Potential-based Shaping in Model-based Reinforcement Learning

John Asmuth and Michael L. Littman and Robert Zinkov
RL^3 Laboratory, Department of Computer Science
Rutgers University, Piscataway, NJ USA
jasmuth@gmail.com and mlittman@cs.rutgers.edu and rzinkov@eden.rutgers.edu

Abstract

Potential-based shaping was designed as a way of introducing background knowledge into model-free reinforcement-learning algorithms. By identifying states that are likely to have high value, this approach can decrease experience complexity—the number of trials needed to find near-optimal behavior. An orthogonal way of decreasing experience complexity is to use a model-based learning approach, building and exploiting an explicit transition model. In this paper, we show how potential-based shaping can be redefined to work in the model-based setting to produce an algorithm that shares the benefits of both ideas.

Introduction

Like *reinforcement learning*, the term *shaping* comes from the animal-learning literature. In training animals, shaping is the idea of directly rewarding behaviors that are needed for the main task being learned. Similarly, in reinforcement-learning parlance, shaping means introducing "hints" about optimal behavior into the reward function so as to accelerate the learning process.

This paper examines potential-based shaping functions, which introduce artificial rewards in a particular form that is guaranteed to leave the optimal behavior unchanged yet can influence the agent's exploration behavior to decrease the time spent trying suboptimal actions. It addresses how shaping functions can be used with model-based learning, specifically the Rmax algorithm (Brafman & Tennenholtz 2002).

The paper makes three main contributions. First, it demonstrates a concrete way of using shaping functions with model-based learning algorithms and relates it to model-free shaping and the Rmax algorithm. Second, it argues that "admissible" shaping functions are of particular interest because they do not interfere with Rmax's ability to identify near-optimal policies quickly. Third, it presents computational experiments that show how model-based learning, shaping, and their combination can speed up learning.

The next section provides background on reinforcement-learning algorithms. The following one defines different notions of return as a way of showing the effect of different ways of incorporating shaping into both model-based and model-free algorithms. Next, admissible shaping functions are defined and analyzed. Finally, the experimental section applies both model-based/model-free algorithms in both shaped/unshaped forms to two simple benchmark problems to illustrate the benefits of combining shaping with model-based learning.

Background

We use the following notation. A *Markov decision process* (MDP) (Puterman 1994) is defined by a set of states S, actions A, reward function $R : S \times A \rightarrow \Re$, transition function $T : S \times A \rightarrow \Pi(S)$ ($\Pi(\cdot)$ is a discrete probability distribution over the given set), and discount factor $0 \leq \gamma \leq 1$ for downweighting future rewards. If $\gamma = 1$, we assume all non-terminal rewards are non-positive and that there is a zero-reward absorbing state (goal) that can be reached from all other states. Agents attempt to maximize cumulative expected discounted reward (expected return). We assume that all reward values are bounded above by the value $R_{\max}$. We define $v_{\max}$ to be the largest expected return attainable from any state—if it is unknown, it can be derived from $v_{\max} = R_{\max}/(1-\gamma)$ if $\gamma < 1$ and $v_{\max} = R_{\max}$ if $\gamma = 1$ (because of the assumptions described above).

An MDP can be solved to find an optimal policy—a way of choosing actions in states to maximize expected return. For any state $s \in S$, it is optimal to choose any $a \in A$ that maximizes $Q(s, a)$ defined by the simultaneous equations:

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q(s',a').$$

In the reinforcement-learning (RL) setting, an agent interacts with the MDP, taking actions and observing state transitions and rewards. Two well-studied families of RL algorithms are *model-free* and *model-based* algorithms. Although the distinction can be fuzzy, in this paper, we examine classic examples of these families—Q-learning (Watkins & Dayan 1992) and Rmax (Brafman & Tennenholtz 2002).

An RL algorithm takes experience tuples as input and produces action decisions as output. An experience tuple $\langle s, a, s', r \rangle$ represents a single transition from state s to s' under the influence of action a. The value r is the immediate reward received on this transition. We next describe two concrete RL algorithms.

