The Use of Cases as Heuristics to speed up Multiagent Reinforcement Learning

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

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

Abstract. This work presents a new approach that allows the use of cases in a case base as heuristics to speed up Multiagent Reinforcement Learning algorithms, combining Case Based Reasoning (CBR) and Multiagent Reinforcement Learning (MRL) techniques. This approach, called Case Based Heuristically Accelerated Multiagent Reinforcement Learning (CB-HAMRL), builds upon an emerging technique, Heuristic Accelerated Reinforcement Learning (HARL), in which RL methods are accelerated by making use of heuristic information. CB-HAMRL is a subset of MRL that makes use of a heuristic function $\mathcal{H}$ derived from a case base, in a Case Based Reasoning manner. An algorithm that incorporates CBR techniques into the Heuristically Accelerated Minimax–Q is also proposed and a set of empirical evaluations were conducted in a simulator for the robot soccer domain, comparing the three solutions for this problem: MRL, HAMRL and CB-HAMRL. Experimental results show that using CB-HAMRL, the agents learn faster than using RL or HAMRL methods.

1 Introduction

Heuristic Accelerated Reinforcement Learning (HARL) (1; 2) is an emerging technique in which RL methods are sped up by making use of a conveniently chosen heuristic function, which is used for selecting appropriate actions to perform in order to guide exploration during the learning process. HARL 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 Multiagent Reinforcement Learning (MRL) techniques, with the goal of speeding up MRL algorithms by using previous domain knowledge, stored as a case base.

To do so, we propose a new algorithm, the Case Based Heuristically Accelerated Minimax–Q (CB-HAMMQ), which incorporates Case Based Reasoning techniques into an existing HAMRL algorithm, the Heuristically Accelerated Minimax–Q (HAMMQ).

Soccer competitions, such as RoboCup, which has been proven to be an important challenge domain for research, and where RL techniques have been widely used. The application domain of this paper is that of a a simulator for the robot soccer domain that extends the one proposed by Littman (3), called "Expanded Littman's Soccer". Nevertheless, the technique proposed in this work is domain independent.

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

2 Heuristic Accelerated Multiagent Reinforcement Learning

Systems where multiple agents compete among themselves to accomplish their tasks can be modeled as a discrete time, finite state, finite action Markov Game (MG) – also known as Stochastic Game (SG). The goal of an agent in a MRL problem is to learn an optimal policy $\pi : \mathcal{S} \times \mathcal{A}_1 \times \ldots \times \mathcal{A}_k$ that maps the current state s into a desirable action(s) a to be performed in s, from any starting state. In MRL, 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).

This paper considers a well-studied specialization of MGs in which there are only two players, called agent and opponent, having opposite goals. Such specialization, called a zero-sum Markov Game (ZSMG), allows the definition of only one reward function that the learning agent tries to maximize while the opponent tries to minimize. A two player ZSMG (3) is defined by the quintuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R} \rangle$, where:

- $\mathcal{S}$: a finite set of environment states.
- $\mathcal{A}$: a finite set of actions that the agent can perform.
- $\mathcal{O}$: a finite set of actions that the opponent can perform.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \rightarrow \Pi(\mathcal{S})$: the state transition function, where $\Pi(\mathcal{S})$ is a probability distribution over the set of states $\mathcal{S}$. $T(s, a, o, s')$ defines a probability of transition from state s to state s' (at a time $t+1$) when the learning agent executes action a and the opponent performs action o.

- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \rightarrow \Re$: the reward function that specifies the reward received by the agent when it executes action a and its opponent performs action o, in state s.

To solve a ZSMG, Littman (3) proposed the use of a strategy similar to Minimax for choosing an action in the Q-Learning algorithm, the Minimax–Q algorithm, which works in the same way as Q-Learning does. The action-value function of an action a in a state s when the opponent takes an action o is can be computed iteratively by:

$$\begin{aligned}\hat{Q}_{t+1}(s,a,o) \leftarrow & \hat{Q}_t(s,a,o) + \\ & \alpha \left[r(s,a,o) + \gamma V_t(s') - \hat{Q}_t(s,a,o)\right], \end{aligned} \tag{1}$$

where α is the learning rate, γ is the discount factor and the value $V_t(s)$ of a state can be computed using the equation:

