On-Policy Concurrent Reinforcement Learning

Bikramjit Banerjee
EECS Dept., Tulane University
New Orleans, LA 70118, USA
banerjee@eecs.tulane.edu

Sandip Sen
MCS Dept., University of Tulsa
Tulsa, OK 74104, USA
sandip@kolkata.mcs.utulsa.edu

Jing Peng
EECS Dept., Tulane University
New Orleans, LA 70118, USA
jp@eecs.tulane.edu

Abstract

When an agent learns in a multiagent environment, the payoff it receives is dependent on the behavior of the other agents. If the other agents are also learning, its reward distribution becomes non-stationary. This makes learning in multiagent systems more difficult than single-agent learning. Prior attempts at value-function based learning in such domains have used off-policy Q-learning that do not scale well as the cornerstone, with restricted success. We study on-policy modifications of such algorithms, with the promise of scalability and efficiency. In particular we prove that these hybrid techniques are guaranteed to converge to their desired fixed points under some restrictions. We also show experimentally, that the new techniques can learn (from self-play) better policies than the previous algorithms (also in self-play) during some phases of the exploration.

1 Introduction

The reinforcement learning (RL) paradigm provides techniques using which an individual agent can optimize its environmental payoff. However, the presence of a non-stationary environment, central to multi-agent learning (MAL), violates the assumptions underlying convergence proofs of single agent RL techniques. As a result, standard reinforcement learning techniques like Q-learning [Watkins1989] are not guaranteed to converge in a multi-agent environment. The focus of convergence for multiple, concurrent learners is on an equilibrium strategy-profile rather than optimal strategies for the individual agents.

The stochastic-game (or Markov Games) framework, a generalization of Markov Decision Processes for multiple controllers, has been used to model learning by agents in purely competitive domains [Littman1994, Sandholm and Crites1996, Hu and Wellman1998, Bowling and Veloso2002]. In [Littman1994] the author has presented a minimax-Q learning algorithm and evaluated its performance experimentally. Hu and Wellman (1998) (along with further research in [Bowling2000, Littman2001]) extended Littman's framework to enable the agents to converge to a mixed-strategy Nash equilibrium in general-sum games. The Q-learning algorithm, being a model-free and online learning technique, is particularly suited for multi-stage games, and as such, an attractive candidate for multiagent-learning in uncertain environments. However, one major problem with Q-learning is scalability. The table based representation proves intractable in large problems. In multi-agent environments in particular, this is even more pronounced since the state-action space grows exponentially in the number of agents. At the same time, Q-learning with function approximation has been shown to diverge [Baird1995], while SARSA(0), an on-policy version of Q-learning [Rummery1994, Sutton and Barto1998], with function approximation has been demonstrated to converge to a bounded region at worst [Gordon2000]. Consequently, SARSA(0) looks attractive for hybridising with previous Q-learning based MAL algorithms. In this paper we present SARSA(0) versions of concurrent Q-learning for competitive as well as general-sum domains, and prove that they can converge to respectively minimax (reported earlier in [Banerjee *et al.*2001]) and Nash equilibrium value-functions in the limit, under appropriate assumptions. We also show experimentally that the new method can not only learn better policies in competitive domains, but can also learn faster in general-sum domains.

The rest of the paper is organized as follows. Section 2 discusses the existing work in multiagent Q-learning which we build upon. Section 3 presents the on-policy hybridisation of these algorithms and proves their convergence to desired fixed points under appropriate assumptions. Section 4 presents another hybrid technique for experimental comparison with the other hybrids. Sections 5 and 6 present two experimental domains - soccer and tightly coupled navigation - with corresponding results and discussions. Finally our conclusions and a glimpse into our ongoing and future research in this area are outlined in section 6.

2 Multiagent Q-learning

First we present some basic definitions relevant to our work.

Definition 1 *A Markov Decision Process (MDP) is a quadruple* $\{S, A, T, R\}$*, where* S *is the set of states,* A *is the set of actions,* T *is the transition function,* $T : S \times A \rightarrow PD(S)$*,* PD *being a probability distribution, and R is the reward function,* $R : S \times A \rightarrow \Re$.

A multiagent reinforcement-learning task can be modeled as a stochastic game which is an extension of an MDP, with S specifying the joint-state of the agents, A being the joint-actions of the agents, $A_1 \times A_2 \times \ldots A_n$ where A_i is the set of actions available to the ith agent, T as the joint-transition function, and the reward function is redefined as $R : S \times A \rightarrow \Re^n$. The functions T and

R are usually unknown. The goal of the ith agent is to find a strategy π_i that maximizes its expected sum of discounted rewards, given the others' strategies π_{-i}

$$v(s, \pi_i) = \sum_{t=0}^{\infty} \gamma^t E(r_t^i | \pi_i, \pi_{-i}, s_0 = s) \tag{1}$$

where s_0 is the initial joint-state, r_t^i is the reward of the ith agent at time t, and $\gamma \in [0, 1)$ is the discount factor

Definition 2 *A bimatrix game is given by a pair of matrices, (M_1, M_2), (each of size $|A_1| \times |A_2|$ for a two-agent game) where the payoff of the kth agent for the joint action (a_1, a_2) is given by the entry $M_k(a_1, a_2),\ \forall (a_1, a_2) \in A_1 \times A_2,\ k = 1, 2$.*

Each stage of a stochastic game for two agents (it can be extended to n agents using n-dimensional tables instead of matrices) can be looked upon as a bimatrix game. A zero-sum game is a special bimatrix game where $M_1(a_1, a_2) + M_2(a_1, a_2) = 0,\ \forall (a_1, a_2) \in A_1 \times A_2$.

Definition 3 *A mixed-strategy Nash Equilibrium for a bimatrix game (M_1, M_2) is a pair of probability vectors (π_1^*, π_2^*) such that*

$$\pi_1^{*T} M_1 \pi_2^* \geq \pi_1^T M_1 \pi_2^* \ \ \forall \pi_1 \in PD(A_1).$$

$$\pi_1^{*T} M_2 \pi_2^* \geq \pi_1^{*T} M_2 \pi_2 \ \ \forall \pi_2 \in PD(A_2).$$

where $PD(A_i)$ is the set of probability-distributions over the ith agent's action space.

No player in this game has any incentive for unilateral deviation from the Nash equilibrium strategy, given the other's strategy. There always exists at least one such equilibrium profile for an arbitrary finite bimatrix game [Nash1951].

