Journal of Machine Learning Research ? (????) ??-?? Submitted ?/??; Published ?/??

# R-MAX – A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning *

**Ronen I. Brafman** BRAFMAN@CS.BGU.AC.IL
*Computer Science Department*
*Ben-Gurion University*
*Beer-Sheva, Israel 84105*

**Moshe Tennenholtz** MOSHE@ROBOTICS.STANFORD.EDU
*Computer Science Department*
*Stanford University*
*Stanford, CA 94305*

**Editor:** Leslie Pack Kaelbling

## Abstract

R-MAX is a very simple model-based reinforcement learning algorithm which can attain near-optimal average reward in polynomial time. In R-MAX, the agent always maintains a complete, but possibly inaccurate model of its environment and acts based on the optimal policy derived from this model. The model is initialized in an optimistic fashion: all actions in all states return the maximal possible reward (hence the name). During execution, it is updated based on the agent's observations. R-MAX improves upon several previous algorithms: (1) It is simpler and more general than Kearns and Singh's $E^3$ algorithm, covering zero-sum stochastic games. (2) It has a built-in mechanism for resolving the exploration vs. exploitation dilemma. (3) It formally justifies the "optimism under uncertainty" bias used in many RL algorithms. (4) It is simpler, more general, and more efficient than Brafman and Tennenholtz's LSG algorithm for learning in single controller stochastic games. (5) It generalizes the algorithm by Monderer and Tennenholtz for learning in repeated games. (6) It is the only algorithm for learning in repeated games, to date, which is provably efficient, considerably improving and simplifying previous algorithms by Banos and by Megiddo.

**Keywords:** Reinforcement Learning, Learning in Games, Stochastic Games, Markov Decision Processes, Provably Efficient Learning

## 1. Introduction

Reinforcement learning has attracted the attention of researchers in AI and related fields for quite some time. Many reinforcement learning algorithms exist and for some of them convergence rates are known. However, Kearns and Singh's $E^3$ algorithm (Kearns and Singh, 1998) was the first provably near-optimal polynomial time algorithm for learning in Markov decision processes (MDPs). $E^3$ was extended later to handle single controller stochastic games (SCSGs) (Brafman and Tennenholtz, 2000) as well as structured MDPs (Kearns and Koller, 1999). In $E^3$ the agent learns by updating a model of its environment

*. The second author's permanent address is: Faculty of Industrial Engineering and Management, Technion–Israel Institute of Technology, Haifa 32000, Israel.

using statistics it collects. This learning process continues as long as it can be done relatively efficiently. Once this is no longer the case, the agent uses its learned model to compute an optimal policy and follows it. The success of this approach rests on two important properties: the agent can determine online whether an efficient learning policy exists, and if such a policy does not exist, it is guaranteed that the optimal policy with respect to the learned model will be approximately optimal with respect to the real world.

The difficulty in generalizing $E^3$ to adverserial contexts, i.e., to different classes of games, stems from the adversary's ability to influence the probability of reaching different states. In a game, the agent does not control its adversary's choices, nor can it predict them with any accuracy. Therefore, it has difficulty predicting the outcome of its actions and whether or not they will lead to new information. Consequently, we believe that a straightforward generalization of $E^3$ to stochastic games is unlikely to be feasible – it would imply that at each point during the learning process, the agent can either choose an exploration policy or an exploitation policy. We believe that there are situations in which the agent cannot force this choice on its opponent.[1] In particular, during the earlier stages of the learning process when the agent is biased towards exploration, the adversary may be able to prevent learning from taking place by choosing its behavior appropriately.

To overcome this problem, we propose a different approach in which the choice between exploration and exploitation is implicit. Our agent will always attempt to optimize its behavior, albeit with respect to a fictitious model. Roughly speaking, this model assumes that the reward the agent obtains in any situation it is not too familiar with is its maximal possible reward, denoted by $R_{max}$. Often, optimal behavior with respect to this fictitious model results in exploration with respect to the real model, and thus, to learning. The major insight behind the R-max algorithm is that the optimal policy with respect to the agent's *fictitious* model has a very interesting and useful property with respect to the *real* model: it is always either optimal or it leads to efficient learning. In many situations, the agent will not know ahead of time whether its behavior will lead to optimizing behavior with respect to the real model or to learning – this depends on how the adversary will act. However, it knows that it will either optimize or learn efficiently.

Since there is only a polynomial number of parameters to learn, as long as learning is done efficiently we can ensure that the agent spends a polynomial number of steps exploring, and the rest of the time will be spent exploiting. Thus, the resulting algorithm may be said to use an *implicit* explore or exploit approach, as opposed to Kearns and Singh's *explicit* explore or exploit approach.

This learning algorithm, which we call R-max, is very simple to understand and to implement. The algorithm converges in polynomial-time to a near-optimal solution. Moreover, R-max is described in the context of zero-sum stochastic games, a model that is more general than Markov Decision Processes. As a consequence, R-max is more general and more efficient than a number of previous results. It generalizes the results of Kearns and Singh (1998) to adverserial contexts and to situations where the agent considers a stochastic model of the environment inappropriate, opting for a non-deterministic model instead. R-max can handle more classes of stochastic games than the LSG algorithm (Brafman and Tennenholtz, 2000). In addition, it attains a higher expected average reward than LSG.

1. Of course, the agent always has a perfect exploitation policy at its disposal – the optimal one – but it has no way of knowing it.

We note that, like these other algorithms, R-max is polynomial in the size of the *explicit* state-space model. (The details of the complexity parameters are discussed later.)

R-max also improves upon previous algorithms for learning in repeated games (Aumann and Maschler, 1995), such as those by Megiddo (1980) and Banos (1968). It is the only polynomial time algorithm for this class of games that we know of. It is also much simpler than these previous algorithms. Finally, R-max generalizes the results of Monderer and Tennenholtz (1997) to handle the general probabilistic maximin (safety level) decision criterion.

The approach taken by R-max is not new. It has been referred to as *the optimism in the face of uncertainty* heuristic, and was considered an ad-hoc, though useful, approach (e.g., see Section 2.2.1 in Kaelbling et al. (1996), where it appears under the heading "Ad-Hoc Techniques" and Section 2.7 in Sutton and Barto (1998) where this approach is called *optimistic initial values* and is referred to as a "simple trick that can be quite effective on stationary problems"). This optimistic bias has been used in a number of well-known reinforcement learning algorithms, e.g. Kaelbling's interval exploration method (Kaelbling, 1993), the exploration bonus in Dyna (Sutton, 1990), the curiosity-driven exploration of Schmidhuber (1991), and the exploration mechanism in prioritized sweeping (Moore and Atkenson, 1993). More recently, Tadepalli and Ok (1998) presented a reinforcement learning algorithm that works in the context of the undiscounted average-reward model used in this paper. In particular, one variant of their algorithm, called AH-learning, is very similar to R-max. However, as we noted above, none of this work provides theoretical justification for this very natural bias. Thus, an additional contribution of this paper is a formal justification for the *optimism under uncertainty* bias.

The paper is organized as follows: in Section 2 we define the learning problem more precisely and the relevant parameters. In Section 3 we describe the R-max algorithm. In Section 4 we prove that it yields near-optimal reward in polynomial time. We conclude in Section 5.

## 2. Preliminaries

We present R-max in the context of a model that is called a *stochastic game*. This model is more general than a Markov decision process because it does not necessarily assume that the environment acts stochastically (although it can). In what follows we define the basic model, describe the set of assumptions under which our algorithm operates, and define the parameters influencing its running time.

