Near-Optimal Reinforcement Learning in Polynomial Time

Michael Kearns
AT&T Labs
180 Park Avenue, Room A235
Florham Park, New Jersey 07932
mkearns@research.att.com

Satinder Singh
Department of Computer Science
University of Colorado
Boulder, Colorado 80309
baveja@cs.colorado.edu

Abstract

We present new algorithms for reinforcement learning, and prove that they have polynomial bounds on the resources required to achieve near-optimal return in general Markov decision processes. After observing that the number of actions required to approach the optimal return is lower bounded by the mixing time T of the optimal policy (in the undiscounted case) or by the horizon time T (in the discounted case), we then give algorithms requiring a number of actions and total computation time that are only polynomial in T and the number of states, for both the undiscounted and discounted cases. An interesting aspect of our algorithms is their explicit handling of the Exploration-Exploitation trade-off.

1 Introduction

In reinforcement learning, an agent interacts with an unknown environment, and attempts to choose actions that maximize its cumulative payoff (Sutton & Barto, 1998; Barto et al., 1990, Bertsekas & Tsitsiklis, 1996). The environment is typically modeled as a Markov decision process (MDP), and it is assumed that the agent does not know the parameters of this process, but has to *learn* how to act directly from experience. Thus, the reinforcement learning agent faces a fundamental trade-off between *exploitation* and *exploration* (Thrun, 1992; Sutton & Barto, 1998): should the agent exploit its cumulative experience so far, by executing the action that currently seems best, or should it execute a different action, with the hope of gaining information or experience that could lead to higher future payoffs? Too little exploration can prevent the agent from ever converging to the optimal behavior, while too much exploration can prevent the agent from gaining near-optimal payoff in a timely fashion.

There is a large literature on reinforcement learning, which has been growing rapidly in the last decade. To the best of our knowledge, all previous results on reinforcement learning in *general* MDP's are asymptotic in nature, providing no explicit guarantees on either the number of actions or the computation time the agent requires to achieve near-optimal performance (Sutton, 1988; Watkins & Dayan, 1992; Jaakkola et al., 1994; Tsitsiklis, 1994; Gullapalli & Barto, 1994). On the other hand, finite-time results become available if one considers *restricted* classes of MDP's, if the model of learning is modified from the standard one, or if one changes the criteria for success (Saul & Singh, 1996; Fiechter, 1994; Fiechter, 1997; Schapire & Warmuth, 1994; Singh & Dayan, in press). Fiechter (1994,1997), whose results are closest in spirit to ours, considers only the discounted case, and makes the learning protocol easier by assuming the availability of a "reset" button that allows the agent to return to a fixed set of start states at any time.

Thus, despite the many interesting previous results in reinforcement learning, the literature has lacked algorithms for learning optimal behavior in *general* MDP's with provably *finite* bounds on the resources (actions and computation time) required, under the standard model of learning in which the agent wanders continuously in the unknown environment. The results presented in this paper fill this void in what is essentially the strongest possible sense.

We present new algorithms for reinforcement learning, and prove that they have *polynomial* bounds on the resources required to achieve near-optimal payoff in *general* MDP's. The bounds are polynomial in the number of states, and also in the mixing time of the optimal policy (undiscounted case), or the horizon time $1/(1-\gamma)$ (discounted case). One of the contributions of this work is in simply identifying the fact that finite-time convergence results *must* depend on these parameters of the underlying MDP. An interesting aspect of our algorithms is their rather explicit handling of the exploration-exploitation trade-off.

For lack of space, here we present only our results for the more difficult undiscounted case. The analogous results for the discounted case are covered in a forthcoming longer paper; interested readers can retrieve the latest version from the web page http://www.research.att.com/~mkearns.

2 Preliminaries and Definitions

We begin with the basic definitions for MDP's.

Definition 1 *A* **Markov decision process (MDP)** M *on* **states** $1, \ldots, N$ *and with* **actions** $a_1, \ldots, a_k$, *consists of:*

Transition probabilities $P^a_M(ij) \geq 0$, *which for any action a, and any states i and j, specify the probability of reaching state j after executing action a from state i in M. Thus, $\sum_j P^a_M(ij) = 1$ for any state i and action a.*

Payoff distributions, *for each state i, with mean $R_M(i)$ (where $R_{max} \geq R_M(i) \geq 0$), and variance $Var_M(i) \leq Var_{max}$. These distributions determine the random payoff received when state i is visited.*

For simplicity, we will assume that the number of actions k is a constant; it will be easily verified that if k is a parameter, the resources required by our algorithms scale polynomially with k.

Several comments regarding some benign technical assumptions that we will make on payoffs are in order here. First, it is common to assume that payoffs are actually associated with state-action pairs, rather than with states alone. Our choice of the latter is entirely for technical simplicity, and all of the results of this paper hold for the standard state-action payoffs model as well. Second, we have assumed fixed upper bounds R_{max} and Var_{max} on the means and variances of the payoff distributions; such a restriction is necessary for finite-time convergence results. Third, we have assumed that expected payoffs are always non-negative for convenience, but this is easily removed by adding the minimum expected payoff to every payoff.

