Should I trust my teammates? An experiment in Heuristic Multiagent Reinforcement Learning

Reinaldo A. C. Bianchi
Centro Universitário da FEI,
Av. Humberto de A. C. Branco, 3972
São Bernardo do Campo, 09850-901, Brazil.
Phone: +55 11 4353 2910
rbianchi@fei.edu.br

Ramón López de Mántaras
Artificial Intelligence Research Institute (IIIA-CSIC),
Campus de la UAB, E-08193
Bellaterra, Catalonia (Spain)
Phone: +34 93 580 9570
mantaras@iiia.csic.es

Abstract

Trust and reputation are concepts that have been traditionally studied in domains such as electronic markets, e-commerce, game theory and bibliometrics, among others. More recently, researchers started to investigate the benefits of using these concepts in multi-robot domains: when one robot has to decide if it should cooperate with another one to accomplish a task, should the trust in the other be taken into account? This paper proposes the use of a trust model to define when one agent can take an action that depends on other agents of his team. To implement this idea, a Heuristic Multiagent Reinforcement Learning algorithm is modified to take into account the trust in the other agents, before selecting an action that depends on them. Simulations were made in a robot soccer domain, which extends a very well known one proposed by Littman by expanding its size, the number of agents and by using heterogeneous agents. Based on the results it is possible to show the performance of the learning algorithm can be improved even when using very simple trust models.

1 Introduction

Reinforcement Learning (RL) techniques are very attractive in the context of Multiagent systems: they are easy to use, have guarantee of convergence to equilibrium in the limit (provided that some conditions are satisfied, such as a large number of visits to every state-action pair [Watkins, 1989]), are based on sound theoretical foundations [Littman and Szepesvári, 1996], 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* [Kaelbling *et al.*, 1996; Munos and Bourgine, 1998].

Unfortunately, convergence of any RL algorithm may only be achieved after extensive exploration of the state-action space, which can be very time consuming, a problem that is worsened by the existence of multiple agents. Despite that, Multiagent Reinforcement Learning (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 recently proposed 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 an useful way [Bianchi *et al.*, 2008]. In this proposal, called heuristically Accelerated Reinforcement Learning (HARL), the heuristic function is associated with a preference policy that indicates that a certain action must be taken instead of another. This proposal was also extended to deal with Multiagent problems [Bianchi *et al.*, 2007], but without taking into account that different agents may not perform in the way the heuristic action demands.

This paper investigates the use of a trust model to define when one agent can take an action that depends on other agents of his team. To implement this idea, a Heuristic Multiagent Reinforcement Learning algorithm called Heuristically Accelerated Minimax-Q (HAMMQ) was modified to take into account the trust one agent have in the other agents, before selecting an action that depends on them.

The remainder of this paper is organized as follows: Section 2 briefly reviews the Multiagent Reinforcement Learning problem and the Distributed Q–Learning algorithm, while Section 3 describes the heuristic approach to RL. Section 4 shows how to incorporate a simple trust model in the Heuristically Accelerated Minimax-Q algorithm. Section 5 presents the experiments performed and shows the results obtained. Finally, Section 6 provides our conclusions and outlines future work.

2 Multiagent Reinforcement Learning

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

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

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

```
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.

along time.

To solve a MG, 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). The action-value function of an action a in a state s when the opponent takes an action o is given by:

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

and the value of a state can be computed using linear programming via the equation:

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

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 players take their actions in consecutive turns is called an Alternating Markov Game (AMG). In this 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:

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

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 $\epsilon - Greedy$:

$$\pi(s) = \begin{cases} \arg\max\limits_a \min\limits_o \hat{Q}(s,a,o) & \text{if } q \leq p, \\ a_{random} & \text{otherwise,} \end{cases} \qquad (4)$$

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].

```
Initialize $\hat{Q}_t(s,a,o)$ and $H_t(s,a,o)$.
Repeat:
    Visit state $s$.
    Select an action $a$ using the modified $\epsilon{-}Greedy$ 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 $\hat{Q}(s,a,o)$ 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.

3 Heuristically Accelerated Multiagent Reinforcement Learning

Several algorithms that speed up Multiagent Reinforcement Learning (MRL) have been proposed. One of them is the Heuristically Accelerated Minimax Q (HAMMQ) algorithm [Bianchi *et al.*, 2007], which can be defined as a way of solving a ZSMG by making explicit use of a heuristic function $\mathcal{H} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \rightarrow \Re$ 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 action choice rule used in HAMMQ is a modification of the standard $\epsilon - Greedy$ rule that includes the heuristic function:

$$\pi(s) = \begin{cases} \arg\max\limits_a \min\limits_o \left[\hat{Q}(s,a,o) + \xi H_t(s,a,o)\right] \text{ if } q \leq p, \\ a_{random} \text{ otherwise,} \end{cases} \qquad (5)$$

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 $0 \leq p \leq 1$ is a parameter that defines the exploration/exploitation tradeoff. The subscript t indicates that it can be non-stationary and ξ is a real variable used to weight the influence of the heuristic.

