Improving Reinforcement Learning by using Case Based Heuristics

Reinaldo A. C. Bianchi[1,2], Raquel Ros[2], and Ramón López de Mántaras[2]

[1] Centro Universitário da FEI, São Bernardo do Campo, Brazil.
rbianchi@fei.edu.br,
[2] Artificial Intelligence Research Institute (IIIA-CSIC), Bellaterra, Spain.
{ros,mantaras}@iiia.csic.es

Abstract. This work presents a new approach that allows the use of cases in a case base as heuristics to speed up Reinforcement Learning algorithms, combining Case Based Reasoning (CBR) and Reinforcement Learning (RL) techniques. This approach, called Case Based Heuristically Accelerated Reinforcement Learning (CB-HARL), builds upon an emerging technique, the Heuristic Accelerated Reinforcement Learning (HARL), in which RL methods are accelerated by making use of heuristic information. CB-HARL is a subset of RL that makes use of a heuristic function derived from a case base, in a Case Based Reasoning manner. An algorithm that incorporates CBR techniques into the Heuristically Accelerated Q–Learning is also proposed. Empirical evaluations were conducted in a simulator for the RoboCup Four-Legged Soccer Competition, and results obtained shows that using CB-HARL, the agents learn faster than using either RL or HARL methods.

1 Introduction

Case Based Reasoning (1; 2) techniques have been shown to be useful in a multitude of domains, with widespread applications ranging from the optimization of autoclave loading (3), the diagnosis and treatment of many medical problems (4), to the synthesis of high quality expressive music (5).

Reinforcement Learning (RL) is also a very successful Artificial Intelligence technique. RL algorithms are very useful for solving a wide variety problems when their models are not available a priori, since many of them are known to have guarantees of convergence to equilibrium (6; 7). Unfortunately, the convergence of a RL algorithm may only be achieved after an extensive exploration of the state-action space, which is usually very time consuming.

One way to speed up the convergence of RL algorithms is making use of a conveniently chosen heuristic function, which can be used for selecting the appropriate actions to perform in order to guide exploration during the learning process. Several Heuristically Accelerated Reinforcement Learning (HARL) methods that makes use of a heuristic function have been recently proposed (8; 9). These techniques are very attractive: as RL, they are based on firm theoretical foundations. As the heuristic function is used only in the choice of the action to be taken, many of the conclusions obtained for RL remain valid for HARL algorithms, such as the guarantee of convergence to equilibrium in the limit and the definition of an upper bound for the error.

Although several methods have been successfully applied for defining the heuristic function, a very interesting option has not been explored yet: the reuse of previously learned policies, using a Case Based Reasoning approach. This paper investigates the combination of Case Based Reasoning (CBR) and Reinforcement Learning (RL) techniques, with the goal of speeding up RL algorithms by using previous domain knowledge, stored as a case base. To do so, we propose a new algorithm, the Case Based Heuristically Accelerated Q–Learning (CB-HAQL), which incorporates Case Based Reasoning techniques into an existing HARL algorithm, the Heuristically Accelerated Q–Learning (HAQL).

The application domain of this paper is that of the RoboCup Standard Platform League, Four-Legged Soccer Competition (10), where teams consisting of four Sony AIBO robots operating fully autonomously and communicating through a wireless network compete in a 6 x 4 m field. This domain is one of many RoboCup challenges, which has been proven to be an important domain for research, and where RL techniques have been widely used. Nevertheless, the technique proposed in this work is domain independent.

The paper is organized as follows: Section 2 briefly reviews the Reinforcement Learning problem; Section 3 describes HARL approach and the HAQL algorithm, while section 4 describes Case Based Reasoning. Section 5 shows how to incorporate CBR techniques into HARL algorithms, in a modified formulation of the HAQL algorithm. Section 6 describes the robotic soccer domain used in the experiments, presents the experiments performed, and shows the results obtained. Finally, conclusions are presented in Section 7.

2 Reinforcement Learning and the Q–Learning algorithm

Reinforcement Learning (RL) algorithms have been applied successfully to the on-line learning of optimal control policies in Markov Decision Processes (MDPs). In RL, this policy is learned through trial-and-error interactions of the agent with its environment: on each interaction step the agent senses the current state s of the environment, chooses an action a to perform, executes this action, altering the state s of the environment, and receives a scalar reinforcement signal r (a reward or penalty).

The RL problem can be formulated as a discrete time, finite state, finite action Markov Decision Process (MDP). The learning environment can be modeled by a 4-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, where:

- $\mathcal{S}$: is a finite set of states.
- $\mathcal{A}$: is a finite set of actions that the agent can perform.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$: is a state transition function, where $\Pi(\mathcal{S})$ is a probability distribution over $\mathcal{S}$. $T(s, a, s')$ represents the probability of moving from state s to s' by performing action a.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$: is a scalar reward function.