Q-learning is a model-free algorithm that maintains an estimate $\hat{Q}$ of the Q function Q. Starting from some initial values, each experience tuple $\langle s, a, s', r \rangle$ changes the Q function estimate via

$$\hat{Q}(s,a) \leftarrow_\alpha r + \gamma \max_{a'} \hat{Q}(s',a'),$$

where α is a learning rate and "$x \leftarrow_\alpha y$" means that x is assigned the value $(1-\alpha)x + \alpha y$. The Q function estimate is used to make an action decision in state $s \in S$, usually by choosing the $a \in A$ such that $\hat{Q}(s,a)$ is maximized. Q-learning is actually a family algorithms. A complete Q-learning specification includes rules for initializing $\hat{Q}$, for changing α over time, and for selecting actions. The primary theoretical guarantee provided by Q-learning is that $\hat{Q}$ converges to Q in the limit of infinite experience if α is decreased at the right rate and all s, a pairs begin an infinite number of experience tuples.

A prototypical model-based algorithm is Rmax, which creates "optimistic" estimates $\hat{T}$ and $\hat{R}$ of T and R. Specifically, Rmax hypothesizes a state $s_{\max}$ such that $\hat{T}(s_{\max}, a, s_{\max}) = 1$, $\hat{R}(s_{\max}, a) = 0$ for all a. It also initializes $\hat{T}(s, a, s_{\max}) = 1$ and $\hat{R}(s,a) = v_{\max}$ for all s and a. (Unmentioned transitions have probability 0.) It keeps a transition count $c(s,a,s')$ and a reward total $t(s,a)$, both initialized to zero. In its practical form, it also has an integer parameter $m \geq 0$. Each experience tuple $\langle s, a, s', r \rangle$ results in an update $c(s,a,s') \leftarrow c(s,a,s') + 1$ and $t(s,a) \leftarrow t(s,a) + r$. Then, if $\sum_{s'} c(s,a,s') = m$, the algorithm modifies $\hat{T}(s,a,s') = c(s,a,s')/m$ and $\hat{R}(s,a) = t(s,a)/m$. At this point, the algorithm computes

$$\hat{Q}(s,a) = \hat{R}(s,a) + \gamma \sum_{s'} \hat{T}(s,a,s') \max_{a'} \hat{Q}(s',a')$$

for all state–actions pairs. When in state s, the algorithm chooses the action that maximizes $\hat{Q}(s,a)$. Rmax guarantees, with high probability, that it will take near-optimal actions on all but a polynomial number of steps if m is set sufficiently high (Kakade 2003).

We refer to any state–action pair s, a such that $\sum_{s'} c(s,a,s') < m$ as *unknown* and otherwise *known*. Notice that, each time a state–action pair becomes known, Rmax performs a great deal of computation, solving an approximate MDP model. In contrast, Q-learning consistently has low per-experience computational complexity. The principle advantage of Rmax, then, is that it makes more efficient use of the experience it gathers at the cost of increased computational complexity.

Shaping and model-based learning have both been shown to decrease experience complexity compared to a straightforward implementation of Q-learning. In this paper, we show that the benefits of these two ideas are orthogonal—their combination is more effective than either approach alone. The next section formalizes the effect of potential-based shaping in both model-free and model-based settings.

Comparing Definitions of Return

We next discuss several notions of how a trajectory is summarized by a single value, the *return*. We address the original notion of return, shaped return, Rmax return, and finally shaped Rmax return.

Original Return

Let's consider an infinite sequence of states, actions, and rewards: $\bar{s} = s_0, a_0, r_0, \ldots, s_t, a_t, r_t, \ldots$. In the discounted framework, the return for this sequence is taken to be $U(\bar{s}) = \sum_{i=0}^{\infty} \gamma^i r_i$.

Note that the Q function for any policy can be decomposed into an average of returns like this:

$$Q(s,a) = \sum_{\bar{s}} \Pr(\bar{s}|s,a) U(\bar{s}).$$

Here, with the averaging weights $\Pr(\bar{s}|s,a)$ depend on the likelihood of the sequence as determined by the dynamics of the MDP and the agent's policy.

Shaping Return

In potential-based shaping (Ng, Harada, & Russell 1999), the system designer provides the agent with a *shaping function* $\Phi(s)$, which maps each state to a real value. For shaping to be successful, it is noted that Φ should be higher for states with higher return.

