# Active Reinforcement Learning

**Arkady Epshteyn** AEPSHTEY@GOOGLE.COM
Google Inc., 4720 Forbes Ave, Pittsburgh, PA 15213 USA

**Adam Vogel** ACVOGEL@STANFORD.EDU
**Gerald DeJong** MREBL@UIUC.EDU
Computer Science Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA

## Abstract

When the transition probabilities and rewards of a Markov Decision Process (MDP) are known, an agent can obtain the optimal policy without any interaction with the environment. However, exact transition probabilities are difficult for experts to specify. One option left to an agent is a long and potentially costly exploration of the environment. In this paper, we propose another alternative: given initial (possibly inaccurate) specification of the MDP, the agent determines the sensitivity of the optimal policy to changes in transitions and rewards. It then focuses its exploration on the regions of space to which the optimal policy is most sensitive. We show that the proposed exploration strategy performs well on several control and planning problems.

## 1. Introduction

When the transition probabilities and rewards of an MDP are known, the optimal policy can be computed offline. However, it is unrealistic to expect a domain expert to accurately specify thousands of MDP parameters. The optimal policy computed offline in an imperfectly modeled world may turn out to be suboptimal when executed in the actual environment. To fix this problem in practice, both rewards and transition probabilities are tweaked by domain experts until the desired performance is achieved. An alternative approach is to allow the agent to explore the world in a model-free fashion using reinforcement learning (RL). However, reinforcement learning in the actual environment is time-consuming, expensive, and sometimes dangerous (Abbeel and Ng (2005), for example, describe a helicopter crash which occurred during an overly aggressive exploration).

In this work, we introduce an approach called active reinforcement learning which combines the strengths of offline planning and online exploration. In particular, our framework allows domain experts to specify possibly inaccurate models of the world offline. However, instead of using this model for planning, our algorithm uses it as a blueprint for exploration. Our approach is based on the observation that, while all of the transition probabilities and rewards in the model may be misspecified, it is not important to know all of them to determine the optimal policy. Consider a surveillance helicopter flying agent. Does it make a difference if it crashes with probability 0.9 or 0.95 when it flies close to the ground? It seems unlikely that the optimal policy would be very sensitive to this value. However, the probability of the agent taking a good photograph of its target from a given viewing angle is extremely important. Therefore, the primary goal of the agent's experimentation, given a description of the problem, should be to determine the probabilities of capturing a photo of the target as opposed to trying to determine the exact probability of crashing. Active reinforcement learning enables this type of exploration. It uses sensitivity analysis to determine how the optimal policy in the expert-specified MDP is affected by changes in transition probabilities and rewards of individual actions. This analysis guides the exploration process by forcing the agent to sample the most sensitive actions first. We will present experimental results demonstrating the effectiveness of active RL. In addition, we will show that, while our algorithm is approximate, it produces near-optimal results in polynomial time for a special class of MDPs.

## 2. Related Work

Many strategies have been proposed to address the difficulty of specifying MDPs offline. Givan's bounded-parameter MDP framework (Givan et al., 2000) allows

Appearing in *Proceedings of the 25$^{th}$ International Conference on Machine Learning*, Helsinki, Finland, 2008. 

the designer to specify uncertainty intervals around MDP's transition probabilities and rewards. The agent then finds the best policy in a game against adversarial nature which picks the worst possible world in which to evaluate it. (Alternative specifications of prior uncertainty in the same framework are given in Nilim and Ghaoui (2003).) While intuitively appealing, this approach may pick an overly conservative policy which is far from optimal for a given environment. Moreover, it places an excessive demand on the designer to quantify not only the prior model, but also his uncertainty about its transition probabilities.

In the online RL setting, there are plenty of reinforcement learning approaches that use optimistic exploration in face of uncertainty, such as $E^3$ (Kearns & Singh, 2002), $R-MAX$ (Brafman & Tennenholtz, 2002), and model-based interval exploration (Strehl & Littman, 2005). The main idea of these algorithms is to explore the actions the agent has experienced the fewest number of times in the past. This criterion does not apply to prior knowledge. In this paper, we focus on the problem of determining which states are worth exploring based solely on the prior MDP specification.

An approach similar in spirit to ours is Bayesian reinforcement learning (Dearden et al., 1999), which imposes a prior distribution over possible worlds and updates it based on interactions with the environment. However, this approach makes use of unrealistic assumptions on the shapes of probability distributions and approximate sampling to ensure tractability. The largest problem to which it was applied is two orders of magnitude smaller than the problems we solve in this work. In addition, we present an approximate version of our algorithm which is able to handle much larger (possibly continuous) state/action spaces.

The idea of using a prior MDP specification to reduce the amount of exploration in RL has also been explored by Abbeel et al. (2006). However, they only handle deterministic environments and their exploration is driven by the perceived optimal policy, not sensitivity analysis.