### 2.1 Stochastic Games

A game is a model of multi-agent interaction. In a game, we have a set of players, each of whom chooses some action to perform from a given set of actions. As a result of the players' combined choices, some outcome is obtained which is described numerically in the form of a payoff vector, i.e., a vector of values, one for each of the players. We concentrate on two-player, fixed-sum games (i.e., games in which the sum of values in the payoff vector is constant). We refer to the player under our control as *the agent*, whereas the other player will be called *the adversary.*[2]

A common description of a game is as a matrix. This is called a game in *strategic form*. The rows of the matrix correspond to the agent's actions and the columns correspond to the adversary's actions. The entry in row $i$ and column $j$ in the game matrix contains the rewards obtained by the agent and the adversary if the agent plays his $i^{th}$ action and the adversary plays his $j^{th}$ action. We make the simplifying assumption that the size of the action set of both the agent and the adversary is identical. However, an extension to sets of different sizes is trivial.

In a *stochastic game* (SG) the players play a (possibly infinite) sequence of standard games from some given set of games. After playing each game, the players receive the appropriate payoff, as dictated by that game's matrix, and move to a new game. The identity of this new game depends, stochastically, on the previous game and on the players' actions in that previous game. Formally:

**Definition 1** *A* fixed-sum, two player, stochastic-game [SG] $M$ *on states* $S = \{1, \ldots, N\}$, *and actions* $A = \{a_1, \ldots, a_k\}$, *consists of:*

- **Stage Games:** *each state* $s \in S$ *is associated with a two-player, fixed-sum game in strategic form, where the action set of each player is* $A$. *We use* $R^i$ *to denote the reward matrix associated with stage-game* $i$.
- **Probabilistic Transition Function:** $P_M(s, t, a, a')$ *is the probability of a transition from state* $s$ *to state* $t$ *given that the first player (the* agent*) plays* $a$ *and the second player (the* adversary*) plays* $a'$.

An SG is similar to an MDP. In both models, actions lead to transitions between states of the world. The main difference is that in an MDP the transition depends on the action of a single agent whereas in an SG the transition depends on the joint-action of the agent and the adversary. In addition, in an SG, the reward obtained by the agent for performing an action depends on its action *and* the action of the adversary. To model this, we associate a game with every state. Therefore, we shall use the terms *state* and *game* interchangeably.

Stochastic games are useful not only in multi-agent contexts. They can be used instead of MDPs when we do not wish to model the environment (or certain aspects of it) stochastically. In that case, we can view the environment as an agent that can choose among different alternatives, without assuming that its choice is based on some probability distribution. This leads to behavior maximizing the worst-case scenario. In addition, the adversaries that the agent meets in each of the stage-games could be different entities.

R-max is formulated as an algorithm for learning in fixed-sum Stochastic Games. However, it is immediately applicable to fixed-sum repeated games and to MDPs because both of these models are degenerate forms of SGs. A repeated game is an SG with a single state and an MDP is an SG in which the adversary has a single action at each state.

2. From the point of view of the probabilistic maximin value of the game, which is what we attempt to attain, the case of $n$-player zero sum game does not differ significantly, as all other players can be viewed as a single adversary.

For ease of exposition we normalize both players' payoffs in each stage game to be non-negative reals between 0 and some constant $R_{max}$. We also take the number of actions to be constant. The set of possible histories of length $t$ is $(S \times A^2 \times R)^t \times S$, i.e., a sequence consisting of $t$ state-action-reward triplets, followed by the state reached. The set of possible histories, $H$, is the union of the sets of possible histories for all $t \geq 0$, where the set of possible histories of length 0 is $S$.

Given an SG, a policy for the agent is a mapping from $H$ to the set of possible probability distributions over $A$. Hence, a policy determines the probability of choosing each particular action for each possible history.

We define the *value* of a policy using the *average expected reward criterion* as follows: Given an SG $M$ and a natural number $T$, we denote the expected $T$-step undiscounted average reward of a policy $\pi$ when the adversary follows a policy $\rho$, and where both $\pi$ and $\rho$ are executed starting from a state $s \in S$, by $U_M(s, \pi, \rho, T)$ (we omit subscripts denoting the SG when this causes no confusion). Let $U_M(s, \pi, T) = \min_{\rho \textit{ is a policy}} U_M(s, \pi, \rho, T)$ denote the value that a policy $\pi$ can guarantee in $T$ steps starting from $s$. We define $U_M(s, \pi) = \liminf_{T \to \infty} U_M(s, \pi, T)$. Finally, we define $U_M(\pi) = \min_{s \in S} U_M(s, \pi)$.[3]

### 2.2 Assumptions, Complexity and Optimality

We make two central assumptions: First, we assume that the agent always recognizes the identity of the state (or stage-game) it reached (but not its associated payoffs and transition probabilities) and that after playing a game, it knows what actions were taken by its adversary and what payoffs were obtained. Second, we assume that the maximal possible reward $R_{max}$ is known ahead of time. We believe that this latter assumption can be removed.

Next, we wish to discuss the central parameter in the analysis of the complexity of R-max – the *mixing time*, first identified by Kearns and Singh (1998). Kearns and Singh argue that it is unreasonable to refer to the efficiency of learning algorithms without referring to the efficiency of convergence to a desired value. They defined the *$\epsilon$-return mixing time* of a policy $\pi$ to be the smallest value of $T$ after which $\pi$ *guarantees* an expected payoff of at least $U(\pi) - \epsilon$. In our case, we have to take into account the existence of an adversary. Therefore, we adjust this definition slightly as follows: a policy $\pi$ belongs to the set $\Pi(\epsilon, T)$ of policies whose $\epsilon$-return mixing time is at most $T$, if for any starting state $s$, for any adversary behavior $\rho$, and for every $t \geq T$ we have that $U(s, \pi, \rho, t) > U(\pi) - \epsilon$.

That is, if a policy $\pi \in \Pi(\epsilon, T)$ then no matter what the initial state is and what the adversary does, the policy $\pi$ will yield after any $t \geq T$ steps an expected average reward that is $\epsilon$ close to its value. The $\epsilon$-return mixing time of a policy $\pi$ is the smallest $T$ for which $\pi \in \Pi(\epsilon, T)$ holds. Notice that this means that an agent with perfect information about the nature of the games and the transition function will require at least $T$ steps, on the average, to obtain an optimal value using an optimal policy $\pi$ whose $\epsilon$-return mixing time is $T$. Clearly, one cannot expect an agent lacking this information to perform better.

We denote by $Opt(\Pi(\epsilon, T))$ the optimal expected $T$-step undiscounted average return from among the policies in $\Pi(\epsilon, T)$. When looking for an optimal policy (with respect to policies that mix at time $T$, for a given $\epsilon > 0$), we will be interested in approaching this value

3. We discuss this choice below.

in time polynomial in $T$, in $1/\epsilon$, in $\ln(1/\delta)$ (where $\epsilon$ and $\delta$ are the desired error bounds), and in the size of the description of the game.

The reader may have noticed that we defined $U_M(\pi)$ as $\min_{s\in S} U_M(s,\pi)$. It may appear that this choice makes the learning task too easy. For instance, one may ask why shouldn't we try to attain the maximal value over all possible states, or at least the value of our initial state? We claim that the above is the only reasonable choice, and that it leads to results that are as strong as those achieved by $E^3$.

