An Intrinsic Reward Mechanism for Efficient Exploration

Özgür Şimşek OZGUR@CS.UMASS.EDU
Andrew G. Barto BARTO@CS.UMASS.EDU
Department of Computer Science, University of Massachusetts, Amherst, MA 01003

Abstract

How should a reinforcement learning agent act if its sole purpose is to efficiently learn an optimal policy for later use? In other words, how should it explore, to be able to exploit later? We formulate this problem as a Markov Decision Process by explicitly modeling the internal state of the agent and propose a principled heuristic for its solution. We present experimental results in a number of domains, also exploring the algorithm's use for learning a policy for a skill given its reward function—an important but neglected component of skill discovery.

1. Introduction

We consider the following problem: How should a reinforcement learning agent act if it is not concerned with accumulating reward, but with the ability to do so when desired? In other words, how can it efficiently learn how to accumulate reward without necessarily collecting high reward during the process?

This problem is distinct from the one traditionally addressed in reinforcement learning—accumulating as much reward as possible within the agent's lifetime—which requires the agent to perform a trade-off between exploration and exploitation. In contrast, we seek to perform *optimal exploration* for the sake of learning a policy that will enable exploitation when needed at a later time.

We are not proposing to replace the usual overall objective of a reinforcement learning agent. However, using this two-stage structure—in which a training period is present before performance becomes the main concern—is a natural way to approach many practical problems. It also provides a useful conceptual structure for designing reinforcement learning agents that are able to efficiently accumulate a collection of reusable skills with experience. By a skill we mean a closed-loop policy over one-stage actions, for example an *option* (Sutton et al., 1999). Most skill discovery methods first identify a reward function that the skill should maximize, then learn a corresponding policy. The latter, if done during a period devoted exclusively to learning a satisfactory skill policy, is an instance of the exploration problem we consider here. Such an active approach to skill acquisition has not been explored in the literature, but we believe that a short-term indifference to reward accumulation will prove to be beneficial in this context.

We assume that the agent is facing a Markov Decision Process (MDP) and refer to this as the *task MDP*. We formulate the optimal exploration problem as a different MDP, one whose states have two components: an external state that designates the state of the task MDP and an internal state that refers to the internal data structures of the agent. We call this the *derived MDP*. When the agent acts according to the optimal policy of the derived MDP, it performs optimal exploration for learning an optimal policy of the task MDP. This formulation is adapted from that of Duff (2003), where the internal state of the agent is a probability distribution over possible models of the task MDP. Instead of trying to optimally explore for the sake of identifying the task MDP, as in Duff's work, we are interested in optimally exploring to form an optimal policy for the task MDP.

We propose an approximate solution to the derived MDP that produces a simple and intuitive algorithm, schematically represented in Figure 1. The external state and reward are used in the usual manner to update the value function for the task MDP. Unlike the usual case, however, behavior is directed by a second value function, one that corresponds to the derived MDP. The updates to the behavior value function ignore external reward but use a different reward signal that depends on the evolution of the task value func-

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

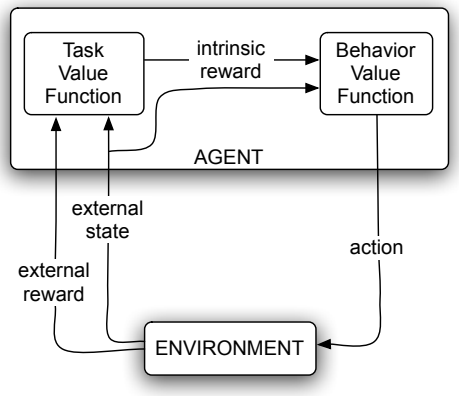

Figure 1. A Schematic Representation of our Approach. External state and reward are used to update the task value function. This update produces an intrinsic reward that is used to update the behavior value function.

tion. Because this reward signal is a function of the internal state of the agent, it is an *intrinsic* reward as defined by Barto et al. (2004) and Singh et al. (2005).

