their assigned actions, a positioning process takes place, i.e. the robots move to the initial positions specified in the description of the case being reused. However, in this kind of domains where saving time is essential, a conservative strategy as presented in (Ros et al., 2007) is not suitable. Therefore, we propose an alternative positioning strategy for domains where the time invested for satisfying the initial conditions is crucial. Finally, we show empirical results that confirm the effectiveness of our collaborative approach compared to an individualistic approach. The scenarios include two moving opponents, which is a significant step forward in complexity with respect to our previous work (Ros et al., 2007), where there was only one static opponent (the goalie).

Application Domain

The RoboCup Soccer competition involves several leagues. One of them is the one we apply our work to: the Four-Legged League. In this 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. Communication among

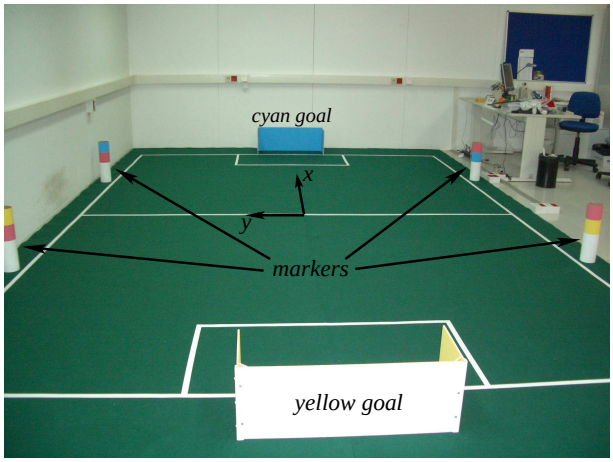


Fig. 1. Snapshot of the Four-Legged League field (image extracted from the IIIA lab).

robots of the same team is allowed through wireless. The field dimensions are 6m long and 4m wide and represents a Cartesian plane as shown in Figure 1. There are two goals (cyan and yellow) and four colored markers the robots use to localize themselves on the field. A game consists of two 10 minutes halves parts. 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 paper is organized as follows. We review related work within the robot soccer domain in Section 2. Next, we describe the CBR approach we have designed, starting with the case definition in Section 3, and continuing with the case retrieval and case reuse through a multi-robot architecture in Sections 4 and 5 respectively. Section 6 details the empirical evaluation we have performed, and Section 7 summarizes the conclusions of this work, as well as proposing future work.

2 Related Work

We have divided this section in three subsections as follows. First, we review related work where CBR techniques have been applied within the RoboCup domain related to the action selection problem. Next, we take a look at works where other approaches besides CBR, such as Reinforcement Learning, 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.

Very close to our work, Karol et al. (2004) presented an initial attempt for including a CBR system in the action selection of 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 is only a first introduction of the model, no experiments were reported.

The work presented by Lin et al. (2002) and Chen and Liu (2002) is applied to the Simulation League, where they presented 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)). The knowledge acquisition is done through first order predicate logic (FOPL), which they claim it is easy for an expert to transmit knowledge. Similar to our work, they introduce the concept of *escape conditions*: a new case is retrieved only if the escape conditions are satisfied. This way, the deliberative system monitors the current execution and the retrieval process only takes place when necessary.

Recent work has been initiated by Berger and Lämmel (2007) where they propose the use of a CBR system to decide whether a "wall-pass" should be performed or not. 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. Their work is applied to the Simulation League. A case represents the positions of the most relevant players on both teams in a given situation.

Since the initiation of RoboCup, Wendler et al. have addressed different problems within this domain. The first one, and more related to our work is presented in (Wendler and Lenz, 1998). In this work, they proposed to learn about the opponents and, based on the observations, adapt the actions of the players within the Simulation League. More precisely, the system indicates the positions where the players should move according to the opponents actions. Continuing with the ideas of studying the opponent team to improve the performance of the team players, 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. Finally, in (Wendler et al.,

2001) a fault-tolerant self localization approach was proposed by means of CBR techniques within the Four-Legged League. Given a picture of the environment from the robot's point of view, the position of the robot on the field is obtained.

Three CBR reasoners prototypes were presented by Marling et al. (2003): the first one is 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 validated in simulation only. 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.

A common drawback of CBR systems usually discussed among researchers is the difficulties for fast retrieval in large case bases. Focusing on this issue, Ahmadi et al. (2003) presented a two-layered CBR system for prediction in the Simulation League. The importance of a player position varies based on its relation with the ball location. Thus, a case is evaluated based on weighted features. The upper CBR layer is in charge of assigning these weights. Therefore, every lower layer case must be adapted to propose different solutions based on the areas of the field where the situation is taking place.

The last work reviewed in this section corresponds to the work presented by Steffens (2004) addressed to opponent modeling in the Simulation League. Similarly to the work presented by Ahmadi above, he argues 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 less features when comparing the current problem to solve with the cases. The relevance of the attributes is based on the positions and roles of the agents.

2.2 Other approaches applied to RoboCup

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) 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. 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 scenes (we include teammates which interact among them, while they only include one teamate); the objects locations (robots and ball are within fixed regions of field, whereas we deal with variable regions); modeling uncertainty (in our work we include fuzzy functions to this end); and the solution of the problem (we deal with a sequence of actions for each teammate instead of a single action).

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 ones propose learning the levels sequentially, i.e. first learn the skills and then how to use them, the latter ones opt 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. 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 the action to perform, i.e. either continue with the default navigation behavior, or to switch to the *shoot* action. Combined approaches such as the one presented by Duan et al. (2007) have been also 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 is applied to the Small-Size League, where a centralized coordinator assigns the roles to the players. 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, although experiments are only shown in a simulated environment. 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. 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) presented their work addressed to the action selection using a fuzzy extension of the Naive Bayesian Networks. 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 sentence 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.

2.3 Summary

We can classify the work reviewed based on the problem the work is addressed to and the league where it is applied to. The problems we have focused our attention to are: *action selection*, *opponent modeling* or *state recognition* (team formation recognition), *role assignment*, *positioning* (locations on the field where the robots should move), *localization* (computation of the robot's position on the field by itself) 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 addressed to 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). In our work, we propose a multi-robot architecture and a coordination mechanism to handle cooperation. Finally, to evaluate our approach, we have performed experiments consisting of two vs. two scenarios both in simulation and with real robots, where the two attackers collaboratively play against a defender and a goalie (non-random opponents) making use of explicit passes, when possible.

3 Case Definition

A case represents a snapshot of the environment at a given time from a single robot point of view. 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, which defines

the applicability boundaries of cases used in the retrieval step. 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.

