Experience Generalization for Concurrent Reinforcement Learners: the Minimax-QS Algorithm

Carlos H. C. Ribeiro
Divisão de Ciência da Computação
Instituto Tecnológico de Aeronáutica
Pr. Mal. Eduardo Gomes, 50
12228-900 São José dos Campos, Brazil
carlos@comp.ita.br

Renê Pegoraro
Departamento de Computação
Universidade Estadual Paulista
Av. Luiz Edmundo Carrijo Coube
17033-360 Bauru, Brazil
pegoraro@fc.unesp.br

Anna H. Reali Costa
Laboratório de Técnicas Inteligentes
Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Trav. 3 no. 158
05508-900 São Paulo, Brazil
anna.reali@poli.usp.br

ABSTRACT

This paper investigates the use of experience generalization on concurrent and on-line policy learning in multi-agent scenarios, using reinforcement learning algorithms. Agents learning concurrently implies in a non-stationary scenario, since the reward received by one agent (for applying an action in a state) depends on the behavior of the other agents. Non-stationary scenarios can be viewed as a two-player game in which an agent and the other *player* (which represents the other agents and the environment) select actions from the available actions in the current state; these actions define the possible next state. An RL algorithm that can be applied to such a scenario is the Minimax-Q algorithm, which is known to guarantee convergence to equilibrium in the limit. However, finding optimal control policies using any RL algorithm (Minimax-Q included) can be very time consuming. We investigate the use of experience generalization for increasing the rate of convergence of RL algorithms, and contribute a new learning algorithm, Minimax-QS, which incorporates experience generalization to the Minimax-Q algorithm. We also prove its convergence to Minimax-Q values under suitable conditions.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Knowledge acquisition*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General

General Terms

Algorithms, Theory

Keywords

reinforcement learning, Markov games, Minimax-Q, experience generalization, robot soccer.

1. INTRODUCTION

Reinforcement learning (RL) algorithms are very attractive to be used in solving a wide variety of control and planning problems, since some of them are known to guarantee convergence to equilibrium in the limit [9] and provide model-free learning of control strategies. In RL, learning is carried out on-line, through trial-and-error interactions of the agent with the environment. Unfortunatelly, convergence of any RL algorithm may only be achieved after extensive exploration of the state-action space, which can be very time consuming.

However, the rate of convergence of an RL algorithm can be increased by using experience generalization, an approach in which a single experience (*i.e.*, a single loop of the algorithm) can 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 $\langle s, a, s_{t+1}, r_t \rangle$ [7].

This paper investigates the use of experience generalization on concurrent and on-line policy learning in multi-agent scenarios, using reinforcement learning (RL) algorithms.

Agents learning concurrently implies in a non-stationary scenario, since the reward received by one agent (for applying an action in a state) depends on the behavior of the other agents. Non-stationary scenarios can be viewed as a two-player game in which an agent and the other *player* (which represents the other agents and the environment) select actions from the available actions in the current state; these actions define the possible next state. An RL algorithm that can be applied to such a scenario is the Minimax-Q algorithm, proposed by Littman in [3].

In this paper we present a new algorithm, Minimax-QS, which incorporates experience generalization to Minimax-Q. We run a series of empirical evaluation of the algorithm in a simplified simulator for the soccer domain. We show that even using very simple domain-dependent rules, the performance of the learning algorithm can be improved. We also prove its convergence to Minimax-Q values under suitable conditions.

Experience generalization is related to state aggregation methods analyzed by some authors [11],[10],[8] in the context of function approximation for reinforcement learning. These authors have been considering the effects of adaptive aggregation when compact representations are used. In particular, it can be shown [9] that some RL algorithms acting on a set of aggregate states converge, provided a persistently exciting action policy is used. The trouble, however, is that the set of action values asymptotically reached will depend on the limit distribution defined by this action policy. If the action policy itself depends on the estimated action values, there is no simple way to ensure convergence. In this paper, we deal with this problem by considering a parameter that controls the degree of experience generalization along time.

2. MULTI-AGENT RL LEARNING

In this section we first introduce the Markov Game framework(MG), which can be viewed as an extended Markov Decision Process (MDP) to multiple agents. We then present the Minimax-Q algorithm for solving MGs. Minimax-Q is based on both Q-Learning, an RL technique for solving MDPs, and the Minimax algorithm, which is applied to finding game solutions.

2.1 Markov Games

Let us consider n agents interacting with the environment via perception and action. On each interaction step each agent i senses the current state s_t of the environment, and chooses an action a_i to perform. The set of actions $a_1, \ldots, a_n$ alter the state s_t of the environment, and a scalar reinforcement signal r_i (a reward or penalty) is provided to each agent i to indicate the desirability of the resulting state.