We devote considerable attention to the formulation of the derived MDP although our approximation is a relatively simple one. We believe that an explicit representation of full information state, consisting of internal and external state components, is the correct theoretical basis for understanding our algorithm. This formulation not only reveals how our algorithm approximates the solution to a well-defined optimization problem but also provides a basis for directions of future development.

In the following sections, we first define the optimal exploration problem precisely, describe our formulation and solution method, and experimentally evaluate its performance. We then explore its use in skill acquisition, concluding with a discussion of related work, our contributions, and future directions.

2. Background

We use the MDP framework to represent the agent's interaction with its environment. A finite MDP is a tuple $\langle S, A, T, R, D, \gamma \rangle$, where S is a finite set of states, A is a finite state of actions, $T : S \times A \times S \to [0, 1]$ is a transition function, $R : S \times A \times S \to \Re$ is a reward function, D is the initial state distribution from which the start state is drawn, and γ is a discount factor, $0 \leq \gamma \leq 1$. At each decision stage, the agent observes a state $s \in S$ and executes an action $a \in A$ with probability $\pi(s, a)$, where $\pi : S \times A \to [0, 1]$ is a stationary stochastic policy. With probability $T(s, a, s')$, the agent observes state s' in the next decision stage and receives an immediate reward with expected value $R(s, a, s')$. The value function of policy π is a map $V^\pi : S \to \Re$ that specifies the expected return for executing π starting from state s, where return is the discounted sum of future rewards. An optimal policy is one that maximizes the value function over all states. The actual value of state s is distinct from the agent's estimate of it. To refer to the latter at decision stage t, we use $V_t(s)$. For every policy π, we also define its *policy value*, $V(\pi)$, with respect to the initial state distribution:

$$V(\pi) = \sum_{s \in S} D(s) V^\pi(s). \tag{1}$$

3. The Optimal Exploration Problem

The *optimal exploration problem* is to devise an action selection mechanism for generating trajectories in the task MDP such that the policy learned by the end of a given number of training experiences has as high a value as possible as defined by Equation 1. We assume that the learning algorithm maintains a value function, but otherwise we treat it as a black box, seeking to devise a method that will adapt to the particular algorithm being used. We assume that the structure of the task MDP is unknown, and, furthermore, that it is not possible to sample a transition from an arbitrary state but only from its current state.

In this context, an action has two direct consequences. First, it generates a training experience for the learning algorithm by revealing an immediate reward and the next state. Second, it changes the external state, i.e., the state of the task MDP, determining which external state will be sampled next. To explicitly consider the impact of both on future updates to the value function, we model the external state of the environment as well as the agent's internal representation of the value function and other relevant data structures.

We observe that the joint evolution of external state and the agent's internal state satisfies the Markov property and formulate the optimal exploration problem as a derived MDP each of whose states has two components: external state (s_e) and internal state (s_i). The actions in the derived MDP and their effect on the external state component are identical to those in the task MDP. The internal state includes the policy derived from the agent's current value function for the task MDP, which we denote by π_{s_i}, and all other information that may impact changes to this in the future. The exact representation of internal state and the transition dynamics of the derived MDP depend

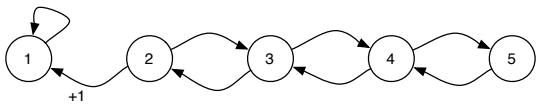

Figure 2. A Deterministic MDP with Five States. All transitions yield zero reward, except for transitions from state 2 into state 1, which yield a reward of +1. The initial state is state 5 with probability 1.

on the algorithm used to learn the task MDP value function. We do not need to specify them here because we do not solve the derived MDP exactly but exploit certain of its structural properties to find an approximate solution.

To make matters concrete, consider the deterministic MDP shown in Figure 2. Assume the agent uses Q-learning with step size $\alpha = 1$, $\gamma < 1$, and initial Q-values set to zero. Assume also that when the agent reaches the absorbing state, the environment is initialized to the start state (state 5). In this context, it is adequate to consider the agent's internal state to be its current greedy policy. In Figure 3, we show the state transition dynamics of the derived MDP, depicting an internal state with the corresponding greedy policy. As the policy for the task MDP is improved, transitions move the state of derived MDP toward the bottom of the diagram. Note that external state 1 is not part of the derived MDP because the agent makes no decisions at this state.