3.1 Problem Description

The problem description corresponds to a set of features that describe the current state of the game from a single robot's perspective. We call this robot the *reference* robot, since the information in the case is based on its perception and internal state (its beliefs). In the robot soccer domain we consider the following features as the most relevant for describing the state of the game:

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

where:

(1) R: reference robot's global position (x_0, y_0)

$$x_0 \in [-2700..2700]\text{mm},\ x_0 \in \mathbb{Z} \qquad y_0 \in [-1800..1800]\text{mm},\ y_0 \in \mathbb{Z}$$

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

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

(3) G: defending goal

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

(4) Tm: teammates' global positions

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

where tm_i is the robot identifier and $n \in [1, 3]$ for teams of 4 robots. This set could be empty for cases where no teammates are implied in the case solution. We only model in the problem description those robots that are relevant for the case, although actually more robots may be available on the field.

(5) Opp: opponents' global positions.

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

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 described in the

case. Similarly to the teammates feature, we only consider the relevant opponent robots, and not all those that are currently on the field.

Although only global coordinates have been introduced in the problem description, for some computations of the retrieval process we use the relative coordinates with respect to the ball's position instead, which are easily derived from the information in the problem description.

3.2 Solution Description

The solution of a case corresponds to the sequences of actions each robot 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:

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

where $n \in [0, 3]$ is the robot identifier, and p_i the number of actions teammate tm_i performs (tm_0 corresponds to the reference robot). 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 global 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_1 must pass the ball to robot tm_0, and robot tm_2 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_1 starts moving towards the ball to get it, while robot tm_0 waits for robot tm_1 to executes the pass. Once robot tm_1 has done the pass, tm_0 receives the ball and kicks it forwards. In the meantime, since robot tm_2 has no preconditions, it starts moving to point p independently from the state in which the other robots are. For this example, the solution would be:

$$A = \begin{pmatrix} tm_0 : [\ wait,\ receive\_ball(tm_1),\ kick(\text{forward})], \\ tm_1 : [\ get\_ball,\ pass\_ball(tm_0)], \\ tm_2 : [\ go\_to\_point(p)] \end{pmatrix}$$

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 opponent robots 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 regions centered in the object's position with radius τ^x and τ^y. The case scope is defined as:

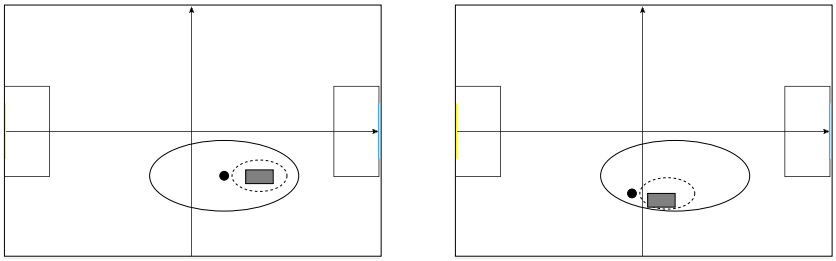
$$K = (ball : (\tau_B^x, \tau_B^y), opp_1 : (\tau_1^x, \tau_1^y), \ldots, opp_m : (\tau_m^x, \tau_m^y))$$

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 the robots that are opponents, and not the ones belonging to the team, i.e. reference robot and teammates. As we will explain during the retrieval process, we define two different measures for each type of robots. 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 4. 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 relative coordinates and their scopes 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, specially, as we have said, because of the high uncertainty in the incoming information, and moreover, because it facilitates the generalization of similar situations. For instance, it is more general to reason about an opponent being in front of the ball, rather than the opponent 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 opponent is considered to be in front of the ball because it is located within the opponent's scope. Note that the opponent's scope has been translated with respect to the current position of the ball in the problem. This is because we



(a) (b)

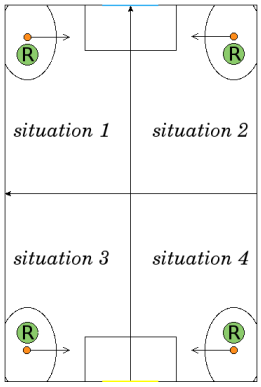
Fig. 2. (a) Example of the scope of a case. The black circle represents the ball and the gray rectangle represents the opponent. The ellipses correspond to the ball's scope (solid ellipse) and the opponent's scope (dashed ellipse). (b) Example of a simplified problem description. The opponent's scope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

use the opponents' relative representation instead of the global one. Since the ball is also situated within the ball's scope of the case, we can state that the case in Figure 2a is a potential solution to the problem in Figure 2b.

3.4 Domain Properties

We can observe two symmetric properties of the ball's and robot's 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, 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) as shown in Figure 3a. 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 every case in the case base, we can compute its symmetric descriptions, obtaining three more cases. Figure 3b shows an example of the case described in *situation 1*.

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



(a)

$$case = \left(\begin{array}{c} P = \left(\begin{array}{l} R_G = (1990, 1470), R = (-250, 0), \\ B = (2240, 1470), G = \text{yellow}, \\ Tm_G = \oslash, Tm = \oslash, \\ Opp_G = \oslash, Opp = \oslash, \\ t = 5, S = 1 \end{array} \right), \\ S = \left(tm_0 : \left[\begin{array}{l} get\_ball, \\ kick(\text{right}) \end{array} \right] \right), \\ K = \left(ball : (609, 406) \right) \end{array} \right)$$

(b)

Fig. 3. (a) Situation 1 corresponds to the original description of the case. While situations 2, 3 and 4 correspond to their 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. .

space in large case bases. In this work, we use simple techniques to this end, as we will explain in next section.

4 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 more detail these measures we first define two types of features describing the problem:

- *controllable* features, i.e. position of the reference robot and the teammates. (the robots can move to more appropriate positions if needed).
- *non-controllable* features, i.e. the ball's and opponents' positions and the defending goal (which we cannot directly modify).

The idea of separating the features into controllable and non-controllable 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. 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 must be evaluated through similarity

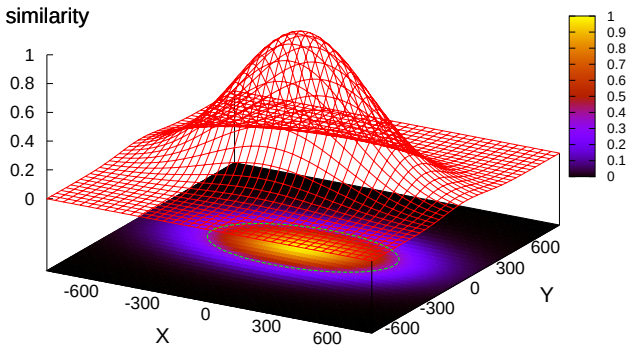


Fig. 4. 2D Gaussian centered in the origin with $\tau^x = 450$ and $\tau^y = 250$. The solid ellipse on the plane XY corresponds to $G(x, y) = 0.367$.

without altering 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 to reuse.

