

Heuristic Selection of Actions in Multiagent Reinforcement Learning *

Reinaldo A. C. Bianchi

FEI University Center
Electrical Engineering Department
São Bernardo do Campo, Brazil.
rbianchi@fei.edu.br

Carlos H. C. Ribeiro

Technological Institute of Aeronautics
Computer Science Division
São José dos Campos, Brazil.
carlos@comp.ita.br

Anna H. R. Costa

Escola Politécnica
University of São Paulo
São Paulo, Brazil.
anna.reali@poli.usp.br

Abstract

This work presents a new algorithm, called Heuristically Accelerated Minimax-Q (HAMMQ), that allows the use of heuristics to speed up the well-known Multiagent Reinforcement Learning algorithm Minimax-Q. A heuristic function \mathcal{H} that influences the choice of the actions characterises the HAMMQ algorithm. This function is associated with a preference policy that indicates that a certain action must be taken instead of another. A set of empirical evaluations were conducted for the proposed algorithm in a simplified simulator for the robot soccer domain, and experimental results show that even very simple heuristics enhances significantly the performance of the multiagent reinforcement learning algorithm.

1 Introduction

Reinforcement Learning (RL) techniques are very attractive in the context of multiagent systems. The reasons frequently cited for such attractiveness are: many RL algorithms guarantee convergence to equilibrium in the limit [Szepesvari and Littman, 1996], they are based on sound theoretical foundations, and they provide model-free learning of adequate sub-optimal control strategies. Besides that, they are also easy to use and have been applied to solve a wide variety of control and planning problems when neither an analytical model nor a sampling model is available *a priori*.

In RL, learning is carried out on line, through trial-and-error interactions of the agent with its environment, receiving a reward (or penalty) at each interaction. In a multiagent system, the reward received by one agent depends on the behaviour of other agents, and a Multiagent Reinforcement Learning (MRL) algorithm needs to address non-stationary scenarios in which both the environment and other agents must be represented. Unfortunately, convergence of any RL algorithm may only be achieved after extensive exploration of the state-action space, which can be very time consuming. The existence of multiple agents increases by itself the size

of the state-action space, worsening the RL algorithms performance when applied to multiagent problems. Despite that, several MRL algorithms have been proposed and successfully applied to some simple problems, such as the Minimax-Q [Littman, 1994], the Friend-or-Foe Q-Learning [Littman, 2001] and the Nash Q-Learning [Hu and Wellman, 2003].

An alternative way of increasing the convergence rate of an RL algorithm is to use heuristic functions for selecting actions in order to guide the exploration of the state-action space in a useful way. The use of heuristics to speed up reinforcement learning algorithms has been proposed recently by Bianchi *et al.* [2004], and it has been successfully used in policy learning for simulated robotic scenarios.

This paper investigates the use of heuristic selection of actions in concurrent and on-policy learning in multiagent domains, and proposes a new algorithm that incorporates heuristics into the Minimax-Q algorithm, the Heuristically Accelerated Minimax-Q (HAMMQ). A set of empirical evaluation of HAMMQ were carried out in a simplified simulator for the robot soccer domain [Littman, 1994]. Based on the results of these simulations, it is possible to show that using even very simple heuristic functions the performance of the learning algorithm can be improved.

The paper is organised as follows: section 2 briefly reviews the MRL approach and describes the Minimax-Q algorithm, while section 3 presents some approaches to speed up MRL. Section 4 shows how the learning rate can be improved by using heuristics to select actions to be performed during the learning process, in a modified formulation of the Minimax-Q 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 Multiagent Reinforcement Learning

Markov Games (MGs) – also known as Stochastic Games (SGs) – are an extension of Markov Decision Processes (MDPs) that uses elements from Game Theory and allows the modelling of systems where multiple agents compete among themselves to accomplish their tasks.

Formally, an MG is defined by [Littman, 1994]:

- \mathcal{S} : a finite set of environment states.

*The support of FAPESP (Proc. 2006/05667-0), CNPq (Proc. 305558/2004-8) and CAPES/GRICES (Proc. 099/03) is acknowledged.