The reward for the derived MDP obtained upon a transition from state (s_e, s_i) to (s'_e, s'_i) is the difference between the values of their associated policies for the task MDP: $V(\pi_{s'_i}) - V(\pi_{s_i})$. It follows that, with $\gamma = 1$, the return obtained in a trajectory of finite length is the difference between the values of policies associated with the last and the first state in the trajectory. Consequently, with $\gamma = 1$ and a horizon that equals the number of training experiences available, the optimal solution to the derived MDP specifies an optimal exploration policy—one that yields a policy with as high a policy value as possible after the specified number of transitions.

It is worthwhile to make a few observations here on the transition structure of the derived MDP. We do not make any assumptions about the structure of the task MDP, so the transitions along the external state component can be arbitrary. But transitions along the internal state component have a particular structure. Internal state changes very little from one decision stage to the next because a single training experience changes the value function only slightly. Furthermore,

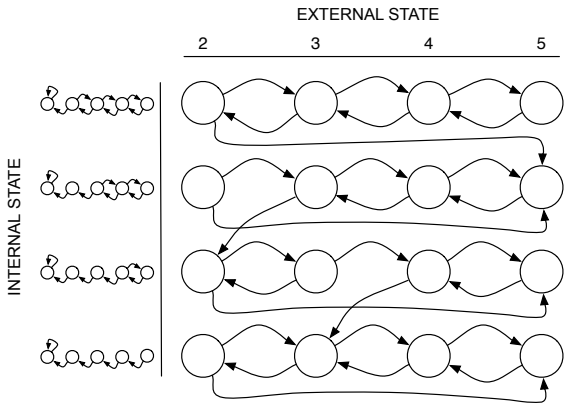

Figure 3. State Transition Graph of the Derived MDP corresponding to the Task MDP of Figure 2. The horizontal axis shows the external state while the vertical axis shows the internal state, depicting an internal state with the associated greedy policy for the task MDP.

in general (but not always), the policy value increases with more experiences. As a consequence, rather than jumping arbitrarily along the internal state dimension, the agent goes through a progression of internal states that typically increase in value.

Furthermore, if we refer to a set of states with the same internal state component as a *layer*, we observe that the connectivity and reward structure of layers that are directly connected are very similar. This is due to the incremental nature of learning updates. If an external state transition produces a change in the value function, it is likely to produce a similar change the next time it is experienced. In addition, transitions that are close to such transitions are likely to soon produce changes themselves because the changes in the value function propagate in the state space.

4. Our Approach

The derived MDP models the agent's learning process and its optimal policy specifies an optimal exploration policy for learning to solve the task MDP. It is, however, not practical to solve the derived MDP exactly. Here we enumerate the major difficulties and explain how we address them to derive a principled heuristic.

The agent cannot generate simulated experience for learning trials. The transition probabilities of both the task MDP and the derived MDP are unknown. As a consequence, the only experiences available to solve the derived MDP are those obtained during the training period itself—which the derived MDP is supposed to optimize! Unless one takes a Bayesian approach

to explicitly represent the uncertainty in the transition probabilities, which is intractable even for very small problems, the only viable alternative is to learn to solve both MDPs simultaneously, using the current solution to the derived MDP to select actions. This is the approach we take.

The state set of the derived MDP is enormous, even if one could easily identify an appropriate internal state representation. In any problem of reasonable size, the agent is unlikely to observe a state of the derived MDP more than once. We address this by using state approximation. We ignore the internal state component and learn a behavior policy as a function of external state only. When the internal state component is hidden, the derived MDP appears as a non-stationary MDP, with expected reward associated with transitions varying over time as the agent jumps from one layer of the derived MDP to another. This non-stationarity, however, is slowly varying because the directly connected layers are very similar. It should therefore be possible to "track" this slowly-varying non-stationary MDP in the sense of maintaining a nearly optimal (or at least, good) policy over time.

