Discovering Options That Minimize Average Planning Time

Alexander Ivanov[1], **Akhil Bagaria**[2*†], **George Konidaris**[1†]

[1] Brown University, Providence, RI, USA
[2] Amazon, New York, NY, USA
alexander_ivanov@brown.edu, akhilbg@amazon.com, gdk@cs.brown.edu

Abstract

We present an option discovery algorithm that accelerates planning by minimizing the shortest distance between any two states in the MDP. The proposed algorithm produces options that approximately minimize planning time in the multi-goal setting: it is shown to be a worst-case $(4\alpha, 2)$-approximation of the optimal option set, where α is the approximation ratio of the k-medians with penalties subroutine. We then present a variation, *Fast Average Options*, with improved run-time and describe a general means of producing similar algorithms based on the selection of a k-medians subroutine. We empirically evaluate our method on four discrete and two continuous control planning domains where it outperforms other leading option discovery algorithms.

1 Introduction

Temporally extended actions, or *options*, can be used to solve long-horizon decision problems (Sutton, Precup, and Singh 1999; Barto and Mahadevan 2003). When designed by a domain expert, these options lead to deep exploration (Bellemare et al. 2020) and extend the planning horizon of the agent (Sutton, Precup, and Singh 1999; Konidaris, Kaelbling, and Lozano-Pérez 2018). An important question in both reinforcement learning and planning is that of option *discovery*: how can agents use the data gathered in an environment to construct useful temporal abstractions?

The majority of work in this area proposes intuitive heuristics for discovery; examples include identifying relatively novel states (Şimşek and Barto 2004), identifying bottleneck states (Şimşek and Barto 2008), finding repeated policy fragments (Pickett and Barto 2002), identifying well connected regions (Ramesh, Tomar, and Ravindran 2019), finding states that often occur on successful trajectories (McGovern and Barto 2001), and lowering the computation time of planning (Young and Sutton 2023; Wan and Sutton 2022). While these are sometimes effective, heuristic approaches provide no guarantee that similar performance should be expected for a new task (Solway et al. 2014; Brunskill and Li 2014). As a result, a few recent approaches have taken a more formal approach where performance bounds can be derived for an explicit agent-level performance criterion (Jinnai et al. 2019a,b). In particular, Jinnai et al. (2019a) discovers options that bound worst-case planning time, which is the maximum number of iterations required to solve a single task. However, they do not address the case where the agent must solve a family of tasks; so, we aim to reduce *expected planning time*, where the expectation is over the agent's task distribution. This quantity is better suited for the multi-task (Kaelbling 1993; Solway et al. 2014) and lifelong learning (Brunskill and Li 2014) settings.

Consider an agent that must solve a family of goal-reaching tasks, where each task is modeled as an MDP and differs in its start-goal configuration. In such a task family, it would be useful for the agent to have options that reduce planning time on *average* over all possible tasks. As a result, we provide an algorithm that minimizes the average distance between all possible start-goal pairs.

Our core insight is that option discovery in the multi-goal context is equivalent to the classical k-medians with penalties problem (k-MP) (An and Svensson 2017; Dohan, Karp, and Matejek 2015). Using k-MPs and a result from graph theory (Meyerson and Tagiku 2009), we find options that reduce the overall expected cost of transitioning between any two states. This connection between option discovery and k-MPs allows us to design efficient approximation algorithms with strong theoretical guarantees.

We present *Average Options*, a formal option discovery algorithm for planning in multi-goal environments; we prove that this algorithm bounds planning time. We then present a more scalable, approximate version of our algorithm, *Fast Average Options*. We test both algorithms on six control problems and show that they outperform other option discovery algorithms (Jinnai et al. 2019b; Machado et al. 2018) with a strong theoretical basis.

2 Background and Related Work

The interaction between an agent and its environment can be modeled as a Markov Decision Process (MDP) $M = (S, A, P, R, \gamma)$, where S is the set of states, A is the set of actions, P is the transition function, R is the reward function, and γ is the discount factor (Sutton and Barto 2018). When transitions are deterministic, the transition function $P : S \times A \times S \to [0, 1]$ is replaced by $T : S \times A \to S$.

*Work done while at Brown University.
†These authors advised equally.

We say that an MDP has "invertible actions" if for each pair $s \in S, a \in A$, there exists $a' \in A$ such that $T(s', a') = s$.

An MDP can be represented as a graph with the states S as vertices and with the states s and s' being connected by an edge if there exists $a \in A$ such that $T(s, a) = s'$. If an MDP has invertible actions then this graph is undirected.

In planning, the Value Iteration (VI) algorithm can be used to find the optimal policy π^*. VI proceeds with the following initialization and update rule for $V_k : S \to \mathbb{R}$:

$$V_0(s) = 0,$$

$$V_{k+1}(s) = \max_{a \in A} \left(\sum_{s' \in S} P(s, a, s')(R(s, a, s') + \gamma V_k(s')) \right). \tag{1}$$

The update step is repeated until $V_{k+1}(s) - V_k(s) < \epsilon$ for all $s \in S$. To compare different option discovery methods, we follow Silver and Ciosek (2012) and measure the number of iterations required for convergence of VI.