## 3. Preliminaries

The Markov decision process is defined by a tuple $(S, A, T, Next, R, \alpha)$, where $S = \{1, .., |S|\}$ is a finite set of states, $A$ is a finite set of actions, $R(s,a)$ is a reward function, $T(s'|s,a)$ is a transition probability function, $\alpha \in (0,1)$ is the discount factor. $Next(s,a) = \{s' : T(s'|s,a) > 0\}$ defines a set of states reachable in one step with nonzero probability after taking action $a \in A$ in a state $s \in S$. Since transition probabilities of all the states in $Next(s,a)$ are constrained to lie in the probability simplex $\Delta(Next(s,a))$, $T(\cdot|s,a)$ is a function with $|Next(s,a)|-1$ degrees of freedom. To make this explicit, let $\underline{Next}(s,a)$ denote an arbitrary state such that $T(\underline{Next}(s,a)|s,a) = 1 - \sum_{s' \in \overline{Next}(s,a)} T(s'|s,a)$, where $\overline{Next}(s,a) = Next(s,a) \backslash \underline{Next}(s,a)$ is the set of states in $Next(s,a)$ other than the state $\underline{Next}(s,a)$, and let $T|_{\overline{s,a}}$ denote the restriction of $T(\cdot|s,a)$ to $\overline{Next}(s,a)$.

Let $\pi(s)$ define a deterministic policy which maps states to actions. Let $T^\pi(s,s') = T(s'|s,\pi(s))$ be the $|S| \times |S|$ transition probability matrix and $R^\pi(s) = R(s,\pi(s))$ be the $|S| \times 1$ reward vector under $\pi$. Then the value matrix $V^\pi(T,R)$ is given by the Bellman equation $V^\pi = \alpha T^\pi V^\pi + R^\pi$. $V^\pi$ can be computed efficiently via iterative application of the Bellman equation, known as policy evaluation.

The utility of a policy $\pi$, $U^\pi(T,R)$, is given by the expected discounted rewards: $U^\pi(T,R) = E_{s_0 \sim D} V^\pi(s_0; T, R)$, with initial state $s_0$ drawn from the distribution $D$. *The utility of a policy explicitly depends on the transition and reward model of the MDP.*

We need one more piece of notation to describe the algorithm. We want to be able to take a transition probability function $T$ and replace the transition probabilities $T(\cdot|\hat{s},\hat{a})$ of a fixed state/action pair $\hat{s},\hat{a}$ with a given probability distribution $X \in \Delta(Next(\hat{s},\hat{a}))$, leaving the rest of the probabilities the same. To do this, we define the replacement function $W_{\hat{s},\hat{a}}[T,X](s'|s,a) \triangleq \begin{cases} X(s'), & \text{if } s,a = \hat{s},\hat{a} \\ T(s'|s,a), & \text{otherwise} \end{cases}$. Similarly, the function $Y_{\hat{s},\hat{a}}[R,r](s,a) \triangleq \{r \text{ if } s,a = \hat{s},\hat{a} \text{ and } R(s,a) \text{ otherwise}\}$ replaces the reward of the chosen state/action pair with $r$.

## 4. Active RL Algorithm

In this section, we give a general overview of the active reinforcement learning algorithm.

Let $T_0, R_0$ be the user-supplied model of transition probabilities and rewards for an MDP. We can use Taylor's approximation to model the local sensitivity of $U^\pi(T_0,R_0)$ as the transition probabilities $\mathbf{X} \in \Delta(Next(\hat{s},\hat{a}))$ are perturbed around some specified value $\mathbf{T_1}$ for a *single* state/action pair $\hat{s},\hat{a}$:

$$\hat{U}^\pi_{T_1}(W_{\hat{s},\hat{a}}[T_0,\mathbf{X}]) \approx U^\pi(W_{\hat{s},\hat{a}}[T_0,\mathbf{T_1}],R_0) + \nabla_{\mathbf{X}|_{\overline{\hat{s},\hat{a}}}} U^\pi(W_{\hat{s},\hat{a}}[T_0,\mathbf{T_1}],R_0)(\mathbf{X}|_{\overline{\hat{s},\hat{a}}} - \mathbf{T_1}|_{\overline{\hat{s},\hat{a}}})$$

Transition probabilities for all the actions other than the action $\hat{a}$ in state $\hat{s}$ are held fixed at the values

defined by the user-supplied model $T_0$. Similarly, the rewards of all the actions are held fixed at the user-supplied values $R_0$. Sensitivity of the utility function to changes in rewards $R(s, a)$ of individual actions is modeled in an analogous fashion.

The above approximation fixes the transition probabilities for all the actions except one, and approximates the utility of the best policy around any specified point $\mathbf{T_1}$ in the transition probability simplex of that one action. In this section, we assume that it is possible to compute this Taylor's expansion efficiently and explain the main idea of our algorithm, deferring the details of computing the gradient to Section 5.

Taylor's approximation makes it possible to determine how the payoff from following a fixed policy is affected by the changes in the MDP parameters. However, even large changes in payoffs do not necessarily mean that the agent is acting suboptimally. An extreme illustration of this is a gridworld agent who is rewarded only upon getting to the goal state. Even if the agent is wrong about the magnitude of the reward, its optimal policy remains the same: always move towards the goal. Thus, an agent could be completely wrong about the environment and still act optimally. The key goal of the sensitivity analysis is to determine how the *optimal policy* changes in response to the changes in the transition probabilities and rewards. One way to measure this sensitivity is by asking the question: how much do transition probabilities/rewards of a given action have to change before the currently optimal policy becomes suboptimal?