An MG is formally represented by a tuple:

$$\langle n, S, A_1, A_2, \ldots, A_n, r_1, r_2, \ldots, r_n, P\rangle$$

(see [3]), where:
n is the number of agents;
S is a set of states;
$A_1, \ldots, A_n$ is a collection of sets A_i of the actions available to agent i;
$r_i : S \times A_1 \ldots \times A_n \rightarrow \mathcal{R}$ is a scalar reinforcement function for the ith agent,
$P : S \times A_1 \ldots \times A_n \rightarrow \Pi(S)$ is a state transition function, where a member of $\Pi(S)$ is a probability distribution over S. $P(s_{t+1}|s_t, a_1, \ldots, a_n)$ represents the probability of moving from state s_t to s_{t+1} when the n agents perform respectively actions $a_1, \ldots, a_n$ at state s_t.

2.2 Minimax-Q

Let us consider a specialization of the MG framework, which consists of two agents performing actions in alternating turns, in a zero-sum game. Let A be the set of possible actions that the playing agent $\mathbf{A}$ can choose from, and $\bar{A}$ the set of actions for the opponent player $\mathbf{B}$. $r_{s_t,a_t,\bar{a}_t}$ is the immediate reinforcement $\mathbf{A}$ receives for performing action $a_t \in A$ in state $s_t \in S$ when its opponent $\mathbf{B}$ performs action $\bar{a}_t \in \bar{A}$.

The goal of $\mathbf{A}$ is to learn an optimal policy of actions that maximizes its expected cumulative sum of discounted reinforcements. Learning this policy is very difficult, since it depends on the actions the opponent performs. The solution to this problem is to evaluate each policy with respect to the opponent's strategy that makes it look the worst. This idea is the core of the Minimax-Q algorithm [3], which is known to guarantee convergence to equilibrium in the limit [9].

For deterministic action policies, the optimal value of a state $s_t \in S$ in an MG is:

$$V^*(s_t) = \max_{a \in A} \min_{\bar{a} \in \bar{A}} Q(s_t, a, \bar{a}) \qquad (1)$$

and the Minimax-Q learning rule is:

$$Q_{t+1}(s_t, a_t, \bar{a}_t) = Q_t(s_t, a_t, \bar{a}_t) + \alpha_t[r_{s_t,a_t,\bar{a}_t} + \gamma \hat{V}_t(s_{t+1}) - Q_t(s_t, a_t, \bar{a}_t)] \qquad (2)$$

where:
s_t is the current state,
a_t is the action performed by $\mathbf{A}$ in s_t,
$\bar{a}_t$ is the action performed by $\mathbf{B}$ in s_t,
$Q_{t+1}(s_t, a_t, \bar{a}_t)$ is the expected discounted reinforcement for taking action a_t when $\mathbf{B}$ performs $\bar{a}_t$ in state s_t, and continuing the optimal policy thereafter,
$r_{s_t,a_t,\bar{a}_t}$ is the reinforcement received by $\mathbf{A}$,
s_{t+1} is the consequent state,
$\hat{V}_t(s_{t+1})$ is the current estimate of the optimal expected discounted reward $V^*(s_{t+1})$,
γ is the discount factor,
α_t is the learning rate.

For non-deterministic action policies, a general formulation of Minimax-Q has been defined elsewhere [3], [1].

3. GENERALIZING MINIMAX-Q

Ribeiro [6] argues that embedding a priori knowledge in an RL algorithm may improve its convergence rate. He proposes the use of a spreading mechanism in which a single experience (*i.e.*, a single loop of the algorithm) can update more than a single action value. This better use of experience in RL algorithms is known as experience generalization, in which 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 $\langle s, a, s_{t+1}, r_t\rangle$. Considering this spreading mechanism, we propose a variant of the Minimax-Q algorithm, named Minimax-QS.

Formally, in Minimax-QS for alternating MG, at time t:
1. $\mathbf{A}$ and $\mathbf{B}$ observe the current state s_t.
2. $\mathbf{A}$ selects an action $a_t \in A$ and executes it.
3. $\mathbf{B}$ selects an action $\bar{a}_t \in \bar{A}$ and executes it.
4. $\mathbf{A}$ and $\mathbf{B}$ observe the next state s_{t+1} and receive the reinforcement $r_{s_t,a_t,\bar{a}_t}$.
5. The Q values for every state-$\mathbf{A}$ action-$\mathbf{B}$ action 3-tuple $(s, a, \bar{a})$ are updated according to:

$$Q_{t+1}(s, a, \bar{a}) = Q_t(s, a, \bar{a}) + \alpha_t \sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a})[r_{s_t,a_t,\bar{a}_t} + \gamma \hat{V}_t(s_{t+1}) - Q_t(s, a, \bar{a})] \qquad (3)$$