Reward for the derived MDP cannot be computed exactly. Recall that the reward for transitioning from state (s_e, s_i) to state (s'_e, s'_i) is $V(\pi_{s'_i}) - V(\pi_{s_i})$, or equivalently

$$\sum_{s \in S_e} D(s) \left(V^{\pi_{s'_i}}(s) - V^{\pi_{s_i}}(s) \right), \tag{2}$$

which we obtain using Equation 1 and where S_e is the state set of the task MDP. This expression is a weighted sum, over all external states, of the change in actual state value from one decision stage to the next, brought about by the updates to the value function of the task MDP. It can not be computed exactly because the actual state values are unknown, but a reasonable estimate can be obtained using the evolution of estimated state values, $V_t(s)$, as we now explain.

We distinguish between two types of updates to the value function: those that use the new training experience directly as a new sample, for example updates performed by Q-learning with or without eligibility traces, and those that propagate the direct updates in the state space, for example model-based planning updates of Dyna (Sutton, 1990). When both updates are present, our estimate is

$$\sum_{s \in S_e} D(s)(V_t(s) - V_{t-1}(s)), \tag{3}$$

where we assume that the transition takes place from decision stage $t-1$ to t. We arrive at this estimate by assuming that the change in estimated value of a state equals the change in its actual value. When only direct updates are present, it is not clear how they would propagate in the state space. Our estimate in this case is

$$\sum_{s \in S_e} (V_t(s) - V_{t-1}(s)), \tag{4}$$

which is independent of the initial state distribution. One can view this as the result of assuming the extreme case that a direct update would propagate undiminished to all the other states, in other words that it would change the estimated value of all other states by the same amount.

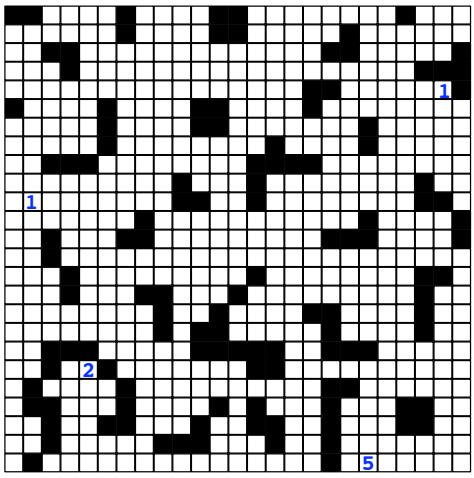

Figure 4. The Maze Task. Terminal states are marked with the amount of reward they generate.

With the approximation to the reward function given by Expression 3 or 4, there are two issues to consider. First, for a given state s, $V_t(s)$ may show high fluctuations over time. It may therefore be desirable to use a smoother estimate of state value using the history of value functions, for example a moving average or the maximum estimate over history.[1] Second, it is important to have pessimistic initial values because the underlying assumption is that increases in estimated value reflect increases in actual value.

5. Proposed Algorithm

The ideas in the preceding sections produce a simple and intuitive algorithm schematically represented in Figure 1. The agent maintains two value functions:

[1] This may be considered an instance of optimism under uncertainty, a heuristic frequently employed in reinforcement learning, because it assumes that the value of a state is the highest single estimate in the agent's lifetime.

