Reinforcement Learning for Weakly-Coupled MDPs and an Application to Planetary Rover Control

Daniel S. Bernstein and Shlomo Zilberstein

Department of Computer Science, University of Massachusetts,
Amherst, Massachusetts 01003
{bern, shlomo}@cs.umass.edu

Abstract. Weakly-coupled Markov decision processes can be decomposed into subprocesses that interact only through a small set of bottleneck states. We study a hierarchical reinforcement learning algorithm designed to take advantage of this particular type of decomposability. To test our algorithm, we use a decision-making problem faced by autonomous planetary rovers. In this problem, a Mars rover must decide which activities to perform and when to traverse between science sites in order to make the best use of its limited resources. In our experiments, the hierarchical algorithm performs better than Q-learning in the early stages of learning, but unlike Q-learning it converges to a suboptimal policy. This suggests that it may be advantageous to use the hierarchical algorithm when training time is limited.

1 Introduction

The Markov decision processes (MDP) framework is widely used to model problems in decision-theoretic planning and reinforcement learning [6]. Recently there has been increased interest in delimiting classes of MDPs that are naturally decomposable and developing special-purpose techniques for these classes [1]. In this paper, we focus on reinforcement learning for weakly-coupled MDPs. A weakly-coupled MDP is an MDP that has a natural decomposition into a set of subprocesses. The transition from one subprocess to another requires entry into one of a small set of bottleneck states. Because the subprocesses are only connected through a small set of states, they are "almost" independent. The common intuition is that weakly-coupled MDPs should require less computational effort to solve than arbitrary MDPs.

The algorithm that we investigate is a reinforcement learning version of a previously studied planning algorithm for weakly-coupled MDPs [2]. The planning algorithm is model based, whereas our algorithm requires only information from experience trajectories and knowledge about which states are the bottleneck states. This can be beneficial for problems where only a simulator or actual experience are available. Our algorithm fits into the category of hierarchical reinforcement learning (see, e.g., [7]) because it learns simultaneously at the state level and at the subprocess level. We note that other researchers have proposed methods for solving weakly-coupled MDPs [3–5], but very little work has been done in a reinforcement learning context.

For experimentation we use a problem from autonomous planetary rover control that can be modeled as a weakly-coupled MDP. In our decision-making scenario, a rover on

Mars must explore a number of sites over the course of a day without stopping to establish communication with Earth. Using only a list of sites, information about its resource levels, and information about the goals of the mission, the rover must decide which activities to perform and when to move from one site to the next. Limited resources and nondeterministic action effects make the problem nontrivial. In the main body of the paper, we describe in detail how this problem can be modeled as a weakly-coupled MDP, with each site being a separate subprocess.

We compare the hierarchical algorithm with Q-learning, and we see that the hierarchical algorithm performs better initially but fails to converge to the optimal policy. A third algorithm which is given the optimal values for the bottleneck states at the start actually learns more slowly than both of the aforementioned algorithms. We give possible explanations for the observed behavior and suggestions for future work.

2 MDPs and Reinforcement Learning

A *Markov decision process (MDP)* models an agent acting in a stochastic environment with the aim of maximizing its expected long-term reward. The type of MDP we consider contains a finite set S of states, with s_0 being the start state. For each state $s \in S$, A_s is a finite set of actions available to the agent. P is the table of transition probabilities, where $P(s'|s,a)$ is the probability of a transition to state s' given that the agent performed action a in state s. R is the reward function, where $R(s,a)$ is the reward received by the agent given that it chose action a in state s.

A *policy* π is a mapping from states to actions. Solving an MDP amounts to finding a policy that maximizes the expected long-term reward. In this paper, we use the infinite-horizon discounted optimality criterion. Formally, the agent should maximize

$$E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\right],$$

where $\gamma \in [0,1]$ is the discount factor. In order to model *episodic* tasks, we can include an absorbing state from which the agent can only receive an immediate reward of zero; a transition to the absorbing state corresponds to the end of an episode.

Algorithms for MDPs often solve for *value functions*. For a policy π, the state value function, $V^\pi(s)$, gives the expected total reward starting from state s and executing π. The state-action value function, $Q^\pi(s,a)$, gives the expected total reward starting from state s, executing action a, and executing π from then on.

When an explicit model is available, MDPs can be solved using standard dynamic programming techniques such as policy iteration or value iteration. When only a simulator or real experience are available, reinforcement learning methods are a reasonable choice. With these techniques, experience trajectories are used to learn a value function for a good policy. Actions taken on a trajectory are usually greedy with respect to the current value function, but *exploratory* actions must also be taken in order to discover better policies. One widely-used reinforcement learning algorithm is Q-learning [8], which updates the state-action value function after each transition from s to s' under

action a with the following rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right],$$

where α is called the learning rate.