6. Repeat steps above until stopping criterion is met.

where $\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a})$ is the *spreading function* ($0 \leq \sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) \leq 1$). The standard Minimax-Q algorithm corresponds to Equation 3 with $\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) = \delta(s_t, s)\delta(a_t, a)\delta(\bar{a}_t, \bar{a})$, where $\delta(.,.)$ is the Kronecker delta function (that is, $\delta(u, v) = 1$ if $u = v$, otherwise $\delta(u, v) = 0$).

By using $\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a})$ it is possible to reduce the learning time of the Minimax-Q algorithm, since the consequence of choosing action a_t when the opponent chooses $\bar{a}_t$ in state s_t can spread to all others similar 3-tuple $(s, a, \bar{a})$.

For the experiments reported, we consider a spreading mechanism that produces generalization only along the state space S, that is, $\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) = g_t(s_t, s)\delta(a_t, a)\delta(\bar{a}_t, \bar{a})$, where $g_t(s_t, s)$ is a state similarity function. We define $g_t(s_t, s) = \tau^d$, where τ is a constant and d is a similarity — distance — measure between the current state s_t and a similar state s.

4. CONVERGENCE OF MINIMAX-QS

We now present a proof of convergence for the Minimax-QS algorithm. The proposition to be proved is the following:

PROPOSITION 1. *Assume that the conditions for convergence of Minimax-Q to the action value function Q are satisfied. Then the action values generated by the Minimax-QS algorithm converge to Q with probability one if the convergence $\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) - \delta(s, s_t)\delta(a, a_t)\delta(\bar{a}, \bar{a}_t) \to 0$ is of order $O(\alpha_t)$.*

What this proposition basically says is that if the spreading mechanism vanishes at least as quickly as the learning rate α_t, then Minimax-QS converges to the action values generated by Minimax-Q.

A related proof for the QS-algorithm — a spreading-based formulation of Q-learning for single agents — can be found in [7].

4.1 Simultaneous Approximations on RL

Finding an optimal cost function in RL algorithms implies two approximations going on simultaneously: one of them approximates a Dynamic Programming operator, and the second one uses the resulting approximation itself to estimate the optimal costs. To illustrate, let us consider the update equation for Q-learning:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t[r_{s_t,a_t} + \gamma \hat{V}_t(s_{t+1}) - Q_t(s_t, a_t)] \quad (4)$$

This equation iteratively approximates the Action Value operator applied to $Q_t(s_t, a_t)$

$$(\mathrm{T}^q Q)(s_t, a_t) = r_{s_t,a_t} + \gamma \sum_{s_{t+1} \in S} P(s_{t+1}|s_t, a_t) \max_a Q(s_{t+1}, a) \quad (5)$$

whilst using the resulting approximation itself to approximate the optimal costs $V^*(s_{t+1}) = \max_a Q(s_{t+1}, a)$ through $\hat{V}_t(s_{t+1}) = \max_a Q_t(s_{t+1}, a)$.

As an additional example, let us consider the Minimax-Q algorithm for alternating Markov games. The corresponding equation

$$Q_{t+1}(s_t, a_t, \bar{a}_t) = Q_t(s_t, a_t, \bar{a}_t) + \alpha_t[r_{s_t,a_t,\bar{a}_t} + \gamma \hat{V}_t(s_{t+1}) - Q_t(s_t, a_t, \bar{a}_t)] \quad (6)$$

approximates the operator

$$(\mathrm{T}^{mq} Q)(s_t, a_t, \bar{a}_t) = r_{s_t,a_t,\bar{a}_t} + \gamma \sum_{s_{t+1} \in S} P(s_{t+1}|s_t, a_t, \bar{a}_t) V^*(s_{t+1}) \quad (7)$$

applied to $Q_t(s_t, a_t, \bar{a}_t)$. The resulting approximation itself is then used to approximate the optimal costs $V^*(s_{t+1})$ via $\hat{V}_t(s_{t+1}) = \max_a \min_{\bar{a}} Q_t(s_{t+1}, a, \bar{a})$.

It might be convenient to relate stochastic approximations characteristic of RL algorithms — such as those exemplified above — and simpler and better studied approximations. This would allow us to simplify or propose theorems for the more complicated cases by relating them to their simpler counterparts. Indeed, we adopt this line of reasoning for proving the convergence of Minimax-QS. First, however, it is important to review a theorem that relates stochastic approximations.