An individual learner may, but need not, use a model of the environment to learn the transition and reward functions. Q-learning is one example of model-free learning. In greedy policy Q-learning, an agent starts with arbitrary initial $Q(s,a)$ (or action values) for each state-action pair (s,a), and repeatedly chooses actions, noting its rewards and transitions and updating Q as

$$Q^{t+1}(s_t,a_t) = (1-\alpha_t)Q^t(s_t,a_t) + \alpha_t[r_t + \gamma v^t(s_{t+1})]$$

where

$$v^t(s_{t+1}) = \max_a Q^t(s_{t+1},a). \tag{2}$$

and $\alpha_t \in [0,1)$ is the learning rate. Watkins and Dayan (1992) have proved that the above iteration converges to optimal action values under infinite sampling of each state-action pair and a particular schedule of the learning rate, given by

$$0 \leq \alpha_t(x) \leq 1,\ \sum_t \alpha_t(x) = \infty,\ \sum_t \alpha_t^2(x) < \infty.$$

In the case of multiagent learning, the above iteration would not work, since the maximization over one's action is insufficient in the presence of multiple actors. However, if the reward function of the opponent is negatively correlated, then actions can be selected by solving the bimatrix-game $(M(s), -M(s))$ greedily for the opponent, and pessimistically for oneself, to guarantee a minimum expected payoff. This produces Littman's minimax-Q algorithm for simultaneous-move zero-sum games, for which the value function for agent 1 is

$$v_1^t(s_{t+1}) = \max_{\pi \in PD(A)} \min_{o \in O} \pi^T Q_1^t(s_{t+1}, ., o), \tag{3}$$

where A and O are the action sets of the learning agent (agent 1) and its opponent respectively, and $Q_1^t(s_{t+1}, ., o)$ is a vector of action values of the learner corresponding to its opponent's action o. The current policy $\pi(s)$ can be solved by linear programming for the constrained, minimax optimization on $Q(s, ., .)$. The minimax-Q learning algorithm has been proved to converge to optimal action values [Szepesvári and Littman1999].

For general-sum games, however, the ith agent needs to know π_{-i}, in absence of which it has to model its opponents. In such games, each agent can observe the other agent's actions and rewards and maintains separate action values for each of them in addition to its own [Hu and Wellman1998].

The value function for agent 1 in this case is

$$v_1^t(s_{t+1}) = \pi_1^*(s_{t+1})^T Q_1^t(s_{t+1}, ., .)\pi_2^*(s_{t+1}), \tag{4}$$

where (π_1^*, π_2^*) are the Nash-strategies of the agents for the bimatrix game

$$\{Q_1^t(s_{t+1}, ., .), Q_2^t(s_{t+1}, ., .)\},$$

which can be solved by quadratic programming technique [Mangasarian and Stone1964]. In zero-sum games, the value-function in (4) simplifies to

$$v_1^t(s_{t+1}) = \max_{\pi_1 \in PD(A_1)} \min_{\pi_2 \in PD(A_2)} \pi_1^T Q_1^t(s_{t+1}, ., .)\pi_2. \tag{5}$$

This algorithm converges to a Nash equilibrium, for a restrictive class of Nash equilibria [Hu and Wellman1998, Bowling2000], though each agent can do so independently of the others. As a result, the agents are not guaranteed to converge to their portions of a single equilibrium (in case there are multiple such) even in self-play. We note in passing, that the two methods learn identical value-functions in purely competitive domains, though they may learn different policies.

3 On-policy concurrent Q-learning

The techniques discussed so far were all off-policy learning algorithms in that the update of $Q(s, a^1, a^2)$ depends on v that relies on actions that are not taken. Minimax-Q learning has been shown to converge to optimal minimax values in (5) [Szepesvári and Littman1999]. The SARSA(0) algorithm is, on the other hand, an on-policy learning method that depends heavily on the actual learning policy followed [Rummery1994, Sutton and Barto1998]. In general, off-policy algorithms can separate control from exploration while on-policy reinforcement learning algorithms cannot. Despite this, on-policy algorithms with function approximation in single agent learning appear to be superior to off-policy algorithms in control as well as prediction problems [Baird1995, Boyan and Moore1995, Gordon2000, Sutton1996, Tsitsiklis and Roy1997]. In particular, it is shown that the SARSA(0) algorithm with function approximation converges at worst to a bounded region [Gordon2000]. In contrast, Q-learning with function approximation has been shown to diverge [Baird1995]. On-policy algorithms can learn to behave consistently with exploration [Sutton and Barto1998]. Often off-policy algorithms compute a policy that they do not follow. Moreover, on-policy algorithms are more natural to combine with eligibility traces than off-policy algorithms are. This raises the following question that this research effort endeavors to answer: Can on-policy RL algorithms perform equally well in multiagent domains? In that case a hybrid of the SARSA technique and Q-learning algorithms (viz. minimax-SARSA and Nash-SARSA) might provide convergence with generalizability for larger and otherwise intractable problems. Below, we provide theoretical proofs of convergence of the two algorithms under duly stated assumptions.

3.1 Minimax-SARSA learning

In a simple Q-learning scenario, the SARSA technique modifies the update rule (2) as $v^t(s_{t+1}) = Q^t(s_{t+1}, a_{t+1})$. Thus a SARSA algorithm learns the values of its own actions, and can converge to optimal values only if the learning policy chooses optimal actions in the limit. In a multiagent minimax-Q setting, the rule (3) would be replaced, for agent 1, by

$$v_1^t(s_{t+1}) = Q_1^t(s_{t+1}, a_{t+1}, o_{t+1}),$$

while the policy to choose actions would still be computed by the original minimax-Q algorithm. To achieve convergence of this algorithm to minimax-Q values, we follow an ϵ-minimax strategy that satisfies the need of infinite exploration while being minimax in the limit, i.e. ϵ decays to 0 in the limit. We call such exploration 'Minimax in the limit with infinite exploration' or MLIE. Our convergence result rests on the following lemma established by [Singh *et al.*2000].

Lemma 1 *Consider a stochastic process (α_t, Δ^t, F^t), $t \geq 0$, where $\alpha_t, \Delta^t, F^t : X \rightarrow \Re$ satisfy the equations*

$$\Delta^{t+1}(x) = (1 - \alpha_t(x))\Delta^t(x) + \alpha_t(x)F^t(x),$$

where $x \in X,\ t = 0, 1, 2, \ldots$. Let P_t be a sequence of increasing σ-fields such that α_0 and Δ^0 are P_0 measurable and $\alpha_t,\ \Delta^t$ and F^{t-1} are P_t measurable, $t = 1, 2, \ldots$. Assume that the following hold:

1. X is finite.

2. $0 \leq \alpha_t(x) \leq 1,\ \sum_t \alpha_t(x) = \infty,\ \sum_t \alpha_t^2(x) < \infty$ w.p.1.

3. $\|E\{F^t(.)|P_t\}\|_W \leq \delta\|\Delta^t\|_W + c_t$, where $\delta \in [0, 1)$ and c_t converges to 0 w.p.1. and $\|.\|_W$ is a max-norm for a uniform weight-vector, W.