$$V(s) = \max_{\pi \in \Pi(\mathcal{A})} \min_{o \in \mathcal{O}} \sum_{a \in \mathcal{A}} Q(s,a,o)\pi_a, \tag{2}$$

where the agent's policy π is a probability distribution over actions, and π_a is the probability of taking the action a against the opponent's action o. In an Alternating Markov Game (AMG), where two players take their actions in consecutive turns, the policy becomes deterministic and equation 2 can be simplified:

$$V(s) = \max_{a \in \mathcal{A}} \min_{o \in \mathcal{O}} Q(s,a,o). \tag{3}$$

Formally, a Heuristically Accelerated Multiagent Reinforcement Learning (HAMRL) algorithm is a way to solve a MG problem with explicit use of a heuristic function $\mathcal{H} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \rightarrow \Re$ to influence the choice of actions during the learning process. $H(s,a,o)$ defines a heuristic that indicates the desirability of performing action a when the agent is in state s and the opponent executes action o.

The first HAMRL algorithm proposed was the Heuristically Accelerated Minimax Q (HAMMQ) (2), as an extension of the Minimax–Q algorithm. The only difference between them is that in the HAMMQ the heuristic function is used in the action choice rule, which defines which action a_t must be executed when the agent is in state s_t. The action choice rule used in the HAMMQ is a modification of the standard $\epsilon - Greedy$ rule used in Minimax–Q, to include the heuristic function:

$$\pi(s) = \begin{cases} \arg\max\limits_{a} \min\limits_{o} \left[\hat{Q}(s,a,o) + \xi H_t(s,a,o)\right] & \text{if } q \le p, \\ a_{random} & \text{otherwise,} \end{cases} \tag{4}$$

where $\mathcal{H} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \rightarrow \Re$ is the heuristic function, q is a random value uniformly distributed over $[0,1]$ and $0 \le p \le 1$ is a parameter that defines the exploration/exploitation tradeoff. The subscript t indicates that it can be non-stationary (it can be computed only once, or be continually recomputed) and $0 \le \xi \le 1$ is a real variable used to weight the influence of the heuristic.

Table 1. The HAMMQ algorithm.

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

As a general rule, the value of $H_t(s,a,o)$ used in HAMMQ should be higher than the variation among the $\hat{Q}(s,a,o)$ values for the same $s \in \mathcal{S}$, $o \in \mathcal{O}$, 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:

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

where η is a small real value (usually 1) and $\pi^H(s)$ is the action suggested by the heuristic policy. Convergence of this algorithm is presented by Bianchi, Ribeiro and Costa (2), together with the definition of an upper bound for the error. The complete HAMMQ algorithm is presented in Table 1.

Despite the fact that RL is a method that has been traditionally applied in the Robotic Soccer domain, only recently have HARL methods been used in this domain. Bianchi, Ribeiro and Costa (2) investigated the use of a multiagent HARL algorithm in a simplified simulator for the robot soccer domain and Celiberto, Ribeiro, Costa and Bianchi (4) studied the use of the HARL algorithms to speed up learning in the RoboCup 2D Simulation domain. The heuristic used in the both papers were very simple ones: in the first paper the heuristic was 'if the agent is with the ball, go to the opponent's goal', and in the second paper it was simply 'go to the ball'.

3 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) (5; 6) 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* (6), 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 (7; 8; 9; 10; 11; 12), 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:

$$case = (P, A, K). \tag{6}$$

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:

$$P = \{x_B, y_B, x_{R_0}, y_{R_0}, \ldots, x_{R_n}, y_{R_n}\}. \tag{7}$$

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

$$A = \{R_0 : [a_{0_1}, a_{0_2}, ..., a_{0_{p0}}], \ldots, R_n : [a_{n_1}, a_{n_2}, ..., a_{n_{pn}}]\}, \tag{8}$$

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 to 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 (12) 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 circles or ellipsoids centered on the ball's and opponents' positions indicated in the problem description. It can be defined as:

$$K = \{\tau_B, \tau_{R_0}, \ldots, \tau_{R_n}\}, \tag{9}$$

where τ_B is the radius of the region around the ball and $\tau_{R_0} \ldots \tau_{R_n}$ the radius of the regions around the n robots in the game (teammates and opponents). The case retrieval process consists in obtaining from the base the most similar case, the retrieved case. Therefore, it is necessary to compute the similarity between the current problem and the cases in the base. 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.