Table 1. The Q–Learning algorithm.

```
Initialize $\hat{Q}_t(s,a)$ arbitrarily.
 Repeat (for each episode):
    Initialize $s$.
    Repeat (for each step):
         Select an action $a$ using equation 2.
         Execute the action $a$, observe $r(s,a)$, $s'$.
         Update the values of $Q(s,a)$ according to equation 1.
         $s \leftarrow s'$.
     Until $s$ is terminal.
Until some stop criteria is reached.
```

The goal of the agent in a RL problem is to learn an optimal policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ that maps the current state s into the most desirable action a to be performed in s. One strategy to learn the optimal policy π^* is to allow the agent to learn the evaluation function $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$. Each action value $Q(s,a)$ represents the expected cost incurred by the agent when taking action a at state s and following an optimal policy thereafter.

The Q–learning algorithm (11) is a well-know RL technique that uses a strategy to learn an optimal policy π^* via learning of the action values. It iteratively approximates Q, provided the system can be modeled as an MDP, the reinforcement function is bounded, and actions are chosen so that every state-action pair is visited an infinite number of times (the complete algorithm is presented in Table 1). The Q learning update rule is:

$$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha \left[r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right], \tag{1}$$

where s is the current state; a is the action performed in s; r is the reward received; s' is the new state; γ is the discount factor ($0 \leq \gamma < 1$); and α, is the learning rate. To select an action to be executed, the Q–Learning algorithm usually considers an $\epsilon - Greedy$ strategy:

$$\pi(s) = \begin{cases} \arg\max_a \hat{Q}(s,a) & \text{if } q \leq p, \\ a_{random} & \text{otherwise} \end{cases} \tag{2}$$

where:

- q is a random value uniformly distributed over $[0,1]$ and p ($0 \leq p \leq 1$) is a parameter that defines the exploration/exploitation tradeoff: the larger p, the smaller is the probability of executing a random exploratory action.
- a_{random} is an action randomly chosen among those available in state s.

In RL, learning is carried out online, through trial-and-error interactions of the agent with the environment. Unfortunately, convergence of any RL algorithm may only be achieved after extensive exploration of the state-action space. In the next section we show one way to speed up the convergence of RL algorithms, by making use of a heuristic function in a manner similar to the use of heuristics in informed search algorithms.

3 Heuristic Accelerated Reinforcement Learning and the HAQL Algorithm

Formally, a Heuristically Accelerated Reinforcement Learning (HARL) algorithm (8) is a way to solve a MDP problem with explicit use of a heuristic function $\mathcal{H} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$ for influencing the choice of actions by the learning agent. $H(s,a)$ defines the heuristic that indicates the importance of performing the action a when visiting state s. The heuristic function is strongly associated with the policy: every heuristic indicates that an action must be taken regardless of others.

The heuristic function is an action policy modifier, which does not interfere with the standard bootstrap-like update mechanism of RL algorithms. A possible strategy for action choice is an $\epsilon - greedy$ mechanism where a heuristic mechanism formalized as a function $H(s,a)$ is considered, thus:

$$\pi(s) = \begin{cases} \arg\max_a \left[\mathsf{F}(s,a) \bowtie \xi H(s,a)^{\beta} \right] & \text{if } q \leq p, \\ a_{random} & \text{otherwise} \end{cases} \tag{3}$$

where:

- $\mathcal{F} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$ is an estimate of a value function that defines the expected cumulative reward. If $\mathsf{F}(s,a) \equiv \hat{Q}(s,a)$ we have an algorithm similar to standard *Q–Learning*.
- $\mathcal{H} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$ is the heuristic function that plays a role in the action choice. $H(s,a)$ defines the importance of executing action s in state s.
- $\bowtie$ is a function that operates on real numbers and produces a value from an ordered set which supports a maximization operation.
- ξ and β are design parameters that control the influence of the heuristic function.
- q is a parameter that defines the exploration/exploitation tradeoff.
- a_{random} is an action randomly chosen among those available in state s.

The first HARL algorithm proposed was the Heuristically Accelerated Q–Learning (HAQL) (8), as an extension of the Q–Learning algorithm. The only difference between them is that in the HAQL makes use of an heuristic function in the action choice rule defined in Equation (3), where $\mathcal{F} = Q$, the $\bowtie$ operator is the sum, and $\beta = 1$:

$$\pi(s) = \begin{cases} \arg\max_a \left[\hat{Q}(s,a) + \xi H(s,a) \right] & \text{if } q \leq p, \\ a_{random} & \text{otherwise,} \end{cases} \tag{4}$$

where all variables are defined as in Equation (3).