4. $Var\{F^t(x)|P_t\} \leq \beta(1+\|\Delta^t\|_W)^2$*, for some constant* β*.*

Then, Δ^t *converges to 0 with probability 1 (w.p.1).*

The update rule for SARSA for agent 1 say, is

$$\begin{aligned} Q_1^{t+1}(s_t,a_t,o_t) &= (1-\alpha_t)Q_1^t(s_t,a_t,o_t) + \\ &\quad \alpha_t[r_t^1 + \gamma Q_1^t(s_{t+1},a_{t+1},o_{t+1})]. \end{aligned} \tag{6}$$

We also note that the fixed point of the minimax-Q algorithm [Szepesvári and Littman1999], for agent 1, is

$$\begin{aligned} Q_1^*(s_t,a_t,o_t) &= R_1(s_t,a_t,o_t) + \\ &\quad E_y[\max_{\pi_1}\min_o \pi_1^T Q_1^*(y,.,o)]. \end{aligned} \tag{7}$$

Now we state and prove the theorem for convergence of minimax-SARSA learning using Lemma 1.

Theorem 1 *The learning rule specified in (6) converges to the values in equation (7) with probability 1 provided* a_t *is chosen using an MLIE scheme at each step* t*, the immediate rewards are bounded and have finite variance, the action values are stored in lookup tables, the learning rate,* α_t*, satisfies condition 2 in Lemma 1, and the opponent plays greedily in the limit.*

Proof: Writing x in Lemma 1 as (s_t,a_t,o_t) and Δ^t as $Q_1^t(s_t,a_t,o_t) - Q_1^*(s_t,a_t,o_t)$, and defining $\alpha_t(s,a,o) = 0$ unless $(s,a,o) = (s_t,a_t,o_t)\ \forall t$, we have

$$\begin{aligned} \Delta^{t+1}(s_t,a_t,o_t) &= Q_1^{t+1}(s_t,a_t,o_t) - Q_1^*(s_t,a_t,o_t) \\ &= (1-\alpha_t)Q_1^t(s_t,a_t,o_t) + \alpha_t[r_t^1 + \gamma Q_1^t(s_{t+1},a_{t+1},o_{t+1})] - Q_1^*(s_t,a_t,o_t)(\text{from 6}) \\ &= (1-\alpha_t)[Q_1^t(s_t,a_t,o_t) - Q_1^*(s_t,a_t,o_t)] \end{aligned}$$

$$
\begin{aligned}
& +\alpha_t[r_t^1 + \gamma Q_1^t(s_{t+1}, a_{t+1}, o_{t+1}) - Q_1^*(s_t, a_t, o_t)] \\
= \quad & (1-\alpha_t)\Delta^t(s_t, a_t, o_t) + \alpha_t F^t(s_t, a_t, o_t) \qquad (8)
\end{aligned}
$$

where $F^t(s_t, a_t, o_t)$ can be further expanded as

$$
\begin{aligned}
F^t(s_t, a_t, o_t) \quad = \quad & r_t^1 + \gamma \max_{\pi_1} \min_{o} \pi_1^T Q_1^t(s_{t+1}, ., o) \\
& -Q_1^*(s_t, a_t, o_t) + \gamma Q_1^t(s_{t+1}, a_{t+1}, o_{t+1}) \\
& -\gamma \max_{\pi_1} \min_{o} \pi_1^T Q_1^t(s_{t+1}, ., o)], \qquad (9)
\end{aligned}
$$

which gives rise to

$$F^t(s_t, a_t, o_t) = F_M^t(s_t, a_t, o_t) + \gamma[d_t(s_t, a_t, o_t)].$$

where $F_M^t(s_t, a_t, o_t) = r_t^1 + \gamma \max_{\pi_1} \min_o \pi_1^T Q_1^t(s_{t+1}, ., o) - Q_1^*(s_t, a_t, o_t)$. Consequently, we have

$$\|E\{F^t(., ., .)|P_t\}\| \leq \|E\{F_M^t(., ., .)|P_t\}\| + \gamma\|E\{d_t(., ., .)|P_t\}\|$$

It can be shown that the measurability and variance conditions are satisfied and that

$$\|E\{F_M^t(., ., .)|P_t\}\| \leq \gamma_M \|\Delta^t\|$$

for some $\gamma_M \in [0, 1)$ (since minimax-Q operator is a contraction), according to the outline provided by [Singh *et al.*2000]. Therefore

$$\|E\{F^t(., ., .)|P_t\}\| \leq \gamma_M \|\Delta^t\| + \gamma\|E\{d_t(., ., .)|P_t\}\|$$

Thus far, the proof was identical to [Singh *et al.*2000]. The next part of the proof is crucial for establishing that $c_t = \gamma \| E\{d_t(.,.,.)|P_t\}\|$ vanishes in the limit, under MLIE exploration, and is a novelty. We consider the following cases:

Case 1: $\underline{Q_1^t(s_{t+1}, a_{t+1}, o_{t+1}) \geq \max_{\pi_1} \min_o \pi_1^T Q_1^t(s_{t+1}, ., o)}$. Since

$$\max_{\pi_1} \min_o \pi_1^T Q_1^t(s_{t+1}, ., o) \geq \min_o Q_1^t(s_{t+1}, a_{t+1}, o),$$

we have

$$d_t(s_t, a_t, o_t) = |d_t(s_t, a_t, o_t)| \leq Q_1^t(s_{t+1}, a_{t+1}, o_{t+1}) - \min_o Q_1^t(s_{t+1}, a_{t+1}, o)$$

and the corresponding expected value vanishes in the limit if the opponent plays greedily in the limit.

Case 2: $\underline{\max_{\pi_1} \min_o \pi_1^T Q_1^t(s_{t+1}, ., o) \geq Q_1^t(s_{t+1}, a_{t+1}, o_{t+1})}$. Again,

$$\max_{\pi_1} \min_o \pi_1^T Q_1^t(s_{t+1}, ., o) \leq \pi_1^{*T} Q_1^t(s_{t+1}, ., o_{t+1})$$

where $\pi_1^* = \arg\max_{\pi_1} \min_o \pi_1^T Q_1^t(s_{t+1}, ., o)$. Hence

$$-d_t(s_t, a_t, o_t) = |d_t(s_t, a_t, o_t)| \leq \pi_1^{*T} Q_1^t(s_{t+1}, ., o_{t+1}) - Q_1^t(s_{t+1}, a_{t+1}, o_{t+1}).$$

The associated expected value vanishes again in the limit due to the assumption of an MLIE policy on part of agent 1, and independent of the opponent's behavior.

Let $C_t(s_t, a_t, o_t)$ be the maximum of the two upper limits on $|d_t(s_t, a_t, o_t)|$ established above. We see that

$$E\{|d_t(s_t, a_t, o_t)|\} \leq E\{C_t(s_t, a_t, o_t)\}$$

and the right hand side vanishes for each state-action tuple. Hence, $E\{|d_t(s_t, a_t, o_t)|\}$ vanishes for each state-action tuple, which implies that $\|E\{d_t(.,.,.)|P_t\}\|$ vanishes in the limit under MLIE exploration and greedy play by the opponent in the limit. Setting c_t in Lemma 1 to $\gamma\|E\{d_t(.,.,.)|P_t\}\|$, we conclude that minimax-SARSA algorithm converges to the minimax-Q values under MLIE exploration with probability 1, if the opponent plays greedily in the limit, and under appropriate structure of α_t. [Q.E.D.]

