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 MOSHET@IE.TECHNION.AC.IL
Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Haifa, 32000 Israel

Abstract

R-MAX is an extremely 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.

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 & 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 & Tennenholtz, 2000) as well as structured MDPs (Kearns & Koller, 1999). In E^3 the agent learns by updating a model of its environment 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

*. This paper appeared as Technical Report 01-2001, Department of Computer Science, Ben-Gurion University.

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, it is unlikely that an agent can explicitly choose between an exploration and an exploitation policy. For this reason, the only extension of E^3 to adverserial contexts used the restricted SCSG model in which the adversary influences the reward of a game only, and not its dynamics.

To overcome this problem, we suggest a different approach in which the agent never attempts to learn explicitly. Our agent always attempts to optimize its behavior, albeit with respect to a fictitious model in which optimal behavior often leads to learning. This model assumes that the reward the agent obtains in any situation it is not too familiar with, is the maximal possible reward – R_{max}. 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. The agent does *not* know whether it is optimizing or learning efficiently, but it always does one or the other. Thus, the agent will always either exploit or explore efficiently, without knowing ahead of time which of the two will occur. 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 game, 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 & Tennenholtz, 2000). In addition, it attains a higher expected average reward than LSG. R-MAX also improves upon previous algorithms for learning in repeated games (Aumann & Maschler, 1995), such as Megiddo's (Megiddo, 1980) and Banos (Banos, 1968). It is the only polynomial time algorithm for this class of games that we know of, and it is much simpler, too. Finally, R-MAX generalizes the results of Monderer and Tennenholtz (Monderer & 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, Littman, & Moore, 1996), where it appears under the heading "Ad-Hoc Techniques" and Section 2.7 in (Sutton & 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 & Atkenson, 1993). More recently, Tadepalli and Ok (Tadepalli & 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*.

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 a 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 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.

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 $(S \times A^2)^t \times S$, and 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 π when the adversary follows a policy ρ, and where both π and ρ 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 $U_M(s,\pi,T) = \min_{\rho \text{ is a policy}} U_M(s,\pi,\rho,T)$ denote the value that a policy π 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 $U_M(\pi) = \min_{s\in S} U_M(s,\pi)$.[1]

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

1. We discuss this choice below.

by its adversary and what payoffs were obtained. Second, we assume that the maximal possible reward R_{max} is known ahead of time. This latter assumption can be removed.[2]

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 ϵ-*return mixing time* of a policy π to be the smallest value of T after which π *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 π belongs to the set $\Pi(\epsilon, T)$ of policies whose ϵ-return mixing time is at most T, if for any starting state s and for any adversary behavior ρ, we have that $U(s, \pi, \rho, T) > U(s, \pi) - \epsilon$.

That is, no matter what the initial state is and what the adversary does, on the average, we have to apply policy π no more than T steps before our average accumulated reward is sufficiently close to the value of π for the state s. 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 π whose ϵ-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 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 in time polynomial in T, in $1/\epsilon$, in $1/\delta$ (where ϵ and δ 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 previous algorithms.

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

2. We would need to run the algorithm repeatedly for increasing values of R_{max}. The resulting algorithm remains polynomials in the relevant parameters.n

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 & 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 associated a reward matrix R^i with each game, and we 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 ϵ-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:

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.[3] 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); (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.

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

- If a, a' are 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, the optimal 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 the policy and repeat the above steps.

4. Optimality and Convergence

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

Theorem 1 *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 ϵ-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 in which non-optimal reward is obtained is bounded by some polynomial function of the input parameters, say T'. This implies that by performing T-step iterations $D = T'R_{max}/\theta$ times, we get that the loss obtained by non-optimal execution (where exploration is performed) is bounded by θ, for any $0 < \theta < 1$.

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

Definition 2 *Let M and $\bar{M}$ be SGs over the same state and action spaces. We say that $\bar{M}$ is an α-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.*

Lemma 1 *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 π, and adversary policy ρ, 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 $\bar{M}$), π, and ρ, and $U_M(p), (U_{\bar{M}}(p))$ is the average payoff along this path. Because the average payoff is bound 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 ρ and τ; 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 $\sum_p |\Pr_{h_i}(p) - \Pr_{h_{i+1}}(p)| \leq \epsilon / T R_{max}$, 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 $\sum_p |\Pr_{h_i}(p) - \Pr_{h_{i+1}}(p)| \leq \epsilon / T R_{max}$. We can sum over all path 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 in (Kearns & Singh, 1998) except for the use of R-MAX. The induced SG is the model used by the agent to determine its policy.

Definition 3 *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 which 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.*[4]

4. Any alternative definition of M_L assigning to arbitrary entries in L the same reward as that associated with their counterparts in M is suitable as well.

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 2 *Let M be an SG, let L and M_L be as above. Let ρ be an arbitrary policy for the adversary, let s be some state, and let $0 < \alpha < 1$. Then either (1) $|Opt(\Pi_M(\epsilon, T)) - V^s_{R-max}| < \alpha$, where V^s_{R-max} is the expected T-step average reward for the R^{M_L}-max policy on M starting from s; 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 ρ, 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.

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 reward for the agent 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:

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

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:

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

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

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

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

The first difference:

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

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 and the fact that we can appropriately associate different paths within these models with equal probabilities.

