Discovering Options for Exploration by Minimizing Cover Time

Yuu Jinnai [1] **Jee Won Park** [1] **David Abel** [1] **George Konidaris** [1]

Abstract

One of the main challenges in reinforcement learning is solving tasks with sparse reward. We show that the difficulty of discovering a distant rewarding state in an MDP is bounded by the expected cover time of a random walk over the graph induced by the MDP's transition dynamics. We therefore propose to accelerate exploration by constructing options that minimize cover time. We introduce a new option discovery algorithm that diminishes the expected cover time by connecting the most distant states in the state-space graph with options. We show empirically that the proposed algorithm improves learning in several domains with sparse rewards.

1. Introduction

A major challenge in reinforcement learning is how to explore in environments where reward is sparse.

One approach is the construction of temporally extended actions, or *options* (Sutton et al., 1999) for exploration (Machado et al., 2017). However, existing approaches lack a principled theoretical grounding and their effectiveness can only be evaluated empirically.

We introduce an option discovery method that explicitly aims to improve exploration in sparse reward domains by minimizing the expected number of steps required to reach an unknown rewarding state. First, we model the behavior of an agent early in its learning process (that is, before observing the reward signal) as a uniform random walk over the graph induced by the MDP's transition dynamics. We show that minimizing the graph *cover time*—the number of steps required for a random walk to visit every state (Broder & Karlin, 1989)—reduces the expected number of steps required to reach an unknown rewarding state. We then introduce a polynomial time algorithm to find a set of options guaranteed to reduce the expected cover time using the transition function either given to or learned by the agent. Finding a set of edges that minimizes expected cover time is an extremely hard combinatorial optimization problem (Braess, 1968; Braess et al., 2005). Thus, our algorithm instead seeks to minimize the upper bound of the expected cover time given as a function of the algebraic connectivity of the graph Laplacian (Fiedler, 1973; Broder & Karlin, 1989; Chung, 1996) using the heuristic method by Ghosh & Boyd (2006).

Finally, we evaluate our option discovery algorithm in six discrete domains where the agent is given the state-space graph but must learn the location of the reward online. Our empirical results demonstrate that the approach outperforms previous state-of-the-art methods.

[1]Brown University, Providence, RI, United States. Correspondence to: Yuu Jinnai <yuu_jinnai@brown.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

2. Background

2.1. Reinforcement Learning

Reinforcement learning is the problem of learning a policy that maximizes the total expected reward obtained by an agent interacting with an environment. The environment is often modeled as a Markov Decision Process (MDP) (Puterman, 1994). An MDP is a five tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a state transition function, $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward function, $\gamma \rightarrow [0, 1]$ is a discount factor.

The agent selects actions according to a policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ mapping states to actions. The expected total discounted reward from state s following a policy π is the value of the state:

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s, \pi(s), s') V^\pi(s'). \quad (1)$$

This function is called a *value function*. The action-value function of a policy is an expected total discount reward received by executing an action a and then follow policy π:

$$Q^\pi(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s') V^\pi(s'). \quad (2)$$

The goal of the agent is to learn an optimal policy π^* which maximizes the total discounted reward: $\pi^* =$

$\arg\max_\pi V^\pi$, with corresponding optimal value functions $V^* = \max_\pi V^\pi$ and $Q^* = \max_\pi Q^\pi$.

A state-transition in an MDP by a stationary policy π can be modeled as a Markov chain $\{X_t\}$ where $P(X_{t+1}|X_t) = \sum_{a \in A} \pi(a|s)T(s,a,s')|_{X_{t+1}=s', X_t=s}$. The state-transition graph $G = (V, E)$ of an MDP is a graph with nodes representing the states in the MDP and the edges representing state adjacency in the MDP. More precisely, $V = S$, $e(s, s') \in E$ iff $\exists a T(s,a,s') > 0 \vee T(s', a, s) > 0$. An adjacency matrix is a square matrix of size $|S| \times |S|$ with (i, j)-value being 1 if $e(s_i, s_j) \in E$ and 0 otherwise.

2.2. Options

Temporally extended actions offer great potential for mitigating the difficulty of solving difficult MDPs in planning and reinforcement learning (Sutton et al., 1999). We use one such framework, the *options framework*, which defines a temporally-extended action as follows.

Definition 1 (option): *An option o is defined by a triple: $(\mathcal{I}, \pi, \beta)$ where:*

- *$\mathcal{I} \subseteq \mathcal{S}$ is a set of states where the option can initiate,*
- *$\pi : \mathcal{S} \rightarrow \Pr(\mathcal{A})$ is a policy,*
- *$\beta : \mathcal{S} \rightarrow [0, 1]$, is a termination condition.*

Many previous approaches propose heuristic methods to generate options where effectiveness can only be demonstrated empirically (Iba, 1989; McGovern & Barto, 2001; Menache et al., 2002; Stolle & Precup, 2002; Şimşek & Barto, 2004; Simsek et al., 2005; Şimşek & Barto, 2009; Konidaris & Barto, 2009; Machado & Bowling, 2016; Kompella et al., 2017; Machado et al., 2017; 2018; Eysenbach et al., 2019; Nair et al., 2018; Riedmiller et al., 2018).

As the options framework is very general and difficult to analyze, we focus on *point options* (Jinnai et al., 2019), a simple subclass of options where both the initiation set and termination condition consist of a single state.