Note that **Case 1** needs the boundedness of Q_1^t that can be established along the same lines as in [Singh *et al.*2000]. We could imagine a minimax-Q learning process that updates the same state-action values as the SARSA process and in the same order, thus making the former action values upper bounds for the corresponding values in the latter. Similarly a minmax update rule in place of a maxmin rule would establish the corresponding lower bounds and since both are contractions, the convergence of the baseline maxmin (minmax) processes are guaranteed. It might be argued that the condition of greedy play by the opponent in the limit is restrictive. However, this is typical of a convergence proof of on-policy algorithms that requires more details of the actions taken by the agents. Gains from on-policy algorithms in terms of learning efficiency and cost offset the condition of greedy play in the limit. We also note that the condition of of greedy play makes sense in competitive games only. In general-sum domains the opponent playing a minimizing strategy is not rationally justifiable, hence the convergence proof is not useful in such cases.

3.2 Nash-SARSA learning

The extension of the SARSA method to Nash learning in general-sum domains is trivial. We only note that the desired fixed point here (counterpart of equation 7) is

$$\begin{aligned} Q_1^*(s_t, a_t, o_t) &= R_1(s_t, a_t, o_t) + \\ &\quad E_y[\pi_1^{*T} Q_1^*(y, ., .)\pi_2^*]. \end{aligned} \tag{10}$$

where (π_1^*, π_2^*) are some Nash pair for the game $\{Q_1^*, Q_2^*\}$. The same strategy as in minimax-SARSA, can be used to prove the convergence of Nash-SARSA, but the restrictions introduced to the nature of the games in the process, limits the applicability of the result. To demonstrate this, we focus on the two cases as above (since the rest of the proof is identical, with the observation that Nash operator is a contraction according to [Hu and Wellman1998, Bowling2000], under certain assumptions, which are therefore necessary in our proof as well).

Case 1: $\underline{Q_1^t(s_{t+1}, a_{t+1}, o_{t+1}) \geq \pi_1^{*T} Q_1^t(s_{t+1}, ., .)\pi_2^*}$, where (π_1^*, π_2^*) are some Nash pair for the game $\{Q_1^t, Q_2^t\}$. Since by definition of a Nash equilibrium,

$$\pi_1^{*T} Q_1^t(s_{t+1}, ., .)\pi_2^* \geq Q_1^t(s_{t+1}, a_{t+1}, .)\pi_2^*,$$

we have

$$d_t(s_t, a_t, o_t) = |d_t(s_t, a_t, o_t)| \leq Q_1^t(s_{t+1}, a_{t+1}, o_{t+1}) - Q_1^t(s_{t+1}, a_{t+1}, .)\pi_2^*$$

and the corresponding expected value vanishes in the limit if the opponent plays its portion of the same Nash equilibrium in the limit.

Case 2: $\underline{\pi_1^{*T} Q_1^t(s_{t+1}, ., .)\pi_2^* \geq Q_1^t(s_{t+1}, a_{t+1}, o_{t+1})}$. Now, if the game ($\forall t$) is such that any deviation by the opponent from its portion of the Nash equilibrium strategy <u>increases</u> the payoff of the learner, then

$$\pi_1^{*T} Q_1^t(s_{t+1}, ., .)\pi_2^* \leq \pi_1^{*T} Q_1^t(s_{t+1}, ., o_{t+1})$$

since o_{t+1} is a deviation by the opponent from its Nash equilibrium strategy. This condition is similar to [Hu and Wellman1998], but they (and later in [Bowling2000]) used more restrictions in conjunction or disjunction with this. We also use one more restriction that the opponent must play the <u>same Nash equilibrium strategy</u> in the limit. Rather than being a further impediment, this actually allows the players to converge to the same equilibrium in self-play. It is also justified in self-play since the players are using the same algorithm. Now we have,

$$-d_t(s_t, a_t, o_t) = |d_t(s_t, a_t, o_t)| \leq \pi_1^{*T} Q_1^t(s_{t+1}, ., o_{t+1}) - Q_1^t(s_{t+1}, a_{t+1}, o_{t+1}).$$

The associated expected value vanishes again in the limit due to the assumption of an NELIE (Nash equilibrium in the limit with infinite exploration) policy on part of agent 1, and independent of the opponent's behavior.

Thus Nash-SARSA converges to a Nash equilibrium under the same assumptions as in [Hu and Wellman1998, Bowling2000] and an additional assumption that the players follow NELIE strategy which is trivially true in self-play.

4 Minimax-Q(λ)

Since Q-learning updates only one action value at a time in a typical lookup table representation, it is slow in learning the action values. A well-known technique to speed up single agent Q-learning is to integrate it with the TD(λ) estimators for estimating the action values, resulting in the multi-step Q(λ)-learning algorithm [Peng and Williams1996]. Here the parameter λ in Q(λ) represents the degree of bootstrapping (i.e. Q(λ)-learning recursively builds its estimates upon themselves), with $\lambda = 0$, as in Q or minimax-Q learning, representing the most extreme form of bootstrapping, and $\lambda = 1$, as in Monte Carlo methods, representing no bootstrapping (i.e. actual returns are used as their basis for building other estimates). While there is a range of results that suggest that Q(λ) learning is generally more effective than simple Q-learning in single agent domains, we are interested to know if similar performance characteristics can be observed in multi-agent environments, and in particular its comparison with the on-policy methods. We evaluate the minimax-Q(λ) algorithm primarily for interesting experimental comparisons with the algorithms presented earlier in the paper.

We note that for experimentation, we have used a computationally more efficient version of the Peng-Williams' algorithm, where the action value updates are 'lazily' postponed until necessary [Wiering and Schmidhuber1998]. Q(λ) can be applied to each of the two learning schemes in section 2, by defining $v(s_{t+1})$ in the Q(λ) learning algorithm by the equation in (3) or (4) as the case may be. The guarantee of convergence, however, may no longer hold.

5 Experiments in a competitive domain

To evaluate the proposed schemes, we used the purely competitive soccer domain [Littman1994]. It is a 4×5 grid containing two agents, A and B, as shown in figure 1, that always occupy distinct squares. The goal of agent A is on the left, and that of B on right. The figure 1 shows the initial positions of the agents, with the ball being given to an agent at random at the start of each game (agent B in figure). Each agent can choose from a fixed set of five actions at each state: going up, left, down or right, or staying where it is.