To understand this point, consider the following situation: we start learning at some state $s$ in which the optimal action is $a$. If we do not execute the action $a$ in $s$, we reach some state $s'$ that has a very low value. A learning algorithm without any prior knowledge cannot be expected to immediately guess that $a$ should be done in $s$. In fact, without such prior knowledge, it cannot conclude that $a$ is a good action unless it tries the other actions in $s$ and compares their outcome to that of $a$. Thus, one can expect an agent to learn a near-optimal policy only if the agent can visit state $s$ sufficiently many times to learn about the consequences of different options in $s$. In a finite SG, there will be some set of states that we can sample sufficiently many times, and it is for such states that we can learn to behave.

In fact, it probably makes sense to restrict our attention to a subset of the states such that from each state in this set it is not too hard to get to any other state. In the context of MDPs, Kearns and Singh refer to this as the *ergodicity* assumption. In the context of SGs, Hoffman and Karp (1966) refer to this as the *irreducibility* assumption. An SG is said to be *irreducible* if the Markov-chain obtained by fixing any two (pure) stationary strategies for each of the players is irreducible (i.e., each state is reachable from each other state). In the special case of an MDP, irreducibility is precisely the ergodicity property used by Kearns and Singh in their analysis of $E^3$.

Irreducible SGs have a number of nice properties, as shown by Hoffman and Karp (1966). First, the maximal long-term average reward is independent of the starting state, implying that $\max_\pi \min_{s\in S} U_M(s,\pi) = \max_\pi \max_{s\in S} U_M(s,\pi)$. Second, this optimal value can be obtained by a stationary policy (i.e., one that depends on the current stage-game only). Thus, although we are not restricting ourselves to irreducible games, we believe that our results are primarily interesting in this class of games.

## 3. The R-max Algorithm

Recall that we consider a stochastic game $M$ consisting of a set $S = \{G_1,\ldots,G_N\}$ of stage-games in each of which both the agent and the adversary have a set $A = \{a_1,\ldots,a_k\}$ of possible actions. We associate a reward matrix $R^i$ with each game, and use $R^i_{m,l}$ to denote a pair consisting of the reward obtained by the agent and the adversary after playing actions $a_m$ and $a_l$ in game $G_i$, respectively. In addition, we have a probabilistic transition function, $P_M$, such that $P_M(s,t,a,a')$ is the probability of making a transition from $G_s$ to $G_t$ given that the agent played $a$ and the adversary played $a'$. It is convenient to think of $P_M(i,\cdot,a,a')$ as a function associated with the entry $(a,a')$ in the stage-game $G_i$. This way, all model parameters, both rewards and transitions, are associated with joint actions of a particular game. Let $\epsilon > 0$. For ease of exposition, we assume throughout most of the analysis that the $\epsilon$-return mixing time of the optimal policy, $T$, is known. Later, we show how this assumption can be relaxed.

The R-max algorithm is defined as follows:

**Input:** $N$ – the number of stage games, $k$ – the number of actions at each stage game, $\epsilon$ – the error bound, $\delta$ – the algorithm's failure probability, $R_{max}$ – an upper bound on the reward function, $T$ – the $\epsilon$-return mixing time of an optimal policy.

**Initialize:** Construct the following model $M'$ consisting of $N+1$ stage-games, $\{G_0, G_1, \ldots, G_N\}$, and $k$ actions, $\{a_1, \ldots, a_k\}$. Here, $G_1, \ldots, G_N$ correspond to the real games, $\{a_1, \ldots, a_k\}$ correspond to the real actions, and $G_0$ is an additional fictitious game. Initialize all game matrices to have $(R_{max}, 0)$ in all entries.[4] Initialize $P_M(G_i, G_0, a, a') = 1$ for all $i = 0, \ldots, N$ and for all actions $a, a'$.

In addition, maintain the following information for each entry in each game $G_1, \ldots, G_N$: (1) a boolean value *known/unknown*, initialized to *unknown*; (2) the states reached by playing the joint action corresponding to this entry (and how many times each state was reached); (3) the reward obtained (by both players) when playing the joint action corresponding to this entry. Items 2 and 3 are initially empty.

**Repeat:**

**Compute and Act:** Compute an optimal $T$-step policy for the current state, and execute it for $T$-steps or until a new entry becomes known.

**Observe and update:** Following each joint action do as follows: Let $a$ be the action you performed in $G_i$ and let $a'$ be the adversary's action.

- If the joint action $(a, a')$ is performed for the first time in $G_i$, update the reward associated with $(a, a')$ in $G_i$, as observed.
- Update the set of states reached by playing $(a, a')$ in $G_i$.
- If at this point your record of states reached from this entry contains $K_1 = \max((\lceil \frac{4NTR_{max}}{\epsilon})^3\rceil, \lceil -6ln^3(\frac{\delta}{6Nk^2})\rceil) + 1$ elements, mark this entry as *known*, and update the transition probabilities for this entry according to the observed frequencies.

As can be seen, R-max is quite simple. It starts with an initial estimate for the model parameters that assumes all states and all joint actions yield maximal reward and lead with probability 1 to the fictitious stage-game $G_0$. Based on the current model, an optimal $T$-step policy is computed and followed. Following each joint action the agent arrives at a new stage-game, and this transition is recorded in the appropriate place. Once we have enough information about where some joint action leads to from some stage-game, we update the entries associated with this stage-game and this joint action in our model. After each model update, we recompute an optimal policy and repeat the above steps.

4. The value 0 given to the adversary does not play an important role here.

## 4. Optimality and Convergence

In this section we provide the tools that ultimately lead to the proof of the following theorem:

**Theorem 2** *Let $M$ be an SG with $N$ states and $k$ actions. Let $0 < \delta < 1$, and $\epsilon > 0$ be constants. Denote the policies for $M$ whose $\epsilon$-return mixing time is $T$ by $\Pi_M(\epsilon, T)$, and denote the optimal expected return achievable by such policies by $Opt(\Pi_M(\epsilon, T))$. Then, with probability of no less than $1 - \delta$ the* R-MAX *algorithm will attain an expected return of $Opt_M(\Pi(\epsilon, T)) - 2\epsilon$ within a number of steps polynomial in $N, k, T, \frac{1}{\epsilon}$, and $\frac{1}{\delta}$.*

In the main lemma required for proving this theorem we show the following: if the agent follows a policy that is optimal with respect to the model it maintains for $T$ steps, it will either attain near-optimal average reward, as desired, or it will update its statistics for one of the unknown slots with sufficiently high probability. This can be called the *implicit* explore or exploit property of R-MAX: The agent does not know ahead of time whether it is exploring or exploiting – this depends in a large part on the adversary's behavior which it cannot control or predict. However, it knows that it does one or the other, no matter what the adversary does. Using this result we can proceed as follows: As we will show, the number of samples required to mark a slot as known is polynomial in the problem parameters, and so is the total number of entries. Therefore, the number of $T$-step iterations used for learning the parameters is bounded by some polynomial function of the input parameters, say $L$. That is, after $L$ learning iterations we have an (almost) perfect model and can compute a near-optimal policy. Of course, during the learning iterations, our average reward can be low. Thus, we need to perform an $D$ $T$-step iterations in order to compensate for our losses during the $L$ learning episodes. If we choose $D = L \cdot R_{max}/\epsilon$, our average reward will be $\epsilon$-close to optimal. Finally, to compensate for the fact that some of our attempts to explore or to exploit could fail, we will need to actually run the algorithm for $D'$ episodes, where $D'$ is some polynomial function of $D$.