Definition 2 (Point option): *A* **point option** *is any option whose initiation set and termination set are each true for exactly one state each:*

$$\begin{aligned} |\{s \in \mathcal{S} : \mathcal{I}(s) = 1\}| &= 1, \\ |\{s \in \mathcal{S} : \beta(s) > 0\}| &= 1, \\ |\{s \in \mathcal{S} : \beta(s) = 1\}| &= 1. \end{aligned} \tag{3}$$

Adding a point option corresponds to inserting a single edge into the graph induced by the MDP dynamics. We refer to the state with $\beta(s) = 1$ as the subgoal state. Point options are a useful subclass for several reasons. A point option is a simple model of a temporally extended action whose effect on the state-space graph is easy to specify, and whose policy can often be efficiently computed. Moreover, any option with a single termination state can be represented as a collection of point options.

3. Cover Time

In this section we model the behavior of the agent *in the first episode* as a random walk induced by a fixed stationary policy, and show an upper bound to *the expected cover time* of the random walk. We model the behavior of a fixed stationary policy for two reasons. First, it is a reasonable model for an agent with no prior knowledge of the task. Second, it serves as a *worst-case* analysis: it is reasonable to assume that most of the cases efficient exploration algorithms such as UCRL (Ortner & Auer, 2007; Jaksch et al., 2010) explore faster than a fixed stationary policy. Thus the upper bound we show for the expected cover time is applicable to other algorithms.

Intuitively, the expected cover time is the time required for a random walk to visit all the vertices in a graph (Broder & Karlin, 1989). To define it formally, we first define the hitting time of any discrete Markov chain $\{X_t\}$. Let us assume this Markov chain has the state space of V, the vertices of graph G. The hitting time H_{ij}, where $i, j \in V$, is

$$H_{ij} = \inf \{t : X_t = j | X_0 = i\}. \tag{4}$$

In other words, H_{ij} is the greatest lower bound on the number of time step t required to reach state j after starting at state i. Cover time starting from state i is:

$$C_i = \max_{j \in V} H_{ij}, \tag{5}$$

and the expectation of cover time, $\mathbb{E}[C(G)]$, is the expected cover time of trajectories induced by the random walk, maximized over the starting states (Broder & Karlin, 1989). As such, the expected cover time bounds how likely a random walk leads to a rewarding state.

Theorem 1. *Assume a stochastic shortest path problem to reach a goal g where a non-positive reward $r_c \leq 0$ is given for non-goal states and $\gamma = 1$. Let P be a random walk transition matrix, $P(s, s') = \sum_{a \in A} \pi(s)T(s, a, s')$ then:*

$$\forall g : V_g^\pi(s) \geq r_c \mathbb{E}[C(G)]. \tag{6}$$

where $\mathbb{E}[C(G)]$ is the expected cover time of a transition matrix P.

Proof. The value of state s is r_c times the expected number of steps to reach the goal state. Thus,

$$\begin{aligned} V_g^\pi(s) &= r_c \mathbb{E}[H_{sg}] \\ &\geq r_c \mathbb{E}[\max_{s' \in S} H_{ss'}] \\ &= r_c \mathbb{E}[C_s(G)] \\ &\geq r_c \mathbb{E}[C(G)] \end{aligned} \tag{7}$$

□

The theorem suggests that *a smaller expected cover time means easier exploration*. Now the question is how to reduce the expected cover time without prior reward information.

Let P be a random walk induced by a fixed policy π in an MDP. Broder & Karlin (1989) showed that the expected cover time $\mathbb{E}[C(G)]$ of a random walk P can be bounded using the second largest eigenvalue of the random walk matrix $\lambda_{k-1}(P)$:

$$\mathbb{E}[C(G)] \leq \frac{n^2 \ln n}{1 - \lambda_{k-1}(P)}(1 + o(1)), \tag{8}$$

where $n = |V|$ and k is the number of eigenvalues. The normalized graph Laplacian of an unweighted undirected graph is defined as:

$$\mathcal{L} = I - T^{-1/2} A T^{-1/2}, \tag{9}$$

where I is an identity matrix (Chung, 1996). The random walk matrix can be written in terms of the Laplacian:

$$P = T^{-1} A = T^{-1/2}(I - \mathcal{L}) T^{1/2}. \tag{10}$$

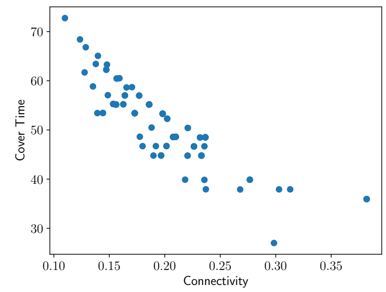
Because P and $I - \mathcal{L}$ are similar matrices, they have the same eigenvalues and eigenvectors. Thus, $\lambda_{k-1}(P) = 1 - \lambda_2(\mathcal{L})$, where $\lambda_2(\mathcal{L})$ is the second smallest eigenvalue of $\mathcal{L}$. From Equation 8,

$$\mathbb{E}[C(G)] \leq \frac{n^2 \ln n}{\lambda_2(\mathcal{L})}(1 + o(1)). \tag{11}$$

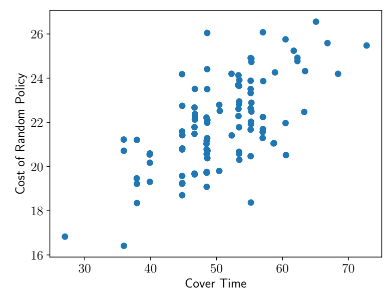
Thus, *the larger the $\lambda_2(\mathcal{L})$ is, the smaller the upper bound of the expected cover time*.