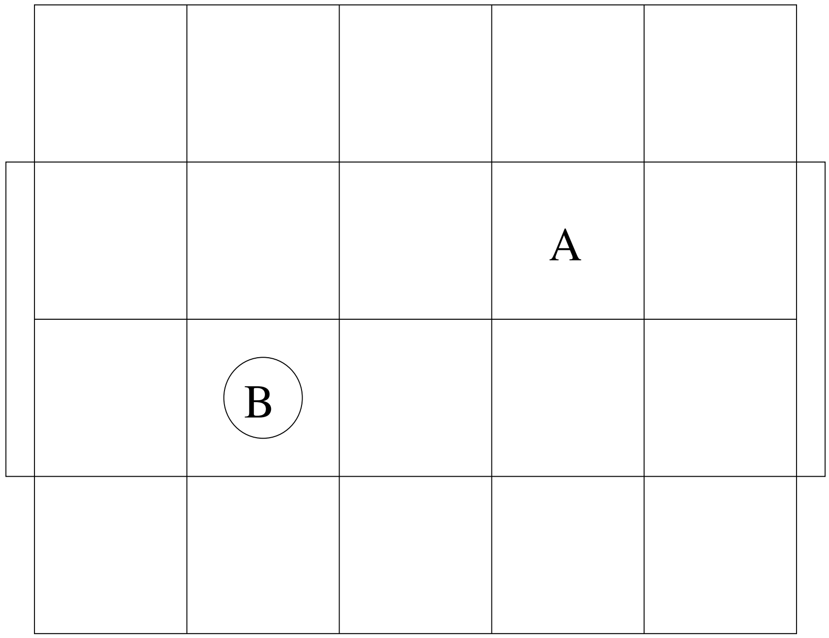

Figure 1: The experimental soccer domain.

When both the agents have selected their actions, these actions are executed in a random order. If an agent bumps onto another, the stationary agent receives the ball, and the movement fails. An agent receives reinforcements of +1 for a goal scored or a self-goal by the opponent and -1 for a self-goal or a goal scored by the opponent. This payoff scheme maintains the zero-sum character of the game. In all other cases the reinforcement received by an agent is zero. Whenever a non-zero reward is received, the game resets to the initial configuration.

We refer to an agent following Littman's minimax-Q algorithm as an M-agent. In the training phase of the experiments, we performed symmetric training between two ordinary M-agents, two M-agents both using the $Q(\lambda)$ algorithm, and two M-agents both using the SARSA algorithm. The respective policies learnt, are denoted as $MM_i,\ \lambda MM_i,\ sMM_i$, which are recorded at the end of each $i \times 10000$ iterations. Each training lasted 100,000 iterations. We used identical exploration-probabilities as that by Littman (1994) and the decay-factor for the learning-rate was set to 0.999954.

In the test phase, we allowed an sMM_i policy to play against an MM_i policy, for $i = 1 \ldots 10$.

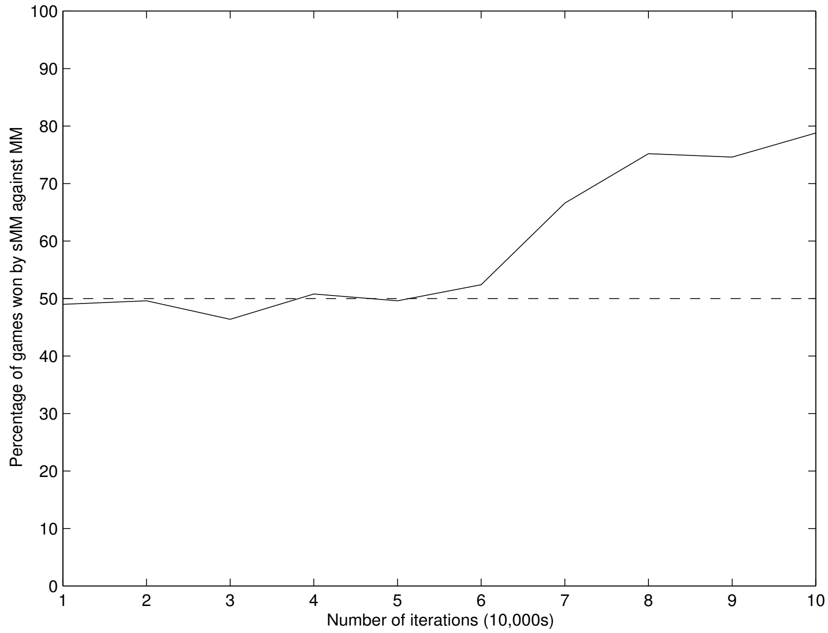

Figure 2: Games are played by sMM_i against MM_i for various values of i (horizontal axis). The percentages of wins (vertical axis) by the former for various i are plotted (averaged over 10 runs).

Each test iteration results in a draw with a probability of 0.05, to break possible deadlocks. 100,000 such iterations were conducted in each run and the resultant percentages of win by the sMM_i policies over its opponent averaged over 10 runs are reported in figure 2. The approximate trend suggests that ordinary minimax-Q initially dominates but minimax-SARSA gradually catches up and outperforms the former. In figure 3, the corresponding results from playing λMM_i against MM_i are shown. In this case the minimax-Q(λ) algorithm outperforms the ordinary minimax-Q algorithm from the very beginning. However, the λMM policies gradually lose their edge as the ordinary minimax-Q algorithm learns better progressively. The figure 4 corroborates these observations, as λMM_i performs well against sMM_i, but this performance decays with increasing i. λ was set to 0.7 in both experiments.

We note that minimax-Q(λ) learns better policies than ordinary minimax, early on. But surprisingly, minimax-SARSA also learns better policies than the latter, during the later part of the experimental phase. We also stress that the results reported are far from convergence, at which all the algorithms

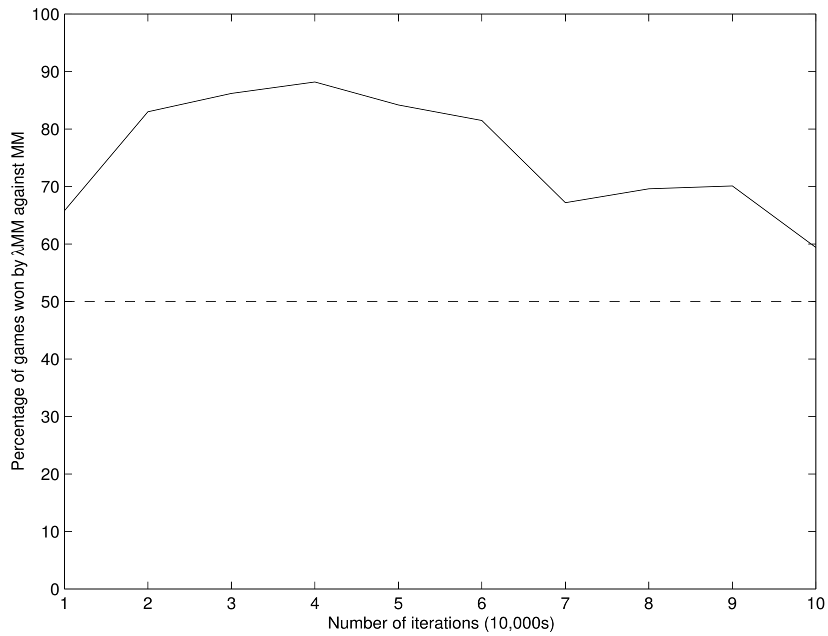

Figure 3: Games are played by λMM_i against MM_i for various values of i (horizontal axis). The percentages of wins (vertical axis) by the former for various i are plotted (averaged over 10 runs).

should perform equally well.