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:

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

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 *et al.* [2007], together with the definition of an upper bound for the error. The complete HAMMQ algorithm is presented in Table 2.

One important characteristic of the HARL algorithms is that, as the heuristic function is explicit, the learning algorithm is able to further refine it, quickly removing any error that the heuristic may contain. 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 *et al.* [2007] investigated the use of the HAMMQ in a Multiagent domain, a simplified simulator for the robot soccer domain; Celiberto *et al.* [2007] studied the use of the HAMRL algorithms to speed up learning in the RoboCup 2D Simulation domain.

4 Combining Trust and MRL

One problem with the HAMMQ algorithm is that, in actions that involve more than one agent, one is never sure if the other agents will collaborate and perform as the heuristic demands. Also, in the case where heterogeneous agents exists, one agent cannot be sure that the other will be capable of completing the task. One way to tackle this problem is to use an explicit value that weights the influence of the heuristic, deciding if it should be used or not.

This paper proposes the use of a trust model to weight the influence of the heuristic. Among several trust models and definitions in the literature [Ramchurn *et al.*, 2004], we choose to implement an observed individual model of trust, following the one defined in [Mui *et al.*, 2002]. In this work, the trust in an agent a_j in the eyes of a_i is a real variable $0 < T_{ij} < 1$ which is the number of successful cooperation observed by a_i over the total number of observations made by a_i of a_j collaborations. $T_{i,j}$ is only factored in actions that includes a collaboration between agents i and j; actions that do not include a collaboration between two agents have $T_{i,j} = 1$.

To implement this model, the action choice rule used in the t-HAMMQ is a modification of the original one, where the trust value is used to weight the influence of the heuristic:

$$\pi(s) = \begin{cases} \arg\max\limits_{a} \min\limits_{o} \left[\hat{Q}(s,a,o) + T_{i,j} H(s,a,o) \right] & \text{if } q \leq p, \\ a_{random} & \text{otherwise.} \end{cases}$$

5 Robotic Soccer using t-HAMMQ

A set of empirical evaluations of t-HAMMQ were carried out in a proposed simulator for the robot soccer domain that extends the one proposed by Littman [1994]. In this domain, two teams, A and B, of three players compete in a 15 x 12 grid. Each cell can be occupied by one of the players, which can take an action at a turn. 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. 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 looses 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 a random position and the possession of the ball is randomly determined, with the player that holds the ball making the first move.

The agents in this extended simulator are heterogeneous in the sense that they have different perception and execution capabilities: some agents can perceive the whole field, while others can perceive only a small grid around them; some agents runs faster than others, and some agents are capable of kicking the ball further.

In this experiment each team is composed by the goalie, the defender and the attacker. The goalie only perceives a 5 x 5 grid around itself, while the other agents perceive the whole field. The attacker runs at twice the speed of the other agents, and the goalkeeper can only kick the ball as far as the middle of the field. The reinforce the agents receive are: the goalie receives -100 every time a goal is scored against it; the defender receives $+100$ every time it gets the ball and -100 every time it loses it; and the attacker receives $+100$ if a goal is scored by his team. The difference in the rewards they receive makes them learn different roles.

The heuristic policy used was defined using a simple rule: pass the ball to the agent closest to the goal. Note that the heuristic policy does not take into account the opponents position, leaving the task of how to avoid them to the learning process. In this example, the trust T_{ij} in an agent a_j in the eyes of a_i is the number of goal made by a_j observed by a_i over the total number of passes a_j received. T_{ij} starts the game with the value of 0.5.

Thirty training sessions were run for the Minimax-Q, the HAMMQ and the t-HAMMQ, with each session consisting of 3000 games of 10 trials. A trial finishes whenever a goal is scored or when 500 moves are completed.

Figure 1 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 t-HAMMQ is better 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 t-HAMMQ is better than Minimax-Q until the 1500^{th} game, with a level of confidence greater than 5%. After the 1500^{th} game the results are comparable, since both converge to equilibrium. (Tests were made until the 10.000^{th} game to verify if the algorithms had reached their equilibrium). The same comparison, between the t-HAMMQ and the HAMMQ, shows that the first is better than the latter until the 500^{th} game.