If M is an MDP over states $1, \ldots, N$ and with actions $a_1, \ldots, a_k$, a **policy** in M is a mapping $\pi : \{1, \ldots, N\} \rightarrow \{a_1, \ldots, a_k\}$. An MDP M, combined with a policy π, yields a standard Markov process on the states, and we will say that π is *ergodic* if the Markov process resulting from π is ergodic (that is, has a well-defined stationary distribution). For the development and exposition, it will be easiest to consider MDP's for which *every* policy is ergodic, the so-called *unichain* MDP's (Puterman, 1994). Considering the unichain case simply allows us to discuss the stationary distribution of any policy without cumbersome technical details, and as it turns out, the result for unichains already forces the main technical ideas upon us. Also, note that the unichain assumption does *not* imply that every policy will eventually visit every state, or even that there exists a single policy that will do so quickly; thus, the exploration-exploitation dilemma remains with us strongly. We discuss the extension to the *multichain* case in the longer version of this paper.

If M is an MDP, then a T**-path in** M is a sequence p of $T+1$ states (that is, T transitions) of M: $p = i_1, i_2, \ldots, i_T, i_{T+1}$. The probability that p is traversed in M upon starting in state i_1 and executing policy π is $\mathbf{Pr}^\pi_M[p] = \Pi_{k=1}^T P_M^{\pi(i_k)}(i_k i_{k+1})$. The (expected) undiscounted return **along** p in M is $U_M(p) = (1/T)(R_{i_1} + \cdots + R_{i_T})$ and the T-step undiscounted return from state i is $U^\pi_M(i,T) = \sum_p \mathbf{Pr}^\pi_M[p] U_M(p)$, where the sum is over all T-paths p in M that start at i. We define $U^\pi_M(i) = \lim_{T\rightarrow\infty} U^\pi_M(i,T)$. Since we are in the unichain case, $U^\pi_M(i)$ is independent of i, and we will simply write U^π_M. Furthermore, we define the **optimal** T-step undiscounted return from i in M by $U^*_M(i,T) = \max_\pi\{U^\pi_M(i,T)\}$. Also, $U^*_M(i) = \lim_{T\rightarrow\infty} U^*_M(i,T)$. Finally, we observe that the maximum possible T-step return is R_{max}.

3 Mixing Times for Policies

It is easy to see that if we are seeking results about the undiscounted return of a learning algorithm after a finite number of steps, we need to take into account some notion of the *mixing times* of policies in the MDP. To put it simply, for finite-time results, there may no longer be an unambiguous notion of "the" optimal policy. There may be some policies which will eventually yield high return (for instance, by finally reaching some remote, high-payoff state), but take many steps to approach this high return, and other policies which yield lower asymptotic return but

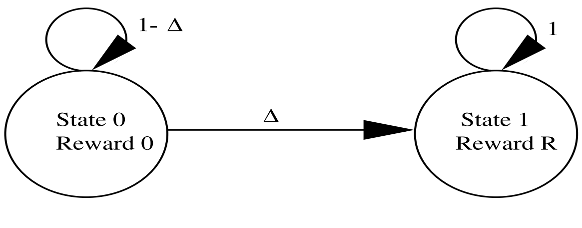

Figure 1: A simple Markov process demonstrating that finite-time convergence results must account for mixing times.

higher short-term return. Such policies are simply incomparable, and the best we could hope for is an algorithm that "competes" favorably with *any* policy, in an amount of time that is comparable to the mixing time *of that policy*.

Definition 2 *Let M be an MDP, and let π be an ergodic policy in M. Then the* **ϵ-return mixing time** *of π is the smallest T such that for all $T' \geq T$, $|U_M^\pi(i, T') - U_M^\pi| \leq \epsilon$ for all i* [1].

Suppose we are simply told that there is a policy π whose asymptotic return U_M^π exceeds R in an unknown MDP M, and that the ϵ-return mixing time of π is T. In principle, a sufficiently clever learning algorithm (for instance, one that managed to discover π "quickly") could achieve return close to $U_M^\pi - \epsilon$ in not much more than T steps. Conversely, without further assumptions on M or π, it is not reasonable to expect *any* learning algorithm to approach return U_M^π in many *fewer* than T steps. This is simply because it may take the assumed policy π itself on the order of T steps to approach its asymptotic return. For example, suppose that M has just two states and only one action (see Figure 1): state 0 with payoff 0, self-loop probability $1 - \Delta$, and probability Δ of going to state 1; and absorbing state 1 with payoff $R >> 0$. Then for small ϵ and Δ, the ϵ-return mixing time is on the order of $1/\Delta$; but starting from state 0, it really will require on the order of $1/\Delta$ steps to reach the absorbing state 1 and start approaching the asymptotic return R. (A more formal lower bound along the lines of this argument will be given in the long version.)

[1]In the long version, we relate the notion of ϵ-return mixing time to the standard notion of mixing time to stationary distributions (Puterman, 1994). The important point here is that the ϵ-return mixing time is polynomially bounded by the standard mixing time, but may in some cases be substantially smaller.