The reason why sMM beats MM can be understood in the context of Q updates. While sMM uses the actual action value from the next state to update the current state, MM still uses the minimax value from the next state, which postpones relying on the individual table-entries. For example, consider the state transition from S_1 to S_2 in figure 5. The bold arrows mark the actions chosen by the two agents. Then the update for MM will be

$$Q(S_1, a_{12}, o_{13}) = (1 - \alpha_t)Q(S_1, a_{12}, o_{13}) + \alpha_t[r + \gamma * 0]$$

since the minimax value of S_2 is 0. However, the update for SARSA is

$$Q(S_1, a_{12}, o_{13}) = (1 - \alpha_t)Q(S_1, a_{12}, o_{13}) + \alpha_t[r + \gamma * 10].$$

The zero values in table S_2 may be due to insufficient exploration or may be their true values. Hence,

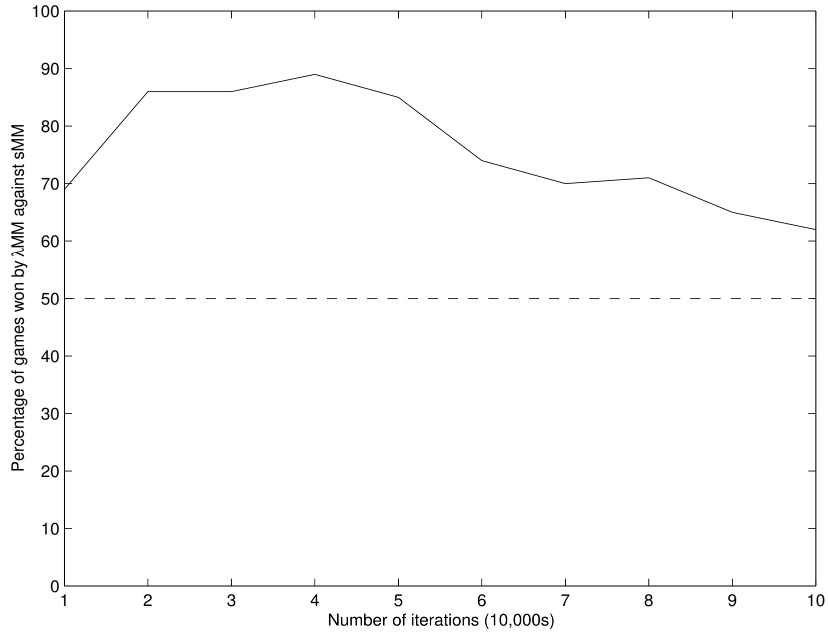

Figure 4: Games are played by λMM_i against sMM_i for various values of i. The percentages of wins (averaged over 10 runs) by the former for various i are plotted.

the information that there is something interesting in S_2 is backpropagated more expeditiously in sMM than in MM, but this happens after an initial lull when both MM and sMM are in the exploratory state and are equally ignorant of the domain. Later with more exploration, we expect MM to catch up (in figure 2) as learning continues. It is noteworthy that the same argument is inapplicable in ordinary Q-learning versus SARSA for a single agent learning scenario. Consider the single columns (o_{13} for S_1 and o_{22} for S_2) in figure 5. The value 10 will be used in the Q-update for both the algorithms, assuming same actions are selected in both cases. Hence, the speedup achieved by sMM over MM is crucially dependent on the pessimistic character of minimax strategy itself. We might not expect a comparable speedup in Nash-SARSA, and experimental verification of this intuition would be interesting.

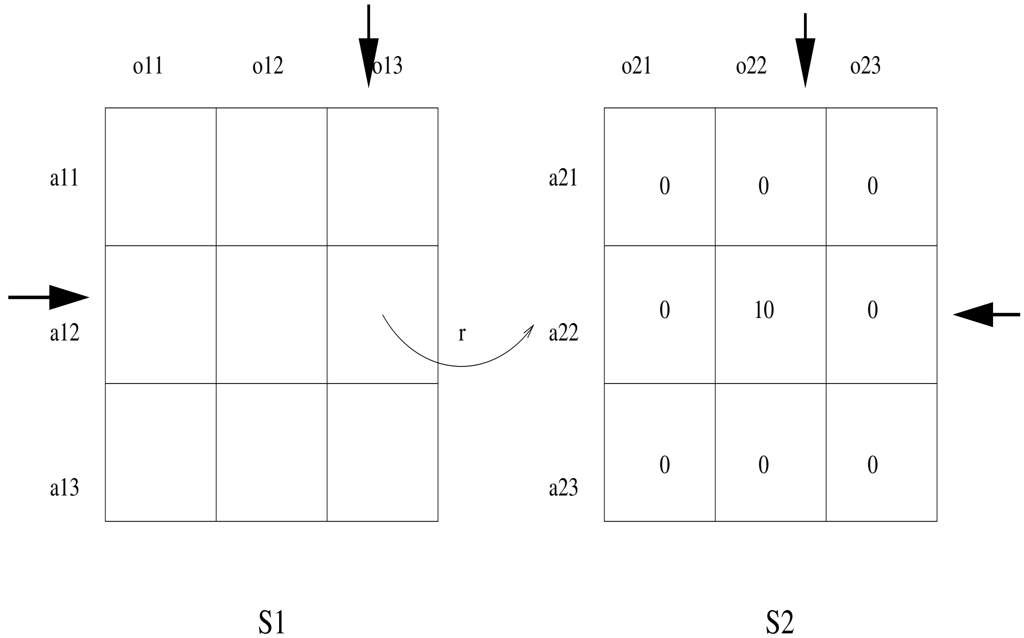

Figure 5: A sample state transition showing action-selection, reward and action values for the row agent.

6 Experiments in a general-sum domain

We note that the minimax-Q algorithm is applicable in general-sum domains as well, where the rationale of the assumption of minimizing policy of the opponent is to guarantee a minimum security level to the learner, instead of maximizing the reward of the opponent itself as in the zero-sum interpretation. The SARSA and $Q(\lambda)$ versions will still work in such domains. For the purpose of experimentation, we introduce a general-sum problem that we call "tightly coupled navigation."