- $\mathcal{A}_1 \dots \mathcal{A}_k$: a collection of sets \mathcal{A}_i with the possible actions of each agent i .
- $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times \dots \times \mathcal{A}_k \rightarrow \Pi(\mathcal{S})$: a state transition function that depends on the current state and on the actions of each agent.
- $\mathcal{R}_i : \mathcal{S} \times \mathcal{A}_1 \times \dots \times \mathcal{A}_k \rightarrow \mathbb{R}$: a set of reward functions specifying the reward that each agent i receives.

Solving an MG consists in computing the policy $\pi : \mathcal{S} \times \mathcal{A}_1 \times \dots \times \mathcal{A}_k$ that maximizes the reward received by an agent along time.

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 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 [Littman, 1994] 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 .
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \rightarrow \mathbb{R}$: 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 .

The choice of an optimal policy for a ZSMG is not trivial because the performance of the agent depends crucially on the choice of the opponent. A solution for this problem can be obtained with the Minimax algorithm [Russell and Norvig, 2002]. It evaluates the policy of an agent regarding all possible actions of the opponent, and then choosing the policy that maximizes its own payoff.

To solve a ZSMG, Littman [1994] proposed the use of a similar strategy to Minimax for choosing an action in the Q-Learning algorithm, the Minimax-Q algorithm (see Table 1). Minimax-Q works essentially 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 given by:

$$Q(s, a, o) = r(s, a, o) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, o, s') V(s'), \quad (1)$$

and the value of a state can be computed using linear programming [Strang, 1988] via the equation:

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

where the agent's policy is a probability distribution over actions, $\pi \in \Pi(\mathcal{A})$, and π_a is the probability of taking the action a against the opponent's action o .

An MG where two players take their actions in consecutive turns is called an Alternating Markov Game (AMG). In this

Initialise $\hat{Q}_t(s, a, o)$.

Repeat:

 Visit state s .

 Select an action a using the $\epsilon - Greedy$ rule (eq. 4).

 Execute a , observe the opponent's action o .

 Receive the reinforcement $r(s, a, o)$

 Observe the next state s' .

 Update the values of $\hat{Q}(s, a, o)$ according to:

$$\hat{Q}_{t+1}(s, a, o) \leftarrow \hat{Q}_t(s, a, o) + \alpha [r(s, a, o) + \gamma V_t(s') - \hat{Q}_t(s, a, o)].$$

$s \leftarrow s'$.

Until some stop criterion is reached.

Table 1: The Minimax-Q algorithm.

case, as the agent knows in advance the action taken by the opponent, the policy becomes deterministic, $\pi : \mathcal{S} \times \mathcal{A} \times \mathcal{O}$ and equation 2 can be simplified:

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

In this case, the optimal policy is $\pi^* \equiv \arg \max_a \min_o Q^*(s, a, o)$. A possible action choice rule to be used is the standard $\epsilon - Greedy$:

$$\pi(s) = \begin{cases} \arg \max_a \min_o \hat{Q}(s, a, o) & \text{if } q \leq p, \\ a_{random} & \text{otherwise,} \end{cases} \quad (4)$$

where q is a random value with uniform probability in $[0, 1]$ and p ($0 \leq p \leq 1$) is a parameter that defines the exploration/exploitation trade-off: the greater the value of p , the smaller is the probability of a random choice, and a_{random} is a random action selected among the possible actions in state s . For non-deterministic action policies, a general formulation of Minimax-Q has been defined elsewhere [Littman, 1994; Banerjee *et al.*, 2001].

Finally, the Minimax-Q algorithm has been extended to cover several domains where MGs are applied, such as Robotic Soccer [Littman, 1994; Bowling and Veloso, 2001] and Economy [Tesauro, 2001].

One of the problems with the Minimax-Q algorithm is that, as the agent iteratively estimates Q , learning at early stages is basically random exploration. Also, update of the Q value is made by one state-action pair at a time, for each interaction with the environment. The larger the environment, the longer trial-and-error exploration takes to approximate the function Q . To alleviate these problems, several techniques, described in the next section, were proposed.