The last term is the probability of reaching an unknown entry multiplied by R_{max}. If this probability is less than $\frac{\alpha}{R_{max}}$ then

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

Denote by π^* an optimal T-step policy, and let $U_M(s,\pi,T)$ be its value. (Note that this value is independent of the adversary strategy ρ, as π^* guarantees at least this value for every adversary behavior.) If $U_M(s,\text{R-max},\rho,T) \geq U_M(s,\pi,T)$ we are done. Suppose that $U_M(s,\text{R-max},\rho,T) < U_M(s,\pi,T)$. We know that $U_{M_L}(s,\text{R-max},\rho,T) \geq U_M(s,\pi,T)$ is no lesser than the optimal T step average reward for 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 ϵ 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 ϵ, 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

be small enough by sampling it a larger (though still polynomial) number of time. 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 provided the agent with near optimal return in polynomial time.

1. First, we have to guarantee that our estimates of the transition probabilities for every slot 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 $|Z_i| \leq 1$. Then, 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 to 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 on 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, $Opt(\Pi_M(T,\epsilon)) - \frac{1}{2}\epsilon$ 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 μ, 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 μ 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 δ.

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.

To remove the assumptions that the ϵ-return mixing time is known, we proceed as in (Kearns & Singh, 1998). From the proof of the algorithm we deduced some polynomial P in the problem parameters such that if T is the mixing-time, then after $P(T)$ steps we are guaranteed, with probability $1-\delta$, the desired return. We repeat the execution of the algorithm for all values of $T = 1, 2, 3, \ldots$, each time performing $P(T)$ steps. Suppose that T_0 is the mixing time, then after $\sum_{i=1}^{T_0} P(i) = O(P(T_0)^2)$ steps, we will obtain the desired return. ∎

Notice that the R-max algorithm does not have a final halting time and will be applied continuously as long as the agent is functioning in its environment. The only caveat is that at some point our current mixing time candidate T will be exponential in the actual mixing time T_0, at which point each step of the algorithm will require an exponential calculation. However, this will occur only after an exponential number of steps. This is true for the E^3 algorithm too.

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 π is the optimal policy given full information about the game, the agent may actually converge to a policy π' that differs from π, but which yields the best return given the adversary's actual behavior. This return will be no smaller than the return guaranteed by π. The mixing time of π' will, in general, differ from the mixing time of π. However, we are guaranteed that if T_0 is the ϵ-return mixing time of π, and v is its value, after time polynomial in T_0, the agent's actual return will be at least v (subject to the deviations afforded by the theorem).

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 a lot of work has been devoted to the study of learning in repeated games (Fudenberg & 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 & 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 (Banos, 1968) and Megiddo (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 – because 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, the R-max algorithm is slightly modified, as we do not need to maintain a fictitious state and we need not maintain statistics on the frequency of various transitions. We describe the precise algorithm below:

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*.

Given $\epsilon > 0$, and $0 < \delta < 1$, we need to show that after polynomially many iterations M, where M is polynomial in the number of entries in the game, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$, we obtain a payoff that is at most ϵ lower than the expected payoff of the optimal strategy in this game, with probability of a least $1 - \delta$.

First notice that the expected payoff at each stage, when we do not expose the value of a new entry, is greater of equal to the expected payoff of the optimal strategy. By choosing $M = Q_1 + Q_2$, where $Q_2 = k^2 R_{max}/M \leq \epsilon/2$ we get that the loss due to learning of new entries is bounded by $\frac{\epsilon}{2}$. Now, we need to guarantee that after Q_1 executions (where Q_1 is polynomial in the problem parameters) of a policy with expected payoff greater or equal to r (where r is the expected payoff of the optimal policy in the original game), our actual payoff is at least $r - \epsilon/2$ with probability of at least $1 - \delta$. This follows from the arguments presented in case 3 of the general proof for SGs.

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.

R-max is an optimistic model-based algorithm that formally justifies the optimism in the face of uncertainty bias. Its analysis is similar, in many respects, to Kearns and Singh's E^3 algorithm. However, unlike the 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,[5] 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 provided that the latter satisfy the conditions we stated initially. In particular, all single-controller stochastic game instances covered in (Brafman & 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 ϕ that does not appear in R-max.

Two other SG learning algorithms appeared in the literature. Littman (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, assymptotic convergence results are presented in (Littman & Szepesvri, 1996). Hu and Wellman (Hu & Wellman, 1998) consider a more general framework of multi-agent general-sum games. This framework is more general than 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 that their algorithm converges to the optimal value (defined, in their case, via the notion of Nash equilibrium). However, 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 that satisfies the conditions of R-max. 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 (Megiddo, 1980) and by Banos (Banos, 1968). Moreover, for these algorithms, only convergence in the limit is proven. A more recent algorithm by Hart and Mas-Colell(Hart & Mas-Colell, 2000) features an algorithm that is much simpler than the algorithms by Banos and Megido. 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

5. In a game, an agent need not play optimally to obtain an optimal reward because it may obtain this reward because of bad choices by the adversary.

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 know 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 iterated games and for stochastic games.

Acknowledgements: 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. The first author is partially supported by the Paul Ivanier Center for Robotics and Production Management.

References

Aumann, R., & Maschler, M. (1995). *Repeated Games with Incomplete Information*. MIT Press.

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

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

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

Hart, S., & Mas-Colell, A. (2000). A reinforcement procedure leading to correlated equilibrium. Tech. rep., Center for Rationality – Hebrew University of Jerusalem.

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

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

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

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

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

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

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

Littman, M. L., & Szepesvri, C. (1996). A generalized reinforcement-learning model: Convergence and apllications. In *Proc. 13th Intl. Conf. on Machine Learning*, pp. 310–318.

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

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

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

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

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

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

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