Improved Automatic Discovery of Subgoals for Options in Hierarchical Reinforcement Learning

R. Matthew Kretchmar, Todd Feil, Rohit Bansal
Department of Mathematics and Computer Science
Denison University
Granville, OH 43023, USA
kretchmar@denison.edu

Abstract

Options have been shown to be a key step in extending reinforcement learning beyond low-level reactionary systems to higher-level, planning systems. Most of the options research involves hand-crafted options; there has been only very limited work in the automated discovery of options. We extend early work in automated option discovery with a flexible and robust method.

Keywords: reinforcement learning, options, subgoal discovery

1 Introduction

Reinforcement Learning has proven to be useful in low-level, control and sense-react systems. Extending the role of reinforcement learning to higher levels of abstraction is a major focus of research. While work in this area falls under multiple names of Hierarchical Reinforcement Learning, Hierarchical Decomposition, Options, Macro-Actions, and Temporal Abstraction, the goal is the same: to move beyond the low-level, sense-and-react systems by abstracting actions to higher levels of reasoning; that is, to apply reinforcement learning in planning-like domains.

Options (we use the term options) have clearly proven to be useful in a number of previously troubling aspects of reinforcement learning including accelerating learning and the transference of knowledge between two similar learning tasks [6, 4]. However, most of the work in this area involves options that are *a priori* hand-crafted to suit the problem domain. This requires prior domain knowledge and, for some tasks, a significant amount of human effort. It is desireable to have the learning agent automatically find and form these options based upon the current learning exerience.

Section 2 provides a very brief discussion of reinforcement learning and definitions for options while Section 3 reviews recent work on automated option creation. In Section 4, we provide an alternative automated method, called the FD Algorithm that has many attractive properties including flexibility across different tasks, application to tasks without physical distance metrics, fewer parameters, and a relative insensitivity to parameter tuning. We present an example of successful option creation in Section 5. Finally, Section 6 concludes with a few remarks.

2 Option Overview

Reinforcement learning is a relatively new domain of machine learning in which a machine attempts to optimize performance at a task via trail-and-error. The learner senses states and chooses from among a set of actions available for each state. The state-action pair produces a next state and also a reward signal. It is the goal of the learning agent to choose actions so as to maximize the accumulative sum of reward signals. The problem is complicated by the fact that different action choices might appear to have lower rewards but they lead the agent to more reinforcement-rich parts of the state space. The agent must properly assign the credit/blame of action choices to the payoff of future rewards. Interested readers should consult [7] as an excellent reference on reinforcement learning.

Options are a set of primitive action choices. An agent may choose to select an option in which case all the actions of that option are executed in succession; thus the option can be viewed as a macro-action. Options have shown promise in allowing the agent to reason at a higher cognitive level by learning over a set of high-level options rather than a set of low-level actions.

Readers should consult [6] for a more comprehensive reference on options.

Formally, an option is a 3-tuple $(\mathcal{I}, \pi, \beta)$ where $\mathcal{I}$ is the option input set: the set of states in which the option may be selected instead of a primitive action. The option includes a policy, π, that indicates how the agent is to act while following the option, and a terminating function, β, that provides a probability of terminating the option per each state in the option.

The work in this paper uses a subset of the general options. Here we consider options with a single state in the option is defined as the *subgoal* of the option. The option's ***purpose*** is to move the agent to the subgoal so as to maximize reward (positive reward cycles are forbidden). All the other option states are part of the input set $\mathcal{I}$. The terminating function, β, is zero over all states in $\mathcal{I}$ and is one for the option subgoal.

Figure 1 shows the structure of an example option. The task has nineteen discrete states – nine in each room and one in the doorway. There are four actions available from each state (up, down, right and left). We have crafted an option that helps us move from a state in the left room toward the right room. The option subgoal is the doorway state (State 10). The option input set consists of all states in the left room (State 1 through State 9). The option terminates in the subgoal, State 10. Formally,

$$\mathcal{I} = \{1, 2, ..., 9\},$$
$$\begin{aligned} \pi(1), \pi(4), \pi(7) &= \downarrow \\ \pi(2), \pi(5), \pi(8) &= \rightarrow \\ \pi(3), \pi(6), \pi(9) &= \uparrow \end{aligned}$$
$$\beta(s) = \begin{cases} 1 & \text{if } s = 10 \\ 0 & \text{otherwise.} \end{cases}$$

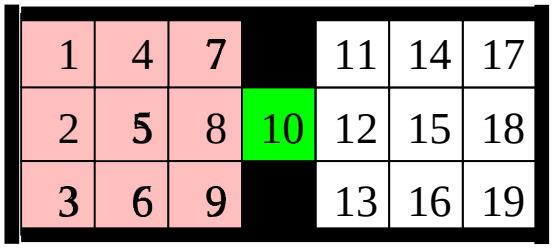

Figure 1: Simple Task with Option in Left Room

3 Automated Subgoal Discovery

The option of Figure 1 is crafted by hand; however it is more useful to be able to discover this option automatically. If the agent were to perform multiple trials (or episodes) by starting in a state in the left room, and then moving to some goal in the right room, the agent should be able to sense common patterns in each trial; the agent should be able to find those sequences of states which are commonly performed in solving these different but related tasks. These common sequences, or trajectories, should be candidates for options.