To make this question precise, let us first focus on the sensitivity to transition probabilities. We will use $\Pi_{T_1;\hat{s},\hat{a}} = \arg\max_\pi U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{T_1}], R_0)$ to denote the optimal policy in the MDP in which all transitions except for those of action $\hat{a}$ in state $\hat{s}$ are held fixed at $T_0$, and transitions of $\hat{s}, \hat{a}$ are given by $\mathbf{T_1}$. Let $C = \{T : U^{\Pi_{T_0;\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{T}], R_0) \geq U^{\Pi_{T;\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{T}], R_0)\}$ define a region in the transition probability space in which the optimal policy $\Pi_{T_0}$ for the user-specified MDP dominates every other policy. The goal of sensitivity analysis is to find the radius of the largest ball which we can position at $\mathbf{T_0}$ and expand without leaving the confines of $C$ along the dimensions $T(\cdot|\hat{s}, \hat{a})$ corresponding to the transitions for a given action $\hat{a}$ in $\hat{s}$. The larger the ball, the more robust the optimal policy is to the changes in the transition probabilities of the given action $\hat{a}$. However, it is not obvious how to compute this quantity exactly. Instead, we approximate it via a variant of Newton's root-finding method which starts out at some point $T_0'$ in the transition probability space outside of $C$ and converges to a point on the boundary of $C$. The method is illustrated in Figure 1. Each application of the method consists of starting out at some point $T_0'$, replacing the utility function $U^{\Pi_{T_0';\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{T_0'}], R_0)$ of the best policy at $T_0'$ with its tangent, finding the "zero" of this tangent, i.e., a point $T_1'$ closest to $T_0$ in the intersection of this tangent and the tangent to the utility function at $T_0$, replacing the initial estimate $T_0'$ with the new estimate $T_1'$, and iterating.

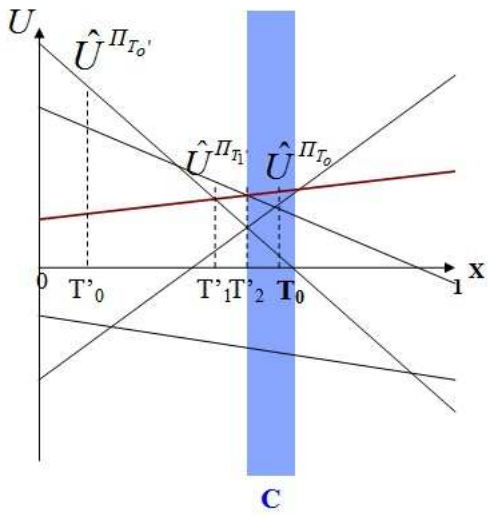


*Figure 1.* Newton's method for sensitivity analysis of MDPs. The $\mathbf{X}$-axis is the probability of transition to $s'$ for a single chosen $\hat{s}, \hat{a}$ pair that leads to one of two states $s'$ or $s''$ (the probability of transition to $s''$ is $1 - \mathbf{X}$). The utilities of policies are linear functions of $\mathbf{X}$. Function $U^{\Pi_T}(\mathbf{X})$ is the utility of the policy which is optimal when $\mathbf{X} = T$. It is given by the upper envelope of the set of all value functions. $C$ is the (blue) region where the optimal policy at $T_0$ (red line) dominates every other policy. The algorithm starts at $T_0'$ and converges to $T_2'$ on the boundary of $C$.

The pseudocode for this algorithm consists of the following steps:

1. Determine the optimal policy $\Pi_{T_0;\hat{s},\hat{a}}$ for the user-provided model using any MDP solver.
2. Determine the Taylor approximation of the utility of this policy $\hat{U}_{T_0}^{\Pi_{T_0;\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}])$ as a function of transition probabilities $\mathbf{X} \in \Delta(Next(\hat{s}, \hat{a}))$.
3. Select the starting point $T_0' \in \Delta(Next(\hat{s}, \hat{a}))$ and let $i \leftarrow 0$.
4. Using any MDP solver, determine the optimal policy $\Pi_{T_i';\hat{s},\hat{a}}$ for the user-provided model with transition probabilities of action $\hat{a}$ in state $\hat{s}$ replaced by $T_i'$.

5. Determine the Taylor approximation of the utility of this policy $\hat{U}_{T_i'}^{\Pi_{T_i';\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}])$ as a function of transition probabilities $\mathbf{X} \in \Delta(Next(\hat{s}, \hat{a}))$.