As a general rule, the value of $H_t(s,a)$ used in HAQL should be higher than the variation among the $\hat{Q}(s,a)$ values for the same $s \in \mathcal{S}$, in such a way that it can influence the choice of actions, and it should be as low as possible in order to minimize the error. It can be defined as:

$$H(s,a,o) = \begin{cases} \max\limits_i \hat{Q}(s,i) - \hat{Q}(s,a) + \eta \text{ if } a = \pi^H(s), \\ 0 \text{ otherwise.} \end{cases} \tag{5}$$

Table 2. The HAQL algorithm.

```
Initialize $\hat{Q}_t(s,a)$ and $H_t(s,a)$ arbitrarily.
 Repeat (for each episode):
    Initialize $s$.
    Repeat (for each step):
        Update the values of $H_t(s,a)$ as desired.
        Select an action $a$ using equation 4.
        Execute the action $a$, observe $r(s,a)$, $s'$.
        Update the values of $Q(s,a)$ according to equation 1.
        $s \leftarrow s'$.
    Until $s$ is terminal.
Until some stop criteria is reached.
```

where η is a small real value (usually 1) and $\pi^H(s)$ is the action suggested by the heuristic policy.

Convergence of the HAQL algorithm was presented by Bianchi, Ribeiro and Costa (8), together with the definition of an upper bound for the error in the estimation of Q. The complete HAQL algorithm is presented in Table 2. The same authors investigated the use of HARL in multiagent domain, proposing a multiagent HARL algorithm – the Heuristically Accelerated Minimax-Q (9) – and testing it in a simplified simulator for the robot soccer domain.

Despite the fact that RL is a method that has been traditionally applied in the Robotic Soccer domain, only recently HARL methods started being used in this domain. Bianchi, Ribeiro and Costa (9) investigated the use of a multiagent HARL algorithm in a simplified simulator for the robot soccer domain; Celiberto, Ribeiro, Costa and Bianchi (12) studied the use of the HARL algorithms to speed up learning in the RoboCup 2D Simulation domain. Finally, Martins and Bianchi (13) studied the use of several HARL algorithms in a simulated Robot soccer environment that reproduces the conditions of a real physical robot, the FIRA Simurosot competition league.

4 Case-based reasoning

Humans frequently try to solve a new problem by remembering a previous similar situation, reasoning about it, and then reusing knowledge of that situation to solve the new problem. Case-based reasoning (CBR) (1; 2) uses knowledge of previous situations (cases) to solve new problems, by finding a similar past case and reusing it in the new problem situation. In the CBR approach, a case usually describes a problem and its solution, i.e., the state of the world in a defined moment and the sequence of actions to perform to solve that problem.

According to López de Mántaras *et al* (2), solving a problem by CBR involves “obtaining a problem description, measuring the similarity of the current problem to previous problems stored in a case base with their known solutions, retrieving one or more similar cases, and attempting to reuse the solution of the retrieved case(s), possibly after adapting it to account for differences in problem descriptions”. Other steps that are usually found in CBR systems are the evaluation of the proposed solution, the revision of the solution, if required in light of its evaluation, and the retention (learning) of a new case, if the system has learned to solve a new problem.

The case definition used in this work is the one proposed by Ros (14; 15; 16; 17; 18), which is composed of three parts: the problem description (P), the solution description (A) and the case scope (K), and is formally described as a 3-tuple:

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

The problem description P corresponds to the situation in which the case can be used. For example, for a simple robotic soccer problem, the description of a case can include the robot position, the ball's position and the positions of the other robots in the game. For a game with n robots, P can be:

$$P = \{x_B, y_B, x_{R_0}, y_{R_0}, \dots, x_{R_n}, y_{R_n}\}.$$

The solution description is composed by the sequence of actions that each robot must perform to solve the problem, and can be defined as:

$$A = \{R_0 : [a_{0_1}, a_{0_2}, ..., a_{0_{p_0}}], \dots, R_n : [a_{n_1}, a_{n_2}, ..., a_{n_{p_n}}]\},$$

where n is the number of robots in the team, a_{0_i} is an individual or joint action that robot R_i must perform and p_i corresponds the number of actions the robot R_i performs.

The case scope defines the applicability boundaries of the cases, to be used in the retrieval step. For example, Ros (18) define it as "the regions of the field within which the ball and the opponents should be positioned in order to retrieve that case". In the case of a simple robot soccer problem, K can be represented as ellipsoids centered on the ball's and opponents' positions indicated in the problem description. It can be defined as:

$$K = \{(\tau_B^x, \tau_B^y), (\tau_{R_0}^x, \tau_{R_0}^y) \dots, (\tau_{R_n}^x, \tau_{R_n}^y)\},$$

where τ_B^x, τ_B^y corresponds to the x and y radius of the ellipsoid region around the ball and $(\tau_{R_0}^x, \tau_{R_0}^y) \dots, (\tau_{R_n}^x, \tau_{R_n}^y)$ the radius of the regions around the n robots in the game (teammates and opponents).

Case retrieval is in general driven by a similarity measure between the new problem and the solved problems in the case base. In this work we use the case retrieval method proposed by Ros (18), where the similarity along three important aspects are evaluated:

- 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.