To understand the actual proof, consider how our learned model differs from the real model. First, there are states in it which are marked *unknown* and they behave quite differently from their corresponding states in the real model. Second, there are states marked *known* and they look very similar to the corresponding states in the real model, although not necessarily identical. The proof handles these differences in two steps. In Lemma 2, we show that the explore or exploit property discussed above holds when the *known* states are identical to their counterparts in the real model. Lemma 1 shows that the optimal policy with respect to one model leads to near-optimal behavior with respect to models that are sufficiently similar. By combining them, we can handle both the *unknown* states (via Lemma 2) and the similar but not identical *known* states (via Lemma 1).

Before proving our main lemma, we state and prove an extension of Kearns and Singh's Simulation Lemma (Kearns and Singh, 1998) to the context of SGs with a slightly improved bound.

**Definition 3** *Let $M$ and $\bar{M}$ be SGs over the same state and action spaces. We say that $\bar{M}$ is an $\alpha$-approximation of $M$ if for every state $s$ we have:*

1. *If $P_M(s, t, a, a')$ and $P_{\bar{M}}(s, t, a, a')$ are the probabilities of transition from state $s$ to state $t$ given that the joint action carried out by the agent and the adversary is $(a, a')$, in $M$ and $\bar{M}$ respectively, then, $P_M(s, t, a, a') - \alpha \leq P_{\bar{M}}(s, t, a, a') \leq P_M(s, t, a, a') + \alpha$*

2. *For every state $s$, the same stage-game is associated with $s$ in $\bar{M}$ and in $M$ (and thus, the rewards will be identical).*

**Lemma 4** *Let $M$ and $\bar{M}$ be SGs over $N$ states, where $\bar{M}$ is an $\frac{\epsilon}{NTR_{max}}$-approximation of $M$, then for every state $s$, agent policy $\pi$, and adversary policy $\rho$, we have that*

$$|U_{\bar{M}}(s,\pi,\rho,T) - U_M(s,\pi,\rho,T)| \leq \epsilon.$$

**Proof** When we fix both players' policies we get, both in MDPs and in general SGs, a probability distribution over $T$-step paths in the state space. This is not a Markov process because the player's policies can be non-stationary. However, the transition probabilities at each point depend on the current state and the actions taken and the probability of each path is a product of the probability of each of the transitions. This is true whether the policies are pure or mixed.

We need to prove that:

$$\sum_p |\Pr_M(p)U_M(p) - \Pr_{\bar{M}}(p)U_{\bar{M}}(p)| \leq \epsilon$$

where $p$ is a $T$-step path starting at $s$, $\Pr_M(p)$ (respectively, $\Pr_{\bar{M}}(p)$) is its probability in the random process induced by $M$ (resp. by $\bar{M}$), $\pi$, and $\rho$, and $U_M(p), (U_{\bar{M}}(p))$ is the average payoff along this path. Because the average payoff is bounded by $R_{max}$ we have:

$$\sum_p |\Pr_M(p)U_M(p) - \Pr_{\bar{M}}(p)U_{\bar{M}}(p)| \leq \sum_p |\Pr_M(p) - \Pr_{\bar{M}}(p)|R_{max}.$$

To conclude our proof, it is sufficient to show that

$$\sum_p |\Pr_M(p) - \Pr_{\bar{M}}(p)| \leq \epsilon/R_{max}$$

Let $h_i$ define the following random processes: start at state $s$ and follow policies $\rho$ and $\pi$; for the first $i$ steps, the transition probabilities are identical to the process defined above on $\bar{M}$, and for the rest of the steps its transition probabilities are identical to $M$. Clearly, when we come to assess the probabilities of $T$-step path, we have that $h_0$ is identical to the original process on $M$, whereas $h_T$ is identical to original process on $\bar{M}$. The triangle inequality implies that

$$\sum_p |\Pr_M(p) - \Pr_{\bar{M}}(p)| = \sum_p |\Pr_{h_0}(p) - \Pr_{h_T}(p)| \leq \sum_{i=0}^{T-1} \sum_p |\Pr_{h_i}(p) - \Pr_{h_{i+1}}(p)|$$

If we show that for any $0 \leq i < T$ we have that $\sum_p |\Pr_{h_i}(p) - \Pr_{h_{i+1}}(p)| \leq \epsilon/TR_{max}$, it will follow that $\sum_p |\Pr_M(p) - \Pr_{\bar{M}}(p)| \leq \epsilon/R_{max}$, which is precisely what we need to show.

We are left with the burden of proving that $\sum_p |\Pr_{h_i}(p) - \Pr_{h_{i+1}}(p)| \leq \epsilon/TR_{max}$. We can sum over all paths $p$ as follows: first we sum over the $N$ possible states that can be reached in $i$ steps. Then we sum over all possible path prefixes that reach each such state. Next, we sum over all possible states reached after step $i+1$, and finally over all possible suffixes that start from each such state. Now, we note that the probability of each particular path $p$ is the product of the probability of its particular prefix, the probability of a transition from $x_i$ to $x_{i+1}$, and the probability of the suffix. We will use $x_i$ to denote the state reached after $i$ steps, $x_{i+1}$ to denote the state reached after $i+1$ steps, $pre(x_i)$ to denote the $i$-step prefixes reaching $x_i$, and $suf(x_j)$ to denote the suffixes starting at $x_j$. Thus,

$$\sum_p |\Pr_{h_i}(p) - \Pr_{h_{i+1}}(p)| = \sum_{x_i} \sum_{pre(x_i)} \sum_{x_{i+1}} \sum_{suf(x_{i+1})} |(\Pr_{h_i}(pre(x_i)) \Pr_{h_i}(x_i \to x_{i+1}) \Pr_{h_i}(suf(x_{i+1})))- (\Pr_{h_{i+1}}(pre(x_i)) \Pr_{h_{i+1}}(x_i \to x_{i+1}) \Pr_{h_{i+1}}(suf(x_{i+1})))|$$

However, the prefix and suffix probabilities are identical in $h_i$ and $h_{i+1}$. Thus, this sum is equal to

$$\sum_{x_i} \sum_{pre(x_i)} \sum_{x_{i+1}} \sum_{suf(x_{i+1})} \Pr_{h_i}(pre(x_i)) \Pr_{h_i}(suf(x_{i+1}))| \Pr_{h_i}(x_i \to x_{i+1}) - \Pr_{h_{i+1}}(x_i \to x_{i+1})| =$$

$$\sum_{x_i} \sum_{pre(x_i)} \Pr_{h_i}(pre(x_i)) \sum_{x_{i+1}} \sum_{suf(x_{i+1})} \Pr_{h_i}(suf(x_{i+1}))| \Pr_{h_i}(x_i \to x_{i+1}) - \Pr_{h_{i+1}}(x_i \to x_{i+1})| \leq$$

$$[\sum_{x_i} \sum_{pre(x_i)} \Pr_{h_i}(pre(x_i))][\sum_{x_{i+1}} \sum_{suf(x_{i+1})} \Pr_{h_i}(suf(x_{i+1}))\epsilon/NTR_{max}]$$

This last expression is a product of two independent terms. The first term is the sum over all possible $i$-step prefixes (i.e., overall all prefixes starting in the given $x_0$ and ending in $x_i$, for any $x_i$). Hence, it is equal to 1. The second term is a sum over all suffixes starting at $x_{i+1}$, for any value of $x_{i+1}$. For any given value of $x_{i+1}$ the probability of any suffix starting at this value is 1. Summing over all possible values of $x_{i+1}$, we get a value of $N$.