Thus, we would like a learning algorithm such that for *any* T, in a number of actions that is polynomial in T, the return of the learning algorithm is close to that achieved by the *best policy among those that mix in time* T. This motivates the following definition.

Definition 3 *Let M be a Markov decision process. We define $\Pi_M^{T,\epsilon}$ to be the class of all ergodic policies π in M whose ϵ-return mixing time is at most T. We let $opt(\Pi_M^{T,\epsilon})$ denote the optimal expected asymptotic undiscounted return among all policies in $\Pi_M^{T,\epsilon}$.*

Our goal in the undiscounted case will be to compete with the policies in $\Pi_M^{T,\epsilon}$ in time that is polynomial in T, $1/\epsilon$ and N. We will eventually give an algorithm that meets this goal for *every* T and ϵ *simultaneously*. An interesting special case is when $T = T^*$, where T^* is the ϵ-mixing time of the *asymptotically* optimal policy, whose asymptotic return is U^*. Then in time polynomial in T^*, $1/\epsilon$ and N, our algorithm will achieve return exceeding $U^* - \epsilon$ with high probability. It should be clear that, modulo the degree of the polynomial running time, such a result is the best that one could hope for in general MDP's. We briefly note that in the case of discounted reward, we can still hope to compete with the *asymptotically* optimal policy in time polynomial in the *horizon time*; this is discussed and achieved in the long version.

4 Main Theorem

We are now ready to describe our learning algorithm, and to state and prove our main theorem: namely, that the new algorithm will, for a general MDP, achieve near-optimal undiscounted performance in polynomial time. *For ease of exposition only*, we will first state the theorem under the assumption that the learning algorithm is given as input a "targeted" mixing time T, and the value $opt(\Pi_M^{T,\epsilon})$ of the optimal return achieved by any policy mixing within T steps. These assumptions are entirely removed in Section 4.6.

Theorem 1 *(Main Theorem) Let M be a Markov decision process over N states. Recall that $\Pi_M^{T,\epsilon}$ is the class of all ergodic policies whose ϵ-return mixing time is bounded by T, and that $opt(\Pi_M^{T,\epsilon})$ is the optimal asymptotic expected undiscounted return achievable in $\Pi_M^{T,\epsilon}$. There exists an algorithm A, taking inputs ϵ, δ, N, T and $opt(\Pi_M^{T,\epsilon})$, such that if the total number of actions and computation time taken by A exceeds a polynomial in $1/\epsilon, 1/\delta, N$, T, and R_{max}, then with probability at least $1 - \delta$, the total undiscounted return of A will exceed $opt(\Pi_M^{T,\epsilon}) - \epsilon$.*

In the long version, we give a similar theorem for the discounted case (via a similar algorithm), with the horizon time playing the role of T. The criterion for success needs to be altered, however, since in the discounted case it is not possible to insist that the *actual* return achieved by the learning algorithm approach the optimal. This is due to the exponentially damped contribution of successive payoffs. Intuitively, in the discounted case it is not possible for a learning algorithm to recover from its "youthful mistakes" as it can in the undiscounted case, so we must settle for an algorithm that simply finds a near-optimal policy from its current state after a short learning period.

The remainder of this section is divided into several subsections, each describing a different and central aspect of the algorithm and proof. The full proof of the theorem is rather technical, but the underlying ideas are quite intuitive, and we sketch them first as an outline.

4.1 Overview of the Proof and Algorithm

Our algorithm will be what is commonly referred to as *indirect* or *model-based*: namely, rather than only maintaining a current policy or value function, the algorithm will actually maintain a model for the transition probabilities and the expected payoffs for some *subset* of the states of the unknown MDP M. It is important to emphasize that although the algorithm maintains a partial model of M, it may choose to *never* build a *complete* model of M, if doing so is not necessary to achieve high return.

It is easiest to imagine the algorithm as starting off by doing what we will call *balanced wandering*. By this we mean that the algorithm, upon arriving in a state it has never visited before, takes an arbitrary action from that state; but upon reaching a state it has visited before, it takes the action it has tried the fewest times from that state (breaking ties between actions randomly). At each state it visits, the algorithm maintains the obvious statistics: the average payoff received at that state so far, and for each action, the empirical distribution of next states reached (that is, the estimated transition probabilities).

A crucial notion for both the algorithm and the analysis is that of a *known state*. Intuitively, this is a state that the algorithm has visited "so many" times (and therefore, due to the balanced wandering, has tried each action from that state many times) that the transition probability and expected payoff estimates for that state are "very close" to their true values in M. An important aspect of this definition is that it is weak enough that "so many" times is still polynomially bounded, yet strong enough to meet the simulation requirements we will outline shortly.

States are thus divided into three categories: known states, states that have been visited before, but are still unknown (due to an insufficient number of visits and therefore unreliable statistics), and states that have not even been visited once. An important observation is that we cannot do balanced wandering indefinitely before at least one state becomes known: by the Pigeonhole Principle, we will soon start to accumulate accurate statistics at some state.