The shaping function is used to modify the reward for a transition from s to s' by adding $F(s,s') = \gamma\Phi(s') - \Phi(s)$ to it. The Φ-shaped return for $\bar{s}$ becomes

$$\begin{aligned}
U_\Phi(\bar{s}) &= \sum_{i=0}^{\infty} \gamma^i (r_i + F(s_i, s_{i+1})) \\
&= \sum_{i=0}^{\infty} \gamma^i (r_i + \gamma\Phi(s_{i+1}) - \Phi(s_i)) \\
&= \sum_{i=0}^{\infty} \gamma^i r_i + \sum_{i=0}^{\infty} \gamma^{i+1}\Phi(s_{i+1}) - \sum_{i=0}^{\infty} \gamma^i \Phi(s_i) \\
&= U(\bar{s}) + \sum_{i=1}^{\infty} \gamma^i \Phi(s_i) - \Phi(s_0) - \sum_{i=1}^{\infty} \gamma^i \Phi(s_i) \\
&= U(\bar{s}) - \Phi(s_0).
\end{aligned}$$

That is, the shaped return is the (unshaped) return minus the shaping function's value for the starting state. Since the shaped return U_Φ does not depend on the actions or any states past the initial state, the shaped Q function is similarly just a shift downward for each state: $Q_\Phi(s,a) = Q(s,a) - \Phi(s)$. Therefore, the optimal policy for the shaped MDP (the MDP with shaping rewards added in) is precisely the same as that of the original MDP (Ng, Harada, & Russell 1999).

Rmax Return

In the context of the Rmax algorithm, at an intermediate stage of learning, the algorithm has built a partial model of the environment. In the known states, accurate estimates of the transitions and rewards are available. When an unknown state is reached, the agent assumes its value is $v_{\max}$, that is, an upper bound on the largest possible value.

For simplicity of analysis, we consider how the notion of return is adapted in the Rmax framework if all rewards take on their true values until an unknown state–action pair is reached. In this setting, the estimated return for a trajectory in Rmax is

$$U_{\text{Rmax}}(\bar{s}) = \sum_{i=0}^{T-1} \gamma^i r_i + \gamma^T v_{\max},$$

where s_T, a_T is the first unknown state–action pair in the sequence $\bar{s}$. Thus, the Rmax return is taken to be the truncated return plus a discounted factor of $v_{\max}$ that depends only on the timestep on which an unknown state is reached. Note that the Rmax return for a trajectory $\bar{s}$ that does not reach an unknown state is precisely the original return, $U_{\text{Rmax}}(\bar{s}) = U(\bar{s})$.

Shaped Rmax Return

We next define a new notion of return that involves shaping and known/unknown states and relate it Rmax.

This definition of return shapes all the rewards from transitions between known states:

$$\begin{aligned} U_{\Phi}^{\text{Rmax}}(\bar{s}) &= \sum_{i=0}^{T-1} \gamma^i (r_i + F(s_i, s_{i+1})) \\ &= \sum_{i=0}^{T-1} \gamma^i r_i - \Phi(s_0) + \gamma^T \Phi(s_T) \\ &= U_{\text{Rmax}}(\bar{s}) - \gamma^T v_{\max} - \Phi(s_0) + \gamma^T \Phi(s_T). \end{aligned}$$

Note that in an MDP in which all state–action pairs are known, this equation is algebraically equivalent to $U_\Phi(\bar{s})$ (T is effectively infinite). So, this rule can be seen as a generalization of shaping to a setting in which there are both known and unknown states.

Also, if we take $\Phi(s) = v_{\max}$, we have $U_{\Phi}^{\text{Rmax}}(\bar{s}) = U_{\text{Rmax}}(\bar{s}) - v_{\max}$. That is, the result is precisely the Rmax return shifted by the constant $v_{\max}$. It is interesting to note that this choice of shaping function results in precisely the original Rmax algorithm. That is, since the Q function is shifted by a constant, an agent following the Rmax algorithm and one following this shaped version of Rmax would make precisely the same sequence of actions. Thus, this rule can also be seen as a generalization of Rmax to a setting in which a more general shaping function can be used.

In its general form, the return of this rule is like that of Rmax, except shifted down by the shaped value of the starting state (which has no impact on the optimal policy) and then shifted *up* by a discounted factor of the first state reached from which an unknown action is taken. This application of the shaping idea encourages the agent to explore unknown states with higher values of $\Phi(s)$. In addition, if $\gamma < 1$, nearby states are more likely to be chosen first. As a result, it captures the same intuition for shaping that drives its use in the Q-learning setting—when faced with an unfamiliar situation, explore states with higher values of the shaping function. For this reason, we refer to this algorithm as "Rmax with shaping".