Thus,

$$\sum_p |\Pr_{h_i}(p) - \Pr_{h_{i+1}}(p)| \leq 1 \cdot \epsilon/NTR_{max} \cdot N$$

This concludes our proof. ■

Next, we define the notion of an *induced SG*. The definition is similar to the definition of an induced MDP given by Kearns and Singh (1998) except for the use of $R_{max}$. The induced SG is the model used by the agent to determine its policy.

**Definition 5** *Let $M$ be an SG. Let $L$ be the set of entries $(G_i, a, a')$ marked* unknown. *That is, if $(G_i, a, a') \in L$ then the entry corresponding to the joint action $(a, a')$ in the stage-game $G_i$ is marked as* unknown. *Define $M_L$ to be the following SG: $M_L$ is identical to $M$, except that $M_L$ contains an additional state $G_0$. Transitions and rewards associated with all entries in $M_L$ that are* not *in $L$ are identical to those in $M$. For any entry in $L$ or in $G_0$, the transitions are with probability 1 to $G_0$, and the reward is $R_{max}$ for the agent and 0 for the adversary.*[5]

5. Note that the formulation of R-max, implies that for some entries in $L$, the reward function is known and can be different from $R_{max}$. This difference does not affect the following result which relies on the fact that the value of the optimal policy in $M_L$ is at least as large as the value of the optimal policy in $M$.

Given an SG $M$ with a set $L$ of unknown states, $R^{M_L}$-max denotes the optimal policy for the induced SG $M_L$. When $M_L$ is clear from the context we will simply use the term R-MAX *policy* instead of $R^{M_L}$-max policy.

We now state and prove the implicit explore or exploit lemma:

**Lemma 6** *Let $M$ be an SG, let $L$ and $M_L$ be as above. Let $\rho$ be an arbitrary policy for the adversary, let $s$ be some state, and let $0 < \alpha < 1$. Then either (1) $V_{R-max} > Opt(\Pi_M(\epsilon, T)) - \alpha$, where $V_{R-max}$ is the expected $T$-step average reward for the $R^{M_L}$-max policy on $M$; or (2) An unknown entry will be played in the course of running* R-MAX *on $M$ for $T$ steps with a probability of at least $\frac{\alpha}{R_{max}}$.*

In practice, we cannot determine $\rho$, the adversary's policy, ahead of time. Thus, we do not know whether R-MAX will attain near-optimal reward or whether it will reach an unknown entry with sufficient probability. The crucial point is that it will do one or the other, no matter what the adversary does.

The intuitive idea behind Lemma 2 is the following: our behavior is biased towards exploration (because of the appealing value of *unknown* states). The value of the exploration policy in our (fictitious) model is very high. This value is a probabilistic maximin. Thus, the adversary cannot prevent us from attaining it. When we explore, this value is actually fictitious, because it overestimates the *unknown* states. If the adversary tries to prevent us from learning new information, it forces us to remain in *known* states. However, these states are modeled correctly, and so their value is correct, not fictitious. Thus, when the adversary tries to prevent us from learning, we attain in practice a value that is close to what we expected to get based on our fictitious world. This value is always near-optimal.

**Proof** First, notice that the value of R-MAX in $M_L$ will be no less than the value of the optimal policy in $M$. This follows from the fact that the rewards for the agent in $M_L$ is at least as large as in $M$, and that the R-max policy is optimal with respect to $M_L$.

In order to prove the claim, we will show that the difference between the reward obtained by the agent in $M$ and in $M_L$ when R-MAX is played is smaller than the exploration probability times $R_{max}$. This will imply that if the exploration probability is small, then R-MAX will attain near-optimal payoff. Conversely, if near-optimal payoff is not attained, the exploration probability will be sufficiently large.

For any policy, we may write:

$$U_M(s, \pi, \rho, T) = \sum_p Pr_M^{\pi,s}[p]U_M(p) = \sum_q Pr_M^{\pi,s}[q]U_M(q) + \sum_r Pr_M^{\pi,s}[r]U_M(r)$$

where the sums are over, respectively, all $T$-paths $p$ in $M$, all $T$-paths $q$ in $M$ such that every entry visited in $q$ is not in $L$, and all $T$-path $r$ in $M$ in which at least one entry visited is in $L$. Hence:

$$|U_M(s, \text{R-max}, \rho, T) - U_{M_L}(s, \text{R-max}, \rho, T)| = |\sum_p Pr_M^{\text{R-max},\rho,s}[p]U_M(p) - \sum_p Pr_{M_L}^{\text{R-max},\rho,s}[p]U_{M_L}(p)|$$

$$= |\sum_q Pr_M^{\text{R-max},\rho,s}[q]U_M(q) + \sum_r Pr_M^{\text{R-max},\rho,s}[r]U_M(r) -$$

$$\sum_q Pr_{M_L}^{\text{R-max},\rho,s}[q]U_{M_L}(q) + \sum_r Pr_{M_L}^{\text{R-max},\rho,s}[r]U_{M_L}(r)|$$

$$\leq |\sum_q Pr_M^{\text{R-max},\rho,s}[q]U_M(q) - \sum_q Pr_{M_L}^{\text{R-max},\rho,s}[q]U_{M_L}(q)| +$$

$$|\sum_r Pr_M^{\text{R-max},\rho,s}[r]U_M(r) - \sum_r Pr_{M_L}^{\text{R-max},\rho,s}[r]U_{M_L}(r)|$$

The first difference:

$$|\sum_q Pr_M^{\text{R-max},\rho,s}[q]U_M(q) - \sum_q Pr_{M_L}^{\text{R-max},\rho,s}[q]U_{M_L}(q)|$$

must be 0. This follows from the fact that in $M$ and in $M_L$, the rewards obtained in a path which do not visit an unknown entry are identical. The probability of each such path is identical as well.

Hence, we have:

$$|U_M(s, \text{R-max}, \rho, T) - U_{M_L}(s, \text{R-max}, \rho, T)| \leq |\sum_r Pr_M^{\text{R-max},\rho,s}[r]U_M(r) - \sum_r Pr_{M_L}^{\text{R-max},\rho,s}[r]U_{M_L}(r)|$$

$$\leq \sum_r Pr_M^{\text{R-max},\rho,s}[r]R_{max}$$

This last inequality stems from the fact that the average reward in any path is no greater than $R_{max}$ and no smaller than 0. Thus, the maximal difference in the value of a path in $M$ and $M_L$ is $R_{max}$.