Perhaps our most important definition is that of the *known-state MDP*. If S is the set of currently known states, the current known-state MDP is simply an MDP M_S that is naturally *induced* on S by the full MDP M; briefly, all transitions in M *between* states in S are preserved in M_S, while all other transitions in M are "redirected" in M_S to lead to a single additional, absorbing state that intuitively represents all of the unknown and unvisited states.

Although the learning algorithm will not have direct access to M_S, by virtue of the definition of the known states, it will have an *approximation* $\widehat{M}_S$. The first of two central technical lemmas that we prove (Section 4.2) shows that, under the appropriate definition of known state, $\widehat{M}_S$ will have good *simulation accuracy*: that is, the expected T-step return of any policy in $\widehat{M}_S$ is close to its expected T-step return in M_S. (Here T is the mixing time.) Thus, at any time, $\widehat{M}_S$ is a *partial* model of M, for that part of M that the algorithm "knows" very well.

The second central technical lemma (Section 4.3) is perhaps the most enlightening part of the analysis, and is named the "Explore or Exploit" Lemma. It formalizes a rather appealing intuition: either the optimal (T-step) policy achieves its high return by staying, with high probability, in the set S of currently known states — which, most importantly, the algorithm can *detect* and *replicate* by finding a high-return *exploitation* policy in the *partial* model $\widehat{M}_S$ — or the optimal policy has significant probability of *leaving* S within T steps — which again the algorithm can detect and replicate by finding an *exploration* policy that quickly reaches the additional absorbing state of the partial model $\widehat{M}_S$.

Thus, by performing two off-line, polynomial-time computations on $\widehat{M}_S$ (Section 4.4), the algorithm is guaranteed to either find a way to get near-optimal return in M quickly, or to find a way to improve the statistics at an unknown or unvisited state. Again by the Pigeonhole Principle, the latter case cannot occur too many times before a new state becomes known, and thus the algorithm is always making progress. In the worst case, the algorithm will build a model of the entire MDP M, but if that does happen, the analysis guarantees that it will happen in polynomial time.

The following subsections flesh out the intuitions sketched above, providing a detailed sketch of the proof of Theorem 1; the full proofs are provided in the long version. In Section 4.6, we discuss how to remove the assumed knowledge of the optimal return and the targeted mixing time.

4.2 The Simulation Lemma

In this section, we prove the first of two key technical lemmas mentioned in the sketch of Section 4.1: namely, that if one MDP $\widehat{M}$ is a sufficiently accurate approximation of another MDP M, then we can actually approximate the T-step return of any policy in M quite accurately by its T-step return in $\widehat{M}$. The important technical point is that the goodness of approximation required depends only polynomially on $1/T$, and thus the definition of known state will require only a polynomial number of visits to the state. Eventually, we will appeal to this lemma to show that we can accurately assess the return of policies in the induced known-state MDP M_S by computing their return in the algorithm's approximation $\widehat{M}_S$ (that is, we will appeal to Lemma 2 below using the settings $M = M_S$ and $\widehat{M} = \widehat{M}_S$).

We begin with the definition of approximation we require.

Definition 4 *Let M and $\widehat{M}$ be Markov decision processes over the same state space. Then we say that $\widehat{M}$ is an* α**-approximation** *of M if for any state i, $R_M(i) - \alpha \leq R_{\widehat{M}}(i) \leq R_M(i) + \alpha$, and for any states i and j, and any action a, $P^a_M(ij) - \alpha \leq P^a_{\widehat{M}}(ij) \leq P^a_M(ij) + \alpha$.*

We now state and prove the Simulation Lemma, which says that provided $\widehat{M}$ is sufficiently close to M in the sense just defined, the T-step return of policies in $\widehat{M}$ and M will be similar.

Lemma 2 *(Simulation Lemma) Let M be any Markov decision process over N states. Let $\widehat{M}$ be an $O((\epsilon/(NTR_{max}))^2)$-approximation of M. Then for any policy π and for any state i,*

$$U^\pi_M(i,T) - \epsilon \leq U^\pi_{\widehat{M}}(i,T) \leq U^\pi_M(i,T) + \epsilon. \quad (1)$$

Proof:(Sketch) Let $\widehat{M}$ be an α-approximation of M, and let us fix a policy π and a start state i. Let us say that a transition from a state i' to a state j' under action a is β**-small** in M if $P^a_M(i'j') \leq \beta$. It is possible to bound the difference between $U^\pi_M(i,T)$ and $U^\pi_{\widehat{M}}(i,T)$ contributed by those T-paths that cross at least one β-small transition by $(\alpha + 2\beta)NTR_{max}$ (details omitted). For the value of α stated in the theorem, our analysis chooses a value of β that yields $(\alpha + 2\beta)NTR_{max} \leq \epsilon/4$.