4.1 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 is, the lower the similarity degree between them. We propose to use a Gaussian function, which besides fullfilling these properties, it is parametrized by its variance. We can use this parameter to model the maximum distance allowed to consider 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, in the robot soccer domain, we define the similarity function for the ball feature as:

$$sim_B(x_p, y_p, x_c, y_c) = G(x_p - x_c, y_p - y_c) = exp\Big(-\Big[\Big(\frac{x_p - x_c}{\tau_B^x}\Big)^2 + \Big(\frac{y_p - y_c}{\tau_B^y}\Big)^2\Big]\Big)$$

where (x_p, y_p) corresponds to the ball's position in problem p, (x_c, y_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 3.3. Figure 4 draws a 2D Gaussian function and its projection on the XY plane (sequence of ellipses with increasing radius as the similarity decreases). As we can observe, the Gaussian's projection with radius

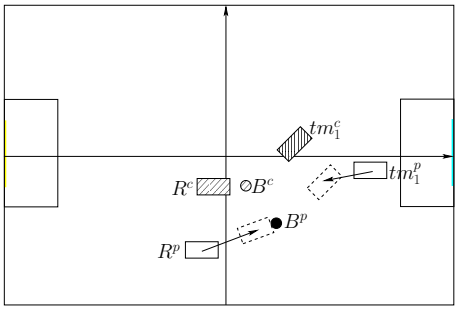


Fig. 5. Case description (R^c, B^c, tm_1^c), and current problem description (R^p, B^p, tm_1^p). The dashed rectangles represent the adapted positions of the robots with respect to the ball's position described in the problem.

τ_B^x and τ_B^y represents the scope of the ball, i.e. the region within which we consider two points to be similar enough, and corresponds to $G(x, y) = 0.367$.

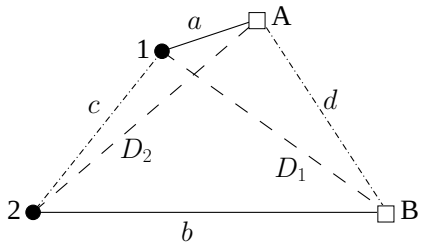
4.2 Cost Measure

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

We refer to the adapted positions as those global locations where the robots should position in order to execute the solution of the case. In general, to compute them we transform the robots' relative coordinates to global coordinates, having the position of the ball in the problem as the reference point. Figure 5 illustrates a simple adaptation example with two robots. Robot R is the one that controls the ball first, while tm_1 waits to receive the pass.

In order to compute the cost of adapting a problem to a case we must first determine the correspondence between the robots described in the problem and the ones described in the case, i.e. which robot r_i from the problem description corresponds to which robot r_j in the case description. Moreover, we must find the best match, i.e. the one that minimizes the cost, including one restriction: the distance between two points must be shorter than a given threshold, thr_c. Due to the domain's dynamism, the distances the robots have to travel must be limited since we cannot allow the robots to move from one point to another for long periods of time, because 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 and fixed ($n = 3$; the goalie is always the same robot, so we only have to find the best match for the remaining three players of the team) we can easily compute all pos-



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

Fig. 6. Trapezoid 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.

sible matches ($3! = 6$). However, as the number of robots becomes larger, the number of combinations increases exponentially. 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 that the resulting paths are easily accessible for each robot (not disturbing other robots of the same team). 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 trapezoid (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.

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:

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

where n is the number of robots that take part of the case solution, $dist$ is the Euclidian distance, r_i is the current position of robot i and $adaptPos_i$, the adapted position for robot i.

4.3 Case Applicability Measure

This last measure makes use of the opponent feature, which has not been included so far. 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 already mentioned previously, 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 two functions:

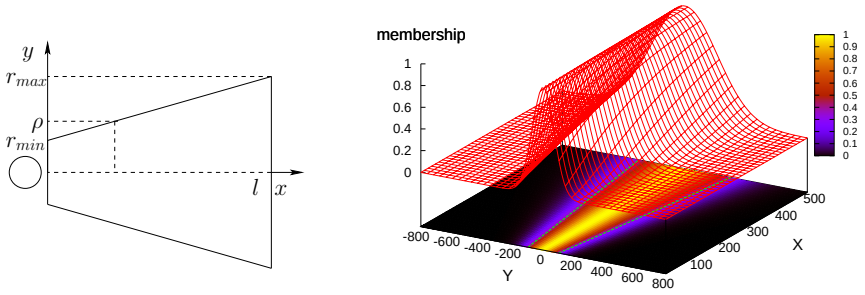
- *free path function*: the trajectory of the ball indicated in the case must be free of opponents to consider the evaluated case to be applicable.
- *opponent similarity*: a case includes information about opponents when these are relevant for the described situation, i.e. they represent a significant 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 similarity between the problem and the case.

Next, we describe both functions in detail.

4.3.1 Free Path Function

Given a case, the *free_path* function indicates whether the trajectories the ball follows during the execution of the case solution is 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 to the trajectory such that the closer the point to the center of the trajectory, the higher the membership. More precisely, this function is defined as a sequence of unidimensional Gaussians along a central axis, where the width of each Gaussian increases from a minimum radius (for $x = 0$) to a maximum one (for $x = l$) defined in the trajectory. The projection of the μ function on the XY plane results in a trapezoid (Figure7a). This trapezoid covers the area of the field where the ball could most likely go through according to the experimentation we have performed. We formally define the membership function for a



(a) (b)

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 solid lines on the plane XY correspond to $\mu(x, y) = 0.367$

trajectory t_j as:

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

where r_{min}, r_{max} correspond to the minimum and maximum radius respectively, and l, 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. Figure 7b shows the membership function described.

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. Formally, we define the free path function as:

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

where,

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

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 (each opponent opp_i is represented by the coordinates (x, y) of its position) in problem p, and $\mu_{t_j} \in [0, 1]$ is the membership function. We consider 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.