6. Let $T'_{i+1}$ be the point in the intersection of $\hat{U}_{T_0}^{\Pi_{T_0;\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}])$ and $\hat{U}_{T_i'}^{\Pi_{T_i';\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}])$ closest to the user-specified model $T_0$. Find $T'_{i+1}$ by solving the following second-order cone program[1]:

$$T'_{i+1} = \arg\min_{\mathbf{X}} \left\| \mathbf{X}|\overline{\hat{s},\hat{a}} - \mathbf{T_0}|\overline{\hat{s},\hat{a}} \right\|$$

$$\text{s.t.} \hat{U}_{T_0}^{\Pi_{T_0;\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}]) = \hat{U}_{T_i'}^{\Pi_{T_i';\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}])$$

$$[\mathbf{X}|\overline{\hat{s},\hat{a}}] \succeq \mathbf{0}; [\mathbf{X}|\overline{\hat{s},\hat{a}}]^T \boldsymbol{e} \leq 1,$$

where $\boldsymbol{e}$ is a vector of all 1's

The last set of constraints ensures that $T'_{i+1}$ is a valid probability distribution.

7. Let $i \leftarrow i+1$ and repeat steps 4-7 while $\Pi_{T_i;\hat{s},\hat{a}} \neq \Pi_{T_0;\hat{s},\hat{a}}$.

8. Return $\left\| T_i'|\overline{\hat{s},\hat{a}} - T_0|\overline{\hat{s},\hat{a}} \right\|$

The algorithm returns an estimate of the maximum-radius sensitivity ball about the user-specified transition model $T_0$. The estimate is the minimum over a set of restarts of Newton's method, initializing $T_0'$ to each vertex of the probability simplex $\Delta(Next(\hat{s}, \hat{a}))$. The quality of the estimate is limited both by the finite number of iterations of Newton's method and by the limited number of restarts. The algorithm is executed for every state/action pair $\hat{s}, \hat{a}$ to find which actions are most sensitive to changes in transition probabilities of the user-supplied model. We confirmed experimentally that the estimates of the radius of the sensitivity ball produced by our algorithm are very close to the true values when the dimensionality of the probability simplex is small.

An analogous algorithm is used to determine the sensitivity of the MDP to perturbations in individual rewards. We have not yet described how to compute the Taylor approximation of the utility function. We will do so in the next section.

## 5. Sensitivity of a Policy

By definition of the gradient and linearity of expectation, the gradient of the utility function is given by $\nabla_{\mathbf{X}|\overline{\hat{s},\hat{a}}} U^{\pi}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0) =$ $[E_{s_0 \sim D} \frac{\partial V^{\pi}(s_0; W_{\hat{s},\hat{a}}[T_0,\mathbf{X}], R_0)}{\partial X(s'_j)}]_{j=1}^{|\overline{Next}(\hat{s},\hat{a})|}; s'_j \in \overline{Next}(\hat{s}, \hat{a})$. To compute the gradient of the value function, we need the following lemma:

**Lemma 5.1.** *For a given policy $\pi$, a [state,action,next state] tuple $[\hat{s}, \hat{a}, s'] : s' \in \overline{Next}(\hat{s}, \hat{a})$, a transition function for the given $[\hat{s}, \hat{a}]$: $\mathbf{X} \in \Delta(Next(s, a))$, reward function $R_0$, let transition function $T = W_{\hat{s},\hat{a}}[T_0, \mathbf{X}]$. Then $\frac{\partial V^{\pi}(s_0;T,R_0)}{\partial X(s')} \triangleq 0$ for $\hat{a} \neq \pi(\hat{s})$. For $\hat{a} = \pi(\hat{s})$, let $V^{\pi}$ be the policy value function which satisfies the Bellman equation and let an $|S| \times |S|$ matrix $L^{\pi}$ define the directional vector for the derivative:* $L^{\pi}(s_i, s_j) \triangleq \begin{cases} 1, & \text{if } s_j = s' \\ -1, & \text{if } s_j = \underline{Next}(\hat{s}, \hat{a}) \\ 0, & \text{otherwise} \end{cases}$ *. Then the partial derivative $\frac{\partial V^{\pi}(s_0;T,R_0)}{\partial X(s')}$ can be computed from the recurrence $\frac{\partial V^{\pi}(T,R_0)}{\partial X(s')} = \alpha T^{\pi} \frac{\partial V^{\pi}(T,R_0)}{\partial X(s')} + \alpha L^{\pi} V^{\pi}$. The form of this recurrence is exactly the same as that of the Bellman equation, with the value function replaced by its derivative and the reward function replaced by $\alpha L^{\pi} V^{\pi}$. Therefore, policy evaluation can be used to compute $\frac{\partial V^{\pi}(s_0;T,R_0)}{\partial X(s')}$.*

*Proof.* (sketch) A slight modification of the analysis given in (Cao, 2003) shows that $\frac{\partial V^{\pi}(T,R_0)}{\partial X(s')} = \alpha(I - \alpha T^{\pi})^{-1} L^{\pi} V^{\pi}$, where $I$ is the $|S| \times |S|$ identity matrix. In order to compute the directional derivatives efficiently, note that this equation can be rewritten as the above recurrence. □

Applying a similar analysis to calculate the derivative of the utility function with respect to the reward $x$ for a given state/action pair $[\hat{s}, \hat{a}]$, we obtain (letting the reward function $R = Y_{\hat{s},\hat{a}}(R_0, x)$): $\frac{\partial V^{\pi}(T_0,R)}{\partial x} = \alpha T_0^{\pi} \frac{\partial V^{\pi}(T_0,R)}{\partial x} + M$ where $M(s_i) \triangleq \begin{cases} 1, \text{ if } s_i = \hat{s} \\ 0, \text{ otherwise} \end{cases}$ for $\hat{a} = \pi(\hat{s})$ and $\frac{\partial V^{\pi}(T,R)}{\partial x} \triangleq 0$ for $\hat{a} \neq \pi(\hat{s})$.