Thus, for now we restrict our attention to the walks of length T that do *not* cross any β-small transtion of M. It can be shown that for any T-path p that, under π, does not cross any β-small transitions of M, we have

$$(1-\Delta)^T \mathbf{Pr}^\pi_M[p] \leq \mathbf{Pr}^\pi_{\widehat{M}}[p] \leq (1+\Delta)^T \mathbf{Pr}^\pi_M[p] \quad (2)$$

where $\Delta = \alpha/\beta$. The approximation error in the payoffs yields

$$U_M(p) - \alpha \leq U_{\widehat{M}}(p) \leq U_M(p) + \alpha. \quad (3)$$

Since these inequalities hold for *any* fixed T-path that does not traverse any β-small transitions in M under π, they also hold when we take expectations over the *distributions* on such T-paths in M and $\widehat{M}$ induced by π. Thus,

$$\begin{aligned}(1-\Delta)^T[U^\pi_M(i,T) - \alpha] - \epsilon/4 \quad &\leq \quad U^\pi_{\widehat{M}}(i,T) \\ &\leq (1+\Delta)^T[U^\pi_M(i,T) + \alpha] + \epsilon/4\end{aligned}$$

where the additive $\epsilon/4$ terms account for the contributions of the T-paths that traverse β-small transitions under π, as bounded above. The desired constraint that the outermost quantities in this chain of inequalities be separated by an additive factor of at most 2ϵ determines choices for Δ and α that yield the theorem (details omitted). □

What role does T play in the Simulation Lemma? As we make T larger, $\widehat{M}$ must be a better approximation of M in order to satisfy the conditions of the Simulation Lemma — but then we are guaranteed of the simulation accuracy of $\widehat{M}$ for a larger number of steps. If we wish to "compete" with the policies in $\Pi^{T,\epsilon}_M$, then by appealing to the Simulation Lemma using T, we ensure that the *asymptotic* return in M of any policy in $\Pi^{T,\epsilon}_M$ is well approximated by its T-*step* return in $\widehat{M}$.

Thus, the Simulation Lemma essentially determines what the definition of known state should be: one that has been visited enough times to ensure (with high probability) that the estimated transition probabilities and the estimated payoffs for the state are all within $O((\epsilon/(NTR_{max}))^2)$ of their true values. A straightforward application of Chernoff bounds shows that the desired approximation will be achieved for those states from which every action has been executed at least

$$m_{known} = O(((NTR_{max})/\epsilon)^4 \, Var_{max} \log(1/\delta)) \quad (4)$$

times, where $Var_{max} = \max(1, \max_i[Var_M(i)])$ is the maximum of 1 and the maximum variance of the random payoffs over all states.

4.3 The "Explore or Exploit" Lemma

The Simulation Lemma indicates the degree of approximation required for sufficient simulation accuracy, and led to the definition of a known state. If we let S denote the set of known states, we now specify the straightforward way in which these known states define an induced MDP. This induced MDP has an additional "new" state, which intuitively represents all of the unknown states and transitions.

Definition 5 *Let M be a Markov decision process, and let S be any subset of the states of M. The* **induced Markov decision process on** *S, denoted M_S, has states $S \cup \{s_0\}$, and transitions and payoffs defined as follows:*

- *For any state $i \in S$, $R_{M_S}(i) = R_M(i)$; all payoffs in M_S are deterministic (zero variance) even if the payoffs in M are stochastic.*
- *$R_{M_S}(s_0) = 0$.*
- *For any action a, $P^a_{M_S}(s_0 s_0) = 1$. Thus, s_0 is an absorbing state.*
- *For any states $i, j \in S$, and any action a, $P^a_{M_S}(ij) = P^a_M(ij)$. Thus, transitions in M between states in S are preserved in M_S.*
- *For any state $i \in S$ and any action a, $P^a_{M_S}(is_0) = \sum_{j \notin S} P^a_M(ij)$. Thus, all transitions in M that are not between states in S are redirected to s_0 in M_S.*

Definition 5 describes an MDP directly induced on S by the true unknown MDP M, and as such preserves the *true* transition probabilities between states in S. Of course, our algorithm will only have *approximations* to these transition probabilities, leading to the following obvious approximation to M_S: if we simply let $\widehat{M}$ denote the empirical approximation to M — that is, the states of $\widehat{M}$ are simply all the states visited so far, the transition probabilities of $\widehat{M}$ are the observed transition frequencies, and the rewards are the observed rewards — then $\widehat{M}_S$ is the natural approximation to M_S. Now if we let S be the set of known states, as defined by Equation (4), then the simulation accuracy of $\widehat{M}_S$ with respect to M_S in the sense of Equation 1 follows immediately from the Simulation Lemma. Let us also observe that any return achievable in M_S (and thus approximately achievable in $\widehat{M}_S$) is also achievable in the "real world" M — that is, for any policy π in M, any state $i \in S$, and any T, $U^\pi_{M_S}(i,T) \le U^\pi_M(i,T)$.

We are now at the heart of the analysis: we have identified a "part" of the unknown MDP M that the algorithm "knows" very well, in the form of the approximation $\widehat{M}_S$ to M_S. The key lemma follows, in which we demonstrate the fact that M_S (and thus, by the Simulation Lemma, $\widehat{M}_S$) must always provide the algorithm with *either* a policy that will yield near-optimal return in the true MDP M, *or* a policy that will allow rapid exploration of an unknown state in M (or both).