3 Approaches to speed up Multiagent Reinforcement Learning

The Minimax-SARSA Algorithm [Banerjee *et al.*, 2001] is a modification of Minimax-Q that admits the next action to be chosen randomly according to a predefined probability, separating the choice of the actions to be taken from the update of the Q values. If a' is chosen according to a greedy policy, Minimax-SARSA becomes equivalent to Minimax-Q. But if

a' is selected randomly according to a predefined probability distribution, the Minimax-SARSA algorithm can achieve better performance than Mimimax-Q [Banerjee *et al.*, 2001].

The same work proposed the Minimax-Q(λ) algorithm, by combining eligibility traces with Minimax-Q. Eligibility traces, proposed initially in the $TD(\lambda)$ algorithm [Sutton, 1988], are used to speed up the learning process by tracking visited states and adding a portion of the reward received to each state that has been visited in an episode. Instead of updating a state-action pair at each iteration, all pairs with eligibilities different from zero are updated, allowing the rewards to be carried over several state-action pairs. Banerjee *et al.* [2001] also proposed Minimax-SARSA(λ), a combination of the the two algorithms that would be more efficient than both of them, because it combines their strengths.

A different approach to speed up the learning process is to use each experience in a more effective way, through temporal, spatial or action generalization. The Minimax-QS algorithm, proposed by Ribeiro *et al.* [2002], accomplishes spatial generalization by combining the Minimax-Q algorithm with spatial spreading in the action-value function. Hence, at receiving a reinforcement, other action-value pairs that were not involved in the experience are also updated. This is done by coding the knowledge of domain similarities in a spreading function, which allows a single experience (*i.e.*, a single loop of the algorithm) to update more than a single cost value. The consequence of taking action a_t at state s_t is spread to other pairs (s, a) as if the real experience at time t actually was $\langle s, a, s_{t+1}, r_t \rangle$.

4 Combining Heuristics and Multiagent Reinforcement Learning: the HAMMQ Algorithm

The Heuristically Accelerated Minimax Q (HAMMQ) algorithm can be defined as a way of solving a ZSMG by making explicit use of a heuristic function $\mathcal{H} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \rightarrow \mathbb{R}$ 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 heuristic function is associated with a preference policy that indicates that a certain action must be taken instead of another. Therefore, it can be said that the heuristic function defines a ‘‘Heuristic Policy’’, that is, a tentative policy used to accelerate the learning process. The heuristic function can be derived directly from prior knowledge of the domain or from clues suggested by the learning process itself and is used only during the selection of the action to be performed by the agent, in the action choice rule that defines which action a should be executed when the agent is in state s . The action choice rule used in HAMMQ is a modification of the standard $\epsilon - Greedy$ rule that includes the heuristic function:

$$\pi(s) = \begin{cases} \arg \max_a \min_o \left[\hat{Q}(s, a, o) + \xi H_t(s, a, o) \right] & \text{if } q \leq p, \\ a_{random} & \text{otherwise,} \end{cases} \quad (5)$$

where $\mathcal{H} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \rightarrow \mathbb{R}$ is the heuristic function. The subscript t indicates that it can be non-stationary and ξ is a real

variable used to weight the influence of the heuristic (usually 1).

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:

$$H(s, a, o) = \begin{cases} \max_i \hat{Q}(s, i, o) - \hat{Q}(s, a, o) + \eta & \text{if } a = \pi^H(s), \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

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

As the heuristic function is used only in the choice of the action to be taken, the proposed algorithm is different from the original Minimax-Q in the way exploration is carried out. Since the RL algorithm operation is not modified (*i.e.*, updates of the function Q are the same as in Minimax-Q), our proposal allows that many of the theoretical conclusions obtained for Minimax-Q remain valid for HAMMQ.

Theorem 1 Consider a HAMMQ agent learning in a deterministic ZSMG, with finite sets of states and actions, bounded rewards ($\exists c \in \mathbb{R}; (\forall s, a, o), |R(s, a, o)| < c$), discount factor γ such that $0 \leq \gamma < 1$ and with values used in the heuristic function bounded by $(\forall s, a, o) h_{min} \leq H(s, a, o) \leq h_{max}$. For this agent, \hat{Q} values will converge to Q^* , with probability one uniformly over all states $s \in \mathcal{S}$, provided that each state-action pair is visited infinitely often (obeys the Minimax-Q infinite visitation condition).