## 6. Convergence and Complexity

In this section, we consider the algorithm's convergence and complexity. The geometric structure of our algorithm is similar to policy iteration which has well-known connections to Newton's method (Puterman, 1994; Madani, 2000). Just like in policy iteration, the known local quadratic convergence of Newton's method does not ensure global polynomial time complexity. Unlike policy iteration, we cannot rely on properties of contractions to establish convergence. However, we can establish convergence for MDPs whose structures (given by transitions with nonzero probabilities) are directed acyclic graphs (DAGs). For

[1] Second-order cone programs (SOCPs) are a special case of semidefinite programs which can be solved more efficiently, see (Lobo et al., 1998) for an overview.

such MDPs, the values $U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0)$ are linear functions of $\mathbf{X}|\hat{s},\hat{a}$ for single state/action pairs $[\hat{s},\hat{a}]$. We have the following result which follows from the Intermediate Value Theorem:

**Theorem 6.1.** *Suppose that for a given $\hat{s},\hat{a}$ and for every policy $\pi$, $U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}]; R_0)$ is a linear function of $\mathbf{X}|\hat{s},\hat{a}$. Then, for any initial point $T'_0 \in \Delta(Next(\hat{s},\hat{a}))$, the sequence $\{T'_i\}$ generated by the active RL algorithm converges to the boundary of $C$ in a finite number of steps.*

For DAG-structured MDPs such that the maximum number of $Next$ states for any state/action pair is two, much stronger guarantees are available. As pointed out in (Madani, 2000), any DAG-structured MDP can be converted in polynomial time into an MDP of this form by introducing extra states and transitions as necessary. The significance of these MDPs is that, since $\overline{Next}(\hat{s},\hat{a})$ is a singleton, $U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0)$ is a linear function of a single variable $\mathbf{X}|\hat{s},\hat{a}$. The following theorem shows that our algorithm can determine the radius of the region $C$ in logarithmic time in this degenerate case:

**Theorem 6.2.** *Define the distance between two policies $\pi$ and $\pi'$ as $max_{\mathbf{X}}|U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0) - U^{\pi'}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0)|$. Let $\gamma$ be the smallest distance larger than 0 between any policy and $\Pi_{T_0;\hat{s},\hat{a}}$. Let the MDP have bounded rewards: $|R_0(s,a)| \leq M$ for $\forall s,a$. Then, after $t = O(\log(\frac{M}{\gamma\epsilon(1-\alpha)}))$ iterations, the iterates $T'_{i\geq t}$ of Newton's method are within $\epsilon$ of the limit point on the boundary of $C$.*

*Proof.* (sketch) The proof follows from a known fact that one-dimensional Newton's method makes progress in each iteration by either exponentially decreasing the height or exponentially increasing the slope of the function $U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0)$. (Madani, 2000). □

If the structure of an MDP is not a DAG, then $U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0)$ is not linear in $\mathbf{X}|\hat{s},\hat{a}$ and the above convergence results no longer apply. However, as the discount factor $\alpha \to 0$, the value function for any MDP will become approximately linear as the influence of distant rewards becomes negligible. Thus, Newton's method offers a way to find an approximate solution to our problem which becomes more accurate as the discount factor decreases. For general root-finding problems, Newton's method need not converge (i.e., it may cycle or diverge to infinity). It is possible that our variant may also exhibit this undesirable behavior when applied to arbitrary MDPs. However, our experimental results in Section 8 demonstrate that our method works well in practice on a wide range of problems even when strong assumptions required to obtain theoretical guarantees for convergence are significantly violated.

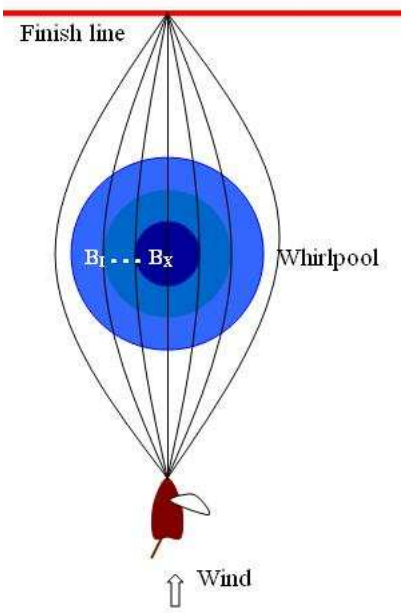


*Figure 2.* Sailboat Domain. Black lines indicate the paths of the boat for a set of possible initial policies. The boat is controlled by the rudder and the sail.

## 7. Approximate Active RL

In worlds with very large or continuous state/action spaces, the exact Active RL algorithm of Section 4 is intractable. Moreover, in continuous state spaces, sensitivity of individual state/action pairs no longer applies. Instead, we consider the case where the state space is partitioned into regions $\mathbb{B}$ with the uncertainty in our transition model for each region generated by a random variable.