The second smallest eigenvalue of $\mathcal{L}$ is known as the *algebraic connectivity* of the graph and its corresponding eigenvector is called *Fiedler vector* (Fiedler, 1973). There are several operations we can apply to the graph to increase the algebraic connectivity. First, adding nodes to the graph can increase the algebraic connectivity. However, this increases the number of nodes n, and thus the cover time does not always improve. Second, we can rewire edges in the graph. However, rewiring edges is undesirable as it amounts to removing primitive actions from the MDP which may damage the agent's ability to optimally solve the MDP. Third, we can add edges to the graph, which in the reinforcement learning setting amounts to adding options to the agent. This strategy preserves optimality as it does not remove any primitive actions. Therefore, adding edges (i.e. options) is a reliable way to reduce the cover time without potentially sacrificing optimality.

As far as we are aware, we are the first to introduce the concept of the cover time to reinforcement learning.



(a) (Connectivity) $\sim$ (cover time)



(b) (cover time) $\sim$ (cost of the policy)

Figure 1: (a) The relationship between the algebraic connectivity (λ_2) and the cover time of a random walk on randomly generated connected graphs. (b) The relationship between the cover time of a random walk and the cost of random policy.

3.1. Empirical Evaluation

The preceding section showed that the bigger the algebraic connectivity, the smaller the upper bound of the expected cover time. We now empirically examine (1) the relationship between the algebraic connectivity and cover time, and (2) the relationship between cover time and the difficulty of an MDP.

We randomly generated shortest path problems and plotted the cost of a random policy, the cover time, and the algebraic connectivity of the state-space graph. We generated 100 random connected graphs with 10 nodes with the edge density fixed to 0.3. To generate a connected graph, we use the following procedure. First, we start with a single node. We pick one node from the existing graph and add an edge to connect to a new node. We follow this procedure for the number of nodes $n-1$, generating a random tree of size n. Then, we pick an edge uniformly randomly from E^c until the edge density reaches the threshold. We approximated the expected cover time of a random walk on a random graph by sampling 10,000 trajectories induced by the random walk and computing their average cover time.

We generated a shortest path problem by picking an initial state and a goal state randomly for each graph. The agent can transition to each neighbor with a cost of 1.

Figure 1a shows the relationship of the algebraic connectivity and the expected cover time of the random walk induced by a uniform random policy. The result shows that the random walk tends to have smaller expected cover time when the underlying state-transition graph has larger algebraic connectivity. Figure 1b shows the expected cost of a random policy from the initial state to reach the goal state. The cost of a random policy is correlated to the cover time.

4. Covering Options

We now describe an algorithm to automatically find options that minimize the expected cover time. The algorithm is approximate, since the problem of finding such a set of options is computationally difficult; it is thought to be NP-hard, though that has not been proven (Aldous & Fill, 2002). Even a good solution is hard to find due to the Braess's paradox (Braess, 1968; Braess et al., 2005) which states that the expected cover time does not monotonically decrease as edges are added to the graph.

Thus, the expected cover time is often minimized indirectly via maximizing the algebraic connectivity (Fiedler, 1973; Chung, 1996). The expected cover time is upper bounded by quantity involving the algebraic connectivity (Equation 11), and by maximizing it the bound can be minimized (Broder & Karlin, 1989). Adding a set of edges to maximize the algebraic connectivity is NP-hard (Mosk-Aoyama, 2008), so we use the approximation method by Ghosh & Boyd (2006):

1. Compute the second smallest eigenvalue and its corresponding eigenvector (i.e., the Fiedler vector) of the Laplacian of the state transition graph G.
2. Let v_i and v_j be the state with largest and smallest value in the eigenvector respectively. Generate two point options; one with $\mathcal{I} = \{v_i\}$ and $\beta = \{v_j\}$ and the other one with $\mathcal{I} = \{v_j\}$ and $\beta = \{v_i\}$. Each option policy is the optimal path from the initial state to the termination state.
3. Set $G \leftarrow G \cup \{(v_i, v_j)\}$ and repeat the process until the number of options reaches k.

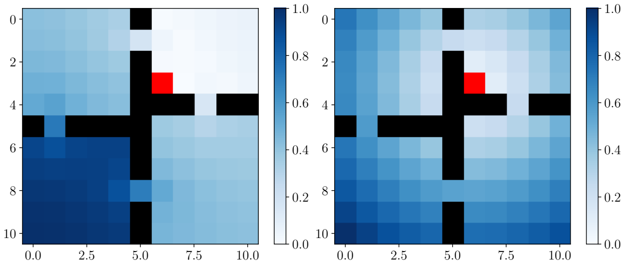


Figure 2: The distance between the red state and all other states, measured via Fiedler vector (left) and Euclidean distance (right). The Fiedler vector captures the connectivity of the graph, so distances measured using it reflect path lengths in the graph; the pair of nodes with the maximum and the minimum value are the farthest apart.