The most promising initial work in automated option discovery is by McGovern and Barto [3, 4, 5]. Their idea is to combine Maron's Diverse Density Algorithm [2] for automated subgoal discovery and then Lin's Experience Replay Algorithm [1] for forming the option policy. The McGovern/Barto Diverse Density Algorithm is sketched below:

1. Start initial learning on a task.
2. Record trajectories (sequences of states) as experienced by the agent.
3. Classify the trajectories as ***positive*** if the agent reaches the goal or *negative* otherwise [1].
4. After accumulating a number of trajectories, perform the Diverse Density Algorithm to compute candidates for the option subgoal. Pick the state with the largest Diverse Density metric as the subgoal [2].
5. Construct the option input set, $\mathcal{I}$ by searching trajectories and adding those states that preceed the subgoal.
6. The termination function, β, is set to 1 for the subgoal and 0 for all other states in the input set.
7. Perform a separate Q-learning problem using the trajectories as experience. Formulate the policy, π, based on the result of this Q-learning over the option's states and trajectories. This step is known as *Experience Replay* [1].

This algorithm is the first viable method of automated subgoal discovery, but it is not without some drawbacks. First, the use of the Diverse Density Algorithm dictates that subgoals cannot be present in any negative trajectories; this has the effect of immediately eliminating any state from subgoal consideration if it appears just once along any non-goal achieving trajectory. In a two room problem similar to Figure 1, it is

[1] An episode might be cut short (and hence be classified as negative) if the agent fails to reach the goal within a predetermined number of steps.

[2] A more sophisticated variation is to successively compute the Diverse Density after each new trajectory is added and employ a running average to find that state that consistently has high Diverse Density scores.

quite possible that a trajectory moves through the doorway but then terminates before it finds the correct goal state in the right-hand room. As more trajectories are added, the effect is exacerbated because it is increasingly likely that a negative trajectory contains an otherwise good candidate for a subgoal; intuitively, this is opposite of the desired effect of increasing the chances of finding desirable subgoals with increased experience.

A second major limitation arises because the Diverse Density Algorithm employs a physical distance metric. This implies that the state space must correlate to physical distances. There are numerous applications without any notion of physical distance; it would not be possible to apply this algorithm to these learning tasks. Furthermore, there are tasks in which two states might *appear* to be physically near, but are in fact quite separate from each other. This is illustrated by State 7 and State 11 in the two-room task of Figure 1; these states appear to be close but are actually separated by a wall – temporally they are further apart.

Thirdly, the McGovern/Barto algorithm is highly sensitive to various parameters. In our experience of applying this algorithm to a larger-version of the two room problem, numerous parameters had to be adjusted precisely before useful subgoals were discovered at all. Slight deviations from these parameters caused the algorithm to fail. The list of parameters includes: the correct number of trajectories, the correct temporal length of trajectories, when to start recording trajectories, and other subtle details. Even when the algorithm did work, it worked sporadically, usually failing because viable subgoal candidates appeared on unsuccessful trajectories.

Finally, a hand-crafted filter is applied to eliminate certain states from consideration as subgoals. After application of the Diverse Density Algorithm to the two room task, the very best candidates for subgoal are the states immediately surrounding the overall goal, the states near the starting state, and then, lastly, those states in the doorway. McGovern/Barto employ a filter to eliminate states within a neighborhood of the overall goal and start states; this is another parameter that requires *a priori* knowledge of the state space.

In the next section, we present an alternative method of automated subgoal discovery based upon the McGovern/Barto algorithm that eliminates or mitigates many of these difficulties. We retain the excellent insight of the McGovern/Barto algorithm, but discard many of the limiting factors associated with the Diverse Density Algorithm.

4 The FD Algorithm for Automated Subgoal Discovery

Our alternative for automated discovery of subgoals is called the FD Algorithm because it uses a combination of a *frequency* metric and a *distance* metric:

1. Collect trajectories. We collect only positive trajectories that reach the task goal state and ignore negative trajectories. We also eliminate all cycles from positive trajectories.

2. Compute candidacy metric. For each state, we compute it's potential as a subgoal and then select the optimum state as the subgoal. This process is described fully below.