4.2 Relating Stochastic Approximations

Let $B(S)$ be the set of cost functions over S and $\mathrm{T} : B(S) \mapsto B(S)$ be an arbitrary contraction mapping with fixed point V^*.

For Dynamic Programming algorithms in general, a contraction mapping T (or T^q) is applied directly to successively approximate V^* (or Q). In model-free RL methods, such operator is not available and the agent must use its own experience to approximate it.

Consider a sequence of operators $\mathrm{T}_t : (B(S) \times B(S)) \mapsto B(S)$ and define $U_{t+1} = \mathrm{T}_t U_t V$ where V and U_0 are arbitrary cost functions. We say that T_t approximates T at V with probability one uniformly over S if U_t converges to $\mathrm{T}V$ uniformly over S.

We can interpret this in the following way: a 'memory' U_t is used as a help to make T_t approximate T. The equivalence between these operators is assessed by using any 'test function' V that subject either to T (through $\mathrm{T}V$) or to T_t (through $\mathrm{T}_t U_t V$) produces the same result.

THEOREM 1. *[9] Let T be an arbitrary mapping with fixed point V^*, and let T_t approximate T at V^* with probability one uniformly over S. Let V_0 be an arbitrary cost function, and define $V_{t+1} = \mathrm{T}_t V_t V_t$. If there are functions $0 \le F_t((x)) \le 1$ and $0 \le G_t((x)) \le 1$ satisfying the conditions below with probability one, then V_t converges to V^* with probability one uniformly over S.*

1. *For every U_1, $U_2 \in B(S)$ and $s \in S$,*

$$|(\mathrm{T}_t U_1 V^*)(s) - (\mathrm{T}_t U_2 V^*)(s)| \le G_t(s)|U_1(s) - U_2(s)| \quad (8)$$

2. *For every U, $V \in B(S)$ and s, $s' \in S$,*

$$|(\mathrm{T}_t U V^*)(s) - (\mathrm{T}_t U V)(s)| \le F_t(s) \sup_{s'} |V^*(s') - V(s')| \quad (9)$$

3. *For all $k > 0$, the product $\prod_{t=k}^{n} G_t(s)$ converges to zero uniformly in s as n increases;*

4. *There is $0 \le \beta \le 1$ such that for all s and large enough t, $F_t(s) \le \beta(1 - G_t(s))$.*

This theorem eliminates the burden of having to prove the convergence of the cost functions V_t to V^*. Instead, it says that it suffices to show that — given the above conditions — the operator T_t approximates the operator T at the fixed point V^*.

A proof and a full discussion on the applicability of this theorem can be found respectively in [4] and [9].

4.3 A Proof of Convergence for Minimax-QS

The proof involves three basic steps. First, an appropriate space and operators T and T_t must be defined. Then, conditions 1 to 4 of Theorem 1 must be verified. Finally, it must be checked if T_t approximates T with probability 1. Ideally, T_t^{mqs} should be defined in such a way that $U_{t+1} = \mathrm{T}_t U_t V$ is an ordinary stochastic approximation process.

Let us reconsider the update equation for the Minimax-QS algorithm. The superscript s indicates that spreading has been used to update the action values:

$$Q^s_{t+1}(s,a,\bar{a}) = Q^s_t(s,a,\bar{a}) + \alpha_t\sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a})[r_{s_t,a_t,\bar{a}_t} + \gamma\hat{V}_t(s_{t+1}) - Q^s_t(s,a,\bar{a})] \quad (10)$$

Assume the function σ_t is such that $\sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a})$ converges to $\delta(s_t,s)\delta(a_t,a)\delta(\bar{a}_t,\bar{a})$ uniformly. This means that the Minimax-QS update rule approaches more and more the standard Minimax-Q equation as learning progresses.

The operator T is the Minimax-Q operator, namely

$$(\mathrm{T}^{mq}Q)(s,a,\bar{a}) = r_{s,a,\bar{a}} + \gamma \sum_{s_{t+1}} P(s_{t+1}|s,a,\bar{a})V^*(s_{t+1}) \quad (11)$$

and the random operator sequence T_t is

$$(\mathrm{T}_t^{mqs}UV)(s,a,\bar{a}) = U(s,a,\bar{a}) + \alpha_t\sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a})[r_{s_t,a_t,\bar{a}_t} + \gamma \max_u \min_{\bar{u}} V(s_{t+1},u,\bar{u}) - U(s,a,\bar{a})] \quad (12)$$

where $U,V : S \times A \times \bar{A} \to \mathbb{R}$. The update equation for Minimax-QS can then be written as $Q^s_{t+1} = \mathrm{T}_t^{mqs} Q^s_t Q^s_t$, and if all the conditions of Theorem 1 are satisfied then this process converges to the fixed point of T^{mq} — namely, the Q values for the Minimax-Q algorithm.