Intuitively, the algebraic connectivity represents how tightly the graph is connected. The Fiedler vector is an embedding of a graph to a line (single real value) where nodes connected by an edge tend to be placed close by (see Figure 2 for example). A pair of nodes with the maximum and minimum value in the Fiedler vector are the most distant nodes in the embedding space. Our method greedily connects the two most distant nodes in the embedding; this operation greedily maximizes the algebraic connectivity to a first order approximation (Ghosh & Boyd, 2006).

Thus, our algorithm generates options which maximize the algebraic connectivity, which in turn minimizes the upper bound of the expected cover time. The algorithm is guaranteed to improve the upper bound and the lower bound of the expected cover time:

Theorem 2. *Assume that a random walk induced by a policy π is a uniform random walk:*

$$P(u,v) := \begin{cases} 1/d_u & \text{if } u \text{ and } v \text{ are adjacent,} \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

where d_u is the degree of the node u. Adding two options by the algorithm improves the upper bound of the cover time if the multiplicity of the second smallest eigenvalue is one:

$$\mathbb{E}[C(G')] \leq \frac{n^2 \ln n}{\lambda_2(\mathcal{L}) + F}(1 + o(1)), \tag{13}$$

where $\mathbb{E}[C(G')]$ is the expected cover time of the augmented graph, $F = \frac{(v_i - v_j)^2}{6/(\lambda_3 - \lambda_2) + 3/2}$, and v_i, v_j are the maximum and minimum values of the Fiedler vector.

Proof. Assume the multiplicity of the second smallest eigenvalue is one. Let $\mathcal{L}'$ be the graph Laplacian of the graph with an edge inserted to $\mathcal{L}$ using the algorithm by Ghosh & Boyd (2006). By adding a single edge, the algebraic connectivity is guaranteed to increase at least by F:

$$\lambda_2 \geq \lambda_2 + \frac{(v_i - v_j)^2}{6/(\lambda_3 - \lambda_2) + 3/2}, \tag{14}$$

and the upper bound of the cover time is guaranteed to decrease:

$$\begin{aligned}\mathbb{E}[C(G')] &\leq \frac{n^2 \ln n}{\lambda_2}(1 + o(1)) \\ &\leq \frac{n^2 \ln n}{\lambda_2 + \frac{(v_i - v_j)^2}{6/(\lambda_3 - \lambda_2) + 3/2}}(1 + o(1)).\end{aligned}$$

As $\frac{(v_i - v_j)^2}{6/(\lambda_3 - \lambda_2) + 3/2}$ is positive,

$$\frac{n^2 \ln n}{\lambda_2 + \frac{(v_i - v_j)^2}{6/(\lambda_3 - \lambda_2) + 3/2}}(1+o(1)) < \frac{n^2 \ln n}{\lambda_2}(1+o(1)), \tag{15}$$

thus the upper bound is guaranteed to decrease. □

Note that if the multiplicity of the second smallest eigenvalue is more than one, then adding any single option cannot improve the algebraic connectivity. Assume the second smallest eigenvalue is more than one. Then, $\lambda_2(\mathcal{L}) = \lambda_3(\mathcal{L})$. From eigenvalue interlacing (Haemers, 1995), for any edge insertion, $\lambda_2(\mathcal{L}) \leq \lambda_2(\mathcal{L}') \leq \lambda_3(\mathcal{L})$. Thus, $\lambda_2(\mathcal{L}') = \lambda_2(\mathcal{L})$.

The state transition graph G must be given to or learned by the agent. We assume that the graph is strongly connected, so every state is reachable from every other state, and also that the graph is undirected.

As in the work by Machado et al. (2017), our algorithm can be generalized to the function approximation case using an incidence matrix instead of an adjacency matrix.

4.1. Comparison to Eigenoptions

Machado et al. (2016; 2017; 2018) proposed a method to generate options using the Laplacian eigenvectors. The proposed algorithm is similar to eigenoptions but different in several aspects. First and foremost, covering options explicitly seek to speed up the exploration by maximizing the algebraic connectivity to improve the upper bounds of the cover time. While the eigenoptions also use the graph Laplacian for option discovery, their method is repurposed from a feature construction method. Second, they are solving a different optimization problem. The k-th covering option is the one minimizing the algebraic connectivity of the graph augmented with 1 to $k-1$-th options. The k-th eigenoption minimizes the algebraic connectivity of the original subject to the constraint that the option has to be orthogonal to 1 to $k-1$-th options. We did not find analytical results for how the orthogonal constraint can contribute to minimizing the algebraic connectivity or the expected cover time. Third, covering options are fast to compute as the algorithms only needs to compute the Fiedler vector. Although computing the whole graph spectrum is a heavy matrix operation, the Fiedler vector can be computed efficiently even for very large graphs (Koren et al., 2002).