3. Use Experience Replay to initially train the option [1].

Specifically, the candidacy metric for state i, referred to as c_i, is computed as:

$$c_i = F_i D_i \qquad \forall i \in S, \tag{1}$$

where F_i is the i^{th} state's frequency measure and D_i is it's distance measure. Suppose the task has n discrete states. We collect T trajectories each of which will have no more than n states (because we eliminate cycles). The frequency measure for state i is simply the percentage of trajectories that contain state i:

$$F_i = \frac{\text{\# of trajectories with state } i}{T}. \tag{2}$$

As correctly pointed out in [5], the difficulty with using a frequency metric alone is that states near the goal tend to have the highest frequency; these are not typically the most desireable candidates for a subgoal. Thus we incorporate a distance component to our metric as well.

The distance metric for each state, D_i, is computed based on the *temporal* distance of each state from undesireable subgoal locations. McGovern/Barto employ a static filter to eliminate states near the goal as candidates for the option subgoal. Instead, the FD distance metric negatively weights states which are closer to the task goal but does not automatically preclude them from consideration.

First we compute a *simplified* distance measure, d_i, as:

$$d_i = 2 \cdot \min_{t \in T} \min_{s \in \{s_0, g\}} \frac{|s - i|}{l_t}. \tag{3}$$

If state i is not in any trajectory of T, then $d_i = 0$. We use $|s - i|$ to indicate the *temporal distance* between state i and state s. That is, if both state i and state s

exist on the same trajectory t, then $|s - i|$ is the number of steps along the trajectory to transition between the two states. In the above equation, state s is either the initial state (s_0) or the task goal state g. We choose the minimum temporal distance between state i and the start state (s_0), or state i and the goal state (g). The minimum temporal distance is normalized by the trajectory length, l_t, so that trajectories of different lengths can be compared. We compute this mininum temporal distance for every trajectory $t \in T$ that contains state i, and then select the smallest normalized temporal distance over all the trajectories. Finally we multiply by 2 so that $0 \leq d_i \leq 1$.

d_i is a linear function that is maximal ($d_i = 1$) at the midpoint along any trajectory and minimal ($d_i = 0$) at the end points (start and goal states) of the trajectory. Figure 2 illustrates this relationship.

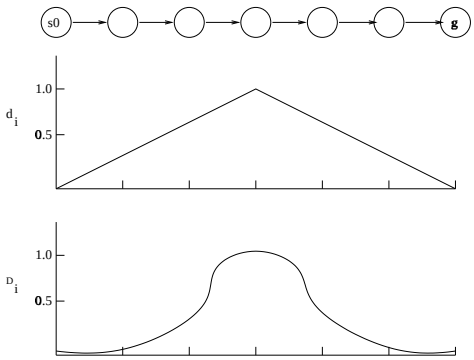

Figure 2: Example simplified distance d_i, and distance D_i

We then compute the distance measure, D_i, by passing d_i through a gaussian function:

$$D_i = e^{-1.0 \cdot \left(\frac{1-d_i}{a}\right)^b} \quad (4)$$

where a and b are parameters to shape the width and slope of the gaussian (typically $a = 0.5, b = 3.5$ unless indicated otherwise).

This metric has several advantages.

- No notion of physical distance.
- Faster to compute than Diverse Density.
- Actively prefer states nearer to the middle of the trajectory while not absolutely precluding states near the ends of the trajectory.
- Favor states that are visited more frequently.
- Fewer parameters and increased robustness with respect to parameter tuning.

5 A Case Study

In this section, we test our FD Algorithm for finding good subgoal candidates. For the purposes of comparison, we apply the algorithm to the reinforcement learning task used in previous studies on automated option creation.

The task featured in the McGovern/Barto work on automated subgoal discovery is shown in Figure 3; it consists of two rooms connected by a 2-state doorway. The overall task goal is a state near the upper corner of the right-hand room indicated by a G in the figure. The agent starts randomly in one of the states of the left-hand room. There are four deterministic actions of up, down, right, and left. The standard SARSA algorithm is applied with various reinforcement learning parameters of $\alpha = 0.1$, $\epsilon = 0.1$, and $\lambda = 0$ [7].

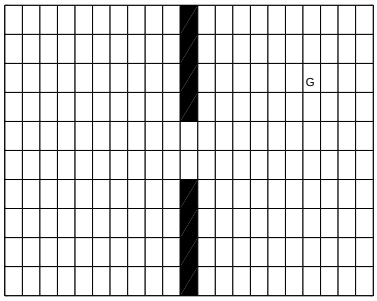

Figure 3: Two Room Task

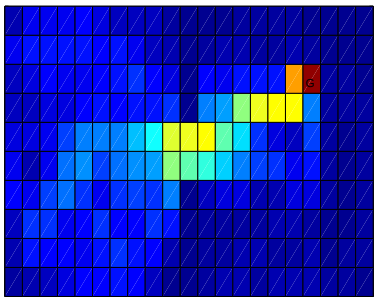

Figure 4: Frequency f_i of States in Trajectories

We collect $T = 50$ trajectories for learning experience. However, we do not collect the first 50 trajectories as these are likely to be longer and less "efficient" at moving toward the goal than later trajectories experienced after some learning has occurred. We wait until the running average trajectory length drops below a predetermined level. For this particular task, we ignore the first 150 or so trajectories and then collect the next 50 for use in the automated subgoal discovery algorithm.