This problem is a 4×3 grid world as shown in figure 6. The values in the lower left corner of each cell in figure 6 is the reward to agent 1 for reaching the state corresponding to that cell. Similarly the values in the upper right corner are those for agent 2. The rewards in this domain are state-based, i.e. the reward corresponding to a cell is received if the agents reach or remain in that cell. Here the agents are tightly coupled as they must always occupy the same cell and the next position and the payoff received by the agents are determined by both of their actions in every step. Contrast this with

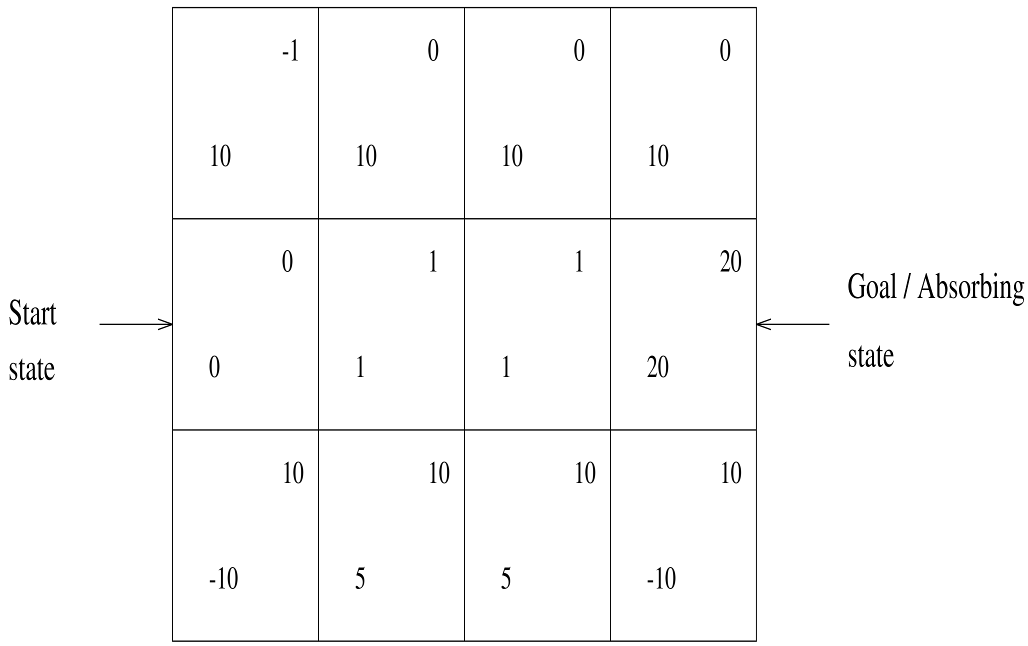

Figure 6: The tightly coupled navigation domain.

the “soccer game” domain, which is less tightly coupled since the position and the payoff received by an agent is dependent on the other agent’s action only when agents are in close proximity. Here each agent has three available actions in each state, viz. up, down, right. However, since they are coupled, they can move only when they choose the same action; otherwise they remain in the same state. The starting and the absorbing states have been shown in the figure 6. When the agents reach the goal state, each receives the reward 20 and without making any update in this iteration, the game restarts with the agents reshifted to the start-state and updates begin once again.

A realistic scenario for this domain is two men carrying a piece of heavy furniture. The furniture moves in a given direction if both the agents move in that direction; otherwise the furniture does not move (falls off from their hands). It should be assumed that they are not coordinating their moves by explicit communication, but are only observing the moves and the subsequent situation of the other. There may be different paths that the agents wish to follow to reach their common goal. However, since they are tightly coupled, they must strike a compromise and find an intermediate path that both can be maximally satisfied with, given the coupling.

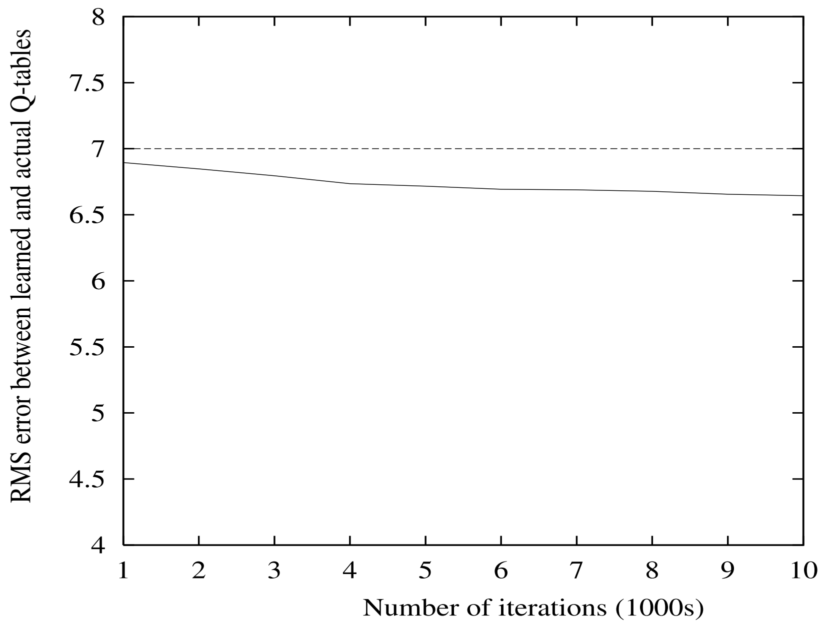

Figure 7: Mean RMS deviation plots of minimax-SARSA (solid) and ordinary minimax-Q for probability of reward-generation = 0.

We have symmetrically trained two minimax-Q and two minimax-SARSA agents in this domain. The exploration probabilities for the agents in each iteration was 0.2, the same as in the soccer domain. We varied the probability of reward-generation in each iteration from 0 to 0.5 to 1.0, where 0 stands for the case where rewards are generated only when the agents reach the goal state and a probability of 1.0 stands for a reward generated at each step. We wanted to study the effect of infrequent rewards, which is a realistic scenario in most practical domains, on the convergence of our algorithms. We expected the convergence rates to fall with more and more infrequent rewards. In order to study the convergence, the exact minimax action values were computed off-line and an average RMS deviation of the learned action values every 1000 training-iterations were plotted. The trainings lasted a total of 10,000 iterations.