Lemma 3 *(Explore or Exploit Lemma) Let M be any Markov decision process, let S be any subset of the states of M, and let M_S be the induced Markov decision process on M. For any $i \in S$ and any T, and any $1 > \alpha > 0$, either there exists a policy π in M_S such that $U^\pi_{M_S}(i,T) \ge U^*_M(i,T) - \alpha$, or there exists a policy π in M_S such that the probability that a walk of T steps following π will terminate in s_0 exceeds α/R_{max}.*

Proof: Let π be a policy in M satisfying $U^\pi_M(i,T) = U^*_M(i,T)$, and suppose that $U^\pi_{M_S}(i,T) < U^*_M(i,T) - \alpha$ (otherwise, π already witnesses the claim of the lemma). We may write

$$\begin{aligned} U^\pi_M(i,T) &= \sum_p \mathbf{Pr}^\pi_M[p] U_M(p) \\ &= \sum_q \mathbf{Pr}^\pi_M[q] U_M(q) + \sum_r \mathbf{Pr}^\pi_M[r] U_M(r) \end{aligned}$$

where the sums are over, respectively, all T-paths p in M, all T-paths q in M in which every state in q is in S, and all T-paths r in M in which at least one state is not in S. Keeping this interpretation of the variables p, q and r fixed, we may write $\sum_q \mathbf{Pr}^\pi_M[q] U_M(q) = \sum_q \mathbf{Pr}^\pi_{M_S}[q] U_{M_S}(q) \le U^\pi_{M_S}(i,T)$. The equality follows from the fact that for any path q in which every state is in S, $\mathbf{Pr}^\pi_M[q] = \mathbf{Pr}^\pi_{M_S}[q]$

and $U_M(q) = U_{M_S}(q)$, and the inequality from the fact that $U^{\pi}_{M_S}(i,T)$ takes the sum over all T-paths in M_S, not just those that avoid the absorbing state s_0. Thus $\sum_q \mathbf{Pr}^{\pi}_M[q]U_M(q) \leq U^*_M(i,T) - \alpha$ which implies that $\sum_r \mathbf{Pr}^{\pi}_M[r]U_M(r) \geq \alpha$. But $\sum_r \mathbf{Pr}^{\pi}_M[r]U_M(r) \leq R_{max}\sum_r \mathbf{Pr}^{\pi}_M[r]$ and so $\sum_r \mathbf{Pr}^{\pi}_M[r] \geq \alpha/R_{max}$ as desired. □

4.4 Off-line Optimal Policy Computations

Let us take a moment to review and synthesize. The combination of the simulation accuracy of $\widehat{M}_S$ and the Explore or Exploit Lemma establishes our basic line of argument:

- At any time, if S is the set of current known states, the T-step return of any policy π in $\widehat{M}_S$ (approximately) lower bounds the T-step return of (any extension of) π in M.

- At any time, there must either be a policy in $\widehat{M}_S$ whose T-step return in M is nearly optimal, or there must be a policy in $\widehat{M}_S$ that will quickly reach the absorbing state — in which case, this same policy, executed in M, will quickly reach a state that is not currently in the known set S.

At certain points in the execution of the algorithm, we will perform T-step value iteration (which takes $O(N^2T)$ computation) off-line twice: once on $\widehat{M}_S$, and a second time on what we will denote $\widehat{M}'_S$. The MDP $\widehat{M}'_S$ has the same transition probabilities as $\widehat{M}_S$, but different payoffs: in $\widehat{M}'_S$, the absorbing state s_0 has payoff R_{max} and all other states have payoff 0. Thus we reward exploration (as represented by visits to s_0) rather than exploitation. If $\widehat{\pi}$ is the policy returned by value iteration on $\widehat{M}_S$ and $\widehat{\pi}'$ is the policy returned by value iteration on $\widehat{M}'_S$, then Lemma 3 guarantees that either the T-step return of $\widehat{\pi}$ from our current known state approaches the optimal achievable in M (which for now we are assuming we know, and can thus detect), or the probability that $\widehat{\pi}'$ reaches s_0, and thus that the execution of $\widehat{\pi}'$ in M reaches an unknown or unvisited state in T steps with significant probability (which we can also detect). Finally, note that even though T-step value iteration produces a non-stationary policy, it is the expected payoff that is important, not whether we follow a stationary or non-stationary policy.

4.5 Putting it All Together

All of the technical pieces we need are now in place, and we now give a more detailed description of the algorithm, and sketch the remainder of the analysis. (Again, full details are provided in the long version.) We emphasize that for now we assume the algorithm is given as input a targeted mixing time T and the optimal return $opt(\Pi^{T,\epsilon}_M)$ achievable in $\Pi^{T,\epsilon}_M$. In Section 4.6, we remove these assumptions.

We call the algorithm *Explicit Explore or Exploit*, or E^3, because whenever the algorithm is not engaged in balanced wandering, it performs an explicit off-line computation on the partial model in order to find a T-step policy guaranteed to either explore or exploit.

Explicit Explore or Exploit ($\mathbf{E^3}$) Algorithm:

- (Initialization) Initially, the set S of known states is empty.

- (Balanced Wandering) Any time the current state is not in S, the algorithm performs balanced wandering.