Proof: In HAMMQ, the update of the value function approximation does not depend explicitly on the value of the heuristic function. Littman and Szepesvary [1996] presented a list of conditions for the convergence of Minimax-Q, and the only condition that HAMMQ could put in jeopardy is the one that depends on the action choice: the necessity of visiting each pair state-action infinitely often. As equation 5 considers an exploration strategy $\epsilon - greedy$ regardless of the fact that the value function is influenced by the heuristic function, this visitation condition is guaranteed and the algorithm converges. *q.e.d.*

The condition that each state-action pair must be visited an infinite number of times can be considered valid in practice – in the same way that it is for Minimax-Q – also by using other strategies:

- Using a Boltzmann exploration strategy [Kaelbling *et al.*, 1996].
- Intercalating steps where the algorithm makes alternate use of the heuristic function and exploration steps.
- Using the heuristic function during a period of time shorter than the total learning time.

The use of a heuristic function made by HAMMQ explores an important characteristic of some RL algorithms: the free choice of training actions. The consequence of this is that a suitable heuristic function speeds up the learning process, otherwise the result is a delay in the learning convergence that does not prevent it from converging to an optimal value.

Initialise $\hat{Q}_t(s, a, o)$ and $H_t(s, a, o)$.

Repeat:

Visit state s .

Select an action a using the modified ϵ -Greedy rule (Equation 5).

Execute a , observe the opponent's action o .

Receive the reinforcement $r(s, a, o)$

Observe the next state s' .

Update the values of $H_t(s, a, o)$.

Update the values of $\hat{Q}_t(s, a, o)$ according to:

$$\hat{Q}_{t+1}(s, a, o) \leftarrow \hat{Q}_t(s, a, o) + \alpha[r(s, a, o) + \gamma V_t(s') - \hat{Q}_t(s, a, o)].$$

$s \leftarrow s'$.

Until some stop criterion is reached.

Table 2: The HAMMQ algorithm.

The complete HAMMQ algorithm is presented on table 2. It is worth noticing that the fundamental difference between HAMMQ and the Minimax-Q algorithm is the action choice rule and the existence of a step for updating the function $H_t(s, a, o)$.

5 Robotic Soccer using HAMMQ

Playing a robotic soccer game is a task for a team of multiple fast-moving robots in a dynamic environment. This domain has become of great relevance in Artificial Intelligence since it possesses several characteristics found in other complex real problems; examples of such problems are: robotic automation systems, that can be seen as a group of robots in an assembly task, and space missions with multiple robots [Tambe, 1998], to mention but a few.

In this paper, experiments were carried out using a simple robotic soccer domain introduced by Littman [1994] and modelled as a ZSMG between two agents. In this domain, two players, A and B, compete in a 4 x 5 grid presented in figure 1. Each cell can be occupied by one of the players, which can take an action at a turn. The action that are allowed indicate the direction of the agent's move – north, south, east and west – or keep the agent still.

The ball is always with one of the players (it is represented by a circle around the agent in figure 1). 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 the agent leads it out the board, the agent stands still.

When a player with the ball gets into the opponent's goal, the move ends and its team scores one point. At the beginning of each game, the agents are positioned in the initial position, depicted in figure 1, and the possession of the ball is randomly determined, with the player that holds the ball making the first move (in this implementation, the moves are alternated between the two agents).

To solve this problem, two algorithms were used:

- Minimax-Q, described in section 2.
- HAMMQ, proposed in section 4.

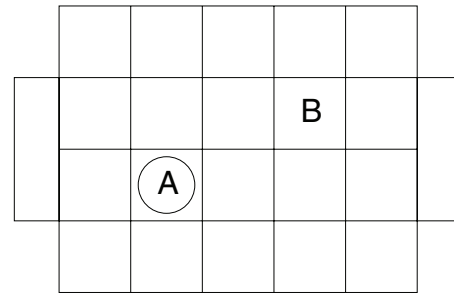


Figure 1: The environment proposed by Littman [1994]. The picture shows the initial position of the agents.