Note that we can achieve equivalent behavior to Rmax with shaping by only modifying the reward for unknown state–action pairs. That is, if we define the reward for the first transition from a known state s to an unknown state s' as being incremented by $\Phi(s')$, precisely the same exploration policy results. This equivalence between two views of model-based shaping approaches is reminiscent of the equivalence between potential-based shaping and value-function initialization in the model-free setting (Wiewiora 2003).

Admissible Shaping Functions

As mentioned earlier, Q-learning comes with a guarantee that the estimated Q values will converge to the true Q values given that all state–action pairs are sampled infinitely often and that the learning rate is decayed appropriately (Watkins & Dayan 1992). Because of the properties of potential-based shaping functions (Ng, Harada, & Russell 1999), these guarantees also hold for Q-learning with shaping. So, adding shaping to Q-learning results in an algorithm with the same asymptotic guarantees *and* the possibility of learning much more rapidly.

Rmax's guarantees are of a different form. Given that the m parameter is set appropriately as a function of parameters $\delta > 0$ and $\epsilon > 0$, Rmax achieves a guarantee known as PAC-MDP (probably approximately correct for MDPs) (Brafman & Tennenholtz 2002; Ng, Harada, & Russell 1999; Strehl, Li, & Littman 2006). Specifically, with probability $1 - \delta$, Rmax will execute a policy that is within ϵ of optimal on all but a polynomial number of steps. The polynomial in question depends on the size of the state space $|S|$, the size of the action space $|A|$, $1/\delta$, $1/\epsilon$, $R_{\max}$, and $1/(1-\gamma)$.

Next, we argue that Rmax with shaping is also PAC-MDP if and only if it uses an *admissible* shaping function.

Definition

It is well known from the field of search that heuristic functions can significantly speed up the process of finding a shortest path to a goal. A heuristic function is called *admissible* if it assigns a value to every state in the search space that is no larger than the shortest distance to the goal for that state. Admissible heuristics can speed up search in the context of the A* algorithm without the possibility of the search finding a longer-than-necessary path (Russell & Norvig 1994). Similar ideas have been used in the MDP setting (Hansen & Zilberstein 1999; Barto, Bradtke, & Singh 1995), so the importance of admissibility in search is well established.

In the context of Rmax, we define a shaping function Φ to be an *admissible shaping function* for an MDP if $\Phi(s) \geq V(s)$, where $V(s) = \max_a Q(s,a)$. That is, the shaping function is an upper bound on the actual value of the state. Note that $\Phi(s) = v_{\max}$ (the unshaped Rmax algorithm) is admissible.

Properties

Admissible shaping functions have two properties that make them particularly interesting. First, if Φ is admissible, then

Rmax with shaping is PAC-MDP. Second, if Φ is not admissible, then Rmax with shaping need not be PAC-MDP.

To justify the first claim, we build on the results of Strehl, Li, & Littman (2006). In particular, Proposition 1 of that paper showed that an algorithm is PAC-MDP if it satisfies three properties—optimism, accuracy, and bounded learning complexity. In the case of Rmax with shaping, accuracy and bounded learning complexity follow precisely from the original Rmax analysis (assuming, reasonably, that $\Phi(s) \leq v_{\max}$). Optimism here means that the Q function estimate $\hat{Q}$ is unlikely to be more than ϵ below the Q function Q (or, in this case, Q_Φ). This property can be proven from the admissibility property of Φ:

$$\begin{aligned}
&\hat{Q}(s,a) \\
&\quad = \sum_{\bar{s}} \Pr(\bar{s}|s,a) U_{\Phi}^{\text{Rmax}}(\bar{s}) \\
&\quad = \sum_{\bar{s}} \Pr(\bar{s}|s,a)(U_{\text{Rmax}}(\bar{s}) - \gamma^T v_{\max} - \Phi(s_0) + \gamma^T \Phi(s_T)) \\
&\quad \geq \sum_{\bar{s}} \Pr(\bar{s}|s,a)(U(\bar{s}) - \Phi(s_0)) \\
&\quad = Q(s,a) - \Phi(s_0) \\
&\quad = Q_\Phi(s,a).
\end{aligned}$$

Thus, running Rmax with any admissible shaping function leads to a PAC-MDP learning algorithm.