Let us choose

$$G_t(s,a,\bar{a}) = 1 - \alpha_t\sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a}) \quad (13)$$

and

$$F_t(s,a) = \alpha_t\sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a}) \quad (14)$$

which respectively satisfy the first two conditions (inequalities 8 and 9) of Theorem 1.

Condition 3 would imply $\prod_{t=k}^{\infty}(1-\alpha_t\sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a})) = 0$. Consider then this result [2]: *if* $0 < m_t < 1$ *then* $\prod_{t=k}^{\infty}(1-m_t) = 0$ *iff* $\sum_{t=k}^{\infty} m_t = \infty$. In our case, this means that $\sum_{t=k}^{\infty}\alpha_t\sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a}) = \infty$ is required. However, if we *assume* $\sum_{t=k}^{\infty}\alpha_t\delta(\mathbf{z},x_t)\delta(u,a_t)\delta(\bar{u},\bar{a}_t) = \infty$ (mandatory to ensure convergence of the Minimax-Q algorithm, see [4]) and $\delta(\mathbf{z},x_t)\delta(u,a_t)\delta(\bar{u},\bar{a}_t) \le C\sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a})$ for some $0 < C \le \infty$, then $C\sum_{t=k}^{\infty}\alpha_t\sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a}) \ge \sum_{t=k}^{\infty}\alpha_t\delta(\mathbf{z},x_t)\delta(u,a_t)\delta(\bar{u},\bar{a}_t) \ge \infty$, satisfying the condition 3.

Condition 4 is satisfied because $F_t(s,a) = 1 - G_t(s,a)$.

Finally, we must show that T_t^{mqs} approximates T^{mq} at Q. As we already know that Minimax-Q converges to Q (and thus its corresponding random operator T_t^{mq} approximates T^{mq} at Q), it suffices to show that, for a fixed $\hat{V}(s_{t+1})$

$$Q^s_{t+1}(s,a,\bar{a}) = Q^s_t(s,a,\bar{a}) + \alpha_t\sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a})[r_{s_t,a_t,\bar{a}_t} + \gamma\hat{V}(s_{t+1}) - Q^s_t(s,a,\bar{a})] \quad (15)$$

and

$$Q_{t+1}(s_t,a_t,\bar{a}_t) = Q_t(s_t,a_t,\bar{a}_t) + \alpha_t[r_{s_t,a_t,\bar{a}_t} + \gamma\hat{V}(s_{t+1}) - Q_t(s_t,a_t,\bar{a}_t)] \quad (16)$$

converge to the same value. In particular, showing this would imply that T_t^{mqs} approximates T_t^{mq} for the fixed point $V^*(s_{t+1})$. As T_t^{mq} itself approximates T^{mq} at this point, that would mean that T_t^{mqs} also approximates T^{mq} at Q.

The proof can be carried out separately for every state-action pair. Let us then fix an arbitrary triplet $(s,a,\bar{a})$ and denote by Q^s_t, Q_t, σ_t, *etc.* the values of $Q^s_t(s,a,\bar{a})$, $Q_t(s,a,\bar{a})$, $\sigma_t(s_t,a_t,\bar{a}_t,s,a,\bar{a})$, *etc.* If $\alpha_t = 0$ then neither Q^s_t nor Q_t changes, so we can consider $\alpha_t > 0$ for all t. Assume now that $\sigma_t - \delta_t = O(\alpha_t)$, *i.e.*, there is a bounded function B_t such that $\sigma_t - \delta_t = B_t\alpha_t$. This assumption means that the spreading function σ_t must converge to δ_t at least as quickly as α_t converges to zero. Some algebraic manipulation on equations 15 yields

$$Q^s_{t+1} = Q^s_t + \alpha_t[r_t + \gamma\hat{V}(s_{t+1}) - Q^s_t] + \alpha_t^2 B_t[r_t + \gamma\hat{V}(s_{t+1}) - Q^s_t] \quad (17)$$

One can see the above equation as a small perturbation of the standard Minimax-Q update, where the additive perturbation term $\alpha_t^2 B_t[r_t + \gamma\hat{V}(s_{t+1}) - Q^s_t]$ can be neglected as α_t gets smaller. Thus, equations 15 and 16 converge to the same value for a fixed $\hat{V}(s_{t+1})$, and therefore the Minimax-QS operator T_t^{mqs} approximates the Minimax-Q operator T^{mq} at Q. Thus, according to theorem 1, T_t^{mqs} converges to Q.