$$Sim(p, c) = dist(B^c, B^p) + \sum_{i=0}^{n} dist({R_i}^c, {R_i}^p), \tag{10}$$

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:

$$dist(a, b) = e^{-\left((a_x - b_x)^2 + (a_y - b_y)^2\right)/2\tau^2}, \tag{11}$$

where τ is the radius of the scope around the object. In this work, τ is the same for the ball and robots positions. The Gaussian distance is used because the larger the distance between two points, the lower the similarity between them. Finally, τ is used as a threshold that defines a maximum distance allowed for two points to have some degree of similarity: if $dist(a, b) > \tau$, $Sim(a, b) = 0$.

Before a case can be reused, it might be necessary to adapt it to the present situation. Adaptation of a case means that the retrieved solution is modified, by translation, rotation or the addition of steps to the sequence of actions in the solution before it can be used. In this work, we assume that rotation and translation costs are small when compared to the cost of the additional steps, because the first two are trivial computations, while the performance of additional steps by the robots are actions that must be executed (in the simulator or in the real world), taking more time. Therefore, we define the cost as the number of steps added to the adapted solution. In this work, the case that will be reused is the one that maximizes the similarity while minimizing the adaptation cost.

In recent years, CBR has been used by several researchers in the Robotic Soccer domain. By far, the Robocup 2D Simulation League is the domain where more work has been done. To mention a few, Lin, Liu and Chen (13) presented a hybrid architecture for soccer players where the deliberative layer corresponds to a CBR system, Ahmadi *et al* (14) presented a two-layered CBR system for prediction for the coach and Berger and Lämmel (15) proposed the use of a CBR system to decide whether a pass should be performed.

CBR has been also used in other Robocup Leagues. In the Small Size League, Srinivasan *et al* (16) proposed a CBR planning for both offense and defense team behavior, for a team of two soccer playing robots; in the work by Marling *et al* (17), CBR is used to help planning individual moves and team strategies. In the Four-Legged League, Karol *et al* (18) presented high level planning strategies including a CBR system. Finally, the works of Ros (12) 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 League (9; 11; 12). A more extensive review of the use of CBR in Robotic Soccer can be found in works by (19) and by Ros (7; 12).

4 Combining Case Based Reasoning and Multiagent Reinforcement Learning

Bianchi, Ribeiro and Costa (1) states that there should be many methods that can be used to define a heuristic function for a HARL algorithm. For example,

Table 2. The CB-HAMMQ algorithm.

```
Initialize $\hat{Q}_t(s, a, o)$ and $H_t(s, a, o)$ 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 $H_t(s, a, o)$ using Equation 5 with the
            actions suggested by the case selected.
    Select an action $a$ using equation 4.
    Execute the action $a$, observe $r(s, a, o)$, $s'$.
    Update the values of $Q(s, a, o)$ according to equation 1.
    $s \leftarrow s'$.
  Until $s$ is terminal.
Until some stop criteria is reached.
```

the same work makes use of information from the learning process itself to infer a heuristic in execution time, proposing a technique that derives a crude estimate of the transition probabilities, and then it propagates – from a final state – the correct policies which lead to that state. Bianchi, Ribeiro and Costa (2) employed prior domain knowledge to establish a very simple ad-hoc heuristic for speeding up learning in a Multiagent Reinforcement Learning domain.

In order to give HAMRL algorithms the capability of reusing previous knowledge from a domain, we propose a new algorithm, the Case Based HAMMQ, that extends the HAMMQ 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. The complete CB-HAMMQ algorithm is presented in Table 2.

Sharma *et al* (20) 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 (21) exploit the use of RL to learn similarity metrics in response to feedback from the environment; Auslander *et al* (22) 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 (23) 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. Bianchi, Ribeiro and Costa (2) proved that if the heuristic used is an admissible one, there will be a speed up in convergence time, if not, the use of the heuristic will not impede the RL method to converge to the optimal policy. As we use the case base as an heuristic, if the case base corresponds to an admissible heuristic 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 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 (2). Finally, this is the first work that uses CBR in a MRL algorithm.