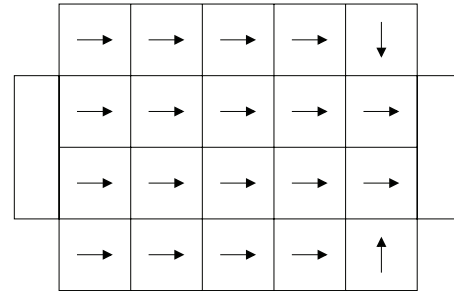


Figure 2: The heuristic policy used for the environment of figure 1. Arrows indicate actions to be performed.

The heuristic policy used was defined using a simple rule: if holding the ball, go to the opponent's goal (figure 2 shows the heuristic policy for player A of figure 1). Note that the heuristic policy does not take into account the opponents position, leaving the task of how to deviate from the opponent to the learning process. The values associated with the heuristic function are defined based on the heuristic policy presented in figure 2 and using equation 6.

The parameters used in the experiments were the same for the two algorithms, Minimax-Q and HAMMQ. The learning rate is initiated with $\alpha = 1.0$ and has a decay of 0.9999954 by each executed action. The exploration/ exploitation rate $p = 0.2$ and the discount factor $\gamma = 0.9$ (these parameters are identical to those used by Littman [1994]). The value of η was set to 1. The reinforcement used was 1000 for reaching the goal and -1000 for having a goal scored by the opponent. Values in the Q table were randomly initiated, with $0 \leq Q(s, a, o) \leq 1$. The experiments were programmed in C++ and executed in a AMD K6-II 500MHz, with 288MB of RAM in a Linux platform.

Thirty training sessions were run for each algorithm, with each session consisting of 500 matches of 10 games. A game finishes whenever a goal is scored by any of the agents or when 50 moves are completed.

Figure 3 shows the learning curves (average of 30 training sessions) for both algorithms when the agent learns how to play against an opponent moving randomly, and presents the average goal balance scored by the learning agent in each

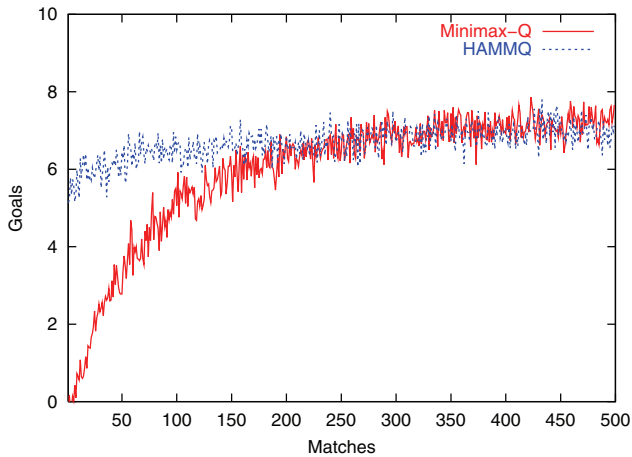


Figure 3: Average goal balance for the Minimax-Q and HAMMQ algorithms against a random opponent for Littman’s Robotic Soccer.

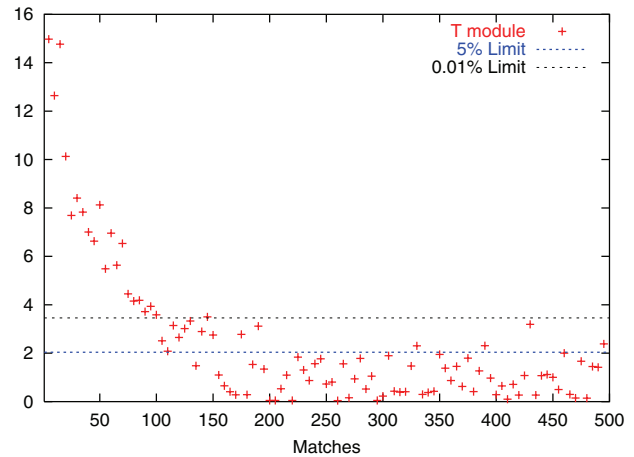


Figure 5: Results from Student’s t test between Minimax-Q and HAMMQ algorithms, training against a random opponent.