The last term is the probability of reaching an unknown entry multiplied by $R_{max}$. If this probability is greater than or equal to $\frac{\alpha}{R_{max}}$ then we are done (since (2) holds). Thus, from now on, let us assume

$$\sum_r Pr_M^{\text{R-max},\rho,s}[r] < \frac{\alpha}{R_{max}}.$$

This implies that

$$|U_M(s, \text{R-max}, \rho, T) - U_{M_L}(s, \text{R-max}, \rho, T)| \leq \alpha$$

Denote by $\pi^*$ an optimal $T$-step policy, and let $U_M(s, \pi^*, T)$ be its value. (Note that this value is, by definition, independent of the adversary strategy $\rho$.) If $V_{R-max} = U_M(s, \text{R-max}, \rho, T) \geq U_M(s, \pi^*, T)$, we are done. (Note: it is possible for $V_{R-max}$ to be greater than $U_M(s, \pi^*, T)$ because the particular adversary behavior $\rho$ may be sub-optimal.) Thus, we are left with the case where $U_M(s, \text{R-max}, \rho, T) < U_M(s, \pi^*, T)$. We now use the fact, noted earlier, that $U_{M_L}(s, \text{R-max}, \rho, T) \geq U_M(s, \pi^*, T)$ (because every agent reward in $M_L$ is at least as large as its comparable reward in $M$). Therefore, we have that

$$|U_M(s, \pi^*, T) - U_M(s, \text{R-max}, \rho, T)| = U_M(s, \pi^*, T) - U_M(s, \text{R-max}, \rho, T)$$

$$\leq U_{M_L}(s, \text{R-max}, \rho, T) - U_M(s, \text{R-max}, \rho, T) \leq \alpha$$

We are now ready to prove Theorem 1. First, we wish to show that the expected average reward is as stated. We must consider three models: $M$, the real model, $M'_L$ the actual model used, and $M'$ where $M'$ is an $\epsilon/2NTR_{max}$-approximation of $M$ such that the SG induced by $M'$ and $L$ is $M'_L$. At each $T$-step iteration of our algorithm we can apply the Implicit Explore or Exploit Lemma to $M'$ and $M'_L$ for the set $L$ applicable at that stage. Hence, at each step either the current R-max policy leads to an average reward that is $\epsilon/2$ close to optimal with respect to the adversary's behavior and the model $M'$ or it leads to an efficient learning policy with respect to the same model. However, because $M'$ is an $\epsilon/2$-approximation of $M$, the simulation lemma guarantees that the policy generated is either $\epsilon$ close to optimal or explores efficiently. We know that the number of $T$-step phases in which we are exploring can be bounded polynomially. This follows from the fact that we have a polynomial number of parameters to learn (in $N$ and $k$) and that the probability that we obtain a new, useful statistic is polynomial in $\epsilon, T$ and $N$. Thus, if we choose a large enough (but still polynomial) number of $T$-step phases, we shall guarantee that our average reward is as close to optimal as we wish.

The above analysis was done assuming we actually obtain the expected value of each random variable. This cannot be guaranteed with probability 1. Yet, we can ensure that the probability that the algorithm fails to attain the expected value of certain parameters is small enough by sampling it a larger (though still polynomial) number of times. This is based on the well-known Chernoff bound. Using this technique one can show that when the variance of some random variable is bounded, we can ensure that we get near its average with probability $1-\delta$ by using a sufficiently large sample that is polynomial in $1/\delta$.

In our algorithm, there are three reasons why the algorithm could fail to provide the agent with near optimal return in polynomial time.

1. First, we have to guarantee that our estimates of the transition probabilities for every entry are sufficiently accurate. Recall that to ensure a loss of no more than $\epsilon/2$ our estimates must be within $\frac{\epsilon}{2NTR_{max}}$ of the real probabilities.

   Consider a set of trials, where the joint action $(a, a')$ is performed in state $s$. Consider the probability of moving from state $s$ to state $t$ given the joint-action $(a, a')$ in a given trial, and denote it by $p$. Notice that there are $Nk^2$ such probabilities (one for each game and pair of agent-adversary actions). Therefore, we would like to show that the probability of failure in estimating $p$ is less than $\frac{\delta}{3Nk^2}$. Let $X_i$ be an indicator random variable, that is 1 iff we moved to state $t$ when we were in state $s$ and selected an action $a$ in trial $i$. Let $Z_i = X_i - p$. Then $E(Z_i) = 0$, and $|Z_i| \leq 1$. Chernoff bound implies that given independent random variables $Z_1, \ldots, Z_n$ where $|Z_i| \leq 1, E(Z_i) = 0$ $(1 \leq i \leq n)$, then $Prob(\sum_{i=1}^{n} Z_i > a) < e^{(-a^2/2n)}$. Hence, Chernoff bound implies that (for any $K_1$) $Prob(\Sigma_{i=1}^{K_1} Z_i > K_1^{\frac{2}{3}}) < e^{\frac{-K_1^{\frac{1}{3}}}{2}}$. This implies that $Prob(\frac{\Sigma_{i=1}^{K_1} X_i}{K_1} - p > K_1^{-\frac{1}{3}}) < e^{\frac{-K_1^{\frac{1}{3}}}{2}}$. Similarly, we can define $Z'_i = p - X_i$, and get by Chernoff bound that $Prob(\Sigma_{i=1}^{K_1} Z'_i > K_1^{\frac{2}{3}}) < e^{\frac{-K_1^{\frac{1}{3}}}{2}}$. This implies that $Prob(p - \frac{\Sigma_{i=1}^{K_1} X_i}{K_1} > K_1^{-\frac{1}{3}}) < e^{\frac{-K_1^{\frac{1}{3}}}{2}}$. Hence, we get that $Prob(|\frac{\Sigma_{i=1}^{K_1} X_i}{K_1} - p| > K_1^{-\frac{1}{3}}) < 2e^{\frac{-K_1^{\frac{1}{3}}}{2}}$.

We now choose $K_1$ such that $K_1^{-\frac{1}{3}} < \frac{\epsilon}{2NTR_{max}}$, and $2e^{\frac{-K_1^{\frac{1}{3}}}{2}} < \frac{\delta}{3Nk^2}$. This is obtained by taking $K_1 = \max((\frac{4NTR_{max})}{\epsilon})^3, -6ln^3(\frac{\delta}{6Nk^2})) + 1$.

The above guarantees that if we sample each slot $K_1$ times the probability that our estimate of the transition probability will be outside our desired bound is less than $\frac{\delta}{3}$.

Using the pigeon-hole principle we know that total number of visits to slots marked unknown is $Nk^2K_1$. After at most this number of visits all slots will be marked known.

2. The Implicit Exploit or Explore Lemma gives a probability of $\frac{\alpha}{R_{max}}$ of getting to explore. We now wish to show that after $K_2$ attempts to explore (i.e. when we do not exploit), we obtain the $K_1$ required visits. Let $X_i$ be an indicator random variable which is 1 if we reach the exploration state ($G_0$ in Lemma 2) when we do not exploit, and 0 otherwise. Let $Z_i = X_i - \frac{\alpha}{R_{max}}$, and let $Z_i' = \frac{\alpha}{R_{max}} - X_i$, and apply Chernoff bound to the sum of $Z_i$'s and $Z_i'$s as before. We get that $Prob(|\Sigma_{i=1}^{K_2} X_i - \frac{K_2\alpha}{R_{max}}| > K_2^{\frac{1}{3}}) < 2e^{\frac{-K_2^{\frac{1}{3}}}{2}}$. We can now choose $K_2$ such that $K_2^{\frac{1}{3}} + K_2\frac{\alpha}{R_{max}} > k^2NK_1$ and $2e^{\frac{-K_2^{\frac{1}{3}}}{2}} < \frac{\delta}{3k^2N}$ to guarantee that we will have a failure probability of less than $\frac{\delta}{3}$ due to this reason.