Formally, we assume that the world is governed by the control model $s_{t+1} = f(s_t, a, d(\mathbb{B}(s_t))$. The agent's state $s$ at time $t+1$ is a (possibly nonlinear) function $f$ of the agent's state at time $t$, its action $a$, and the disturbance input $d$. In every region $\hat{B} \in \mathbb{B}$, the disturbance $d(\hat{B})$ is a random variable with a probability distribution $T \in \Delta(Next(\hat{B}))$ defined on a discrete set $Next(\hat{B}) = \underline{Next}(\hat{B}) \cup \overline{Next}(\hat{B})$. The approximate active RL procedure determines which regions $\hat{B} \in \mathbb{B}$ affect the optimal policy the most.

Consider, for example, the sailing problem illustrated in Figure 2. The agent's goal in this domain is to get

the sailboat to the finish line. The sailboat is controlled by the rudder and the sail. The problem is complicated by a probabilistic whirlpool which could aid the agent by increasing its speed or detain it by deviating the boat from its course. In this case, the function $f$ describes the sailboat dynamics given the boat's position, the rudder/sail action, the deterministic wind, and the whirlpool current $d$. The strength of the current and its direction are given by a distribution $T$ which depends on the band $\hat{B}$ inside the whirlpool in which the agent finds itself. Approximate active RL helps us find a small number of bands in which the current determines which one of the policies shown in Figure 2 is optimal.

The approximate active RL procedure is the same as its exact variant, except that: 1) local policy search is used in place of an MDP solver[2], 2) the utility $\widetilde{U}^{\pi}(T,R)$ of any policy $\pi$ is approximated by Markov Chain Monte Carlo, and 3) the Taylor approximation of a policy's utility is given by $\hat{U}^{\pi}_{T_1}(W_{\hat{B}}[T_0,\mathbf{X}]) \approx \widetilde{U}^{\pi}(W_{\hat{B}}[T_0,\mathbf{T_1}],R_0) + \nabla_{\mathbf{X}|\overline{\hat{B}}} U^{\pi}(W_{\hat{B}}[T_0,\mathbf{T_1}],R_0)(\mathbf{X}|\overline{\hat{B}} - \mathbf{T_1}|\overline{\hat{B}})$, where $W_{\hat{B}}[T_0;\mathbf{T_1}]$ is a world in which the disturbance distribution in band $\hat{B}$ is replaced with $\mathbf{T_1}$, and the gradient of the utility function is approximated linearly by perturbing the transition probabilities by a small value $\epsilon$ in each dimension of the probability simplex: $\nabla_{\mathbf{X}|\overline{\hat{B}}} U^{\pi}(W_{\hat{B}}[T_0,\mathbf{T_1}],R_0) \approx \frac{1}{\epsilon}[\widetilde{U}^{\pi}(W_{\hat{B}}[T_0,\mathbf{T_1^{j;\hat{B}}}],R_0) - \widetilde{U}^{\pi}(W_{\hat{B}}[T_0,\mathbf{T_1}],R_0)]_{j=1}^{\overline{Next}(\hat{B})}$ where $\mathbf{T_1^{j;\hat{B}}}$ denotes the world in which the transitions in region $\hat{B}$ are perturbed by $\epsilon$ as follows:

$$\mathbf{T_1^{j;\hat{B}}}(d') \triangleq \begin{cases} \mathbf{T_1}(d') + \epsilon, & \text{if } d' = d_j \\ \mathbf{T_1}(d') - \epsilon, & \text{if } d' = \underline{Next}(\hat{B}) \\ \mathbf{T_1}(d'), & \text{otherwise} \end{cases}$$

## 8. Experiments

Exact RL experiments were performed on the following domains:

- In the mountain-car task (231 states), the problem is to drive a car up a steep mountain (Sutton & Barto, 1998). The engine power is a uniform random variable on the interval $[.15, .3]$.
- The task in the cart-pole problem (5832 states) is to balance a pole on a moving cart. The power of the cart is random, uniformly distributed in the interval $[20, 75]$.
- Windy gridworld is a simple $10 \times 7$ gridworld with agent's movement affected by stochastic wind (Sutton & Barto, 1998).
- Pizza delivery problem (4769 states) is based on the racetrack example (Sutton & Barto, 1998). The agent's goal is to drive a car to the finish line, while delivering as many pizzas and avoiding as many randomly placed potholes as possible.
- The drunkard's walk problems are two $10 \times 10$ gridworlds with random rewards and penalties. When the agent moves in some direction, it is equally likely to deviate diagonally from it.