Figure 2(a) demonstrates why not using an admissible shaping function can cause problems for Rmax with shaping. Consider the case where s_0 and s_1 are known and s_2 is unknown. Rewards are zero and $v_{\max} = 0$. So, $\hat{Q}(s_0, a_1) = V(s_1)$ and $\hat{Q}(s_0, a_2) = \Phi(s_2)$. If $V(s_1) > \Phi(s_2)$, the agent will never feel the need to take action a_1 when in state s_0, since it believes that such a policy would be suboptimal. However, if $V(s_2) > V(s_1)$, which can happen if Φ is not admissible, the agent will behave suboptimally indefinitely.

Experiments

We evaluated six algorithms to observe the effects shaping has on a model-free algorithm, Q-learning, a model-based algorithm, Rmax, and ARTDP (Barto, Bradtke, & Singh 1995), which has elements of both. We ran the algorithms in two novel test environments.[1]

RTDP (Real-Time Dyanmic Programming) is an algorithm for solving MDPs with known connections to the theory of admissible heuristic. The RL version, Adaptive RTDP (ARTDP), combines learning models and heuristics and is therefore a natural point of comparison for Rmax with shaping. Lacking adequate space for a detailed comparison, we note that ARTDP is like Rmax with shaping except with (1) $m = 1$, (2) Boltzmann exploration, and (3) incremental planning. Difference (1) means ARTDP is not PAC-MDP. Difference (3) means its per-experience computational complexity is closer to that of Q-learning. In our tests, we ran

[1]The benchmark environments used by Ng, Harada, & Russell (1999) were not sufficiently challenging—Q-learning with shaping learned almost instantly in these problems.

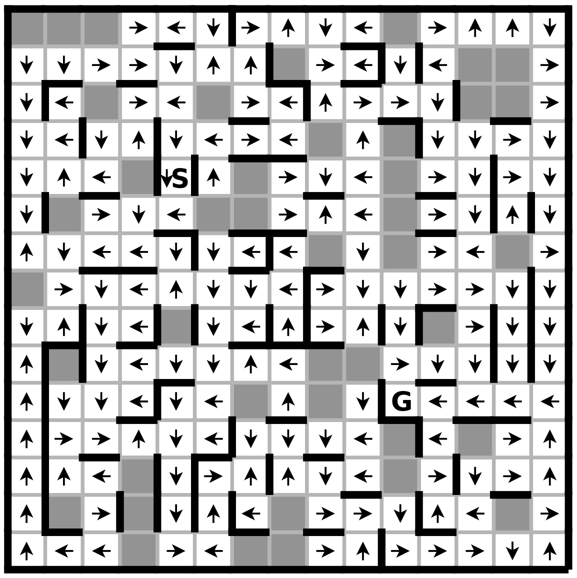

Figure 1: A 15x15 grid with a fixed start (S) and goal (G) location. Gray squares are episode-ending pits.

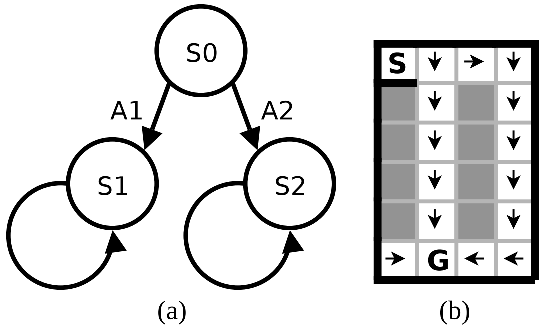

Figure 2: (a) Simple MDP to illustrate why inadmissible shaping functions can lead to learning suboptimal policies. (b) A 4x6 grid.

ARTDP with exploration parameters $T_{\max} = 100$, $T_{\min} = .01$, and $\beta = .996$, which seemed to strike an acceptable balance between exploration and exploitation.

We ran Q-learning with 0.10 greedy exploration (it chose the action with the highest estimated Q value with probability 0.90 and otherwise a random action) and $\alpha = 0.02$. These previously published settings (Ng, Harada, & Russell 1999) were not optimized for the new environments. Rmax used a known-state parameter of $m = 5$ and all reported results were averaged over forty replications to reduce noise.