3. When we perform a $T$-step iteration without learning our expected return is $Opt(\Pi_M(T,\epsilon)) - \epsilon$. However, the actual return may be lower. This point is handled by the fact that after polynomially many local exploitations are carried out, $Opt(\Pi_M(T,\epsilon)) - \frac{3}{2}\epsilon$ (i.e. a value the is $\frac{\epsilon}{2}$ close to that expected payoff) can be obtained with a probability of failure of at most $\frac{\delta}{3}$. This is obtained by standard Chernoff bounds, and makes use of the fact that the standard deviation of the expected reward in a $T$-step policy is bounded because the maximal reward is bounded by $R_{max}$. More specifically, consider $z = MNT$ exploitation stages for some $M > 0$. Denote the average return in an exploitation stage by $\mu$, and let $X_i$ denote the return in the $i$-th exploitation stage ($1 \leq i \leq z$). Let $Y_i = \frac{\mu - X_i}{R_{max}}$. Notice that $|Y_i| \leq 1$, and that $E(Y_i) = 0$. Chernoff bound implies that: $Prob(\Sigma_{j=1}^{z} Y_j > z^{\frac{2}{3}}) < e^{\frac{-z^{\frac{1}{3}}}{2}}$ This implies that the average return along $z$ iterations is at most $\frac{R_{max}}{z^{\frac{1}{3}}}$ lower than $\mu$ with probability of at least $e^{\frac{-z^{\frac{1}{3}}}{2}}$. By choosing $M$ such that $z > (\frac{2R_{max}}{\epsilon})^3$, and $z > 6(ln(\frac{\delta}{3}))^{-3}$, we get the desired result: with probability less than $\frac{\delta}{3}$ the value obtained will not be more than $\frac{\epsilon}{2}$ lower than the expected value.

By making the failure probability less than $\frac{\delta}{3}$ for each of the above stages, we are able to obtain a total failure probability of no more than $\delta$.

From the proof, we can also observe the bounds on running times required to obtain this result. However, notice that in practice, the only bound that we need to consider when implementing the algorithm is the sample size $K_1$. ■

The algorithm we described requires the $\epsilon$-return mixing time as part of its input. We can remove this assumption as follows, using the ideas of Kearns and Singh (1998). Our proof establishes a concrete number of of steps polynomial in the problem parameters, that is sufficient to guarantee near-optimal average reward. Fixing all other input parameters, we can view this number of steps as a function $P(T)$ of the $\epsilon$-return mixing time. To obtain the desired performance when $T$ is unknown, we repeatedly execute the algorithm for $P(1)$ steps, under the assumption $T = 1$. Next, we execute it for $P(2)$ steps under the assumption $T = 2$. We continue this process, each time raising the value of $T$ by 1. At some point, we will reach a $T$ value of $T_0$, where $T_0$ is the true $\epsilon$-return mixing time. Our average reward for this phase is guaranteed to be near-optimal. However, during our previous phases in which $T$ was an under-estimate of $T_0$, we may have suffered certain losses. We need to compensate for them now. This compensation occurs naturally, as from now on, every value of $T$ we choose will be larger than $T_0$, and hence, will result in near-optimal average reward. In particular, if $T_0$ is the mixing time, then after $D' = T_0 R_{max}/\gamma$ phases, corresponding to running the algorithm with mixing time candidates $T_0 + 1, \ldots, T_0 + D'$, our payoff will be $\gamma$ close to optimal.

While this approach is quite standard, and has been already adopted in the case of MDPs by Kearns and Singh (1998), it is worth to notice the following. Consider the three sources of failure that have been mentioned before, and assume that the agent keeps track of all information (e,g, visits to entries of the games, etc.) the two first sources of failure that have to do with "sampling" are not affected by the fact we make repeated use of our procedure for different candidate mixing times. The third source of failure has to do with obtaining an actual payoff that is close to the optimal expected payoff. It is dealt with similarly: We view our collective trials for different values of $T$ greater or equal to the true mixing time as a larger sequence of independent trials. Each such trial has the desired expected value. Therefore, Chernoff bound will be applicable in obtaining the desired result here as well.

There is one caveat to this approach, though. The revised algorithm does not have a final halting time and will be applied continuously as long as the agent is functioning in its environment. Thus, at some point our current mixing time candidate $T$ will be exponential in the actual mixing time $T_0$. At that point, the computation of an optimal policy, which is performed whenever the model is updated, will require exponentially time. We note that this is true for the $E^3$ algorithm too.

This problem does not seem too serious, as it arises only after the algorithm is executed an exponential number of steps, and for the whole prior period, we behave near-optimally. Moreover, in the context of an MDP, it is likely that, at some point, the agent will be able to recognize that its model is sufficiently accurate. This will occur either when the whole model is known, or when a sufficient bound on the contribution of unknown states can be established (i.e., by bounding the probability of reaching these states). In the context of an SG, establishing such a bound is more difficult because of the dependence on adversary behavior.

Another point worth noting is that the agent may never know the values of some of the slots in the game because of the adversary's choices. Consequently, if $\pi$ is the optimal policy given full information about the game, the agent may actually converge to a policy $\pi'$ that differs from $\pi$, but which yields the best return given the adversary's actual behavior. This return will be no smaller than the return guaranteed by $\pi$. The mixing $\pi'$ will, in general, differ from the mixing time of $\pi$. However, we are guaranteed that if $T_0$ is the $\epsilon$-return mixing time of $\pi$, and $v$ is its value, then after time polynomial in $T_0$, the agent's actual return will be at least $v$ (subject to the deviations afforded by the theorem).

Finally, we believe that one can remove the assumption that $R_{max}$ is known by acting as follows. Initialize $R_{max}$ somehow. Whenever you encounter a value that is as large as the current estimate, make your estimate larger. This approach ensures that the bias towards exploration remains. However, it requires a more rigorous justification, as it is possible that too small an estimate may not provide a strong enough bias.

### 4.1 Repeated Games

A stochastic game in which the set of stage games contains a single game is called a *repeated game*. This is an important model in game theory and much work has been devoted to the study of learning in repeated games (Fudenberg and Levine, 1993). There is large class of learning problems associated with repeated games, and the problem as a whole is referred to as *repeated games with incomplete information* (Aumann and Maschler, 1995). The particular class of repeated games with incomplete information we are using (i.e., where the agent gets to observe the adversary's actions and the payoffs, and it knows the value of $R_{max}$) is known as an *Adaptive Competitive Decision Process* and has been studied, e.g., by Banos (1968) and Megiddo (1980).

Because a repeated game contains a single stage game, there are no transition probabilities to learn. However, there is still the task of learning to play optimally. In addition, because there is only a single stage game, the mixing time of any policy is 1 (the agent's expected reward after playing a single stage-game is identical to the policy's expected reward). However, the time required to guarantee this expected reward could be much larger. This stems from the fact that the optimal policy in a game is often mixed. That is, the agent chooses probabilistically, and not deterministically, among different options.

In repeated games, we can attain the same guarantees as in the stochastic-games setting, but with a greatly simplified version of R-MAX: We do not need to maintain a fictitious state or the statistics on the frequency of transitions. We describe the precise algorithm below. Its correctness follows from the more general result for stochastic games.

**Initialization** Initialize the game model with payoffs of $R_{max}$ for every joint action for the agent and 0 for the adversary. Mark all joint actions as *unknown*.

**Play** Repeat the following process:

- **Policy Computation** Compute an optimal policy for the game based on the current model and play it.
- **Update** If the joint action played is marked unknown, update the game matrix with its observed payoffs and mark is *known*.