- (Discovery of New Known States) Any time a state i has been visited m_{known} times during balanced wandering, it enters the known set S, and no longer participates in balanced wandering.

- **Observation:** Clearly, after $N(m_{known} - 1) + 1$ steps of balanced wandering, by the Pigeonhole Principle *some* state becomes known. More generally, if the total number of steps of balanced wandering the algorithm has performed ever exceeds Nm_{known}, then *every* state of M is known (even if these steps of balanced wandering are not consecutive).

- (Off-line Optimizations) Upon reaching a known state $i \in S$ during balanced wandering, the algorithm performs the two off-line optimal policy computations on $\widehat{M}_S$ and $\widehat{M}'_S$ described in Section 4.4:
  - (Attempted Exploitation) If the resulting exploitation policy $\widehat{\pi}$ achieves return from i in $\widehat{M}_S$ that is at least $opt(\Pi^{T,\epsilon}_M) - \epsilon/2$, the algorithm executes $\widehat{\pi}$ for the next T steps.
  - (Attempted Exploration) Otherwise, the algorithm executes the resulting exploration policy $\widehat{\pi}'$ (derived from the off-line computation on $\widehat{M}'_S$) for T steps in M, which by Lemma 3 is guaranteed to have probability at least $\epsilon/(2R_{max})$ of leaving the set S.

- (Balanced Wandering) Any time an attempted exploitation or attempted exploration visits a state not in S, the algorithm immediately resumes balanced wandering.

This concludes the description of the algorithm; we can now wrap up the analysis. One of the main remaining issues is our handling of the confidence parameter δ in the statement of the main theorem: Theorem 1 ensures that a certain performance guarantee is met with probability at least $1 - \delta$. There are essentially three different sources of failure for the algorithm:

- At some known state, the algorithm actually has a poor approximation to the next-state distribution for some action, and thus $\widehat{M}_S$ does *not* have sufficiently strong simulation accuracy for M_S.
- Repeated attempted explorations fail to yield enough steps of balanced wandering to result in a new known state.
- Repeated attempted exploitations fail to result in actual return that is near $opt(\Pi_M^{T,\epsilon})$.

Our handling of the failure probability δ is to simply allocate $\delta/3$ to each of these sources of failure. The fact that we can make the probability of the first source of failure (a "bad" known state) small is handled by a standard Chernoff bound analysis applied to the definition of known states.

For the second source of failure (failed attempted explorations), a standard Chernoff bound analysis again suffices: by Lemma 3, each attempted exploration can be viewed as an independent Bernoulli trial with probability at least $\epsilon/(2R_{max})$ of "success" (at least one step of balanced wandering). In the worst case, we must make *every* state known before we can exploit, requiring Nm_{known} steps of balanced wandering. The probability of having fewer than Nm_{known} steps of balanced wandering will be smaller than $\delta/3$ if the number of (T-step) attempted explorations is $O((R_{max}/\epsilon)N\log(1/\delta)m_{known})$.

Finally, we do not want to simply halt upon finding a policy whose expected return is near $opt(\Pi_M^{T,\epsilon})$, but want to achieve *actual* return approaching $opt(\Pi_M^{T,\epsilon})$, which is where the third source of failure (failed attempted exploitations) enters. We have already argued that the total number of T-step attempted explorations the algorithm can perform before S contains all states of M is polynomially bounded. All other actions of the algorithm must be accounted for by T-step attempted exploitations. Each of these T-step attempted exploitations has expected return at least $opt(\Pi_M^{T,\epsilon}) - \epsilon/2$. The probability that the actual return, *restricted to just these attempted exploitations*, is less than $opt(\Pi_M^{T,\epsilon}) - 3\epsilon/4$, can be made smaller than $\delta/3$ if the number of blocks exceeds $O((1/\epsilon)^2\log(1/\delta))$; this is again by a standard Chernoff bound analysis. However, we also need to make sure that the return restricted to these exploitation blocks is sufficient to dominate the potentially low return of the attempted explorations. It is not difficult to show that provided the number of attempted exploitations exceeds $O(1/\epsilon)$ times the number of attempted explorations, both conditions are satisfied, for a total number of actions bounded by $O(T/\epsilon)$ times the number of attempted explorations, which is $O(NT(R_{max}/\epsilon^2)\log(1/\delta)m_{known})$. The total computation time is thus $O(N^2T/\epsilon)$ times the number of attempted explorations, and thus bounded by

$$O(N^3T(R_{max}/\epsilon^2)\log(1/\delta)m_{known}). \tag{5}$$

This concludes the proof of the main theorem. We remark that no serious attempt to minimize these worst-case bounds has been made; our immediate goal was to simply prove *polynomial* bounds in the most straightforward manner possible. It is likely that a practical implementation based on the algorithmic ideas given here would enjoy performance on natural problems that is considerably better than the current bounds indicate. (See Moore and Atkeson, 1993, for a related *heuristic* algorithm.)