5. Empirical Evaluation

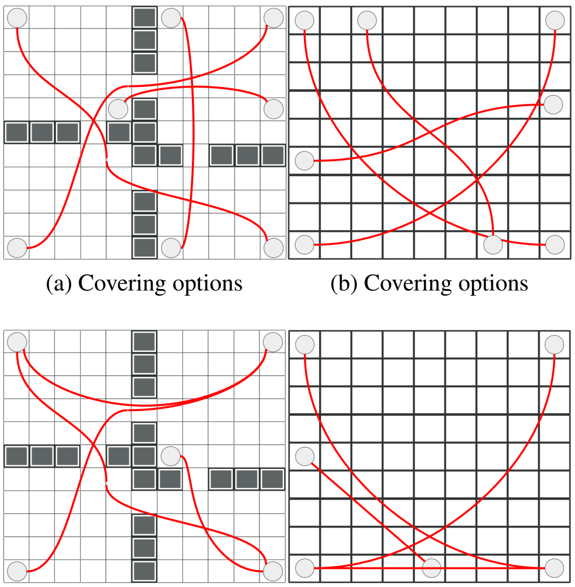
We used six MDPs in our empirical study: a 9x9 grid, a four-room gridworld, Taxi, Towers of Hanoi, Parr's maze, and Race Track. 9x9grid, four-room, and Parr's maze (Parr & Russell, 1998) are 2-dimensional grid pathfinding problems where the task is to reach a specific location. The agent can move in four directions but cannot cross walls. The task in Taxi (Dietterich, 2000) is to pick-up passengers and sends them to their destination. Only one passenger can ride on the taxi at the same time. Towers of Hanoi consists of three pegs of different-size discs sorted in decreasing order of size on one of the pegs. The goal is to move all discs from their initial peg to a goal peg while keeping the constraint that a smaller disc is above a larger one. In the Race Track task the agent must reach the finish line by driving a car. The car position and the velocity are discrete. The agent can change the horizontal and vertical velocity by +1, -1, or 0 in each step. If the car hits the track boundary, it is moved back to the starting position.

We compared the performance of covering options, eigenoptions (Machado et al., 2017), and betweenness options (Şimşek & Barto, 2009). We compare against these methods because they are the state-of-the-art option generation methods which do not require reward information. Machado et al. (2017) proposed to generate a set of options which initialize at every state and terminate at the states which have highest/lowest values for each eigenvector. To make the comparison simple, we consider a point option version of eigenoption method. For k-eigenvectors which correspond to the smallest k eigenvalues, we generate a point option from a state with the highest/lowest value to a state the lowest/highest value in the eigenvector. The point option constructed in this way minimizes the eigenvalue of each corresponding eigenvector.

First, we consider the case where the agent has perfect knowledge of the state-space graph in advance. Then, we consider the case where the agent must sample the state-transition for given amount of steps. Finally, we evaluate an online option generation which discover options while training in the environment.

5.1. Offline Option Discovery

Figure 3 shows the eight options generated by covering options, and eigenoptions on four-room domain and a 9x9



(a) Covering options (b) Covering options

(c) Eigenoptions (d) Eigenoptions

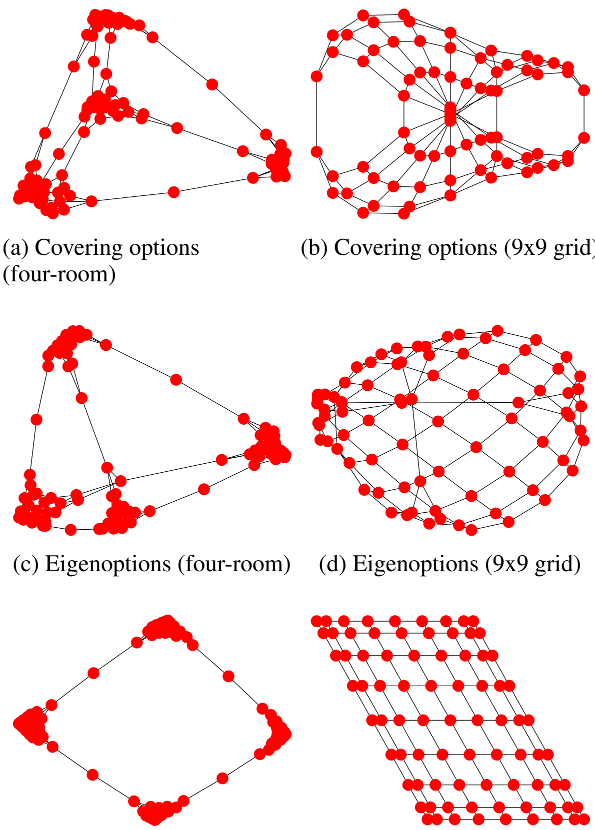
Figure 3: Visualization of covering options vs. eigenoptions on four-room domain and 9x9 grid.

grid-world domain. Note that there are multiple possible set of options acquired by the algorithm and we showed one such set.

Table 1 shows the algebraic connectivity and expected cover time with generated options. In both domains the covering options achieved larger algebraic connectivity and smaller expected cover time than eigenoptions. Figure 4 shows the spectral graph drawing (Koren, 2003) of the state-transition graph augmented with the generated options. The spectral graph drawing is a technique to visualize the graph topology using eigenvectors of the graph Laplacian. Each node n in the state-space graph is placed at $(v_2(n), v_3(n))$ in the (x, y)-coordinate, where v_i is the i-th smallest eigenvector of the graph Laplacian. The figure indicates that the option generation methods are successfully connecting distant states.

four-room	λ_2	Expected Cover Time
Covering options	**0.065**	**672.0**
Eigenoptions	0.054	695.9
No options	0.023	1094.8
9x9 grid	**λ_2**	**Expected Cover Time**
Covering options	**0.24**	**258.6**
Eigenoptions	0.19	261.5
No options	0.12	460.5

Table 1: Comparison of the algebraic connectivity and the expected cover time. For Covering options and eigenoptions we add 8 options.