The similarity function indicates how similar a problem and a case are. In most cases, the function is defined by the distance between the ball and the robots in the problem and in the case.

$$Sim(p, c) = dist(B^c, B^p) + \sum_{i=0}^{n} dist({R_i}^c, {R_i}^p),$$

where B^c is the position of the ball in the case and B^p its position in the problem, ${R_i}^c$ the position of the Robot i in the case and ${R_i}^p$ its position in the problem, and $dist(a, b)$ is the gaussian distance between object a and b. This distance is computed as follows:

$$dist(a, b) = exp\left(-\left[\left(\frac{a_x - b_x}{\tau^x}\right)^2 + \left(\frac{a_y - b_y}{\tau^y}\right)^2\right]\right),$$

where τ^x, τ^y are the radius of the scope around the object (ball and robots positions). The Gaussian distance is used because the larger the distance between two points, the lower the similarity between them. Also, the τ^x, τ^y parameters are used as a threshold that defines a maximum distance allowed for two points to have some degree of similarity: if the distance is greater than a limit, $Sim(a, b) = 0$.

The cost of adapting the problem to the case is computed as a function of the distances between the positions of the team robots in the problem and the positions specified in the case. The adaptation cost is defined as:

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

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.

The applicability of the solution of the case depends on the position of the opponents, and combine two functions: the free path function, which considers if the trajectory of the ball indicated in the case is free of opponents, in order for the evaluated case to be applicable and the opponent similarity, which computes if the opponents 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. This functions and the complete case retrieval algorithm are described in detail in Ros (18).

In recent years, CBR has been used by several researchers in the Robotic Soccer domain. To mention a few, Lin, Liu and Chen (19) presented a hybrid architecture for soccer players where the deliberative layer corresponds to a CBR system, Ahmadi *et al* (20) presented a two-layered CBR system for prediction for the coach, and Karol *et al* (21) presented high level planning strategies including a CBR system. Finally, the works of Ros (18) presents the most ample use of CBR techniques in the Robotic Soccer domain, proposing the use of CBR techniques to handle retrieval, reuse and acquisition of a case base for the action selection problem of a team for the Four-Legged robots.

5 Combining Case Based Reasoning and Reinforcement Learning

Bianchi, Ribeiro and Costa (8) states that there should be many methods that can be used to define a heuristic function for a HARL algorithm. For example, the same work makes use of information from the learning process itself to infer a heuristic in execution time, proposing the "Heuristic from Exploration" method, a technique that derives a crude estimate of the transition probabilities, by annotating the results of every action

Table 3. The CB-HAQL algorithm.

```
Initialize $\hat{Q}_t(s,a)$ and $H_t(s,a)$ arbitrarily.
Repeat (for each episode):
  Initialize $s$.
  Repeat (for each step):
     Compute similarity and cost.
     If there is a case that can be reused:
        Retrieve and Adapt if necessary.
        Compute $H_t(s,a)$ using Equation 5 with the
              actions suggested by the case selected.
     Select an action $a$ using equation 4.
     Execute the action $a$, observe $r(s,a)$, $s'$.
     Update the values of $Q(s,a)$ according to equation 1.
     $s \leftarrow s'$.
  Until $s$ is terminal.
Until some stop criteria is reached.
```

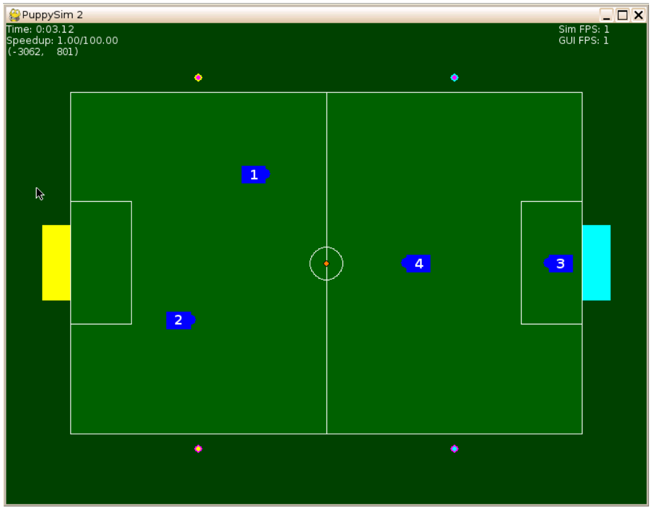
performed by the agent, and then, from this model, it propagates – from a final state – the correct policies which lead to that state.

In order to give HARL algorithms the capability of reusing previous knowledge from a domain, we propose a new algorithm, the Case Based HAQL, that extends the HAQL algorithm, being capable of retrieving a case stored in a base, adapting it to the current situation, and building a heuristic function that corresponds to the case.