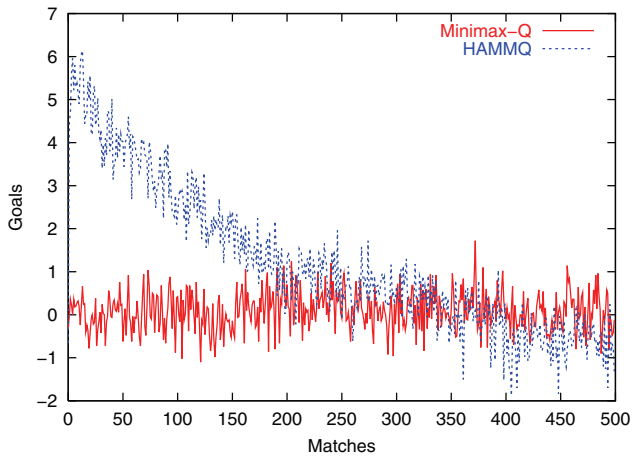


Figure 4: Average goal balance for the Minimax-Q and HAMMQ algorithms against an agent using Minimax-Q for Littman’s Robotic Soccer.

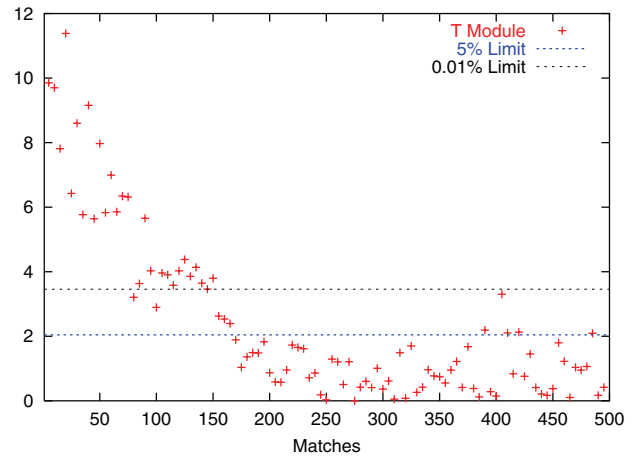


Figure 6: Results from Student’s t test between Minimax-Q and HAMMQ algorithms, training against an agent using Minimax-Q.

match. It is possible to verify that Minimax-Q has worse performance than HAMMQ at the initial learning phase, and that as the matches proceed, the performance of both algorithms become similar, as expected.

Figure 4 presents the learning curves (average of 30 training sessions) for both algorithms when learning while playing against a learning opponent using Minimax-Q. In this case, it can be clearly seen that HAMMQ is better at the beginning of the learning process and that after the 300th match the performance of both agents becomes similar, since both converge to equilibrium.

Student’s *t*-test [Spiegel, 1998] was used to verify the hypothesis that the use of heuristics speeds up the learning process. Figure 5 shows this test for the learning of the Minimax-Q and the HAMMQ algorithms against a random opponent (using data shown in figure 3). In this figure, it is possible to

see that HAMMQ is better than Minimax-Q until the 150th match, after which the results are comparable, with a level of confidence greater than 5%. Figure 6 presents similar results for the learning agent playing against an opponent using Minimax-Q.

Finally, table 3 shows the cumulative goal balance and table 4 presents the number of games won at the end of 500 matches (average of 30 training sessions). The sum of the matches won by the two agents is different from the total number of matches because some of them ended in a draw. What stands out in table 4 is that, due to a greater number of goals scored by HAMMQ at the beginning of the learning process, this algorithm wins more matches against both the random opponent and the Minimax-Q learning opponent.

Training Section	Cumulative goal balance
Minimax-Q × Random	(3401 ± 647) × (529 ± 33)
HAMMQ × Random	(3721 ± 692) × (496 ± 32)
Minimax-Q × Minimax-Q	(2127 ± 430) × (2273 ± 209)
HAMMQ × Minimax-Q	(2668 ± 506) × (2099 ± 117)

Table 3: Cumulative goal balance at the end of 500 matches (average of 30 training sessions).

Training Section	Matches won
Minimax-Q × Random	(479 ± 6) × (11 ± 4)
HAMMQ × Random	(497 ± 2) × (1 ± 1)
Minimax-Q × Minimax-Q	(203 ± 42) × (218 ± 42)
HAMMQ × Minimax-Q	(274 ± 27) × (144 ± 25)

Table 4: Number of matches won at the end of 500 matches (average of 30 training sessions).