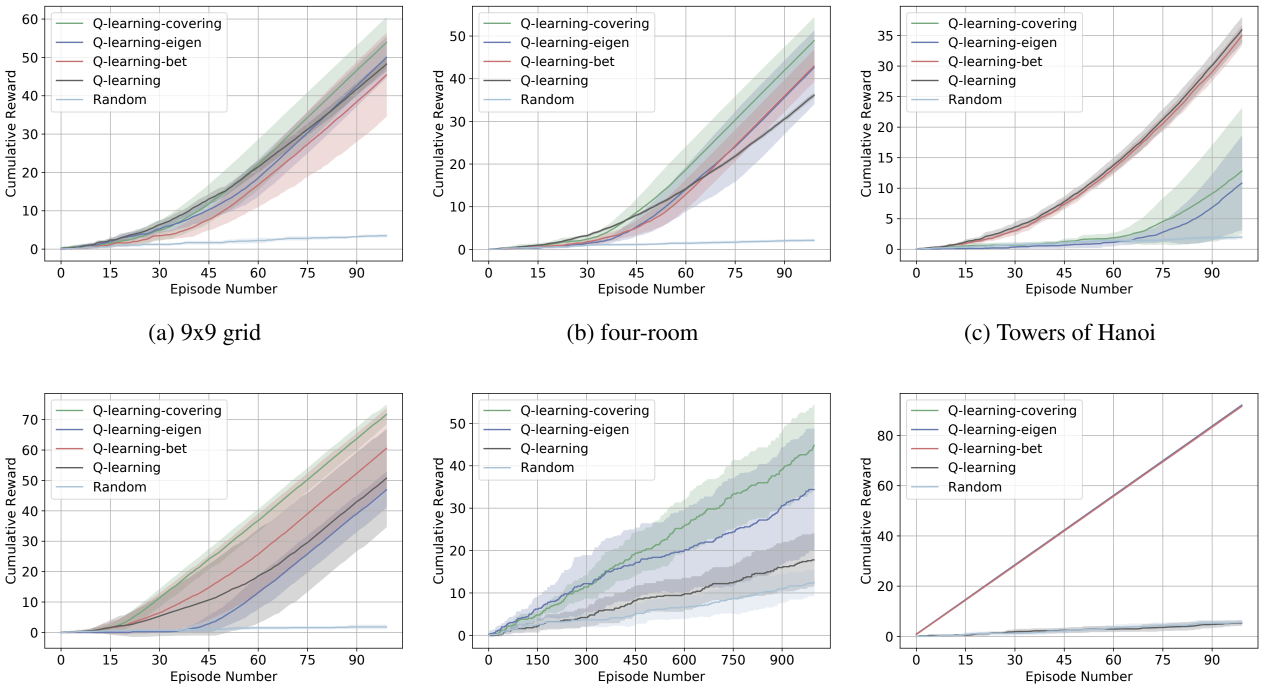
(a) Covering options (four-room) (b) Covering options (9x9 grid)

(c) Eigenoptions (four-room) (d) Eigenoptions (9x9 grid)

(e) No options (four-room) (f) No options (9x9 grid)

Figure 4: Spectral graph drawing of the state-transition graph.

We now evaluate the utility of each discovered options for learning. We used Q-learning (Watkins & Dayan, 1992) ($\alpha = 0.1, \gamma = 0.95$) for 100 episodes, 100 steps for 9x9 grid, 500 steps fourroom, Hanoi, and Taxi. We generated 8 options with each algorithm using the adjacency matrix representing the state-transition of the MDP. Figure 5 compares accumulated rewards averaged over 5 runs. In all experiments, covering options outperformed or was on par with eigenoptions. Figure 6a shows the comparison of accumulated rewards with varying number of covering options on fourroom domain. Overall, adding more options improves performance but the additional utility is diminished. It is to be expected as the target function is a concave function of the number of edges added which roughly means that the first few edges added lead to a much greater increase in algebraic connectivity than those added later (Ghosh & Boyd, 2006). Next, we evaluated the performance of the options with all states in the initiation set. Figure 6b, 6c



(a) 9x9 grid (b) four-room (c) Towers of Hanoi

(d) Taxi (e) Parr's Maze (f) Race Track

Figure 5: Comparison of performance with different option generation methods. Options are generated offline from the adjacency matrix for 9x9grid, four-room, Towers of Hanoi, and Taxi. Options are generated offline from an incidence matrix for Parr's maze and Race Track. Reward information is not used for generating options.

shows the comparison of accumulated rewards on fourroom and 9x9 grid domain.