3 Reinforcement Learning for Weakly-Coupled MDPs

Consider an MDP with a state set S that is partitioned into disjoint subsets $S_1, \ldots, S_m$. The *out-space* of a subset S_i, denoted $O(S_i)$, is defined to be the set of states not in S_i that are reachable in one step from S_i. The set of states $B = O(S_1) \cup \cdots \cup O(S_m)$ that belong to the out-space of at least one subset comprise the set of *bottleneck* states. If the set of bottleneck states is relatively small, we call the MDP *weakly-coupled*.

In [2], the authors describe an algorithm for weakly-coupled MDPs that can be described as a type of policy iteration. Initially, values for the bottleneck states are set arbitrarily. The low-level policy improvement phase involves solving each subproblem, treating the bottleneck state values as terminal rewards. The high-level policy evaluation phase consists of reevaluating the bottleneck states for these policies. Repeating these phases guarantees convergence to the optimal policy in a finite number of iterations.

The rules for backpropagating value information in our reinforcement learning algorithm are derived from the two phases mentioned above. Two benefits of our approach are that it doesn't require an explicit model and that learning can proceed simultaneously at the high level and at the low level.

We maintain two different value functions: a low-level state-action value function Q defined over all state-action pairs and a high-level state value function V_h defined over only bottleneck states. The low-level part of the learning is described as follows. Upon a transition to a non-bottleneck state, the standard Q-learning backup is applied. However, when a bottleneck state $s' \in B$ is encountered, the following backup rule is used:

$$Q(s,a) \leftarrow Q(s,a) + \alpha_l \left[R(s,a) + \gamma V_h(s') - Q(s,a) \right],$$

where α_l is a learning rate. For the purposes of learning, the bottleneck state is treated as a terminal state, and its value is the terminal reward. High-level backups occur only upon a transition to a bottleneck state. The backup rule is:

$$V_h(s) \leftarrow V_h(s) + \alpha_h [R + \gamma^k V_h(s') - V_h(s)],$$

where k denotes the number of time steps elapsed between the two bottleneck states, R is the cumulative discounted reward obtained over this time, and α_h is a learning rate.

It is possible to alternate between phases of low-level and high-level backups or to perform the backups simultaneously. Whether either approach converges to an optimal policy is an open problem. We chose the latter for our experiments because our preliminary work showed it be more promising.

4 Autonomous Planetary Rover Control

4.1 The Model

In this section we describe a simple version of the rover decision-making problem and how it fits within the weakly-coupled MDP framework. In our scenario, a rover is to operate autonomously for a period of time. It has an ordered sequence of sites along with priority information and estimated difficulty of obtaining data, and it must make decisions about which activities to perform and when to move from one site to the next. The goal is to maximize the amount of useful work that gets done during the time period.

The action set consists of taking a picture, performing a spectrometer experiment, and traversing to the next site in the sequence. Spectrometer experiments take more time and are more unpredictable than pictures, but they yield better data. The time to traverse between sites is a noisy function of the distance between the sites. The state features are the time remaining in the day, the current site number (from which priority and estimated difficulty are implicitly determined), the number of pictures taken at the current site, and whether or not satisfactory spectrometer data has been obtained at the current site. Formally, $S = T \times I \times P \times E$, where $T = \{0 \text{ min}, 5 \text{ min}, \ldots, 300 \text{ min}\}$ is the set of time values; $I = \{1, 2, 3, 4, 5\}$ is the set of sites; $P = \{0, 1, 2\}$ is the set of values for pictures taken; and $E = \{0, 1\}$ is the set of values for the quality of the spectrometer data. The start state is $s_0 = \langle 300, 1, 0, 0 \rangle$. The sequence of sites used for our experiments is shown in Table 1.

Table 1. The sequence of sites for the rover to investigate

Site	Priority	Estimated difficulty	Distance to next site
1	8	medium	3 m
2	5	hard	5 m
3	3	easy	7 m
4	2	easy	3 m
5	9	hard	N/A

A nonzero reward can only be obtained upon departure from a site and is a function of the site's priority and the data obtained at the site. The task is episodic with $\gamma = 1$. An episode ends when the time component reaches zero or the rover finishes investigating the last site. The aim is to find a policy that maximizes the expected total reward across all sites investigated during an episode. Because of limited time and nondeterministic action effects, the optimal action is not always obvious.

In order to see how this problem fits into the weakly-coupled MDP framework, consider the set of states resulting from a traversal between sites. In all of these states, the picture and spectrometer components of the state are reset to zero. The set $B = T \times I \times \{0\} \times \{0\}$ is taken to be the set of bottleneck states, and it is over this set that we define the high-level value function. Note that the bottleneck states comprise only 300 of the problem's 1,800 states.