2.1 Options

Options are temporally extended actions (Sutton, Precup, and Singh 1999). An option o is defined as a triple $(\mathcal{I}_o, \pi_o, \beta_o)$. $\mathcal{I}_o \subset S$ is the set of states from which o can be executed, $\pi_o : S \times A \to [0, 1]$ is the option policy, and $\beta_o : S \to [0, 1]$ is the probability that the option terminates in a given state. If $\beta_o(s) \in \{0, 1\}\ \forall s \in S$ it is more convenient for β_o to represent the set of states where the termination probability is 1, $\{s | \beta_o(s) = 1\}$.

Following Jinnai et al. (2019a), we focus on a special case of options called *point options*. Point options initiate and terminate in single states, i.e, $|\mathcal{I}_o| = 1, |\beta_o| = 1$. An option is *bi-directional* if it has a policy to go from $\mathcal{I}_o$ to β_o and a policy to go from β_o to $\mathcal{I}_o$. For brevity, we will refer to bi-directional point options as options. This is a restrictive model of an option, but it is commonplace in formal option discovery as it allows us to apply results from graph theory and precisely bound agent performance. Despite the simplicity of point options, similar algorithms (e.g, Covering Options (Jinnai et al. 2019c)) have served as the foundation for deep RL algorithms for more complex environments (Jinnai et al. 2019c; Klissarov and Machado 2023).

2.2 k-Medians with Penalties Subroutine

The k-medians with penalties (k-MP) (An and Svensson 2017; Dohan, Karp, and Matejek 2015) is an algorithm from Operations Research that is best explained using the analogy of opening a set of *facilities* that *serve* a set of *cities*. For example, consider the situation where a company is trying to decide where to place its warehouses (facilities) so that it can distribute packages (serve) to major American cities with the lowest cost possible. When the company decides to serve a city $c \in C$ using a facility $f \in \mathcal{F}$, it incurs a cost $d(c, f)$; when the company decides to not serve a city c (perhaps because the city is very difficult to access by road), then it incurs a positive real valued penalty p_c. The function $d : C \times F \to \mathbb{R}$ is an arbitrary distance metric, i.e., it is symmetric and satisfies the triangle inequality.

While the k-MP problem succinctly models many real world problems, it is known to be NP-Hard (Kariv and Hakimi 1979). As a result, we must resort to approximation algorithms. An approximation algorithm is an α-approximation if it finds a solution F with cost no higher than α times the cost of the optimal solution, F^*. Many approximation algorithms exist, each with its own set of tradeoffs: while some provide tighter approximations, others have lower run time (Jain et al. 2002; Xu and Xu 2005; Arya et al. 2001). For our theoretical analysis, we will use an approximation algorithm with $\alpha < 2$ (Jain et al. 2002) and for our experiments use a 5-approximation method with improved run-time (Arya et al. 2001).

2.3 Related Option Discovery Algorithms

Option discovery is a challenging open problem in reinforcement learning (Pateria et al. 2021). Recent work has made substantial progress on skill-discovery in large single (Konidaris and Barto 2009; Bacon, Harb, and Precup 2017; Tiwari and Thomas 2019; Nachum et al. 2018; Bagaria and Konidaris 2020; Bagaria et al. 2021) and multi-goal (Levy et al. 2019; Sharma et al. 2020; Bagaria, Senthil, and Konidaris 2021) domains. By contrast, we seek options with strong theoretical bounds on planning time.

Graph-Based Option Discovery. Many other approaches also derive options from graph-theoretic properties of MDPs. Based on the "bottleneck" intuition (Mcgovern 2002; Stolle and Precup 2002), Relative Novelty uses state counting (Şimşek and Barto 2004) and Betweenness Options uses a graph-centrality measure to discover option terminations (Şimşek and Barto 2008). Q-Cut (Menache, Mannor, and Shimkin 2002) and L-Cut (Şimşek, Wolfe, and Barto 2005) use classical algorithms like Max-Flow/Min-Cut to identify subgoals. Evans and Özgür Şimşek (2024) use graph modularity to learn multi-level hierarchies. These methods borrow tools from graph theory but the property of interest is chosen based on intuition.

Spectral Methods. Several methods have used properties of the graph Laplacian to learn options (Mahadevan and Maggioni 2007; Chiu and Soo 2010; Machado et al. 2018; Wu, Tucker, and Nachum 2018; Wang et al. 2021; Jinnai et al. 2019c; Bar, Talmon, and Meir 2020). Among these, Eigen Options has demonstrated the most empirical success; they use the eigenvectors of the Laplacian to create options that traverse the principle directions of the state-space (Machado et al. 2018). Due to their strong theoretical foundation and empirical success, we compare our algorithms against Eigen Options as a representative spectral method.