4.6 Eliminating Knowledge of T and $opt(\Pi_M^{T,\epsilon})$

In order to simplify our presentation of the main theorem and the E^3 algorithm, we made the assumption that the learning algorithm knew the targeted mixing time T and the target optimal return $opt(\Pi_M^{T,\epsilon})$ achievable in this mixing time. In this section, we briefly outline the straightforward way in which these assumptions can be removed without changing the qualitative nature of the results. Details are in the long version of this paper.

In the absence of knowledge of $opt(\Pi_M^{T,\epsilon})$, the Explore or Exploit Lemma (Lemma 3) ensures us that it is safe to have a *bias towards exploration*. More precisely, any time we arrive in a known state i, we will *first* determine the exploration policy $\widehat{\pi}'$ and compute the probability that $\widehat{\pi}'$ will reach the absorbing state s_0 of $\widehat{M}'_S$ in T steps. We can then compare this probability to the lower bound $\epsilon/(2R_{max})$ of Lemma 3. As long as this lower bound is exceeded, we may execute $\widehat{\pi}'$ in an attempt to visit a state not in S. If this lower bound is *not* exceeded, Lemma 3 guarantees that the off-line computation on $\widehat{M}_S$ in the Attempted Exploitation step *must* result in an exploitation policy $\widehat{\pi}$ that is close to optimal. We execute $\widehat{\pi}$ in M and continue.

Note that this exploration-biased solution to removing knowledge of $opt(\Pi_M^{T,\epsilon})$ or $V^*(i)$ results in the algorithm always exploring all states of M that can be reached in a reasonable amount of time, before doing any exploitation. This is a simple way of removing the knowledge while keeping a polynomial-time algorithm; but we explore more practical variants of this strategy in the longer paper.

To remove the assumed knowledge of T, we observe that we already have an algorithm $A(T)$ that, given T as input, runs for $P(T)$ steps for some fixed polynomial $P(\cdot)$ and meets the desired criterion. We now propose a new algorithm A', which does not need T as input, and simply runs A sequentially for $T = 1, 2, 3, \ldots$. For any T, the amount of time A' must be run before A' has executed $A(T)$ is $\sum_{t=1}^{T} P(t) \leq TP(T) = P'(T)$, which is still polynomial in T. We just need to run A' for sufficiently many steps after the first $P'(T)$ steps to dominate any low-return periods that took place in those $P'(T)$ steps, similar to the analysis done for the undiscounted case towards the end of Section 4.5. We again note that this solution, while sufficient for polynomial time, is not the one we would implement in practice.

5 Conclusion

In this paper, we presented the E^3 algorithm, and showed that it achieves near-optimal undiscounted return in general MDP's in polynomial time. In the long version, we show that a slight modification of E^3 gives similar results for the discounted case, that the algorithms can deal with MDP's with terminating states in a natural way, and that they also work in multichain MDP's.

There are a number of interesting lines for further research. We are developing the basic ideas underlying E^3 into a practical algorithm, and hope to report on an implementation and experiments soon. Finding an efficient model-free version of our algorithm, and techniques for dealing with large state spaces, remain for future work.

Acknowledgements

We give warm thanks to Tom Dean, Tom Dietterich, Tommi Jaakkola, Leslie Kaelbling, Michael Littman, Lawrence Saul, Terry Sejnowski, and Rich Sutton for valuable comments. Satinder Singh was supported by NSF grant IIS-9711753.

References

Barto, A. G., Sutton, R. S., Watkins, C. (1990). Sequential decision problems and neural networks. In *NIPS 2*, pages 686–693, Morgan Kaufmann.

Bertsekas, D. P., Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming.* Belmont, MA: Athena Scientific.

Fiechter, C. (1994). Efficient reinforcement learning. In *COLT94*, pages 88–97. ACM Press.

Fiechter, C. (1997). Expected mistake bound model for on-line reinforcement learning. In *Machine Learning: Proceedings of the Fourteenth International Conference (ICML97)*, pages 116–124. Morgan Kaufmann.

Gullapalli, V., Barto, A. G. (1994). Convergence of indirect adaptive asynchronous value iteration algorithms. In NIPS 6, pages 695–702. Morgan Kauffman.

Jaakkola, T., Jordan, M. I., Singh, S. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, *6*(6), 1185–1201.

Moore, A. W., Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, *13*(1).

Puterman, M. L. (1994). *Markov decision processes : discrete stochastic dynamic programming.* New York: John Wiley & Sons.

Saul, L., Singh, S. (1996). Learning curve bounds for Markov decision processes with undiscounted rewards. In *COLT96.*

Schapire, R. E. , Warmuth, M. K. (1994). On the worst-case analysis of temporal-difference learning algorithms. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 266–274. Morgan Kaufmann.

Singh, S., Dayan, P. (in press). Analytical mean squared error curves for temporal difference learning. *Machine Learning.*

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

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

Thrun, S. B. (1992). The role of exploration in learning control. In White, D. A. , Sofge, D. A. (Eds.), *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches.* Florence, Kentucky 41022: Van Nostrand Reinhold.

Tsitsiklis, J. (1994). Asynchronous stochastic approximation and Q-learning. *Machine Learning*, *16*(3), 185–202.

Watkins, C. J. C. H., Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3/4), 279–292.