4.2 Experiments

In our experiments, we tested Q-learning against the hierarchical algorithm on the problem mentioned in the previous section. In addition, we tested an algorithm that we call the *omniscient* hierarchical learning algorithm. This algorithm is the same as the hierarchical algorithm, except that the values for the bottleneck states are fixed to optimal from the start, and only low-level backups are performed. By fixing the bottleneck values, the problem is completely decomposed from the start. Of course, this cannot be done in practice, but it is interesting for the purpose of comparison.

For the experiments, all values were initialized to zero, and we used ϵ-greedy exploration with $\epsilon = 0.1$ [6]. For the results shown, all of the learning rates were set to 0.1 (we obtained qualitatively similar results with learning rates of 0.01, 0.05, and 0.2). Figure 1 shows the total reward per episode plotted against the number of episodes of learning. The points on the curves represent averages over periods 1000 episodes.

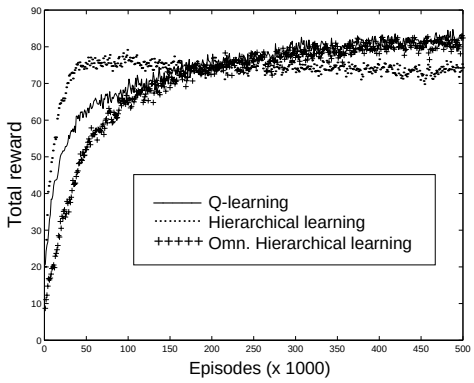

Fig. 1. Learning curves for Q-learning, hierarchical learning, and omniscient hierarchical learning

A somewhat counterintuitive result is that the omniscient hierarchical algorithm performs worse than both the original hierarchical algorithm and Q-learning during the early stages. One factor contributing to this is the initialization of the state-action values to zero. During the early episodes of learning, the value of the "leave" action grows more quickly than the values for the other actions because it is the only one that leads directly to a highly-valued bottleneck state. Thus the agent frequently leaves a site without having gathered any data. This result demonstrates that decomposability doesn't always guarantee a more efficient solution.

The second result to note is that the hierarchical algorithm performs better than Q-learning initially, but then fails to converge to the optimal policy. It is intuitively plausible that the hierarchical algorithm should go faster, since it implicitly forms an abstract process involving bottleneck states and propagates value information over multiple time steps. It also makes sense that the algorithm doesn't converge once we consider that the high-level backups are *off policy*. This means that bottleneck states are evaluated for the policy that is being executed, and this policy always includes non-greedy exploratory

actions. Algorithms such as Q-learning, on the other hand, learn about the policy that is greedy with respect to the value function regardless of which policy is actually being executed.

5 Conclusion

We studied a hierarchical reinforcement learning algorithm for weakly-coupled MDPs, using a problem in planetary rover control as a testbed. Our results indicate that the decomposability of these problems can lead to greater efficiency in learning, but the conditions under which this will happen are not yet well understood. Perhaps experimentation with different low-level and high-level learning rates could shed some insight. Also, experimental results from other weakly-coupled MDPs besides the rover problem would be valuable. Finally, a more detailed theoretical investigation may yield an algorithm similar to ours that is provably convergent.

On the application side, we plan to develop a more realistic and complex simulator of the rover decision-making problem. In this simulator, the rover will choose among multiple sites to traverse to. It will also have to manage its data storage and battery capacity and perform activities during constrained time intervals. The state space of the model will most likely be too large to explicitly store a value for each state. We will instead have to use some form of function approximation.

Acknowledgements

The authors thank Rich Washington, John Bresina, Balaraman Ravindran, and Ted Perkins for helpful conversations. This work was supported in part by the NSF under grants IRI-9624992 and IIS-9907331, and by NASA under grants NAG-2-1394 and NAG-2-1463. Daniel Bernstein was supported by an NSF Graduate Research Fellowship and a NASA SSRP Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the NSF or NASA.

References

1. Boutilier, C., Dean, T. & Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. Journal of Artificial Intelligence Research, 1, 1–93.
2. Dean, T. & Lin, S.-H. (1995). Decomposition techniques for planning in stochastic domains. In IJCAI-95.
3. Foreister, J.-P. & Varaiya, P. (1978). Multilayer control of large Markov chains. IEEE Transactions on Automatic Control, 23(2), 298–304.
4. Hauskrecht, M., Meuleau, N., Kaelbling, L. P., Dean, T. & Boutilier, C. (1998). Hierarchical solution of Markov decision processes using macro-actions. In UAI-98.
5. Parr, R. (1998). Flexible decomposition algorithms for weakly coupled Markov decision problems. In UAI-98.
6. Sutton, R. S. & Barto, A. G. (1998). Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press.
7. Sutton, R. S., Precup, D. & Singh, S. (2000). Between MDPs and Semi-MDPs: Learning, planning, and representing knowledge at multiple temporal scales. Artificial Intelligence, 112, 181–211.
8. Watkins, C. (1989). Learning from Delayed Rewards. PhD thesis, Cambridge University, Cambridge, England.