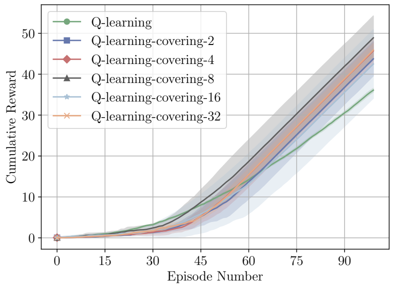
5.2. Offline Approximate Option Discovery

In the previous subsection we assumed that the agents have access to the adjacency matrix of the MDP. However, this may be difficult to achieve when the number of states is too large, as agents are not able to observe the whole state transitions in a reasonable amount of time. Following the evaluation of Machado et al. (2017), we evaluate our method using a sample-based approach for option discovery. Instead of giving the agent an access to the whole adjacency matrix, the agent sampled 100 trajectories of a uniform random policy to generate an incidence matrix. We sampled each trajectory for 1000 steps for Parr's maze and 100 steps for the Race Track domain. We feed the incidence matrix instead of the adjacency matrix to the option generation method. As the agent has no prior knowledge on states outside the states in the incidence matrix, the agent terminates the option if it reached the states not the incidence matrix in addition to the subgoal state. Other experimental settings are the same as the previous subsection. Figure 5 shows the resulting performance. Overall, Covering options is outperforming or on par with eigenoptions. We have no results on betweenness options for Parr's maze as it took more than 20 minutes to generate the options.

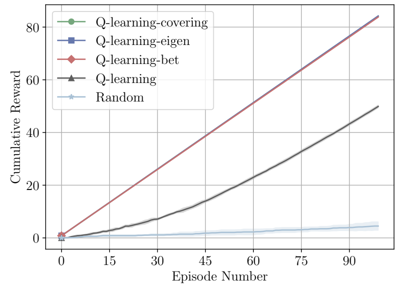
5.3. Online Option Discovery

In the previous two subsections, we evaluated option discovery methods assuming that the agent has access to the state-transition function prior to solving the task itself. This assumption is reasonable in some situations such as multi-task reinforcement learning where the agent is supposed to solve multiple different tasks (reward function) in the same domain (problems with the same transition function).

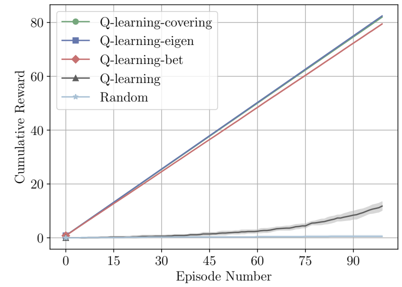
In this section we evaluate our method on online option discovery. The agents generate 4 options to add to their option set every 10000 step for Parr's maze and 500 steps for the Towers of Hanoi and Taxi until the number of options reaches 32. The agents learned for 100 episodes, and episodes were 10,000 steps long for Parr's maze and 100 steps for the Towers of Hanoi and Taxi. We used Q-learning (Watkins & Dayan, 1992) ($\alpha = 0.1, \gamma = 0.95$). To compute the policy of each option, we feed the trajectories sampled by the agent so far to learn Q-values off-policy ($\alpha = 0.1, \gamma = 0.95$). We give an intrinsic reward of 1 to the agent when it reaches the subgoal state and ignore the



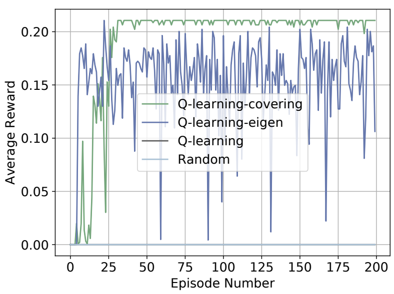
(a) four-room



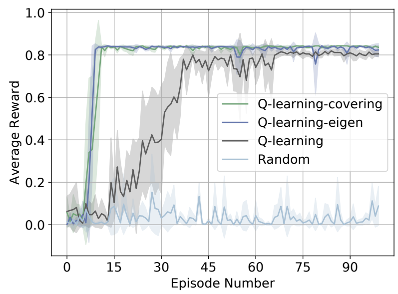
(b) 9x9 grid



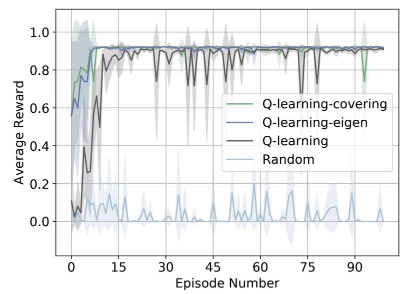
(c) four-room



(d) Parr's maze



(e) Towers of Hanoi



(f) Taxi

Figure 6: (a) Comparison of performance with varying number of covering options. (b-c) Comparison of performance of options which all states are included in the initiation sets. (d-f) Comparison of online option generation methods.

rewards from the environment.

Figure 6d, 6e, and 6f shows the resulting performance. The agents with options are able to learn the policy faster than the agent only with primitive actions. The agents with options can reliably find the goal state even in Parr's maze whereas an agent with primitive actions is unable to find the goal.

6. Related Work

Many option discovery algorithms are based on informative rewards and are thus task dependent. These methods often decompose the trajectories reaching the rewarding states into options. Several works have proposed generating intrinsic rewards from trajectories reaching these rewarding states (McGovern & Barto, 2001; Menache et al., 2002; Konidaris & Barto, 2009), while other approaches use gradient descent to generate options using the observed rewards (Mankowitz et al., 2016; Bacon et al., 2017; Harb et al., 2018).