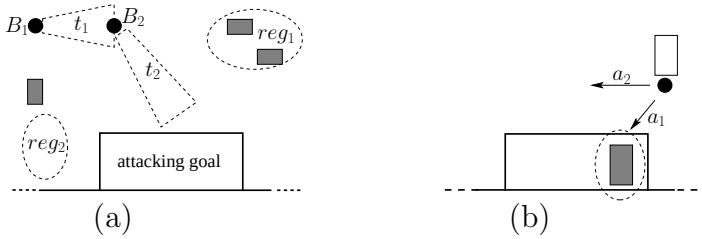


Fig. 8. (a) Example of the ball's path performed by a pass between two players (robots are not depicted for simplicity). The dashed ellipses represent the opponents regions described in the case, and the gray rectangles, the opponents described in the problem to solve. (b) Opponent similarity as a justification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

4.3.2 Opponent's Similarity

Due to the uncertainty and imprecision properties of the domain, opponents on the field are modeled by means of elliptical regions (as defined in Section 3.3), instead of only considering (x, y) positions. The opponent's similarity measure indicates the number of these regions that are occupied by at least one opponent described in the problem to solve. We call them *restrictions*. As more restrictions are satisfied, the more similar the state of the game and the case description are. Figure 8a shows an example where only one restriction is fulfilled, since only one region (reg_1) is occupied by at least one opponent. We define the opponent similarity function between a problem p and a case c as:

$$sim_{opp}(p, c) = |\{reg_j \mid reg_j \in Reg,\ \exists opp_i \in Opp\ (\Omega_{reg_j}(opp_i) > thr_{opp})\}|$$

where,

$$\Omega_{reg_j}(opp_i) = G(x_i - x_j, y_i - y_j) = exp\Big(-\Big[\Big(\frac{x_i - x_j}{\tau_j^x}\Big)^2 + \Big(\frac{y_i - y_j}{\tau_j^y}\Big)^2\Big]\Big)$$

and Reg is the set of elliptic regions in case c (reg_1 and reg_2 in Figure 8a) and Opp is the set of opponents (opp_i is represented by the coordinates (x_i, y_i) of its position) described in problem p. Each region reg_j is defined by an ellipse with radius τ_j^x and τ_j^y centered in (x_j, y_j) (the opponent's scope indicated in the case as defined in Section 3.3). We define Ω as a 2D 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. Thus, to consider that an opponent is within a given region we set the threshold thr_{opp} to 0.367. Once again, we could use the degree of occupation of a given region 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 lies in the candidate cases sorting process 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 basically given by 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.

4.4 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, for the retrieval process, we use a filtering mechanism. 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 an indexed list based on the defending goal feature. Thus, we only evaluate those cases that have equal defending goal 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 ($thr_b = 0.367$). Next, from 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 4.2) is below the cost threshold ($thr_c = 1500$mm). Finally, if the ball's path is free of opponents (line 7) then we consider the evaluated case as a valid candidate (line 8).

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

- *number of teammates that take part in the solution of the case*: we are interested in using cases with multiple robots implied in the solution so we can obtain a cooperative team behavior instead of an individualistic team, where only one robot takes part in the execution of actions. Therefore, the more teammates implied in the gameplay, the better.

Algorithm 1 IsCandidate(p, c)

```
1: if sim_B(B_p, B_c) > thr_b then
2:   for all (robot_p, robot_c) ∈ correspondence(p, c) do
3:     if dist(robot_p, robot_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
```

- *number of fulfilled restrictions* (according to the opponent similarity as explained in Section 4.3.2): the more restrictions satisfied, the better.
- *trade-off between adaptation cost (as explained in Section 4.2) and similarity (as explained in Section 4.1)*: among those cases whose similarities to the problem to solve are within a given rank, the one with lower cost is preferable. 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 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 intervals:

$$H = [0.8, 1] \quad h = [0.6, 0.8) \quad l = [0.4, 0.6) \quad L = (0.0, 0.4)$$

  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 restrictions, (2) number of teammates and (3) trade-off between cost and similarity. This way we ensure that: first, the case is as similar as possible with respect to the opponents positions; second, that the solution involves as much teammates as possible to enhance collaboration among robots, and third, a trade-off between cost and similarity.

After building the sorted list based on the three criteria we obtain:

$$ordered\_list = [c_i, c_j, \ldots c_k]$$

and finally, the selected case corresponds to the first element of the *ordered_list*. The overall retrieval process is presented in Algorithm 2. Its inputs are the problem to solve, p, and the case base, CB.

Algorithm 2 Retrieve(p, CB)

```
1: for c in CB do
2:    if IsCandidate(p, c) then
3:       candidates ← append(c, candidates)
4:    end if
5: end for
6: ordered_list ← sort(candidates)
7: ret_case ← first(ordered_list)
8: return ret_case
```

5 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, the user who queries the reasoner is not a single one, but a team of robots. In this section, we first describe the multi-robot architecture, next, how the team of robots decide which robot retrieves a case, and finally, how they reuse the retrieved case.

5.1 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,...) and to exchange information about their internal state (eg. retrieving, adapting, executing,...).

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.

The first step of the process is to decide which of the *retriever* robots is going to actually retrieve a case to solve the new problem (since only one case can be

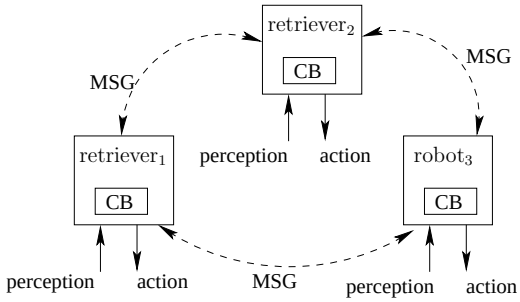


Fig. 9. Multi-robot architecture for $n = 3$ robots and $k = 2$ *retrievers*. Each robot has a library of cases (CB).

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 less 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. They are also constantly sending messages about their current internal beliefs (position, ball's position, etc.) to the rest of the robots. As a consequence, we cannot ensure that all robots agree on who is the one 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 it selects the coordinator, it sends a message to all the robots indicating the Id of the new coordinator.

After the coordinator is selected, it retrieves a case according to the process described in Section 4 and informs the rest of the team which case to reuse. It also informs 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). If no case is retrieved, a default behavior is performed (the designer should decide the most appropriate one based on the domain requirements).