For experiments, we used maze environments with dynamics similar to those published elsewhere (Russell & Norvig 1994; Leffler, Littman, & Edmunds 2007). Our first environment consisted of a 15x15 grid (Figure 1). It has start (S) and goal (G) locations, gray squares representing pits, and dark lines representing impassable walls. The agent receives a reward of $+1$ for reaching the goal, -1 for falling into a pit, and a -0.001 step cost. Reaching the goal or a pit

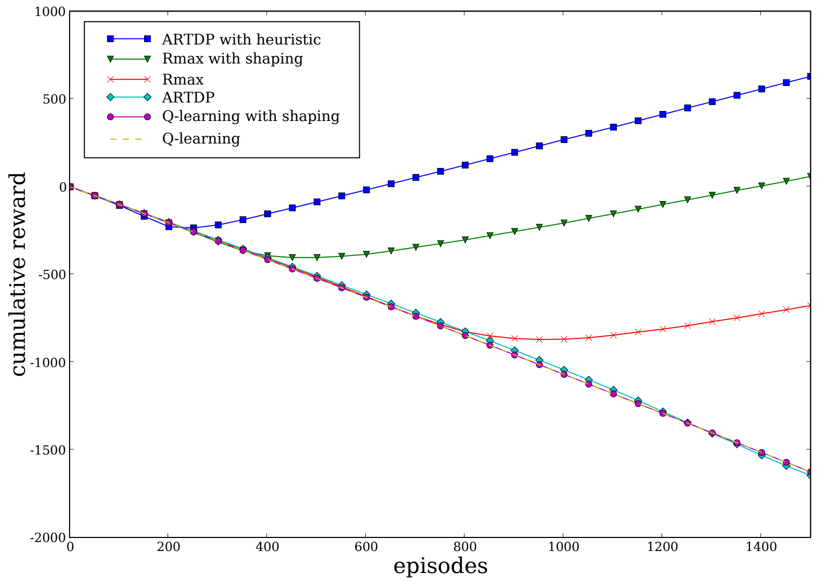

Figure 3: A cumulative plot of the reward received over 1500 learning episodes in the 15x15 grid example.

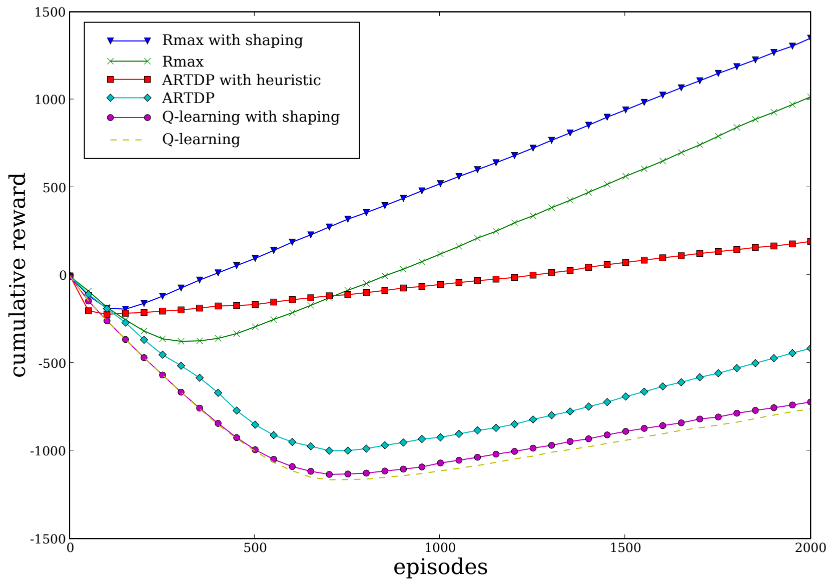

Figure 4: A cumulative plot of the reward received over the first 2000 learning episodes in the 4x6 grid example.

ends an episode and $\gamma = 1$.

Actions move the agent in each of the 4 compass directions, but actions can fail with probability 0.2, resulting in a slip to one side or the other with equal probability. For example, if the agent chooses action "east" from the location right above the start state, it will go east with probability 0.8, south with probability 0.1, and north with probability 0.1. Movement that would take the agent through a wall results in no change of state.