As the problem description P corresponds to one defined state of the set of states $\mathcal{S}$ in an MDP, an algorithm that uses the RL loop can be implemented. Inside this loop, before the action selection, we added steps to compute the similarity of the cases in the base with the current state and the cost of adaptation of these cases. A case is retrieved if the similarity is above a certain threshold, and adaptation cost is low. After a case is retrieved, an heuristic is computed using Equation 5 with the actions suggested by the case selected, and this heuristic is used for a certain amount of time, equals to the number of actions of the retrieved case. After that time, a new case can be retrieved. The complete CB-HAQL algorithm is presented in Table 3.

Some authors have been studying the use of RL together with CBR and the relation between them. Sharma *et al* (22) makes use of CBR as a function approximator for RL, and RL as revision algorithm for CBR in a hybrid architecture system; Juell and Paulson (23) exploit the use of RL to learn similarity metrics in response to feedback from the environment; Auslander *et al* (24) uses CBR to adapt quickly an RL agent to changing conditions of the environment by the use of previously stored policies and Li, Zonghai and Feng (25) proposes an algorithm that makes use of knowledge acquired by reinforcement learning to construct and extend a case base.

Our approach differs from all previous works combining CBR and RL because of the heuristic use of the retrieved case: as the case is used only as a heuristic, if the case base contains a case that can be used in one situation, there will be a speed up in the convergence time. But if the case base does not contain any useful case – or even if it



```
Time: 0:03.12
Speedup: 1.00/100.00
(-3062,   801)
Sim FPS: 1
GUI FPS: 1
```

Fig. 1. The PuppySim2 users' interface showing the robots at their initial positions.

contains cases that implement wrong solutions to the problem, the agent will learn the optimal solution anyway, by using the RL component of the algorithm.

6 Experiments in the Robotic Soccer Domain

Empirical evaluations of the CB-HAQL approach were carried out in an extended version of the PuppySim 2 simulator, created by the CMDash team (26) and extended by Ros (18). This simulator represents the basic aspects of the RoboCup Standard Platform League, Four-Legged Soccer Competition (27), and is used to test the robot's behavioral response under ideal environmental conditions: the robots' perception is noiseless, but the outcome of the actions the robots perform have a certain degree of randomness. PuppySim 2 was implemented using Python.

Using this simulator experiments were performed using two attackers against a defender and a goalie. The attackers are two robots controlled by one of the algorithms to be evaluated: the Q–Learning, described in section 2, the HAQL, described in section 3 or the CB-HAQL, proposed in section 5 and we have also compared them to the results of the CBR system alone obtained by Ros (18). The opponents perform the same reactive behavior when playing against any of the evaluated approaches. The defender and the goalie have 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. Each trial begins with the attackers being positioned in the field in a random position, and the defender, the goalie and the ball in a fixed location (ball in the center, and defender and goalie in the center of their home region). Figure 1 shows the PuppySim 2

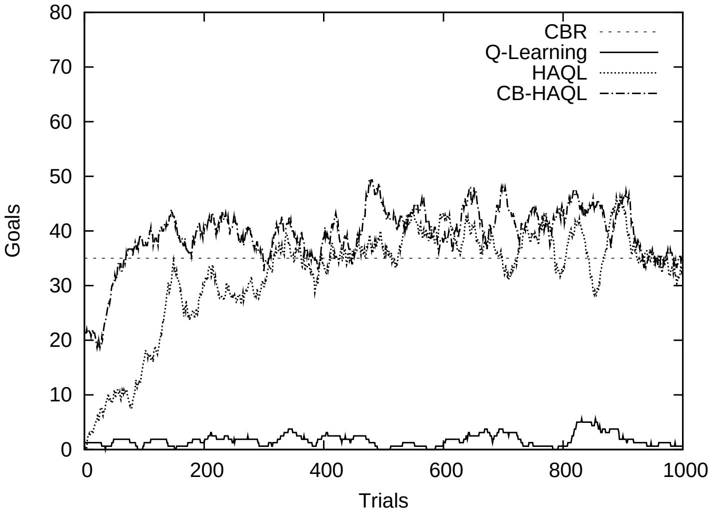


Fig. 2. Percentage of goals scored in each trial using the CBR (constant line at 35%), Q–learning, the HAQL and the CB-HAQL algorithms.

users' interface with one starting configuration. A trial ends either when the attackers score a goal, the ball goes out of the field or the goalie touches it.

The heuristic used in the HAQL algorithm was defined using a simple rule: if holding the ball, go to the opponents goal, not taking into account the opponents positions, leaving the task of how to divert the opponent to the learning process. The heuristic used in the CB-HAQL is computed during the games, as described in section 5. 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. The reward the agents receive are the same for all algorithms: the agent receives -100 every time the ball go out of the field or the goalie touches it and $+100$ a robot scores a goal.

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.