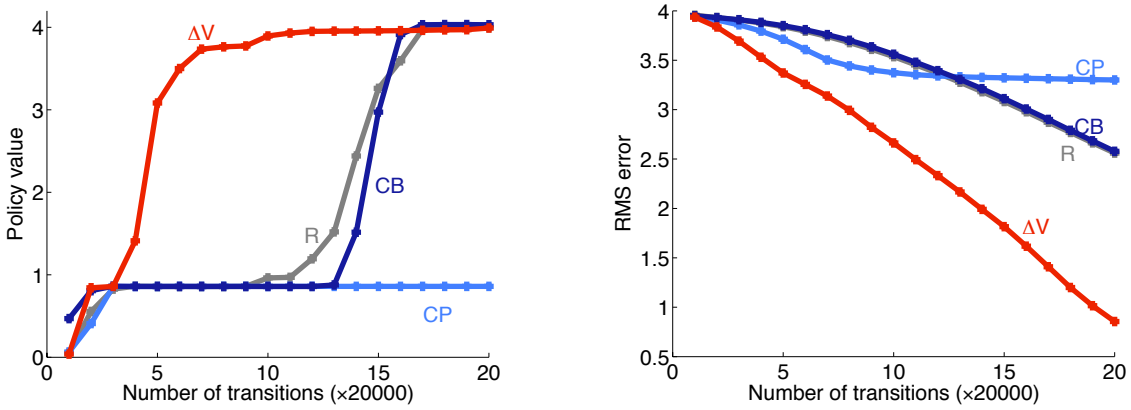

Figure 5. Performance in the maze task: (a) Policy value as defined by Equation 1, (b) RMS error between the current and optimal state values.

one that it can use to solve the task MDP (in the future) and another that it uses to select actions in the present. We call these the *task value function* and the *behavior value function*, respectively, and we call their associated policies the *task policy* and the *behavior policy*.

At decision stage t, the agent executes an action, observes an immediate external reward and the next external state, and updates the task value function. This update produces an intrinsic reward, $r_i(t)$, which the agent uses, together with observed external state, to update the behavior value function. The reinforcement learning algorithm used to update the behavior value function may be different than the algorithm used for learning the task value function. The number of available training experiences does not influence our algorithm, so it does not need to be known in advance.

In the experiments presented here, we defined intrinsic reward as follows:

$$r_i(t) = p + \sum_{s \in S} \left(V_t^{max}(s) - V_{t-1}^{max}(s) \right), \qquad (5)$$

where $V_t^{max}(s) = \max_{T \leq t} V_T(s)$ and $p < 0$ is a small action penalty. The action penalty does not change the optimal policy of the derived MDP but tends to promote faster learning and therefore better tracking of the non-stationary MDP observed when the internal state component is ignored.

6. Performance in a Maze Task

We evaluated the performance of our algorithm in the maze task shown in Figure 4. The available actions in each state are `north`, `south`, `east`, and `west`. These move the agent in the intended direction with probability 0.9 and in a uniform random direction with probability 0.1. If the direction of movement is blocked, the agent remains in the same location. The start state is the square in the center of the grid with probability 1. Reward is -0.001 for each action and an additional 1, 2, or 5 when transitioning into one of the terminal states. When generating training experiences, the agent returned to the start state after reaching a terminal state. The agent used Q-learning with $\alpha = 0.1$ and $\gamma = 0.99$ to learn both the task value function and the behavior value function. Initial Q-values were zero. The behavior policy was the greedy policy with respect to the behavior value function. The penalty term p was -0.005.

We provide comparisons with a number of baselines. *Random (R)* picked actions uniformly randomly. *Counter-based (CB)* picked the action selected the least number of times from the current state, which is a model-free variant of the action selection mechanism in Thrun (1992). *Constant-Penalty (CP)* was identical to our algorithm but as intrinsic reward used only the penalty term p instead of the right hand side of Equation 5.

Figure 5 shows two performance measures: (1) policy value (of the greedy policy with respect to the task value function) computed using Equation 1, and (2) root mean-squared (RMS) error between the current and optimal state values. The figure shows means of 30 trials. In both performance measures, our algorithm (which we denote ΔV) showed clear performance gains over the baselines. These results were typical in a variety of maze tasks with which we experimented. Use of eligibility traces in updating the behavior value function further improved the performance of our algorithm.