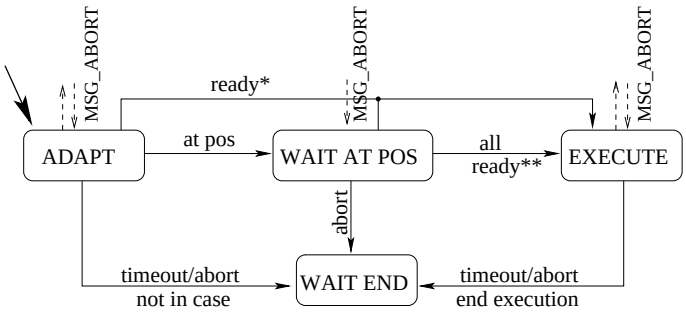


Fig. 10. Finite state machine for the case execution (*independent positioning strategy, **dependent positioning strategy).

5.2 Case Reuse

The case reuse begins when the coordinator informs the team about the retrieved case. Figure 10 describes the finite state machine (FSM) for the case reuse process. First, all robots that take part of the solution of the case (including the coordinator) start moving to their adapted positions, ADAPT state, computed as shown in Section 4.2. The robots that do not take part of the case reuse remain in the WAIT END state (either waiting at their positions or performing an alternative default 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 worst, the opponents may take the ball. As we described in Section 3.2, 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*, 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 as 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 to successfully fulfill the task.

The execution of the case continues until all robots finish their sequences of actions. Finally, they report 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 go back to the initial state of the process, 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 either detects that the retrieved case is not applicable anymore or an unexpected message arrives. 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.

6 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 have special consideration of the opponents. Thus, they try to avoid the opponents by passing the ball between them, 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 respect to 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 it to the rest of the team, and then the rest of the 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*.

6.1 Experiments Setup

Two types of experiments were performed: simulated experiments and real robots experiments. For both experiments we initialize each trial by position-

ing the robots (two attackers vs. a defender and a goalie) and the ball in a fixed location. A trial ends either when the attackers score a goal, the ball goes out of the field or the goalie touches it.

The case base used for the experimentation is composed of 136 cases, which cover the most significant situations that can occur in the evaluation presented in this work. From this set, 34 cases are initially defined, while the remaining ones are automatically generated using spatial transformations exploiting the symmetries of the soccer field as described in 3.4.

6.1.1 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 5 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. However, these situations usually occur when the cost of any of the available cases is over the cost threshold, and not because there are no cases defined. Hence, while the attackers move towards the ball performing the reactive approach, they reduce their distances with respect to the ball. At some point, any of the available cases that was previously filtered out due to cost issues, may now become a candidate solution.

With respect to the opponents, we have implemented a simple behavior for the defender and the goalie. Both perform the same behavior when playing against any of the two evaluated approaches. Each robot has a home region which cannot go beyond. If the ball is within its home region, then the robot moves towards the ball and clears it. Otherwise, the robot remains in the boundary of its home region, facing the ball to maintain it in its point of view.

6.1.2 The Scenarios

We have defined four scenarios for the experimentation stage. In scenarios 1 and 2 (Figures 11a and 11b), the ball (small circle) and the attackers (A and B) are positioned in the middle-back of the field, while the defender (D) and the goalie (G) remain within their home region. 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 11c and 11d), the ball and attackers are located in the middle-front of the field, the goalie remains within the penalty area and the defender is right in front of the ball. These type of scenarios are more interesting from a strategic point of view, since the first decision (action) the

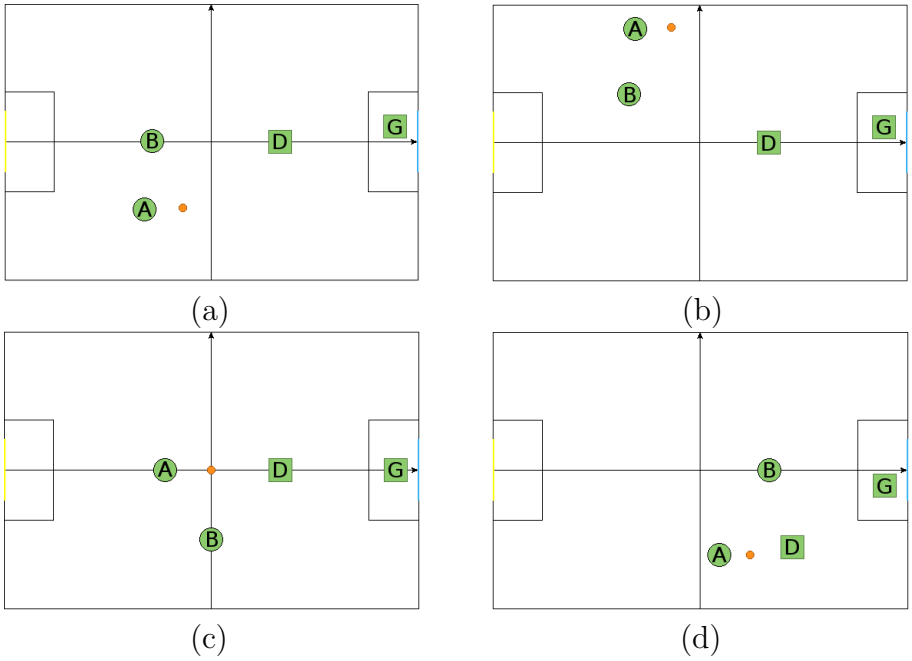


Fig. 11. Scenarios used during the experimentation. Teammates are represented with circles, while opponents, with squares. (a) Scenario 1, (b) scenario 2, (c) scenario 3 and (d) scenario 4.

attackers make (execute) is critical in their aim to reach the goal avoiding the defender whose main task is to intercept or to steal the ball.

We believe that 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 on the actions the robots might perform in any of the two evaluated approaches, since they would perform their symmetric actions instead. We have neither defined any scenario with the ball near the attacking goal because the defender would not be able to do much since it cannot enter the penalty area. Instead, we are interested in having the defender as an active opponent 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 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 for both approaches to perform poorly. Nevertheless, during the experiments the ball can end up in a corner situation, and the approaches must somehow overcome these situations for the robots to achieve their goal.

6.1.3 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 opponent's (the defender) ball possession during the trial.

As mentioned before, a trial ends when either the ball goes out of the field, enters the goal, or the goalie blocks it. 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. More precisely, at most 25cm to the left (right) of the left (right) goalpost.
- *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 defender's ball possession, for every 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 the defender to get the ball, if the pass does not fail. Therefore, the likelihood of successfully completing the task increases.

6.2 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 4.3.1). 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).

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

Table 1
Ball outcome classification and defender's ball possession (simulation).

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

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 specially 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 aimed 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. Furthermore, in the last scenario, it even doubled the average. The higher average values for both approaches correspond to the first three scenarios. This is obvious because in 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

	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 the backing up times per robot (A or B) and scenario.

	scenario 1		scenario 2		scenario 3		scenario 4	
case	single	mult	single	mult	single	mult	single	mult
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 "single" and "multiple" cases used during the experimentations.