Twenty five training sessions were run for the three algorithms, with each session consisting of 1000 trials. Figures 2 to 5 shows the learning curves for all algorithms

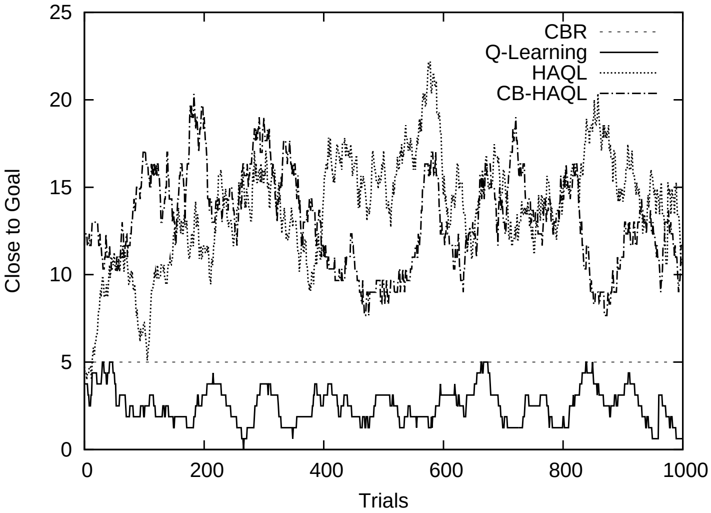


Fig. 3. Percentage of balls that passed close to the goals in each trial, for the CBR (constant line at 5%), Q–learning, the HAQL and the CB-HAQL algorithms.

(the CBR alone and the three learning algorithms) and presents the percent of goals scored (Fig. 2) by the learning team, balls that passed close to the goals (Fig. 3), balls blocked by the defenders (Fig. 4) and balls that went out (Fig. 5) in each trial. It is possible to verify in Fig. 2 that at the beginning of the learning phase HAQL has worse performance than the CB-HAQL, and as the trials proceed, the performance of both algorithms become similar, as expected, since all the algorithms converge to equilibrium. The Q–learning is clearly the one with the worst performance, since it takes much more trials for it to start to learn even basic policies, as not to kick the ball out of the field. In this figure it can also be observed the performance of two agents using only the case base. Student's t–test was used to verify the hypothesis that the use of heuristics speeds up the learning process. Using the data from Fig. 2, the result is that the CB-HAQL is better (makes more goals) than HAQL and Q–Learning until the 300^{th} trial, with a level of confidence greater than 5%. After this trial the results of the CB-HAQL and HAQL are comparable.

Finally, table 4 summarizes the ball classification outcome obtained (results in percentage) using the CBR approach and the three learning algorithms. The results for the CBR approach are the average of 500 trials, and the results for the Q–learning, HAQL and CB-HAQL are the average of 100 trials, using the Q-table that the three algorithms had at the end of the 1000^{th} trial. As we can see the percentage of balls to goal with the CB-HAQL approach is higher compared to either the HAQL or the Q-Learning algorithms. Moreover, the percentage of balls out are lower when using CBR, indicating that the defender had less opportunities to take the ball and kick it out of the field, and that the agent performed less random exploration.

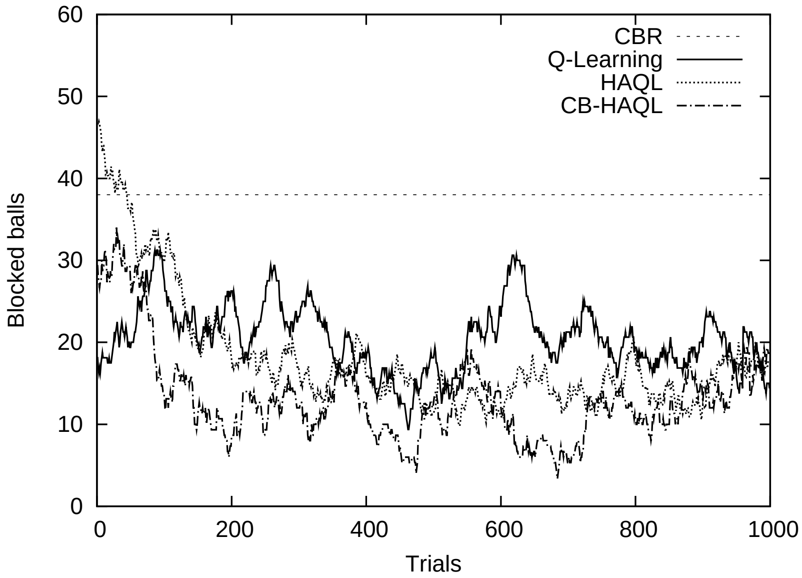


Fig. 4. Percentage of balls blocked by the defenders for the CBR (constant line at 38%), Q–learning, the HAQL and the CB-HAQL algorithms.