5. EXPERIMENTS IN A SOCCER DOMAIN

To perform the experiments, we used the soccer simulator introduced by Littman [3]. It is a two-player zero-sum game played on a 4x5 grid. The players always occupy distinct grid squares. The initial configuration of the board consists of the players **A** and **B** placed in the positions shown in Figure 1, and the possession of the ball is given randomly to **A** or **B** (agent **A** in figure).

At each time step, the players can move around by choosing from 5 actions: N, S, E, W, and stand. When one player attempts to move to the grid square occupied by the other player, the move fails and the second player gets the ball. When a player performs an action that would take it out of the board, the move does not take place. When the player with the ball reaches the goal (right for **A** and left for **B**), it scores and the board is reset to its initial configuration.

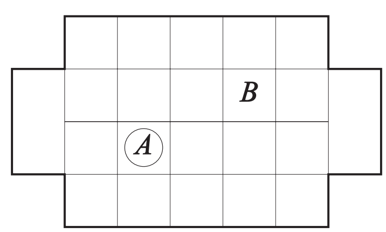

Figure 1: The soccer simulator. An initial board configuration (A with the ball).

A number of state similarity functions can be explored in the soccer domain. We demonstrate the effectiveness of the use of the proposed spreading mechanism considering a state similarity function based on very simple rules derived from particular board configurations.

Referring to Figure 1, it can be noticed that provided player A holds the ball, it does not matter where exactly player B is, as far as the overall game situation is concerned. Thus, we addopted a state similarity function which consider as similar those states in which A holds the ball and B lies in a region around the position it was when the experience took place. However, as the game board is small, similarity conditions must decrease quickly as B moves further away from its real position (because its corresponding state encompasses a large region relative to the game board). Quick spatial decrease was provided by using an exponential form for the state similarity function.

Similarity among configurations was defined as a function of the number of actions required to move the opponent (player **B**) from the position where the real experience took place to the position defined in the similar configuration considered. Formally, this measure of similarity mentioned corresponds to a spreading function $\sigma_t(s_t, a_t, \bar{a}_t, s, a, \bar{a}) = g_t(s_t, s)\delta(a_t, a)\delta(\bar{a}_t, \bar{a})$ (see equation 3) with $g_t(s_t, s) = \tau^d$, where d is the minimal number of actions to get the opponent from s_t to s and τ is either a constant (in some of the experiments reported below it is kept at 0.7) or (for the remaining experiments) a decreasing linear function of the iteraction number (0.7 is the maximum initial value).

We only spread the experience $\langle s_t, a_A, a_B, s_{t+1}, r_t \rangle$ at time t to all states s defined for some neighborhood of $\mathbf{B_{s_t}}$ (see Figure 2). In the experiments reported here this neighborhood is measured by the chessboard metric d_{ch} for two board positions $\mathbf{B_{s_t}} = (x_{B_{s_t}}, y_{B_{s_t}})$, where the real experience took place, and the similar state defined by player **B** at $\mathbf{B_s} = (x_{B_s}, y_{B_s})$:

$$d_{ch}(\mathbf{B_{s_t}}, \mathbf{B_s}) = \max\{|x_{B_{s_t}} - x_{B_s}|, |y_{B_{s_t}} - y_{B_s}|\} \qquad (18)$$

Given a d_{ch} value, a spreading area is defined where each cell of this area represents the **B** position in the similar state considered. Figure 2 depicts the spreading value τ^d for all cells included in the area defined by $d_{ch} = 1$ (Figure 2(a)) and $d_{ch} = 2$ (Figure 2(b)).

5.1 Experimental Setup

We run 9 different experiments in the soccer simulator, 3 using a constant spreading function in the learning algorithm — Minimax-QS, constant spreading — 3 using a linearly decreasing spreading function in the learning algorithm — Minimax-QS, decreasing spreading — and 3 using a traditional implementation of the Minimax-Q algorithm — Minimax-Q.

The learning player **A** was trained against a random opponent **B** (Figures 3 and 4) and against a Minimax-Q opponent **B** (Figure 5).

The parameters used in the algorithms were set as: $\gamma = 0.9$, initial value of $\alpha = 1.0$ (maximum) and decreasing linearly to 0 in the 150000 iteraction, *i.e.*, becomes 0 at aproximately the game 500, initial value of $Q - table = 0$, *random rate of exploration* $= 0.2$, representing the probability of choosing a random action, and *random rate of action execution* $= 0.2$, representing the non-determinism in the action execution.