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 back up again and 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 gets the ball first) 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, "single" cases, with respect to cases with more than one robot "multiple" cases (in this 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 multiple 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 (multiple cases) is more useful since the robots can work together to get closer to the goal. Once they get near the penalty area, it makes more sense to try to score individually (single cases), and not to have passes between teammates.

6.3 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 (the 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 lower uncertainty and thus, increase the accuracy of the case retrieval.

Since working with real robots is harder than in 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 than the first two scenarios because the defender is located in front of the ball, blocking the first movement the attacker could perform.

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.

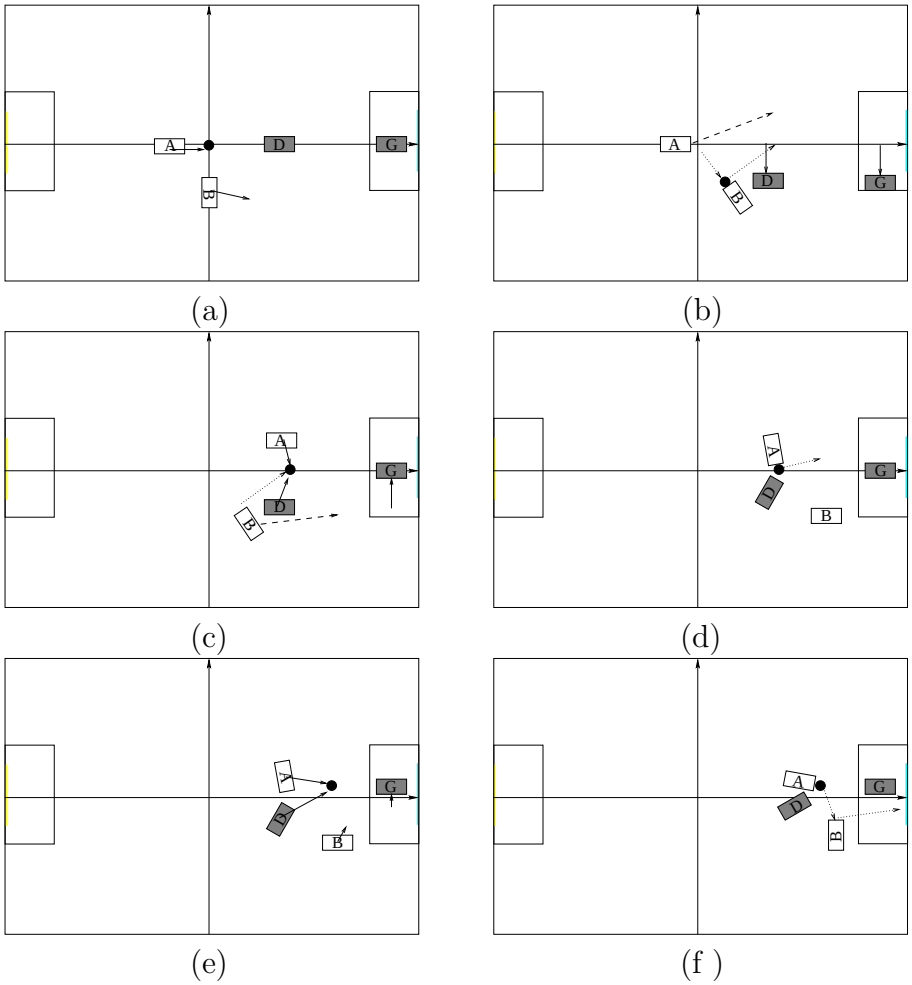
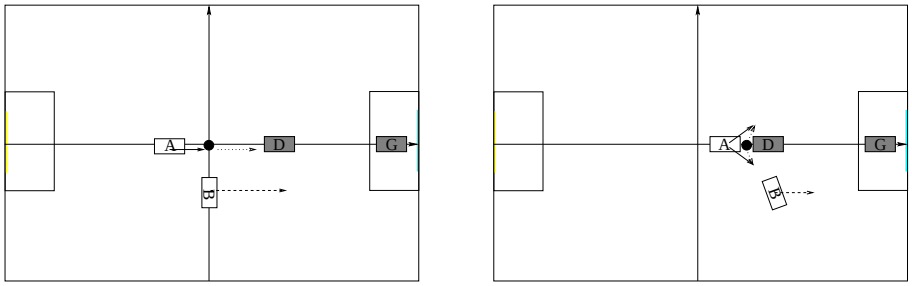


Fig. 12. Sketch performance of the CBR approach in scenario 3. Solid arrows represent the robots movements, dashed arrows, the default behaviors (moving next to the ball), and pointed arrows, the ball's movement.

6.3.1 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 12. As we can observe, given the initial positions of the robots, the first action is to perform a pass to avoid the defender (Figure 12a). 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 12b). The first case reuse ends. The next retrieved case consists in moving the ball forward in order to place it closer to the attacking goal.



(a) (b)

Fig. 13. Sketch performance of the reactive approach in scenario 3.

Hence, as 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 12c and 12d). Finally, the last case is retrieved, which once again consists in having a pass between robots A and B to avoid the goalie (robot G). Hence, robot A moves to take the ball, while robot B waits for the pass (Figure 12e). Once it receives the ball, it kicks it towards the goal (Figure 12e).

The sequence detailed above is a perfect execution, where the attackers manage to score and the trial ends. Unfortunately, because of the high imprecision of the action executions, 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:

- during a pass (Figures 12b and 12f): the pass could fail because (*i*) the ball is sent to the wrong direction (usually due to wrong 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 12c and 12e): 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 takes into account the position of the opponent for making decisions when the opponent is very close to the ball (approximately 40 cm away at most), blocking it from a forward kick. Hence, in the initial trial layout, the defender is too far from the ball to be considered during the decision making and therefore, robot A first performs a forward kick (Figure 13a). In the next timestep, the ball is close enough to the defender and thus, the reactive approach includes it 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 13b. 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

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 4
Ball outcome classification, number of *out* balls by the defender and the attackers, and defender's ball possession (real robots).

kicking the ball forward, or a success for the defender, stealing the ball and clearing it.

The overall performance of the reactive approach is the same in general, 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 either turning or dodging side to side.