The parameters used in the experiments were the same for all the algorithms. The learning rate is $\alpha = 0,9$, the exploration/ exploitation rate was defined as being equal to 0.2 and the discount factor $\gamma = 0.9$ (these parameters are similar to those used by (28). The value of η was set to 1. Values in the Q table were randomly initiated, with $0 \leq Q(s_t, a_t, o_t) \leq 1$. The experiments were programmed in Python and executed in a MacBook Pro, with 4GB of RAM in a Mac OS X platform.

7 Conclusion

This work presented a new algorithm, called Case Based Heuristically Accelerated Q–Learning (CB-HAQL), which allows the use of a cases base to define heuristics

Table 4. Ball outcome classification (results in percentage).

Approach	Goal	Close	To-Goal Goal + Close	Blocked	Out
CBR	35	5	40	38	22
Q–Learning	2	2	4	22	74
HAQL	16	4	20	20	60
CB-HAQL	40	7	47	36	17

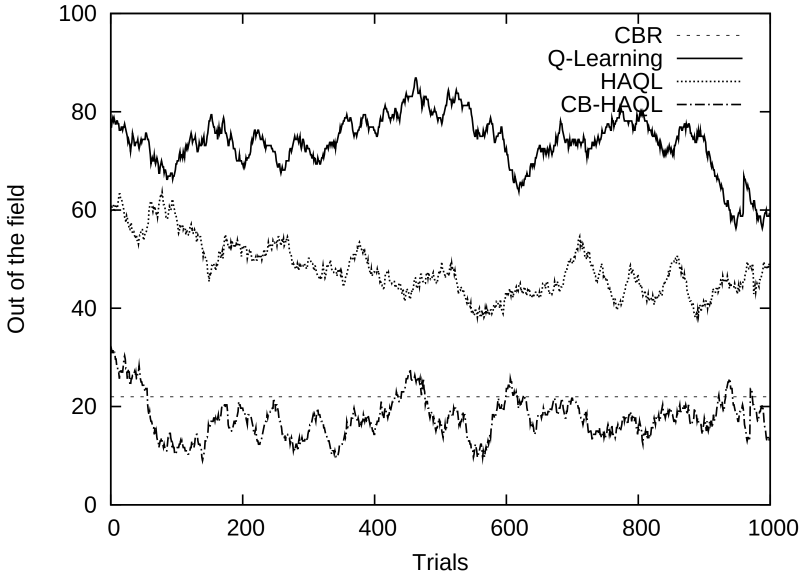


Fig. 5. Percentage of balls that went out of the field in each trial, for the CBR (constant line at 22%), Q–learning, the HAQL and the CB-HAQL algorithms.

to speed up the well-known Reinforcement Learning algorithm Q–Learning. This approach builds upon an emerging technique, the Heuristic Accelerated Reinforcement Learning (HARL), in which RL methods are accelerated by making use of heuristic information.

The experimental results obtained showed that CB-HAQL attained better results than HAQL and Q–Learning for the domain of robotic soccer games. For example, the Q–Learning, after 1000 learning trials, still could not produce policies that scored goals on the opponent, while the HAQL was able to score some goals but significantly less than the CBR alone and the CB-HAQL. Another interesting finding is that the goals scored by the CB-HAQL after 1000 trials was even slightly higher than the number of goals scored by the CBR approach alone, indicating that the learning component of the CB-HAQL algorithm was able to improve the initial case base.

Another important finding of this work is that the CBR approach generated better results than the Q–learning algorithm, for the same experimental setup. Experiments executed until the 10.000 th trial showed that the Q-Learning still had not converged, indicating the slow rate of learning of this algorithm, in this domain.

Finally, since Heuristic functions allow RL algorithms to solve problems where the convergence time is critical, as in many real time applications, in future works we plan to incorporate CBR in other well known RL algorithms, like SARSA, Q(λ), Minimax-Q, Minimax-Q(λ), and Nash-Q, and expanding this framework to deal with General Sum Markov Games.

Bibliography

[1] Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Commun. **7**(1) (1994) 39–59

[2] de Mántaras, R.L., 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.: Retrieval, reuse, revision and retention in case-based reasoning. Knowl. Eng. Rev. **20**(3) (2005) 215–240

[3] Hennessy, D., Hinkle, D.: Applying case-based reasoning to autoclave loading. IEEE Expert: Intelligent Systems and Their Applications **7**(5) (1992) 21–26

[4] Althoff, K.D., Bergmann, R., Wess, S., Manago, M., Auriol, E., Larichev, O.I., Bolotov, A., Zhuravlev, Y.I., Gurov, S.I.: Case-based reasoning for medical decision support tasks: The inreca approach. Artificial Intelligence in Medicine (January 1998) 25–41

[5] de Mántaras, R.L., Cunningham, P., Perner, P.: Emergent case-based reasoning applications. Knowl. Eng. Rev. **20**(3) (2005) 325–328

[6] Szepesvári, C., Littman, M.L.: Generalized markov decision processes: Dynamic-programming and reinforcement-learning algorithms. Technical report, Brown University (1996) CS-96-11.