For each learning algorithm, we run a sequence of 100 sessions, each of them consisting of 600 matches. Each match is composed of 10 games, and each game is ended by a goal scored (either by the learning player or by the opponent) or by reaching a pre-defined number of iterations (empirically set as 50). At the beginning of each match, both players are transported to the initial configuration of the board (see Figure 1). Learning data is reset at the beginning of each session. We computed the average number of goals balance scored by the learning player from 1 to 600 matches, over 100 sessions.

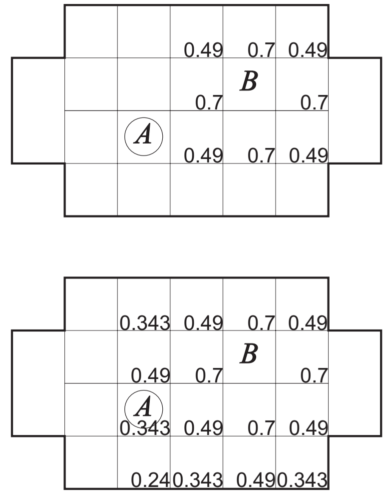

Figure 2: Spreading area and values for the configuration illustrated in Figure 1. (a) Area for chessboard distance $d_{ch} = 1$ (8 neighbors of $\mathbf{B_{s_t}}$) and (b) for $d_{ch} = 2$ (24 neighbors of $\mathbf{B_{s_t}}$).

5.2 Results

The resulting score balances for the learner **A** against the opponent **B** for each experiment are illustrated in Figures 3,4, and 5.

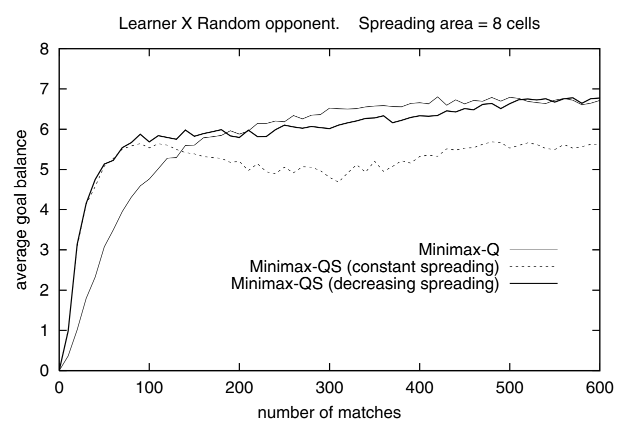

Figure 3: Results for the soccer game using the Minimax-Q algorithm (thin line), the Minimax-QS algorithm with a linearly decreasing spreading fucntion (bold line), and the Minimax-QS algorithm using a constant spreading function (dashed line). The spreading values used are shown in Figure 2(a).

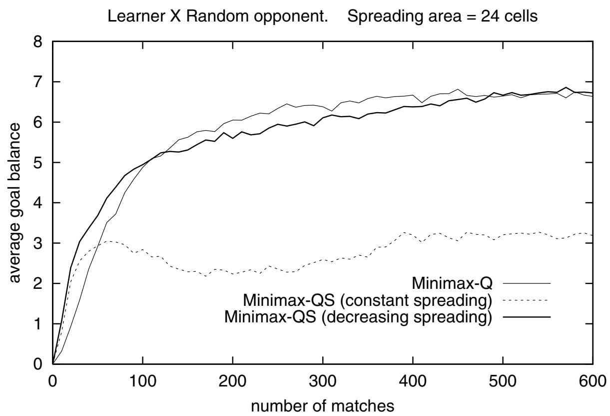

Figure 4: Results for the soccer game using the Minimax-Q algorithm (thin line), the Minimax-QS algorithm with a linearly decreasing spreading function (bold line), and the Minimax-QS algorithm using a constant spreading function (dashed line). The spreading values used are shown in Figure 2(b).

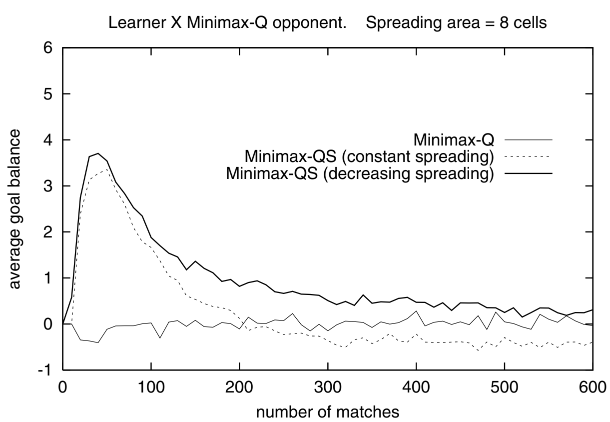

Figure 5: Results for the soccer game using the Minimax-Q algorithm (thin line), the Minimax-QS algorithm with a linearly decreasing spreading function (bold line), and the Minimax-QS algorithm using a constant spreading function (dashed line). The spreading values used are shown in Figure 2(a).