From figures 7, 8 and 9, we can see that the minimax-SARSA algorithm always approaches minimax values faster than the ordinary minimax-Q algorithm. The errors in all the cases decrease

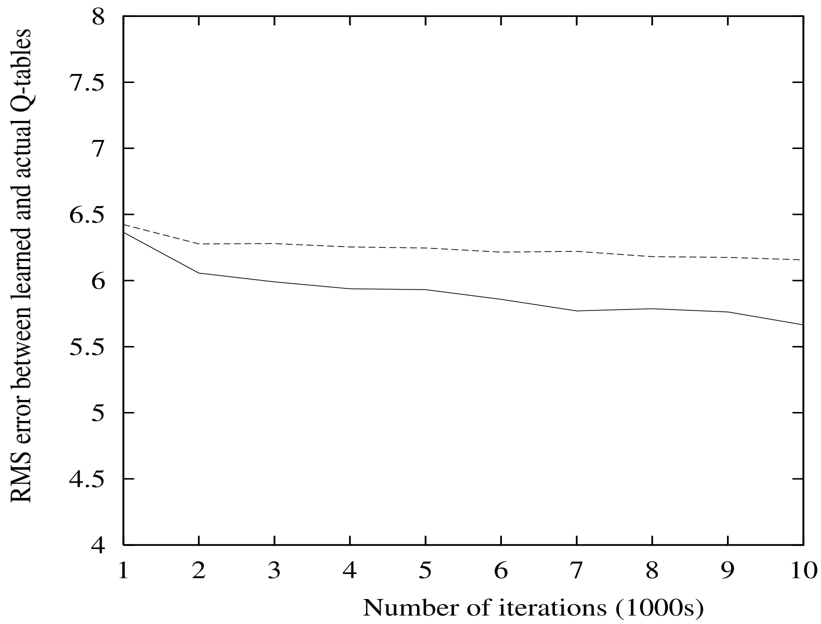

Figure 8: Mean RMS deviation plots of minimax-SARSA (solid) and ordinary minimax-Q for probability of reward-generation = 0.5.

monotonically which suggests that both the algorithms will eventually converge. As expected, the error-levels fall with increasing probability of reward-generation.

7 Conclusion and future work

We conclude that both the SARSA and $Q(\lambda)$ versions of minimax-Q learn better policies early on, than Littman's minimax-Q algorithm, and more so for the $Q(\lambda)$ algorithm. Though this latter algorithm works well, we are not aware of the theoretical convergence properties of this method. Exploring these properties is one open area. We also note that a combination of minimax-SARSA and $Q(\lambda)$ to form what could be called minimax-SARSA(λ), would probably be more efficient than either of the two, by naturally combining their disjoint areas of expediency, seen in the plots in figures 2 and 3. We plan to conduct more experiments with all these hybrid algorithms.

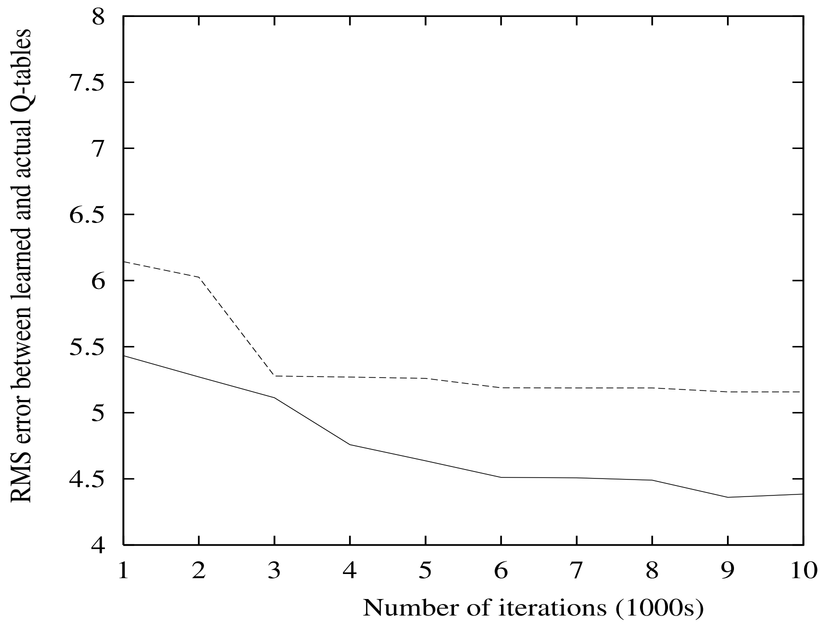

Figure 9: Mean RMS deviation plots of minimax-SARSA (solid) and ordinary minimax-Q for probability of reward-generation = 1.0.

References

[Baird1995] L.C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 30 – 37, San Fransisco, California, 1995. Morgan Kaufmann.

[Banerjee *et al.*2001] B. Banerjee, S. Sen, and J. Peng. Fast concurrent reinforcement learners. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, Seattle, WA, 2001.

[Bowling and Veloso2002] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215 – 250, 2002.

[Bowling2000] Michael Bowling. Convergence Problems of General-sum Multiagent Reinforcement Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 89 – 94, San Fransisco, California, 2000. Morgan Kaufmann.

[Boyan and Moore1995] J.A. Boyan and A.W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems 7*, pages 369–376, 1995.

[Gordon2000] G. J. Gordon. Reinforcement learning with function approximation converges to a region. In *Neural Information Processing Systems*. MIT Press, 2000.

[Hu and Wellman1998] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proc. of the 15th Int. Conf. on Machine Learning (ML'98)*, pages 242–250, San Francisco, CA, 1998. Morgan Kaufmann.

[Littman1994] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. of the 11th Int. Conf. on Machine Learning*, pages 157–163, San Mateo, CA, 1994. Morgan Kaufmann.

[Littman2001] M. L. Littman. Friend-or-foe Q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learnig*, Williams College, MA, USA, 2001.

[Mangasarian and Stone1964] O. L. Mangasarian and H. Stone. Two-person nonzero-sum games and quadratic programming. *Journal of Mathematical Analysis and Applications*, 9:348 – 355, 1964.

[Nash1951] John F. Nash. Non-cooperative games. *Annals of Mathematics*, 54:286 – 295, 1951.

[Peng and Williams1996] J. Peng and R. Williams. Incremental multi-step Q-learning. *Machine Learning*, 22:283 – 290, 1996.

[Rummery1994] G. A. Rummery. *Problem solving with reinforcement learning*. PhD thesis, Cambridge University Engineering Department, 1994.

[Sandholm and Crites1996] T. Sandholm and R. Crites. On multiagent Q-learning in a semi-competitive domain. In G. Weiß and S. Sen, editors, *Adaptation and Learning in Multi-Agent Systems*, pages 191–205. Springer-Verlag, 1996.

[Singh *et al.*2000] Satinder Singh, Tommi Jaakkola, Michael L. Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 39:287–308, 2000.

[Sutton and Barto1998] R. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[Sutton1996] R. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, 1996.

[Szepesvári and Littman1999] Csaba Szepesvári and M.L. Littman. A unified analysis of value-function-based reinforcement-learning algorithms. *Neural Computation*, 11(8):2017 – 2059, 1999.

[Tsitsiklis and Roy1997] J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 1997.

[Watkins1989] C.J.C.H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge University, 1989.

[Wiering and Schmidhuber1998] M. Wiering and J. Schmidhuber. Fast online Q(λ). *Machine Learning, 33(1)*, pages 105–116, 1998.