However, such approaches are often not applicable to sparse reward problems: if rewards are hard to reach using only primitive actions, options are unlikely to be discovered. Thus, some works have investigated generating options without using reward signals. Stolle & Precup (2002) proposed to set states with high visitation count as subgoal states, resulting in identifying bottleneck states in the four-room domain. Şimşek & Barto (2009) generalized the concept of bottleneck states to the (shortest-path) betweenness of the graph to capture how pivotal the state is. Menache et al. (2002) used a learned model of the environment to run a Max-Flow/Min-Cut algorithm to the state-space graph to identify bottleneck states whereas Simsek et al. (2005) proposed to apply spectral cut to identify bottlenecks. These methods generate options to leverage the idea that subgoals are states visited most frequently. On the other hand, Şimşek & Barto (2004) proposed to generate options to encourage exploration by generating options to relatively novel states, encouraging exploration.

7. Conclusions

In this paper, we tackled the sparse reward problem by discovering options that encourage exploration. We introduced the expected cover time which bounds the expected number of steps to reach the undiscovered rewarding state, and introduced an option discovery method, Covering options, which adds options that reduces the expected cover time. We showed analytically that our method guarantees improvement of the upper bound of the expected cover time under certain conditions. We further conduct experiments, finding that Covering options outperforms the previous state-of-the-art in multiple sparse reward tasks.

Acknowledgements

We thank Richard Evan Schwartz who provided insight and expertise that greatly assited the research. This work is supported in part by an AFOSR Young Investigator Grant to George Konidaris, under agreement number FA9550-17-1-0124.

References

Aldous, D. and Fill, J. A. Reversible markov chains and random walks on graphs, 2002. Unfinished monograph, recompiled 2014, available at `http://www.stat.berkeley.edu/$\sim$aldous/RWG/book.html`.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Braess, D. Über ein paradoxon aus der verkehrsplanung. *Unternehmensforschung*, 12(1):258–268, 1968.

Braess, D., Nagurney, A., and Wakolbinger, T. On a paradox of traffic planning. *Transportation science*, 39(4):446–450, 2005.

Broder, A. Z. and Karlin, A. R. Bounds on the cover time. *Journal of Theoretical Probability*, 2(1):101–120, 1989.

Chung, F. R. *Spectral graph theory*. American Mathematical Society, 1996.

Dietterich, T. G. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *Proceedings of the Seventh International Conference on Learning Representations*, 2019.

Fiedler, M. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

Ghosh, A. and Boyd, S. Growing well-connected graphs. In *Decision and Control, 2006 45th IEEE Conference on*, pp. 6605–6611. IEEE, 2006.

Haemers, W. H. Interlacing eigenvalues and graphs. *Linear Algebra and its applications*, 226:593–616, 1995.

Harb, J., Bacon, P.-L., Klissarov, M., and Precup, D. When waiting is not an option: Learning options with a deliberation cost. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Iba, G. A. A heuristic approach to the discovery of macro-operators. *Machine Learning*, 3(4):285–317, 1989.

Jaksch, T., Ortner, R., and Auer, P. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

Jinnai, Y., Abel, D., Hershkowitz, D. E., Littman, M. L., and Konidaris, G. Finding options that minimize planning time. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Kompella, V. R., Stollenga, M., Luciw, M., and Schmidhuber, J. Continual curiosity-driven skill acquisition from high-dimensional video inputs for humanoid robots. *Artificial Intelligence*, 247:313–335, 2017.

Konidaris, G. and Barto, A. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems*, pp. 1015–1023, 2009.

Koren, Y. On spectral graph drawing. In *International Computing and Combinatorics Conference*, pp. 496–508. Springer, 2003.

Koren, Y., Carmel, L., and Harel, D. Ace: A fast multiscale eigenvectors computation for drawing huge graphs. In *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, pp. 137–144. IEEE, 2002.

Machado, M. C. and Bowling, M. Learning purposeful behaviour in the absence of rewards. *arXiv preprint arXiv:1605.07700*, 2016.

Machado, M. C., Bellemare, M. G., and Bowling, M. A Laplacian framework for option discovery in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2295–2304, 2017.

Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesauro, G., and Campbell, M. Eigenoption Discovery through the Deep Successor Representation. In *Proceedings of the International Conference on Learning Representations*, 2018.

Mankowitz, D. J., Mann, T. A., and Mannor, S. Adaptive skills adaptive partitions (ASAP). In *Advances in Neural Information Processing Systems*, pp. 1588–1596, 2016.

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

Menache, I., Mannor, S., and Shimkin, N. Q-cut - dynamic discovery of sub-goals in reinforcement learning. In *Proceedings of the 13th European Conference on Machine Learning*, pp. 295–306. Springer, 2002.

Mosk-Aoyama, D. Maximum algebraic connectivity augmentation is np-hard. *Operations Research Letters*, 36 (6):677–679, 2008.

Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pp. 9209–9220, 2018.

Ortner, P. and Auer, R. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, pp. 49–56, 2007.

Parr, R. and Russell, S. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, pp. 1043–1049, 1998.

Puterman, M. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., van de Wiele, T., Mnih, V., Heess, N., and Springenberg, J. T. Learning by playing solving sparse reward tasks from scratch. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4344–4353, 2018.

Şimşek, Ö. and Barto, A. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning*, pp. 751–758, 2004.

Şimşek, Ö. and Barto, A. G. Skill characterization based on betweenness. In *Advances in neural information processing systems*, pp. 1497–1504, 2009.

Simsek, O., Wolfe, A., and Barto, A. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the Twenty Second International Conference*, pp. 816–823, 2005.

Stolle, M. and Precup, D. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pp. 212–223, 2002.

Sutton, R., , Precup, D., and Singh, S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1): 181–211, 1999.

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