The results (Figures 3 and 4) show an effective increase in the goal balance of the learning player in the beginning of the experiments in which a spreading function was used (Minimax-QS). After 50 matches, Minimax-QS presents a positive goal balance of 5, whilst Minimax-Q presents a positive goal balance of only 2.5. However, in the long term the positive goal balance against a random opponent is not very impressive if a constant spreading function is used. This could mean that, from a certain stage, the spreading mechanism can raise difficulties for the learning process since it can spread information to states that, in fact, should not be considered similar. On the other hand, if the spreading mechanism stops acting after a few matches — about 100 matches in the experiments reported here — Minimax-QS converges to Minimax-Q values. The theoretical proof presented in this paper corroborates this evidence, since it shows that if the spreading mechanism vanishes at least as quickly as the learning rate, then Minimax-QS in fact converges to optimal values.

Results were particularly interesting when a Minimax-Q learner opponent was considered (Figure 5). The Minimax-QS learners — for fixed and variable spreading functions — present significant improvement over the Minimax-Q opponent (positive goal balance of 4 goals around the $50th$. match) in the short term. As expected, in the long term Minimax-QS policies for decreasing spreading converge to the learned Minimax-Q values (zero goal balance), confirming experimentally the optimal convergence property demonstrated in section 4.

6. CONCLUSION

In this paper we have contributed a Minimax-QS algorithm, in which a spreading function is used to improve online learning time of control policies in multi-agent systems. This algorithm enhances Minimax-Q by embedding a priori knowledge in the spreading function, while at the same time keeping the convergence properties of the latter.

We have conducted empirical evaluations of Minimax-QS in a simplified simulator for the soccer domain. The results confirm the usefulness of the spreading function for learning purposes. The positive contributions of the spreading function can be mostly evidentiated in the beginning of the learning process. Even when using a very simple domain-dependent spreading function, the performance of the learning algorithm could be significantly improved. It should be stressed, however, that a wrong choice of state similarity function can significantly degrade performance of Minimax-QS. Fortunately, in many games similarities inherently exist in the environment, making it easy to design simple and useful state similarity functions [5].

We plan to continue investigating the use of experience generalization in hybrid RL algorithms. More specifically, lines of research worth pursuing include evaluation in real domains, integration with other convergence speeding up techniques, and studies on adaptation of the spreading function to the task domain.

7. ACKNOWLEDGMENTS

This research was conducted under the NSF/CNPq-ProTeM CC Project MAPPEL (Grant no. 68003399-8). Carlos H. C. Ribeiro is grateful to FAPESP (proc. 99/05777-2) and CNPq (Grant no. 301228/97-3-NV).

8. REFERENCES

[1] B. Banerjee, S. Sen, and J. Peng. Fast concurrent reinforcement learners. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'2001)*, pages 825–830, 2001.

[2] S. Karlin and H. M. Taylor. *A First Course in Stochastic Processes*. Academic Press, 1975.

[3] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML'94)*, pages 157–163. Morgan Kaufmann, 1994.

[4] M. L. Littman and C. Szepesvári. A generalized reinforcement learning model: Convergence and applications. In *Procs. of the Thirteenth International Conf. on Machine Learning (ICML'96)*, pages 310–318, 1996.

[5] R. Pegoraro. *Agilizando aprendizagem por reforco em robotica movel atraves do uso de conhecimento sobre o dominio*. PhD thesis, University of São Paulo, 2001.

[6] C. H. C. Ribeiro. Embedding a priori knowledge in reinforcement learning. *Journal of Intelligent and Robotic Systems*, 21(1):51–71, January 1998.

[7] C. H. C. Ribeiro and C. Szepesvári. Q-Learning combined with spreading: Convergence and results. In *Procs. of the ISRF-IEE International Conf. on Intelligent and Cognitive Systems (Neural Networks Symposium)*, pages 32–36, 1996.

[8] S. P. Singh, T. Jaakkola, and M. I. Jordan. Reinforcement learning with soft state aggregation. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 361–368. MIT Press, 1995.

[9] C. Szepesvári and M. Littman. Generalized markov decision processes: Dynamic-programming and reinforcement-learning algorithms. Technical Report CS-96-11, Brown University, Department of Computer Science, 1996.

[10] G. Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–67, 1995.

[11] J. N. Tsitsiklis and B. V. Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, May 1997.