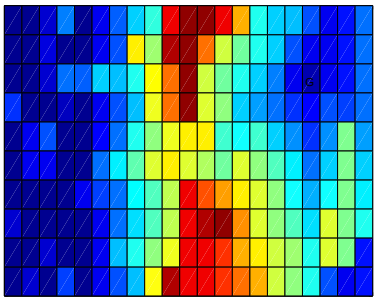

Figure 5: Simplified Distance d_i

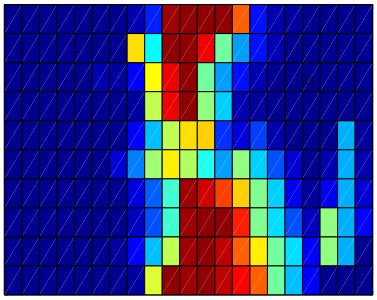

Figure 6: Distance D_i

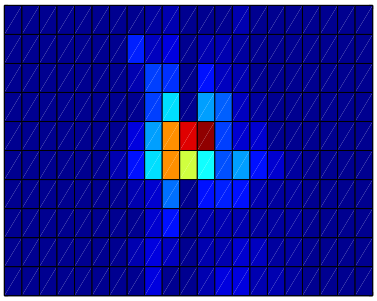

Figure 7: FD Candidacy Metric c_i

For the purposes of illustration, Figures 4 through Figure 7 show all the metrics used in computing our FD Algorithm to find good subgoal candidates. We show the value for the frequency measure f_i, the simplified distance metric d_i, the distance metric D_i, and finally the overall subgoal candidacy metric $c_i = f_i \cdot D_i$. In Figure 4 we see that states near the goal and also (to a slightly lesser extent) states in the doorway have the highest frequency metric. Figure 5 correctly shows that the simplified distance metric d_i is greatest for those states in between the goal and start states and least for those states near the goal or start states. Figure 6 shows the distance metric D_i which is merely d_i passed through a gaussian. Finally Figure 7 shows the overall candidacy metric c_i; clearly one of the two doorway states is identified as the optimal choice for a subgoal.

6 Concluding Remarks and Future Research

The FD Algorithm is able to use reinforcement learning experience to identify candidate states for use as a subgoal in automated option creation. Furthermore, this algorithm has advantages over previous attempts in that it is simpler to apply, less sensitive to parameter tuning, and most importantly, is more flexible in the range of possible tasks.

This work in automated option creation immediately introduces a list of directions for future research. We are currently engaged in the following activities:

- Continue the process of automated option discovery. The FD Algorithm selects a subgoal state and then creates the initial option using experience replay. As the agent continues to interact with its environment, the option can be tuned to better suit the subgoal (states can be added, policies can be tweaked).
- Retain and use the underlying Option Value Function. Each option can store a separate value function (and policy) to measure the cost of moving from an option state to the option subgoal. As pointed out in [6], the option value function can facilitate an off-line "dynamic programming" like approach to computing the value function of the overall task for states both in this option and in other options.
- Include Multiple Options. The creation of multiple options introduces additional problems in effectively achieving good option coverage over the state space while simultaneously limiting unneccessary option overlap. This is related to the problem of distributing local representational resources (ie radial basis functions) in a state space.
- Extend option-based reinforcement learning to POMDPs (partially observable Markov decision processes). These options are fixed to specific states. There are tasks with similar groups of states (consider a 5-room task in which each room looks exactly the same). Here, we would rather relate options to *observations* of the state space rather than to specific states. In this way, the same option can be applied to different but similar locations within the state space.

References

[1] L. J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321, 1992.

[2] O. Maron and T. Lozano-Perez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems, NIPS 98*, pages 570–576, 1998.

[3] A. McGovern and A. G. Barto. Accelerating reinforcement learning through the discovery of useful subgoals. In *Proceedings of teh 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space: i-SAIRAS 2001*, 2001.

[4] A. McGovern and A. G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 361–368, Williams College, MA., 2001.

[5] A. McGovern and A. G. Barto. Linear discriminant diverse density for automatic discovery of subgoals in reinforcement learning. In *Workshop on Hierarchy and Memory in Reinforcement Learning, ICML 2001*, Williams College, MA, 2001.

[6] R. S. Sutton and D. Precup. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.