In all the setups, $\alpha = 0.9$ was used. The structure of the MDPs varies widely from one problem to another (and none of them are DAGs). In the first set of experiments, the effectiveness of active reinforcement learning for transition probabilities was evaluated. The system was provided with an initial description of the problem, as given above. It then performed the sensitivity analysis and sorted state/action pairs based on their sensitivity values. A different problem specification was then generated by randomly perturbing all the transition probabilities. This new specification represented the actual world in which transition probabilities are different from the expert-provided MDP specification. The agent was allowed to sample one action at a time in this actual world[3], replace user-specified transition probabilities with their maximum likelihood estimates (based on 10,000 samples), use an MDP solver to find the optimal policy in this "corrected" MDP, and evaluate this policy in the actual world. We tested two different ways of selecting the order in which actions were tested: 1) the active RL agent which samples the actions in order of decreasing sensitivity, with sensitivities computed by the algorithm in Section 4, and 2) the random agent which samples actions randomly. For comparison, we also tested two agents applying fixed policies: 1) the prior agent which applies the optimal policy for the expert-provided MDP specification, and 2) the omniscient agent which knows the transition probabilities in the actual test world and selects the optimal policy for this world. In addition, we tested the Q-learning agent with $\epsilon-$greedy exploration ($\epsilon = 0.1$) and full backups (full backups means that after taking an action in a state, the agent gets full information about all the

[2] Any local policy search algorithm, such as policy gradient or dynamic programming, can be used for this procedure.

[3] This exploration strategy (sampling with resets) assumes that the agent can execute any action in any state and observe its outcome without having to plan how to get to that state.

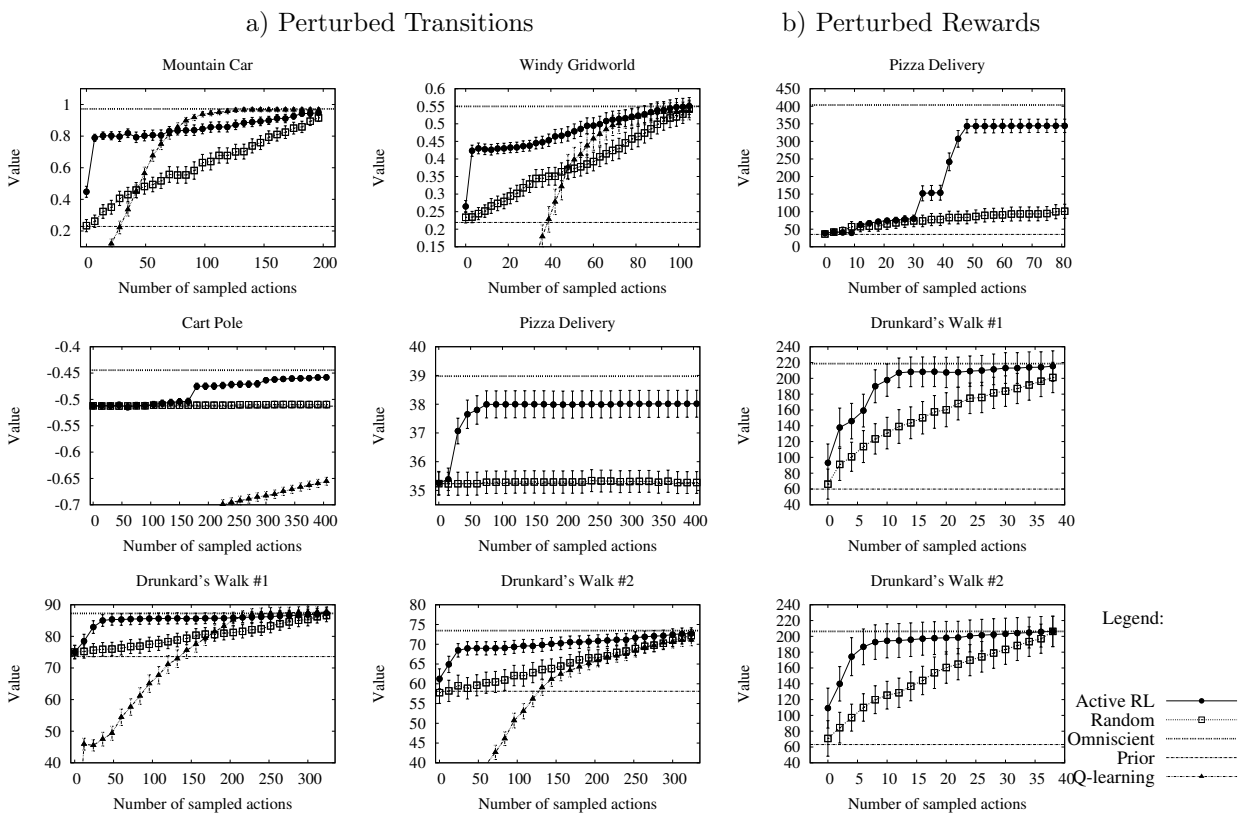


*Figure 3.* Evaluation of exploration strategies on perturbed MDPs. Error bars are based on 95% confidence intervals.

transition probabilities for all the possible next states for that action - this model was used to make the comparison between the Q-learning agent and the Active RL agent fair). The utility of the policy of each of these five agents appears in the plots in Figure 3-(a) as a function of the number of actions tested. The results are averaged over 100 different randomly generated actual worlds[4]. In each domain, the active RL agent outperforms the random sampling agent. The prior agent which relies solely on the expert's specification performs poorly, indicating the need for exploration. As expected, the Q-learning agent performs poorly initially since it has no prior knowledge, but improves with experience. It rarely catches up with the Active RL agent because Q-learning is forced to explore without resets. These experiments indicate that integrating Active RL with Q-learning may result in improvement in the Q-learning agent's performance. This is an important future extension of our work.