5 Experiments in the Robotic Soccer Domain

A set of empirical evaluations of the CB-HAMMQ approach were carried out in a proposed simulator for the robot soccer domain that extends the one proposed by Littman (3). In this domain, called "Expanded Littman's Soccer", two teams, A and B, of three players each compete in a 10 x 15 grid presented in figure 1. Each team is composed by the goalie (g), the defender (d) and the attacker (a). Each cell can be occupied by only one player. The actions that are allowed are: keep the agent still, move – north, south, east and west – or pass the ball to another agent. The action "pass the ball" from agent a_i to a_j is successful if there is no opponent in between them. If there is an opponent, it will catch the ball and the action will fail. Actions are taken in turns: all actions from one team's agents are executed at the same instant, and then the opponents' actions are executed. The ball is always with one of the players. When a player executes an action that would finish in a cell occupied by the opponent, it loses the ball and stays in the same cell. If an action taken by one agent leads it out the board, the agent stands still. When a player with the ball gets into the opponent's goal, the trial ends and its team scores one point. The starting positions of all players are random, and the ball is given to one of the agents in a random fashion at the beginning of a trial.

To solve this problem, three algorithms were used: the Minimax–Q, described in section 2, the HAMMQ, described in section 2 and the CB-HAMMQ, proposed in section 4. Although this domain is still a simple one, it is more complex than the original one proposed by Littman: due to the size of the state space, it is not possible to use a lookup table containing all the states of the problem. In this work a variable resolution table similar to the one proposed by Munos and Moore (24) is used.

The heuristic used in the HAMMQ algorithm was defined using a simple rule: if holding the ball, go to the opponents' goal, not taking into account the teammates' and opponents' positions, leaving tasks such as learning to pass the ball or to divert the opponent to the learning process.

The heuristic value used in the CB-HAMMQ is computed during the games, as described in section 4. The case base used contains a set of basic cases that can be used without adaptation costs. The case base used in this experiment is

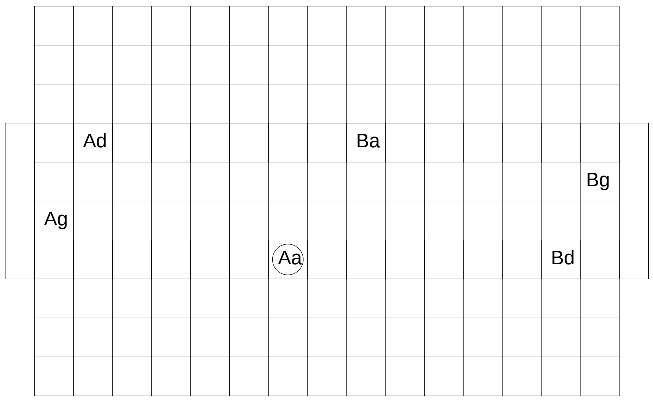

Fig. 1. The "Expanded Littman's Soccer" environment proposed.

composed of 5 basic cases, which cover the most significant situations that are observed during a game in the expanded Littman's Soccer environment. These cases can be described as:

1. If the agent is with the ball and there is no opponent blocking it, then move to the goal.
2. If the agent is with the ball and there is an opponent blocking it, then move up.
3. If the agent is with the ball and there is an opponent blocking it, then move down.
4. If the agent is with the ball and a teammate is closer to the goal, then pass the ball to the other agent.
5. If the ball is with an opponent and the agent is close to the opponent, then stay in front of the opponent.

Is important to notice that this case base does not correspond to the optimal solution of the problem.

The reward the agents receive are the same for all algorithms: the agent that is holding the ball receives +100 every time it reaches the goal. This is a very simple reward scheme, but we decided to use it in this work to avoid the creation of a mismatch between the reward function used in training and the performance measure examined, which is the number of goals scored. Other reward schemes could be used, for example, one that gives rewards to intercepting the ball, losing the ball or correctly passing the ball, such as the one used by Kalyanakrishnan, Liu and Stone (25).

Thirty training sessions were run for the three algorithms, with each session consisting of 20,000 games of 10 trials. Figure 2 shows the learning curves for all algorithms when the learning team plays against an opponent moving randomly, and presents the average goal balance scored by the learning team in each match.