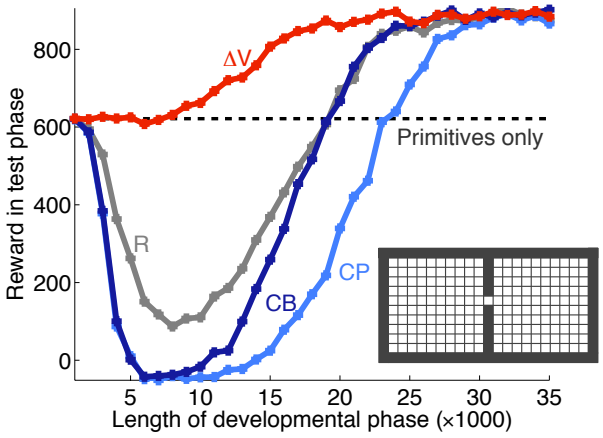

Figure 6. Performance in the gridworld task.

A closer examination of learning trials revealed that the behavior of our algorithm was qualitatively different than the others, as expected. Its behavior was neither random, nor could it be characterized as repeated systematic sweeps of the state space. Instead, it obsessively remained in regions in which the value function was improving, efficiently backing up the rewards in the terminal states to the rest of the state space.

7. Efficient Exploration for Skill Acquisition

The majority of skill discovery methods proceed by generating a reward function that the skill should maximize, typically by identifying a set of states that are useful to reach and defining a skill reward function whose optimal policy efficiently takes the agent to these states, e.g., Hengst (2002), McGovern and Barto (2001). When a new skill reward function is identified, the algorithm we present here may be used to generate the future experiences of the agent until a satisfactory skill policy is acquired. In this context, a brief period of indifference to reward may be a small sacrifice with a high payoff. The sooner the skill is functional, the sooner the agent can start obtaining its benefits.

We evaluated the utility of our algorithm in skill acquisition in the two-room gridworld environment shown in Figure 6, with dynamics identical to the maze domain presented above. A useful skill in this domain, for solving a number of problems, is one that takes the agent efficiently to the doorway. Many existing discovery methods can identify this skill, e.g., Mannor et al. (2004), Şimşek and Barto (2004), Şimşek et al. (2005). But, rather than using one of these algorithms, we isolate the exploration problem from the discovery problem by assuming that the agent has at its disposal an ideal discovery method—one that would identify the doorway as a useful subgoal the first time it is experienced.

Our experimental method consisted of two phases: a developmental phase in which the agent acquired the skill policy and a test phase in which the agent used the acquired skill (in addition to the primitive actions) to maximize an external reward signal. Our performance measure was the total discounted reward obtained in the test phase, during which the agent repeatedly performed an episodic task that started from a random state in the west room and terminated at the southeast corner of the grid. Reward associated with each transition was -0.001, plus an additional $+1$ if the transition took the agent to the terminal state. The skill was made available only from the west room to be able to attribute performance differences only to differences in the quality of the acquired skill policy. Otherwise, poor performance may also be attributed to a well-acquired skill diverting the agent from reaching the terminal state.

To represent skills, we used the options framework (Sutton et al., 1999). A (Markov) *option* is a temporally-extended action, specified by a triple $\langle I, \pi, \beta \rangle$, where I denotes the option's initiation set, i.e., the set of states in which the option can be invoked, π denotes the policy followed when the option is executing, and $\beta : I \to [0, 1]$, denotes the option's termination condition, with $\beta(s)$ giving the probability that the option terminates in state $s \in I$. When the agent observed the doorway state for the first time, it created an option that terminated with probability 1 at the doorway, with an initiation set containing only the state visited prior to visiting the doorway. The initiation set was expanded with subsequent experiences. Each time the agent transitioned to a state in the initiation set from a state s of the west room outside the initiation set, s was added to the initiation set. The skill reward function assigned -0.001 for each transition and an additional $+1$ for transitioning to the doorway.

In the test phase, the agent used intra-option Q-learning with $\alpha = 0.1$, $\epsilon = 0.05$, $\gamma = 1$. The test phase was 60,000 steps, which was the number of transitions required, on average, for an agent using only primitive actions to converge to its maximum performance. The algorithms used in the developmental phase and any unspecified parameters were identical to those in the maze task.