Ball classification The CBR approach outperforms the reactive approach. As summarized in Table 4 the percentage of balls *to goal* is higher in the CBR approach (30%) than in the reactive one (17%), as well as 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 either scoring or trying to score. This fact is also derived from the percentage of balls *out*, where we can observe that the percentage for the reactive approach (53%) even 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) with respect to 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 4 lists the average and standard deviation of the number of times the defender possessed the ball, i.e. either touched or kicked the ball. The average of the defender's ball possession is 2.27 in contrast to the average of 1.40 when playing against the attackers with the CBR approach. This means that in average, at least two times per trial the defender had the opportunity to either block the ball or even worst, to clear the ball from its defending zone (the half side of the field it defends). 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 kicks the ball out of the field more times when

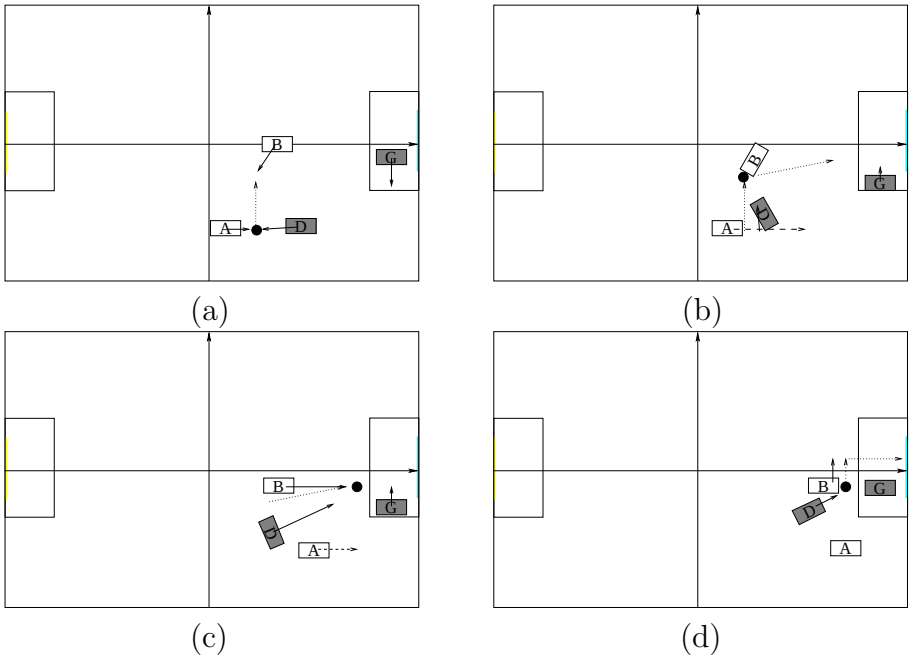


Fig. 14. Sketch performance of the CBR approach in scenario 4.

playing against the reactive approach.

6.3.2 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, while the latter tries to take the ball (Figure 14a and Figure 14b). After the first case reuse, the ball ends close to the penalty area, where the goalie is expecting it as shown in Figure 14c. 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 14d), while the defender comes from the back trying to take the ball on time. 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, in this occasion the opponent is positioned close enough to the ball, so the attacker can detect it. Hence, using the dodging tactic robot A tries to avoid the defender, moving diagonally towards the left and kicking the ball forward (Figure 15a). 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 15b). 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 15c).

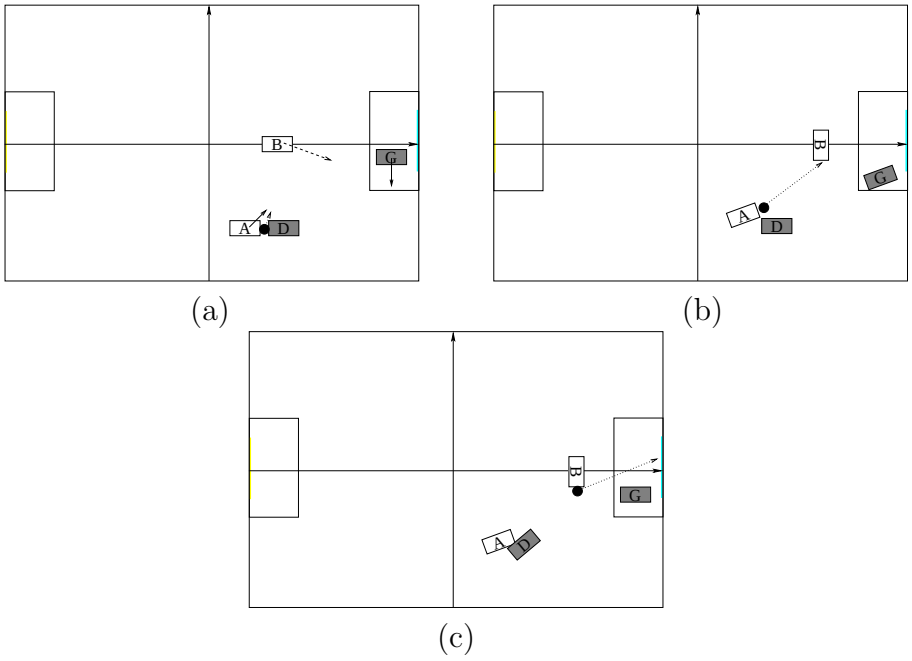


Fig. 15. Sketch performance of the reactive approach in scenario 4.

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

Ball classification The CBR approach is not as efficient as the reactive approach. As we can observe in Table 4 the percentage of balls *to goal* using the reactive approach (37%) is higher than using the CBR approach (23%). However, we must also take special attention to the fact that the percentage of *blocked* balls by the goalie is much higher for the CBR approach (60%, it doubles 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 times they reached the attacking goal and aimed at scoring. This is also reinforced 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 attackers from reaching the attacking goal.

Defender's ball possession Similarly to scenario 3, as Table 4 summarizes, the average number of times the defender intercepted the ball when playing against the reactive approach (1.07) is higher than when playing against the CBR approach (0.60). As mentioned in the approach performance, the first attacker's action using the reactive approach 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 times, the defender manages to block its movements, forcing the

attacker to lose the ball. Therefore, in average, at least once per trial the defender blocks the ball, complicating the task of the attacker to finally move the ball towards the attacking goal, whereas with the CBR approach, it happened about once every two trials in average.