Finally, table 3 shows the average number of goals and table 4 presents the average number of games won at the end of 3000 games. It can be seen 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 t-HAMMQ agents played against Minimax-Q ones, t-HAMMQ team made more goals and won more games.

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 [1994]). 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.

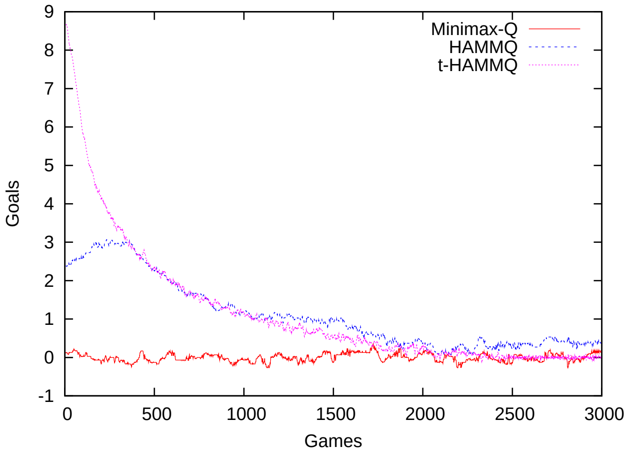

Figure 1: Goal balance for the Minimax-Q, the HAMMQ and the t-HAMMQ algorithms against an agent using Minimax-Q for Extended Littman's Robotic Soccer (average of 30 training sections for each algorithm).

Table 3: Average of goals at the end of 3000 games playing against a Minimax-Q opponent (average and standard deviation of 30 training sections for each algorithm).

Algorithm	Goals made × goals conceded
Minimax-Q	(12382 ± 77) × (12413 ± 85)
HAMMQ	(14704 ± 104) × (11345 ± 84)
t-HAMMQ	(16633 ± 302) × (13366 ± 278)

Table 4: Average number of games won at the end of 3000 games playing against a Minimax-Q opponent (average and standard deviation of 30 training sections for each algorithm).

Algorithm	Games won × games lost
Minimax-Q	(1218 ± 29) × (1226 ± 23)
HAMMQ	(1714 ± 28) × (829 ± 21)
t-HAMMQ	(1813 ± 75) × (649 ± 67)

6 Conclusion

This paper used a trust model to define when one agent can take an action that depends on other agents of his team, and tested it using a Heuristic Multiagent Reinforcement Learning algorithm, the HAMMQ, in an expanded robotic soccer simulation domain.

The experimental results obtained in the domain of robotic soccer games showed that using trust values the t-HAMMQ algorithm performed better than the Minimax-Q or the HAMMQ algorithms, scoring more goals and winning more games than both of them.

This approach can also be incorporated into other well known Multiagent RL algorithms, such as Minimax-SARSA, Minimax-Q(λ), Minimax-QS and Nash-Q. Future works also include working on obtaining results in more complex domains, such as RoboCup 2D and 3D Simulation and Small Size League robots.

References

Bikramjit Banerjee, Sandip Sen, and Jing Peng. Fast concurrent reinforcement learners. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 825–832, 2001.

Reinaldo A. C. Bianchi, Carlos H. C. Ribeiro, and Anna Helena Reali Costa. Heuristic selection of actions in multiagent reinforcement learning. In Manuela M. Veloso, editor, *IJCAI*, pages 690–695, 2007.

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

Luiz A. Celiberto, Carlos H. C. Ribeiro, Anna Helena Reali Costa, and Reinaldo A. C. Bianchi. Heuristic reinforcement learning applied to robocup simulation agents. In Ubbo Visser, Fernando Ribeiro, Takeshi Ohashi, and Frank Dellaert, editors, *RoboCup*, volume 5001 of *Lecture Notes in Computer Science*, pages 220–227. Springer, 2007.

Junling Hu and Michael P. Wellman. Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

Leslie P. Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

Michael L. Littman and Csaba Szepesvári. A generalized reinforcement learning model: convergence and applications. In *Proceedings of the 13th International Conference on Machine Learning (ICML'96)*, pages 310–318, 1996.

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.

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.

Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. Notions of reputation in multi-agents systems: a review. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 280–287, New York, NY, USA, 2002. ACM.

Rémi Munos and Paul Bourgine. Reinforcement learning for continuous stochastic control problems. In *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 1029–1035, Cambridge, MA, USA, 1998. MIT Press.

Sarvapali D. Ramchurn, Dong Huynh, and Nicholas R. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19:2004, 2004.

Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge, 1989.