Autonomous Subgoal Discovery and Hierarchical Abstraction For Reinforcement Learning Using Monte Carlo Method

Mehran Asadi and **Manfred Huber**
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019 U.S.A
{asadi,huber}@cse.uta.edu

Autonomous systems are often difficult to program. Reinforcement learning (RL) is an attractive alternative, as it allows the agent to learn behavior on the basis of sparse, delayed reward signals provided only when the agent reaches desired goals. However, standard reinforcement learning methods do not scale well for larger, more complex tasks. One promising approach to scaling up RL is hierarchical reinforcement learning (HRL) (Sutton, Precup, & Singh 1999; Kim & Dean 2003; Dietterich 2000; Givan, Leach, & Dean 2000; Parr 1998).

Here, low-level policies, which emit the actual, primitive actions at a fast time-scale, solve only parts of the overall task. Higher-level policies solve the overall task, but they may consider only few abstract, high-level observations and actions (often referred to as macro-actions or options), at a slower time scale. This reduces each level's search space and facilitates temporal credit assignment. Another advantage is that low-level policies can be re-used easily, either within the same task or in other tasks. One of the fundamental steps toward HRL is to automatically establish subgoals. Methods for automatically introducing subgoals have been studied in the context of adaptive production systems, where subgoals are created based on examinations of problem-solving protocols. For RL systems, several researchers have proposed methods by which policies learned for a set of related tasks are examined for commonalities or are probabilistically combined to form new policies. Subgoal discovery has been addressed by several researchers such as (McGovern & Barto 2001; Digney 1996; Drummond 1997), However the most closely related research is that of Digney (Digney 1996). In his system, states that are visited frequently or states where the reward gradient is high are chosen as subgoals. Drummond (Drummond 1997) proposed a system where an RL agent detected walls and doorways through the use of vision processing techniques applied to the learned value function. This paper presents a new method for the autonomous construction of hierarchical action and state representations in reinforcement learning, aimed at accelerating learning and extending the scope of such systems. In this approach, the agent uses information acquired while learning one task to discover subgoals by analyzing the learned policy for certain structural properties using Monte Carlo sampling. By creating useful new subgoals and by off-line learning corresponding subtask policies as abstract actions, the agent is able to transfer knowledge to subsequent tasks and to accelerate learning. At the same time, the subgoal actions are used to construct a more abstract state representation using action-dependent approximate state space partitioning. This representation forms a new level in a state space hierarchy and serves as the initial representation for new learning tasks. In order to ensure that tasks are learnable, value functions are built simultaneously at different levels and inconsistencies are used to identify actions to be used to refine relevant portions of the abstract state space. Together these techniques permit the agent to form more abstract action and state representations over time.

The main goal of automatic subgoal discovery is to find useful subgoals that can be defined in the agent's state space. Once they are found, options to those subgoals can be learned and added as actions to the behavioral repertoire of the agent. In the approach to subgoal discovery presented here, subgoals are identified as states with particular structural properties in the context of a given policy. In particular, we define subgoals as states that, under a given policy, lie on a substantially larger number of paths than would be expected by looking at the same number for its successor state. In other words, we are looking for states that work like a funnel for state space trajectories occurring under the learned policy. In a uniformly connected space, where all states are connected to approximately the same expected number of states, every state will also have approximately equal number of direct predecessor under a given policy, except for regions near the goal state or close to the boundaries.

Definition 1:(Goel & Huber 2003) The Count metric $C(s)$ for a state s represents the number of paths that, staring from an arbitrary state, end at state s and is computed by:

$$C(s) = \sum_{i=1}^{n} C_i(s)$$

where

$$C_1(s) = \sum_{s \neq s'} P(s|s', \pi(s'))$$

and

$$C_{i+1}(s) = \sum_{s \neq s'} P(s|s', \pi(s'))C_i(s)$$

Where n is the smallest index such that $C_{n+1} = C_n$ or $n =$ number of states, whichever is smaller. The condition $s \neq s'$ prevents the counting of one step loops. The curve for a path under a given policy generated by a count metric defines a count curve. The curvature of the count curve is $\Delta(t) = C(s_t) - C(s_{t-1})$ and the Gradient Ratio at any state is the ratio of the curvature of the count curve at the current state and the successor state i.e. $\Delta(t)/\Delta(t+1)$. When $\Delta(t) \leq \Delta(t+1)$ the tangent of a curve is decreasing and the number of paths that going through state s_t has not changed or decreased and thus s_t is not a good candidate for being a potential subgoal. However, the algorithm looks for those state for which $\Delta(t) > \Delta(t+1)$. In this case if the ratio is larger than a specified threshold μ, then s_t will be considered a potential subgoal. The specified threshold μ for subgoal discovery affects the number of subgoals, which defines different levels of hierarchy generated using ϵ, δ-reduction which will be described later in this article.
In order to reduce the complexity of the above method, Gradient Ratio is computed, here, using Monte Carlo sampling. Let $h_1, ..., h_n$ be n induced samples by policy π and let $\tilde{C}_{h_i}(s)$ be the predecessor count for each state s in sample h_i, then it can be shown that for a sample size N such that