In the experiments presented, the shaping function was $\Phi(s) = C \times \frac{D(s)}{\rho} + G$, where $C \leq 0$ is the per-step cost, $D(s)$ is the distance of the shortest path from s to the goal, ρ is the probability that the agent goes in the direction intended and G is the reward for reaching the goal. $\frac{D(s)}{\rho}$ is a lower bound on the expected number of steps to reach the goal, since every step the agent makes (using the optimal policy) has chance ρ of succeeding. The ignored chance of failure makes Φ admissible.

Figure 3 shows the cumulative reward obtained by the six algorithms in 1500 learning episodes. In this environment, without shaping, the agents are reduced to exploring randomly from the start state, very often ending up in a pit before stumbling across the goal. ARTDP and Q-learning have a very difficult time locating the goal until around 1500 and 8000 episodes, respectively (not shown). Rmax, in contrast, explores more systematically and begins performing well after approximately 900 episodes.

The admissible shaping function leads the agents toward the goal more effectively. Shaping helps Q-learning master the problem in about half the time. Similarly, Rmax with shaping succeeds in about half the time of Rmax. Using the shaping function as a heuristic (initial Q values), helps ARTDP most of all; its aggressive learning pays off after about 300 steps.

In this example, ARTDP outperformed Rmax using the same heuristic. To achieve its PAC-MDP property, Rmax is conservative. We constructed a modified maze, Figure 2(b), to demonstrate the benefit of this assumption. We modified the environment by changing the step cost to -0.1 and the goal reward to $+5$ and adding a "reset" action that teleported the agent back to the start state incurring the step cost. Otherwise, the structure is the same as the previous maze. The optimal policy is to walk the "bridge" lined by risky pits (akin to a combination lock).

Figure 4 presents the results of the six algorithms. The main difference from the 15x15 maze is that ARTDP is not able to find the optimal path reliably, even using the heuristic. The reason is that there is a high probability that it will fall into a pit the first time it walks on the bridge. It models this outcome as a dead end, resisting future attempts to sample that action. Increasing the exploration rate could help prevent this modeling error but at the cost of the algorithm randomly resetting much more often, hurting its opportunity to explore.

Conclusions

We have defined a novel reinforcement-learning algorithm that extends both Rmax, a model-based approach, and potential-based shaping, a method for introducing "hints" to the learning agent. We argued that, for shaping functions that are "admissible", this algorithm retains the formal guarantees of Rmax, a PAC-MDP algorithm. Finally, we showed that the resulting combination of model learning and reward shaping can learn near optimal policies more quickly than either innovation in isolation.

In this work, we did not examine how shaping can best be combined with methods that generalize experience across states. However, our approach meshes naturally with existing Rmax enhancements along these lines such as factored Rmax (Guestrin, Patrascu, & Schuurmans 2002) and RAM-Rmax (Leffler, Littman, & Edmunds 2007). Of particular interest in future work will be integrating these ideas with more powerful function approximation in the Q functions or the model to help the methods scale to larger state spaces.

Acknowledgments

The authors thank DARPA TL and NSF CISE for their generous support.

References

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72(1):81–138.

Brafman, R. I., and Tennenholtz, M. 2002. R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3:213–231.

Guestrin, C.; Patrascu, R.; and Schuurmans, D. 2002. Algorithm-directed exploration for model-based reinforcement learning in factored MDPs. In *Proceedings of the International Conference on Machine Learning*, 235–242.

Hansen, E. A., and Zilberstein, S. 1999. Solving Markov decision problems using heuristic search. In *Proceedings of the AAAI Spring Symposium on Search Techniques for Problem Solving Under Uncertainty and Incomplete Information*, 42–47.

Kakade, S. M. 2003. *On the Sample Complexity of Reinforcement Learning*. Ph.D. Dissertation, Gatsby Computational Neuroscience Unit, University College London.

Leffler, B. R.; Littman, M. L.; and Edmunds, T. 2007. Efficient reinforcement learning with relocatable action models. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, 278–287.

Puterman, M. L. 1994. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. New York, NY: John Wiley & Sons, Inc.

Russell, S. J., and Norvig, P. 1994. *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice-Hall.

Strehl, A. L.; Li, L.; and Littman, M. L. 2006. PAC reinforcement learning bounds for RTDP and rand-RTDP. In *AAAI 2006 Workshop on Learning For Search*.

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

Wiewiora, E. 2003. Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research* 19:205–208.