## 5. Conclusion

We described R-MAX, a simple reinforcement learning algorithm that is guaranteed to lead to polynomial time convergence to near-optimal average reward in zero-sum stochastic games. In fact, R-max guarantees the safety level (probabilistic maximin) value for the agent in general non-cooperative stochastic games.

R-max is an optimistic model-based algorithm that formally justifies the optimism-under-uncertainty bias. Its analysis is inspired by the approach taken by Kearns and Singh in their $E^3$ algorithm. However, unlike $E^3$, the agent does not need to explicitly contemplate whether to explore or to exploit. In fact, the agent may never learn an optimal policy for the game,[6] or it may play an optimal policy without knowing that it is optimal. The "clever" aspect of the agent's policy is that it "offers" a catch to the adversary: if the adversary plays well, and leads the agent to low payoffs, then the agent will, with sufficient probability, learn something that will allow it to improve its policy. Eventually, without too many "unpleasant" learning phases, the agent will have obtained enough information to generate an optimal policy.

R-max can be applied to MDPs, repeated games, and SGs. In particular, all single-controller stochastic game instances covered in Brafman and Tennenholtz (2000) fall into this category, and R-max can be applied to them. However, R-max is much simpler conceptually and easier to implement than the LSG algorithm described there. Moreover, it also attains higher payoff: In LSG the agent must pay an additional multiplicative factor $\phi$ that does not appear in R-max.

We analyzed R-max in the context of the average-reward criteria. However, this is not the only optimality criteria that appears in the literature. In fact, a more commonly used criteria is that of sum-of-discounted-rewards. The analysis of this criteria is made problematic by the fact that every learning algorithm is expected to display an initial sub-optimal behavior. Because of the discounting employed, it is impossible to compensate later on for the losses incurred initially. Nevertheless, there is a reasonable work-around for this problem. We refer the reader to Kearns and Singh (1998), where this point is discussed. We note that their approach can be replicated here, with similar guarantees.

At this point, we have little intuition about how well R-max will perform in practice. Practical environments are likely to have many states (exponentially many in the number of state variables) and a large mixing time. Thus, convergence-time polynomial in these parameters is not likely to be a useful practical guarantee. In that case, algorithms that take into account the structure of the state space are likely to be required, and the generation and analysis of such algorithms is an important area of future work. Of course, even when structure is taken into account, the mixing-time can remain quite high (e.g., see Kearns & Koller, 1999), and we do not see a direct way of overcoming this problem, as it stems from the inherent stochasticity of the domain, and not from the learning process. However, this does not rule out other forms of analysis or guarantees that may be more reflective of realistic environments. We are currently working on an implementation of R-max with the view of adopting it into a practical approach for reinforcement learning.

Two other SG learning algorithms appeared in the literature. Littman (1994) describes a variant of Q-learning, called minimax Q-learning, designed for 2-person zero-sum stochastic games. That paper presents experimental results, asymptotic convergence results are presented in Littman and Szepesvári (1996). Hu and Wellman (1998) consider a more general framework of multi-agent general-sum games. This framework is more general than

6. In a game, an agent can obtain an optimal reward without following the optimal policy if the adversary plays sub-optimally.

the framework treated in this paper which dealt with fixed-sum, two-player games. Hu and Wellman based their algorithm on Q-learning as well. They prove (albeit, under considerable restrictions) that their algorithm converges to the optimal value (defined, in their case, via the notion of Nash equilibrium). In both cases, convergence is in the limit, i.e., provided that every state and every joint action has been visited infinitely often. Note that an adversary can prevent a learning agent from learning certain aspects of the game indefinitely and that R-max's polynomial time convergence to optimal payoff is guaranteed even if certain states and joint actions have never been encountered.

The class of repeated games is another sub-class of stochastic games for which R-max is applicable. In repeated games, $T = 1$, there are no transition probabilities to learn, and we need not use a fictitious stage-game. Therefore, a much simpler version of R-max can be used. The resulting algorithm is much simpler and much more efficient than previous algorithms by Megiddo (1980) and by Banos (1968). Moreover, for these algorithms, only convergence in the limit is proven. A more recent algorithm by Hart and Mas-Colell (2000) features an algorithm that is much simpler than the algorithms by Banos and by Megiddo. Moreover, this algorithm is *Hannan-Consistent* which means that it not only guarantees the agent its safety level, but it also guarantees that the agent will obtain the maximal average reward given the actual strategy used by the adversary. Hence, if the adversary plays sub-optimally, the agent can get an average reward that is higher than its safety-level. However, it is only known that this algorithm converges almost-surely, and its convergence rate is unknown. An interesting open problem is whether a polynomial time Hannan-consistent near-optimal algorithm exists for repeated games and for stochastic games.

## Acknowledgments

We wish to thank Amos Beimel for his help in improving the error bound in Lemma 1 and the anonymous referees for their useful comments. This research was supported by the Israel Science Foundation under grant #91/02-1. The first author is partially supported by the Paul Ivanier Center for Robotics and Production Management. The second author is partially supported by DARPA grant F30602-98-C-0214-P00005.

## References

R. Aumann and M. Maschler. *Repeated Games with Incomplete Information.* MIT Press, 1995.

A. Banos. On pseudo games. *The Annals of Mathematical Statistics*, 39:1932–1945, 1968.

R. Brafman and M. Tennenholtz. A near-optimal polynomial time algorithm for learning in certain classes of stochastic games. *Artificial Intelligence*, 121(1–2):31–47, 2000.

D. Fudenberg and D.K. Levine. Self-confirming equilibrium. *Econometrica*, 61(3):523–545, 1993.

S. Hart and A. Mas-Colell. A reinforcement procedure leading to correlated equilibrium. Technical report, Center for Rationality – Hebrew University of Jerusalem, 2000.

A.J. Hoffman and R.M. Karp. On Nonterminating Stochastic Games. *Management Science*, 12(5):359–370, 1966.

J. Hu and M.P. Wellman. Multi-agent reinforcement learning: Theoretical framework and an algorithms. In *Proc. 15th International Conference on Machine Learning*, 1998.

L. P. Kaelbling. *Learning in Embedded Systems.* The MIT Press, 1993.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of AI Research*, 4:237–285, 1996.

M. Kearns and D. Koller. Efficient reinforcement learning in factored mdps. In *Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 740–747, 1999.

M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. In *Int. Conf. on Machine Learning*, 1998.

M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. 11th Intl. Conf. on Machine Learning*, pages 157–163, 1994.

M. L. Littman and Csaba Szepesvári. A generalized reinforcement-learning model: Convergence and apllications. In *Proc. 13th Intl. Conf. on Machine Learning*, pages 310–318, 1996.

N. Megiddo. On repeated games with incomplete information played by non-bayesian players. *International Journal of Game Theory*, 9:157–167, 1980.

D. Monderer and M. Tennenholtz. Dynamic Non-Bayesian Decision-Making. *J. of AI Research*, 7:231–248, 1997.

A. W. Moore and C. G. Atkenson. Prioratized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13, 1993.

J. H. Schmidhuber. Curious model-building control systems. In *Proc. Intl. Joint Conf. on Neural Networks*, pages 1458–1463, 1991.

R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proc. of the 7th Intl. Conf. on Machine Learning.* Morgan Kaufmann, 1990.

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

P. Tadepalli and D. Ok. Model-based average reward reinforcement learning. *Artificial Intelligence*, 100:177–224, 1998.