6.4 Discussion

In general, analyzing both, the results obtained in simulation and the ones obtained with the real robots, we can confirm that the Case-Based Reasoning approach indeed improves upon 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 of the third scenario with the real robots confirms the results obtained in simulation. In the fourth scenario, once again the average of balls *out* is higher for the reactive approach, which confirms that the defender had more chances to take 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 14 and 15) we can easily observe that the reactive approach is more efficient on faster moving the ball towards the attacking goal, i.e. with two kicks the robots can score. On the contrary, the attackers with 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 that at least the CBR approach had more chances to get closer to the attacking goal, i.e. succeeded on avoiding the defender in the first step (Figure 14b), while with the reactive approach, the attacker's first action was blocked most of the times (Figure 15a). Otherwise, as the ideal sequences shows, the attackers would have had even more opportunities to try scoring, considerably increasing the percentage of *to goal* balls.

We must analyze a last issue. 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. Hence, no matter that the number of kicks to goal is higher, the CBR can still improve the reactive approach. However, we must always have in mind that the high uncertainty in the perception for the robots is not present in the simulated environment. If we take a look back to Figure 14c, we can observe that after reusing the first case, the ball stops near the penalty area. Although the goalie is not allowed to leave its home region (the penalty area), due to the high uncertainty in the robot's world model, the goalie may believe that the ball is within its

home region, and therefore, try to grab it. In the simulated environment this situation never takes place, 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 be able to score, i.e. a *to goal* ball.

This fourth scenario situation, where the attackers are in front of the opponent's goal, verifies that in this kind of domains (high uncertainty, highly dynamic and real time response requirements) to solve critical situations it is useful to have a reactive strategy rather than a deliberative strategy. When the players are in front of the goal, there is no need of reasoning about interesting plays. Instead, being focused on trying to score as fast as possible is the best strategy. We must also remember that the defenders are forbidden to enter the penalty area, and thus, the opponent team has less opportunities to take the ball. The goalie is the last obstacle to score a goal. Thus, acting fast is crucial. However, a deliberative strategy 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 chances of stealing the ball.

Regarding the defender's performance, we have confirmed that using the CBR approach is a better strategy to avoid the defender 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.

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 highly present. Robot soccer is an example of such domains, and it is the one this work is applied to. Thus, we must also consider one more ingredient when attacking the problem: the adversarial component.

We have successfully extended our previous work 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 sorted based on a set of criteria. After a case is retrieved, 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, where a new positioning strategy is proposed to prevent the robots from being too much time inactive.

Finally, 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 (using the CBR or the reactive approach) played against a defender and a goalie, i.e. two moving opponents, an important step forward with respect to our previous work. The results have shown that the CBR approach outperforms the reactive one in general, not only having more opportunities to score, but also reducing the number of balls out of the field. Moreover, because of the use of passes between teammates, the defender 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 having passes is a good strategy in this type of adversarial domains.

As future work, we are interested in combining both strategies, i.e. a deliberative with a reactive, since as we could observe from the experiments, for crucial situations, acting fast is also important. Hence, by combining both approaches, we can benefit from their advantages. With respect the CBR system, we believe that the time and score of the game should take part of the decision making. 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. Finally, we are also interested in 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 world.

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.), Proceedings of the 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. Springer Verlag, to appear.

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. Vol. 4667 of Lecture Notes in Computer Science. 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 Lecture Notes in Computer Science. Springer, pp. 507–515.

Chen, K.-Y., Liu, A., 2002. A design method for incorporating multidisciplinary requirements for developing a robot soccer player. In: ISMSE. 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.

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 Lecture Notes in Computer Science. Springer, pp. 524–531.

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 Lecture Notes in Computer Science. Springer, pp. 739–747.

Kim, H.-S., Shim, H.-S., Jung, M.-J., Kim, J.-H., 1997. Action selection mechanism for soccer robot. In: Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation. IEEE Computer Society, Washington, DC, USA, p. 390.

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 Lecture Notes in Computer Science. Springer, pp. 126–134.

Konur, S., Ferrein, A., Lakemeyer, G., 2004. Learning decision trees for action selection in soccer agents. In: ECAI-04 Workshop on Agents in dynamic and real-time environments.

Lam, K., Esfandiari, B., Tudino, D., 2006. A scene-based imitation framework for robocup clients. In: Workshop on Modeling Others from Observations (AAAI 2006).

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 Lecture Notes in Computer Science. 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: Proceedings of Computational Intelligence and Security. Vol. 3801 of Lecture Notes in Computer Science. 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, 2002 International Computer Symposium (ICS2002).

López de Màntaras, 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 Lecture Notes in Computer Science. 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: IJCAI-03 Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments: World Modeling, Planning, Learning, and Communicating.

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 Lecture Notes in Computer Science. 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 Lecture Notes in Computer Science. 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. Vol. 4314 of Lecture Notes in Computer Science. Springer, pp. 317–330.

Park, K.-H., Kim, Y.-J., Kim, J.-H., 2001. Modular q-learning based multi-agent 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 Lecture

Notes in Computer Science. Springer, pp. 367–372.

Ros, R., Arcos, J. L., 2007. Acquiring a robust case base for the robot soccer domain. In: Veloso, M. (Ed.), Proceedings of the 20th International Joint Conference on Artificial Intelligence. AAAI Press, pp. 1029–1034, India.

Ros, R., de Màntaras, R. L., is Arcos, J. L., Veloso, M., August 2007. Team playing behavior in robot soccer: A case-based approach. In: Proceedings of the 7th International Conference on Case-Based Reasoning. Vol. 4626 of Lecture Notes in Artificial Intelligence. Springer, pp. 46–60, Northern Ireland.

Ros, R., Veloso, M., May 2007. Executing multi-robot cases through a single coordinator. In: Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems. pp. 1264–1266, USA.

Ros, R., Veloso, M., de Màntaras, R. L., Sierra, C., is Arcos, J. L., September 2006. Retrieving and reusing game plays for robot soccer. In: 8th European Conference on Case-Based Reasoning. Vol. 4106 of Lecture Notes in Artificial Intelligence. 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.

Sng, H. L., Gupta, G. S., Messom, C. H., 2002. Strategy for collaboration in robot soccer. In: DELTA. 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 at AAMAS 2004.

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

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 Lecture Notes in Computer Science. Springer, pp. 729–738.

Wendler, J., Brggert, S., Burkhard, H.-D., Myritz, H., 2001. Fault-tolerant self localization by case-based reasoning. In: Stone, P., Balch, T. R., Kraetzschmar, G. K. (Eds.), RoboCup 2000: Robot Soccer World Cup IV. Vol. 2019 of Lecture Notes in Computer Science. Springer, pp. 259–268.

Wendler, J., Lenz, M., 1998. CBR for Dynamic Situation Assessment in an Agent-Oriented Setting. In: Aha, D., Daniels, J. J. (Eds.), AAAI-98 Workshop on CaseBased Reasoning Integrations.

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.