6 Conclusion

This work presented a new algorithm, called Heuristically Accelerated Minimax-Q (HAMMQ), which allows the use of heuristics to speed up the well-known Multiagent Reinforcement Learning algorithm Minimax-Q.

The experimental results obtained in the domain of robotic soccer games showed that HAMMQ learned faster than Minimax-Q when both were trained against a random strategy player. When HAMMQ and Minimax-Q were trained against each other, HAMMQ outperformed Minimax-Q.

Heuristic functions allow RL algorithms to solve problems where the convergence time is critical, as in many real time applications. This approach can also be incorporated into other well known Multiagent RL algorithms, e.g. Minimax-SARSA, Minimax-Q(λ), Minimax-QS and Nash-Q.

Future works include working on obtaining results in more complex domains, such as RoboCup 2D and 3D Simulation and Small Size League robots [Kitano *et al.*, 1997]. Preliminary results indicate that the use of heuristics makes these problems easier to solve. Devising more convenient heuristics for other domains and analyzing ways of obtaining them automatically are also worthwhile research objectives.

References

- [Banerjee *et al.*, 2001] Bikramjit Banerjee and Sandip Sen and Jing Peng. Fast Concurrent Reinforcement Learners. In: *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 825–830, 2001.
- [Bianchi *et al.*, 2004] Reinaldo A. C. Bianchi, Carlos H. C. Ribeiro and Anna H. R. Costa. Heuristically Accelerated Q-Learning: a New Approach to Speed Up Reinforcement Learning. *Lecture Notes in Artificial Intelligence*, 3171:245–254, 2004.
- [Bowling and Veloso, 2001] Michael H. Bowling and Manuela M. Veloso. Rational and Convergent Learning in Stochastic Games. In: *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 1021–1026, 2001.
- [Hu and Wellman, 2003] Junling Hu and Michael P. Wellman. Nash Q-Learning for General-Sum Stochastic Games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.
- [Kaelbling *et al.*, 1996] Leslie P. Kaelbling, Michael L. Littman and Andrew W. Moore. Reinforcement Learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
- [Kitano *et al.*, 1997] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda and Eiichi Osawa. RoboCup: A Challenge Problem for AI. *AI Magazine*, 18(1):73–85, 1997.
- [Littman, 1994] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In: *Proceedings of the 11th International Conference on Machine Learning (ICML'94)*, pages 157–163, 1994.
- [Littman, 2001] Michael L. Littman. Friend-or-Foe Q-learning in general-sum games. In: *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*, pages 322–328, 2001.
- [Littman and Szepesvári, 1996] Michael L. Littman and Csaba Szepesvári. A Generalized Reinforcement Learning Model: Convergence and Applications. In: *Procs. of the 13th International Conference on Machine Learning (ICML'96)*, pages 310–318, 1996.
- [Ribeiro *et al.*, 2002] Carlos H. C. Ribeiro, René Pegoraro and Anna H. R. Costa. Experience Generalization for Concurrent Reinforcement Learners: the Minimax-QS Algorithm. In: *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, pages 1239–1245, 2002.
- [Russell and Norvig, 2002] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, 2nd edition, 2002.
- [Spiegel, 1998] Murray R. Spiegel. *Statistics*. McGraw-Hill, New York, 1998.
- [Strang, 1988] Gilbert Strang. *Linear algebra and its applications*. Harcourt, Brace, Jovanovich, San Diego, 3rd edition, 1988.
- [Sutton, 1988] Richard S. Sutton. Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3(1):9–44, 1988.
- [Szepesvári and Littman, 1996] Csaba Szepesvári and Michael L. Littman. Generalized Markov Decision Processes: Dynamic-Programming and Reinforcement-Learning Algorithms. Technical Report CS-96-11, Brown University, 1996.
- [Tambe, 1998] Milind Tambe. Implementing Agent Teams in Dynamic Multiagent Environments. *Applied Artificial Intelligence*, 12(2-3):189–210, 1998.
- [Tesauro, 2001] Gerald Tesauro. Pricing in Agent Economies Using Neural Networks and Multi-agent Q-Learning. *Lecture Notes in Computer Science*, 1828:288–307, 2001.