Figure 6 shows the total reward obtained in the test phase as a function of the length of the developmental phase measured in number of transitions. The figure shows means over 100 trials. In addition to the previous baselines, we show the performance of an agent that used only primitive actions, which is independent of the length of the developmental phase. The figure shows that our algorithm not only obtained the maximum performance with much less experience, but also that it never produced a "harmful" skill with which the agent accumulates less reward than it would using only primitive actions.

8. Discussion

Exploration in reinforcement learning has been studied extensively but typically with the objective of maximizing return in an agent's lifetime, which requires a trade-off between exploration and exploitation, e.g., Duff (2002), Kearns and Singh (1998). Doing this optimally is known as the optimal learning problem. In contrast, we study the optimal exploration problem, in which the objective is to learn how to maximize return without necessarily accumulating high reward in the process. Despite this difference, our approach adopts aspects of the full Bayesian adaptive approach to solving the optimal learning problem (Duff, 2002). In particular, we motivate our algorithm through a derived MDP with states factored into internal and external components analogous to, but not the same as, the information and physical state components of the Bayesian adaptive approach.

Our formulation of this problem yields a simple and intuitive algorithm for action selection. An important property of our method is that it is not tied to a specific reinforcement learning algorithm but can be used as long as the agent approximates a value function. Similarly, when used in the context of skill acquisition, our method can be used in conjunction with any skill-discovery mechanism that identifies a reward function to be maximized—the approach taken by most skill-discovery methods in the literature. In addition, it can be used to learn skills specified by a system designer who provides a reward function, which might in some cases be easier than specifying the policy itself.

The behavior achieved by our algorithm is focused exploration in regions of the state space in which the learning updates improve the agent's value function the most. A similar behavior is achieved with Prioritized Sweeping (Moore & Atkeson, 1993) and Queue-Dyna (Peng & Williams, 1993) when a model is used to generate updates to the value function. This is a different problem than ours because, being model-based, states can be selected arbitrarily for backups. Both of these algorithms are concerned with making the most of available training experience, while we are concerned with generating the training experience that will be the most useful. As such, they are complementary to ours and can be used in conjunction with it.

An essential component of our algorithm is a reward function defined as a function of the internal state of the agent. Some other algorithms in the literature that do the same are Kaplan and Oudeyer (2003), Schmidhuber (1991), and Schmidhuber and Storck (1993), which are concerned with learning predictive models of the environment. In contrast, we learn a value function for maximizing an external reward signal. Our formulation may help formalize the intuition behind these related algorithms, although a value function is more suitable for our approach than a model. Because the value of a state is a function of the values of its neighbors, changes in the value of one state propagate throughout the state space, helping to create the structure in the derived MDP that our algorithm exploits. This in general is not the case when learning a model. A model, however, also is typically learned incrementally, so the derived MDP would still have some of the same structure when the goal is to learn a model.

A potential use of our algorithm is as a component of an agent that autonomously builds a hierarchy of reusable skills (Barto et al., 2004; Singh et al., 2005). We advocate an active approach to skill acquisition that has not been pursued in the literature. Most skill discovery methods are *passive*, in that they do not direct the agent to seek experiences that will be useful for acquiring the skill. Instead, they use whatever experiences are available. One exception is the algorithm by Singh et al. (2005) in which an intrinsic reward term is added to the external reward function. While their method does not create a pure exploration policy as we suggest, it has a similar idea of influencing the behavior towards those experiences that would lead to efficient learning of the skill policy. But subsequent research has shown that this intrinsic reward is not effective in achieving such behavior (Barto & Şimşek, 2005). A related potential use of our algorithm is in solving the traditional reinforcement learning problem. As a multi-step exploration policy, it may be useful for addressing the exploration-exploitation trade-off.