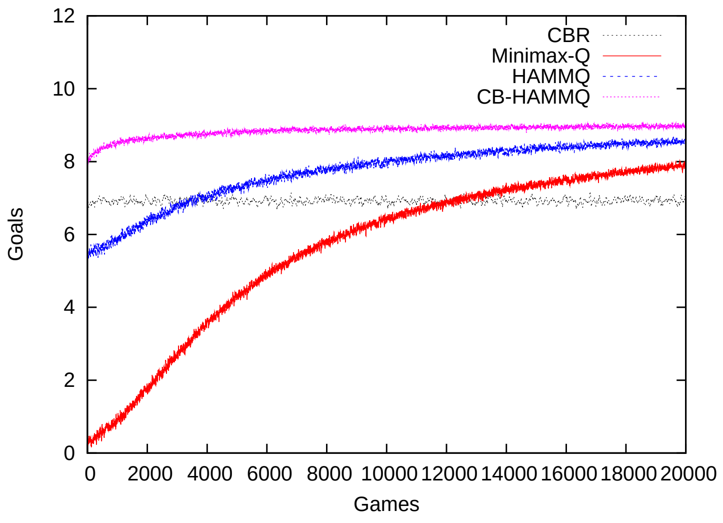

Fig. 2. Goals balance for the CBR, Minimax–Q, the HAMMQ and the CB-HAMMQ algorithms against a random opponent for the Expanded Littman's Robotic Soccer.

It is possible to verify that at the beginning of the learning phase Minimax–Q has worse performance than HAMMQ, and that this has a worse performance than CB-HAMMQ. As the matches proceed, the performance of the three algorithms become similar, as expected. As it can be seen in this figure, the Minimax–Q is still learning after 20,000 games: as it is slower than the other two algorithms, it will only reach the optimal solution after 50,000 games. In this figure it can also be observed the performance of a team of agents using only the case base: a line with values close to 7. As the case base does not contain the optimal solution to the problem, the agents have a performance that is worst than the one presented by the other teams at the end of the learning process.

Figure 3 presents the learning curves (the difference of goals made at the end of a game) for the three algorithms when learning while playing against a learning opponent using Minimax–Q. It can be seen that CB-HAMMQ is better than HAMMQ and Minimax–Q at the beginning of the learning process. Student's t–test was used to verify the hypothesis that the use of heuristics speeds up the learning process. The result is that the CB-HAMMQ is better than HAMMQ until the 7,000th game when playing against a random opponent, and until the 500th game when playing against the Minimax–Q, with a level of confidence greater than 5%. The same test can be made comparing the CB-HAMMQ and the Minimax–Q: in this case, the first outperform the latter until the 20,000th game, while both are playing against a random opponent, and until the 1,000th game when the CB-HAMMQ is playing against the Minimax–Q. After these

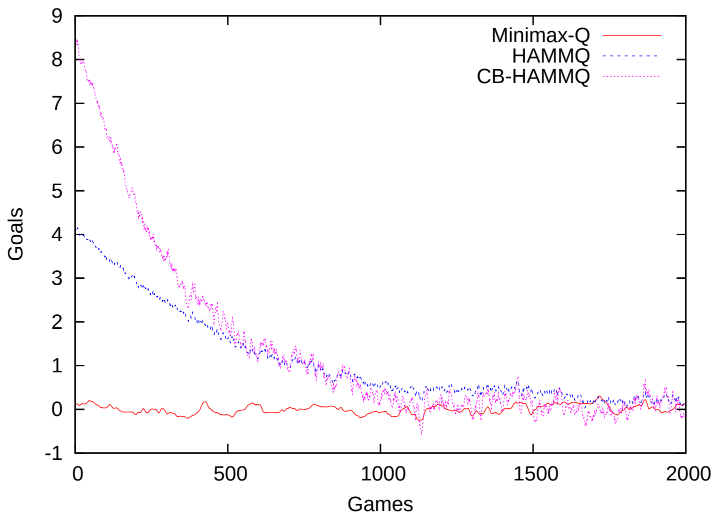

Fig. 3. Goals balance for Minimax–Q, the HAMMQ and the CB-HAMMQ algorithms against an opponent using Minimax–Q for the Expanded Littman's Robotic Soccer.

number of games the results of the algorithms are comparable, since the three algorithms converge to equilibrium.