$$N = O(\frac{\max C(s)}{\epsilon_N} 2^T log(\frac{1}{p}))$$

where T is the finite horizon length,with probability p we have

$$|\tilde{C}_{h_i}(s) - C(s)| \leq \epsilon_N$$

and

$$|\tilde{\Delta}(t) - \Delta(t)| \leq 2\epsilon_N$$

where $\tilde{\Delta}(t)$ is the curvature of the count curve according to the samples. it can be easily verified that

$$\frac{\Delta(t) - 2\epsilon_N}{\Delta(t-1) + 2\epsilon_N} \leq \frac{\tilde{\Delta}(t)}{\tilde{\Delta}(t-1)} \leq \frac{\Delta(t) + 2\epsilon_N}{\Delta(t-1) + 2\epsilon_N}$$

and thus $\tilde{\Delta}(t)$ estimates the potential subgoal as same as $\Delta(t)$ as $\Delta(t)/\Delta(t-1) > 1$ if and only if $\tilde{\Delta}(t)/\tilde{\Delta}(t-1) > 1$. Once the subgoals have been discovered, the policies to each subgoal will be learned in order to establish the options for each discovered subgoals. Then the ϵ, δ-reduction (Asadi & Huber 2004) method is used to construct a state space hierarchy according to the following criterion:

$$|R(s, o_i) - R(s', o_i)| \leq \epsilon \quad (1)$$

and

$$\left| \sum_{s'' \in B_j} P(s''|s, o_i(s)) - \sum_{s'' \in B_j} P(s''|s', o_i(s')) \right| \leq \delta \quad (2)$$

where ϵ and δ are two real numbers whose values are dependent on the environment. The ϵ, δ-reduction method first partitions the state space into blocks $\{B_1, \ldots, B_n\}$ according to the reward function using equation (1) and then refine these partitions using the probability distribution according to equation (2). While the agent tries to solve the task on the abstract partition space, it computes the difference in Q-values between the best actions in the current state in the abstract state space and in the original state space. If the differences between the Q-values of the best actions is more than a constant value then there is a significant difference between different states underlying the particular block that was not captured by the subgoal options. Moreover, it implies that one of the original actions should be chosen in part of this particular block which therefore has to be refined based on the particular action in order to ensure successful completion of the overall task. Several experiments with different tasks and various scenarios have confirmed that this method finds the useful subgoals and constructs a compact state representation, resulting in accelerated learning times.

References

Asadi, M., and Huber, M. 2004. State space reduction for hierarchical reinforcement learning. In *In Proceedings of the 17th International FLAIRS Conference*, 509–514. AAAI.

Dietterich, T. G. 2000. An overview of maxq hierarchical reinforcement learning. *Lecture Notes in Computer Science* 1864.

Digney, B. 1996. Emergent hierarchical control structures: Learning reactive / hierarchical relationships in reinforcement environments. In *Proceedings of the Fourth Conference on the Simulation of Adaptive Behavior*.

Drummond, C. 1997. Using a case base of surfaces to speed-up reinforcement learning. In *Proceedings of the Second International Conference on International Conference on Case-Based Reasoning*, 435–444.

Givan, R.; Leach, S.; and Dean, T. 2000. Bounded-parameter markov decision processes. *Artificial Intelligence* 122(1-2):71–109.

Goel, S., and Huber, M. 2003. Subgoal discovery for hierarchical reinforcement learning using learned policies. In *In Proceedings of the 16th International FLAIRS Conference*, 346–350. AAAI.

Kim, K., and Dean, T. 2003. Solving factored MDPs using non-homogeneous partitions. *Artificial Intelligence* 147:225–251.

McGovern, A., and Barto, A. 2001. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning*, 361–368.

Parr, R. 1998. *Hierarchical Control and Learning for Markov Decision Processes*. Ph.D. Dissertation, University of California, Berkeley, CA.

Sutton, R.; Precup, D.; and Singh, S. 1999. Between MDPs and Semi-MDPs: Learning, planning, and representing knowledge at multiple temporal scales. *Artificial Intelligence* 112:181–211.