Our findings generate a number of open questions, particularly in the skill discovery context. When should the exploration period terminate? What if there are multiple skills to be acquired? Should intrinsic rewards that are generated by each of these skills be combined, or should the agent pursue exploration in service of a

single skill at a time? Perhaps the latter resembles more closely the type of exploration in which people engage, directing their attention to a single purpose until they are satisfied, interrupted by something else, or frustrated because of lack of progress. One idea we are currently investigating is to define an explicit exploration-skill for each skill, whose policy is generated using the algorithm we present here. Our preliminary experiments show that performance gains are possible even with simple threshold rules for activating and terminating this exploration skill, for example those that track how much the skill value function is changing. Another direction of future research is to find better solutions to our formulation, perhaps by identifying useful features for state approximation on the internal state dimension, as in Duff (2003). Finally, important questions remain about integrating our approach with methods that learn models and use them for planning.

In the small domains that we tested our algorithm, we saw marked performance gains when compared to some heuristic methods of action selection. This is somewhat surprising since these domains do not provide many choices for the agent. The active approach we advocate should make a greater difference in large, complex domains in which there is a real need for skill discovery algorithms and in which the need for focused exploration is much greater.

Acknowledgments

We would like to thank Michael Duff, George Konidaris, Michael Littman, Andrew Stout, Chris Vigorito, and Alicia P. Wolfe for useful discussions. This research was supported by the National Science Foundation under Grant No.CCF-0432143 and by a subcontract from Rutgers University, Computer Science Department, under award number HR0011-04-1-0050 from DARPA. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Barto, A. G., & Şimşek, Ö. (2005). Intrinsic motivation for reinforcement learning systems. *Proceedings of the Thirteenth Yale Workshop on Adaptive and Learning Systems*. New Haven, CT, USA.

Barto, A. G., Singh, S., & Chentanez, N. (2004). Intrinsically motivated learning of hierarchical collections of skills. *Proceedings of the Third International Conference on Developmental Learning*.

Duff, M. (2002). *Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes*. Doctoral dissertation, University of Massassachusetts Amherst.

Duff, M. (2003). Design for an optimal probe. *Proceedings of the Twentieth International Conference on Machine Learning*.

Hengst, B. (2002). Discovering hierarchy in reinforcement learning with HEXQ. *Proceedings of the Nineteenth International Conference on Machine Learning*.

Kaplan, F., & Oudeyer, P.-Y. (2003). Motivational principles for visual know-how development. *Proceedings of the Third International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*.

Kearns, M., & Singh, S. (1998). Near-Optimal reinforcement learning in polynomial time. *Proceedings of the Fifteenth International Conference on Machine Learning*.

Mannor, S., Menache, I., Hoze, A., & Klein, U. (2004). Dynamic abstraction in reinforcement learning via clustering. *Proceedings of the Twenty-First International Conference on Machine Learning*.

McGovern, A., & Barto, A. G. (2001). Automatic discovery of subgoals in reinforcement learning using diverse density. *Proceedings of the Eighteenth International Conference on Machine Learning*.

Moore, A., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, *13*, 103–130.

Peng, J., & Williams, R. J. (1993). Efficient learning and planning within the dyna framework. *Adaptive Behavior*, *2*, 437–454.

Schmidhuber, J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers. *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*.

Schmidhuber, J., & Storck, J. (1993). Reinforcement driven information acquisition in nondeterministic environments. Technical report, Fakultat fur Informatik, Technische Universit at Munchen.

Şimşek, Ö., & Barto, A. G. (2004). Using relative novelty to identify useful temporal abstractions in reinforcement learning. *Proceedings of the Twenty-First International Conference on Machine Learning*.

Şimşek, Ö., Wolfe, A. P., & Barto, A. G. (2005). Identifying useful subgoals in reinforcement learning by local graph partitioning. *Proceedings of the Twenty-Second International Conference on Machine Learning*.

Singh, S., Barto, A. G., & Chentanez, N. (2005). Intrinsically motivated reinforcement learning. *Advances in Neural Information Processing Systems*.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *Proceedings of the Seventh International Conference on Machine Learning*.

Sutton, R. S., Precup, D., & Singh, S. P. (1999). Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*, 181–211.

Thrun, S. (1992). *Efficient exploration in reinforcement learning* (Technical Report CMU-CS-92-102). Carnegie-Mellon University.