In the next experiment, the random worlds were generated by perturbing the rewards rather than transition probabilities of the MDP[5]. Performance of the four exploration strategies on the three domains with non-trivial reward structure appear in Figure 3-(b). Once again, the active RL agent significantly outperforms the random sampling agent.

Finally, we experimented with approximate active RL in the sailboat simulation, in which the agent must navigate a whirlpool of water current to reach the finish line. The whirlpool was modeled by ten concentric bands based on the distance from the center of the vertex, and the magnitude of the current varied proportionally to this distance. The expert-specified world reflected uncertainty about the direction of the whirlpool current: the direction of the current was counterclockwise with probability 0.1, clockwise with probability 0.1, and there was no current with prob-

[4]The test worlds were generated by perturbing transition probabilities of each action uniformly in the probability simplex within a radius of 0.6 around the expert-specified values $T_0$. A random variable $T \in \mathbb{R}^n$ uniformly distributed on the $n$-dimensional probability simplex can be generated from $n-1$ random variables $X_1, .., X_{n-1} \sim Uniform(0,1)$ by sorting them into $X_{(0)} \triangleq 0, X_{(1)}, .., X_{(n-1)}, X_{(n)} \triangleq 1$, and letting $T_i = X_{(i)} - X_{(i-1)}$ (Devroye, 1986). Rejection sampling is then used to ensure that $\|T - T_0\| \leq 0.6$.

[5]The test worlds were generated by perturbing all the nonzero rewards uniformly in the interval $[-70, 70]$ around the expert-specified values.

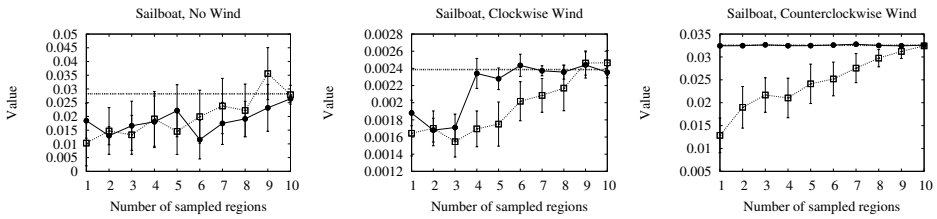


*Figure 4.* Evaluation of exploration strategies in the approximation architecture. The legend is the same as in figure 3, but with Prior strategy not shown (its value is too low to appear on the plots). Error bars are based on 95% confidence intervals.

ability 0.8. In each iteration of the active RL algorithm, local policy search was performed from each of the seven policies shown in Figure 2, and the best policy was selected. Approximate Active RL was tested in three actual worlds: one with a deterministic clockwise current, one with a deterministic counterclockwise current, and one with no current. The results appear in Figure 4. In the two worlds with current, the algorithm which samples bands according to the active RL-prescribed order outperforms the algorithm which samples bands according to a random order. In the world with no current, the performance of the two algorithms is similar.

## 9. Conclusions

In this paper, we presented a new algorithm for combining exploration with prior knowledge in reinforcement learning. We demonstrated that our algorithm can be implemented efficiently using policy iteration and a standard SOCP solver. We also introduced an approximate version of active RL to be applied in domains with large state spaces. In addition to being useful for exploration, the active RL algorithm can be used by an MDP designer to determine which regions of the state space require most precision in specifying transition probabilities and rewards. An important future extension of this work is designing a policy which explores the sensitive regions of the state space without resets.

## References

Abbeel, P., & Ng, A. Y. (2005). Exploration and apprenticeship learning in reinforcement learning. *ICML*.

Abbeel, P., Quigley, M., & Ng, A. (2006). Using inaccurate models in reinforcement learning. *ICML*.

Brafman, R. I., & Tennenholtz, M. (2002). R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, *3*, 213–231.

Cao, X. (2003). From perturbation analysis to markov decision processes and reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, *13*, 9–39.

Dearden, R., Friedman, N., & Andre, D. (1999). Model based bayesian exploration. *Uncertainty in Artificial Intelligence*.

Devroye, L. (1986). *Non-uniform random variate generation*. New York, NY: Springer-Verlag.

Givan, R., Leach, S., & Dean, T. (2000). Bounded-parameter markov decision processes. *Artificial Intelligence*, *122*, 71–109.

Kearns, M., & Singh, S. (2002). Near optimal reinforcement learning in polynomial time. *Machine Learning*, *49*, 209–232.

Lobo, M. S., Vandenberghe, L., Boyd, S., & Lebret, H. (1998). Applications of second-order cone programming. *Linear Algebra and its Applications*, *284*, 193–228.

Madani, O. (2000). *Complexity results for infinite-horizon markov decision processes*. Doctoral dissertation, University of Washington.

Nilim, A., & Ghaoui, L. E. (2003). Robustness in markov decision problems with uncertain transition matrices. *NIPS*.

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

Strehl, A. L., & Littman, M. L. (2005). A theoretical analysis of model-based interval estimation. *ICML*.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.