Finally, table 3 shows the average number of goals and the average number of games won at the end of 20,000 games while playing against a random opponent, and table 4 presents the same data for games played against a Minimax–Q opponent, at the end of 2,000 games. It can be seen in table 4 that when Minimax–Q agents are playing against other Minimax–Q agents, the number of goals made and games won are approximately the same, while when CB-HAMMQ agents played against Minimax–Q ones, CB-HAMMQ team made more goals and won more games. CB-HAMMQ also won more games (1145, losing 425) and made more goals (11109) than the HAMMQ algorithm.

Table 3. Results for games against Random opponent.

Algorithm	Goals made × goals conceded
Minimax–Q	(140207 ± 174) × (38498 ± 164)
HAMMQ	(166208 ± 150) × (22065 ± 153)
CB-HAMMQ	(188168 ± 155) × (11292 ± 140)
	Games won × games lost
Minimax–Q	(18297 ± 33) × (1037 ± 28)
HAMMQ	(19469 ± 9) × (27 ± 4)
CB-HAMMQ	(19997 ± 1) × (0 ± 0)

Table 4. Results for games against Minimax–Q opponent.

Algorithm	Goals made × goals conceded
Minimax–Q	(10299 ± 234) × (9933 ± 240)
HAMMQ	(10467 ± 197) × (9347 ± 197)
CB-HAMMQ	(11109 ± 152) × (8845 ± 153)
	Games won × games lost
Minimax–Q	(848 ± 60) × (696 ± 55)
HAMMQ	(998 ± 50) × (530 ± 43)
CB-HAMMQ	(1145 ± 37) × (426 ± 32)

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 Littman (3). 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 C++ (GNU g++ compiler) and executed in a MacBook Pro, with 4GB of RAM in a Mac OS X platform.

6 Conclusion

This work presented a new algorithm, called Case Based Heuristically Accelerated Minimax–Q (CB-HAMMQ), which allows the use of a case base to define heuristics to speed up the well-known Multiagent Reinforcement Learning algorithm Minimax–Q. This approach builds upon an emerging technique, the Heuristic Accelerated Reinforcement Multiagent Learning, in which MRL methods are accelerated by making use of heuristic information. The experimental results obtained using a new domain proposed for the robotic soccer games showed that CB-HAMMQ attained better results than HAMMQ and Minimax–Q alone.

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

Bibliography

[1] 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

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

[3] 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

[4] 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

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

[6] 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

[7] Ros, R., Arcos, J.L., de Mántaras, R.L., Veloso, M.: A case-based approach for coordinated action selection in robot soccer. Artificial Intelligence (2009) doi:10.1016/j.artint.2009.02.004

[8] 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

[9] 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

[10] 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

[11] 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

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

[13] 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)

[14] 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

[15] Berger, R., Lämmel, G.: Exploiting past experience – case-based decision support for soccer agents. In: KI 2007: Advances in Artificial Intelligence - 30th Annual German Conference on AI. Volume 4667., Springer (2007) 440–443

[16] Srinivasan, T., Aarthi, K., Meenakshi, S.A., Kausalya, M.: Cbrrobosoc: An efficient planning strategy for robotic soccer using case based reasoning. In: CIMCA '06: Proceedings of the International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce, Washington, DC, USA, IEEE Computer Society (2006) 113

[17] Marling, C., Tomko, M., Gillen, M., Alexander, D., Chelberg, D.: 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. (2003)

[18] 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

[19] Burkhard, H.D., Berger, R.: Cases in robotic soccer. In Weber, R., Richter, M.M., eds.: ICCBR. Volume 4626 of Lecture Notes in Computer Science., Springer (2007) 1–15

[20] 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., ed.: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), AAAI Press, AAAI Press (2007) 1041–1046

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

[22] 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

[23] 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

[24] Munos, R., Moore, A.: Variable resolution discretization in optimal control. Machine Learning **49**(2/3) (2002) 291–323

[25] Kalyanakrishnan, S., Liu, Y., Stone, P.: Half field offense in robocup soccer: A multiagent reinforcement learning case study. In Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F., eds.: RoboCup. Volume 5001 of Lecture Notes in Computer Science., Springer (2007) 72–85