[7] Littman, M.L., Szepesvári, C.: A generalized reinforcement learning model: convergence and applications. In: Proceedings of the 13th International Conference on Machine Learning (ICML'96). (1996) 310–318

[8] Bianchi, R.A.C., Ribeiro, C.H.C., Costa, A.H.R.: Accelerating autonomous learning by using heuristic selection of actions. Journal of Heuristics **14**(2) (2008) 135–168

[9] Bianchi, R.A.C., Ribeiro, C.H.C., Costa, A.H.R.: Heuristic selection of actions in multiagent reinforcement learning. In Veloso, M.M., ed.: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007. (2007) 690–695

[10] RoboCup Technical Committee: Standard platform league homepage (2009) http://www.tzi.de/spl.

[11] Watkins, C.J.C.H.: Learning from Delayed Rewards. PhD thesis, University of Cambridge (1989)

[12] Celiberto, L.A., Ribeiro, C.H.C., Costa, A.H.R., Bianchi, R.A.C.: Heuristic reinforcement learning applied to robocup simulation agents. In Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F., eds.: RoboCup. Volume 5001 of Lecture Notes in Computer Science., Springer (2007) 220–227

[13] Martins, M.F., Bianchi, R.A.C.: Comparison of reinforcement learning algorithm performance in the robotic soccer domain (in portuguese). In: Brazilian Symposium on Intelligent Automation – SBAI'2007, Florianopolis, Brazil, SBA (2007) Anais.

[14] Ros, R., Veloso, M., de Mántaras, R.L., Sierra, C., Arcos, J.L.: Retrieving and reusing game plays for robot soccer. Lecture Notes in Artificial Intelligence **4106** (2006) 47–61

[15] Ros, R., Veloso, M., de Mántaras, R.L., Sierra, C., Arcos, J.L.: Beyond individualism: Modeling team playing behavior in robot soccer through case-based reasoning. In: 22nd AAAI Conference on Artificial Intelligence, Vancouver, Canada, AAAI Press, AAAI Press (2007) 1671–1674

[16] Ros, R., de Mántaras, R.L., Arcos, J.L., Veloso, M.: Team playing behavior in robot soccer: A case-based approach. Lecture Notes in Artificial Intelligence **4626** (2007) 46–60

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

[18] Ros, R.: Action Selection in Cooperative Robot Soccer using Case-Based Reasoning. PhD thesis, Universitat Autònoma de Barcelona, Barcelona (2008)

[19] Lin, Y., Liu, A., Chen, K.: A hybrid architecture of case-based reasoning and fuzzy behavioral control applied to robot soccer. In: Workshop on Artificial Intelligence, International Computer Symposium (ICS2002), Hualien, Taiwan, National Dong Hwa University, National Dong Hwa University (2002)

[20] Ahmadi, M., Lamjiri, A.K., Nevisi, M.M., Habibi, J., Badie, K.: Using a two-layered case-based reasoning for prediction in soccer coach. In Arabnia, H.R., Kozerenko, E.B., eds.: MLMTA, CSREA Press (2003) 181–185

[21] Karol, A., Nebel, B., Stanton, C., Williams, M.A.: 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. Volume 3020 of Lecture Notes in Computer Science., Springer (2003) 739–747

[22] Sharma, M., Holmes, M., Santamaría, J.C., Irani, A., Jr., C.L.I., Ram, A.: Transfer learning in real-time strategy games using hybrid cbr/rl. In Veloso, M.M., ed.: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007. (2007) 1041–1046

[23] Juell, P., Paulson, P.: Using reinforcement learning for similarity assessment in case-based systems. IEEE Intelligent Systems **18**(4) (2003) 60–67

[24] Auslander, B., Lee-Urban, S., Hogg, C., Muñoz-Avila, H.: Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning. In Althoff, K.D., Bergmann, R., Minor, M., Hanft, A., eds.: ECCBR. Volume 5239 of Lecture Notes in Computer Science., Springer (2008) 59–73

[25] Li, Y., Zonghai, C., Feng, C.: A case-based reinforcement learning for probe robot path planning. In: 4th World Congress on Intelligent Control and Automation, Shanghai, China. (2002) 1161– 1165

[26] Veloso, M., Rybski, P.E., Chernova, S., McMillen, C., Fasola, J., von Hundelshausen, F., Vail, D., Trevor, A., Hauert, S., Ros, R.: Cmdash'05: Team report. Technical report, School of Computer Science, Carnegie Mellon University (2005)

[27] RoboCup Technical Committee: RoboCup Four-Legged League Rule Book. (2008)

[28] Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Proceedings of the 11th International Conference on Machine Learning (ICML'94). (1994) 157–163