State Aggregation. Option discovery algorithms often cluster the state-space and connect the resulting clusters using options (Agostinelli et al. 2019; Ramesh, Tomar, and Ravindran 2019; Srinivas et al. 2016; Li, Walsh, and Littman 2006; Campos et al. 2020). Unlike our algorithm, these methods cluster states in a continuous vector space $\mathbb{R}^d$, whereas we use k-medians on the *discrete* graph representation of the MDP.

Formal Option Discovery. These methods precisely formulate option discovery (Solway et al. 2014) as an optimization for certain characteristics (like planning time Jinnai et al. 2019a) and then formally bound the agent's performance (Jinnai et al. 2019a,b); our method falls in this category. Brunskill and Li (2014) also find that option discovery is NP-Hard, but their approximation algorithm for discovery does not have theoretical guarantees. "Small world options" (Chaganty, Gaur, and Ravindran 2012) bound the number of steps needed to reach the maximum value state, but their guarantees only apply to lattices, which are a restrictive type of graphs. The A-MIMO algorithm provably bounds planning time, but is applicable only to single-goal planning problems (Jinnai et al. 2019a). By contrast, our algorithm minimizes expected planning time averaged over all goals, and thus can be used in a multi-goal context. Covering Options bounds the expected number of steps needed to reach a rewarding state with a random walk (Jinnai et al. 2019b). Assuming the agent acts randomly gives a bound on the worst-case time to travel between two states. This assumption is overly pessimistic in practice and minimizing cover time applies to option discovery in single task MDPs. Due to the similarity in approach and problem formulation, we empirically compare against Covering Options.

3 Average Options

Consider a graph representation of an MDP where the nodes are states and edges are actions. We can think of (point-) option discovery as the process of identifying which two nodes in the graph should be connected via a single edge. Where might we add an edge to the original graph so that the resulting graph permits faster planning? While previous methods have bounded planning time for single-goal MDPs, we propose a new method that provably minimizes planning time averaged over a distribution of start and goal states.

Fortunately, the problem of finding which new edges would minimize average planning time is closely related to the well-studied k-medians with penalty (k-MP) problem. This problem is NP-Hard, but we can use a good polynomial-time approximation algorithm find a set of k bidirectional point options that minimize the average number of VI iterations needed to converge on the optimal value function, V^*. We first consider the case of deterministic MDPs with invertible actions and then generalize to stochastic communicating MDPs. We will show that our algorithm is able to achieve a $(4\alpha, 2)$- approximation of the optimal set of options in polynomial time $\mathcal{O}(n^2 \log(n) + n^3)$, where $n = |S|$. Then we will show that our algorithm bounds planning time, specifically the number of iterations of VI.

3.1 Deterministic MDPs with Invertible Actions

Let O be a set of options. For a graph G, we denote adding options to the graph as $G+O = G'$. G' has the same vertices as G and for each $o \in O$ an edge is added from the initiation state of o to its termination state. We take the added edges in G' to have length 1 and the distance on G' is defined as the shortest-path distance.

For a graph G with added options O now define $d(G')$ and the average distance, AvgDist(G'), as

$$d(G') = \sum_{s \in S} \sum_{s' \in S} d_{G'}(s, s'), \tag{2}$$

and

$$\text{AvgDist}(G') = d(G')/|S|^2. \tag{3}$$

For a given MDP M and corresponding graph G, we want to find the set of k point options O such that AvgDist(G') is minimized. We prove in Section 3.3 that this is a bound for the number of iterations of VI, so minimizing AvgDist(G') is equivalent to minimizing average planning time.

The length of the shortest path from s to s' in G is written as $d_{G'}(s, s')$ or $d(s, s')$ when the graph is implied. Note that minimizing $d(G)$ is equivalent to minimizing AvgDist(G). To find such a set of k point options we first find a set of states F, $|F| = k+1$, which minimizes cost for the $(k+1)$-medians with penalties problem on an augmented version of the graph G. We augment G so that the cost of a candidate k-MP solution is the average distance in the graph when the k states in F are connected with options. Searching for a cost minimizing k-median with penalties solution thus reduces the bound on average distance. We construct the augmented graph G analogously to Meyerson and Tagiku (2009) and summarize the construction here. Consider a set C constructed by creating a duplicate u_{uv} of each state u in the graph for each other state v and assigning a penalty corresponding to the distance of the state pair $d(u, v)$. Conceptually, the cost of each state u_{uv} in C now corresponds to either the distance to the nearest states in F or its penalty, $d(u, v)$, which is connecting u to v directly without options. Evaluating this cost over all cities u_{uv} and u_{vu} we get that the cost corresponds to the total distance between all pairs of states through primitive actions or options. This gives us the following algorithm as in theorem 4 of Meyerson *et al.*:

1. Given a graph G and distance function d let $\mathcal{F} = S$, the vertices of G. Construct the set of states C by duplicating each node u $(2|S|-2)$ times to form u_{uv} and u_{vu} for each $v \in S$. Define the distance $d(i, u_{vu})$ to be $d(i, u)$ for all $i \in S$ and $u_{vu} \in C$ and assign each u_{vu} a penalty of $d(u, v) - 2$.
2. Solve the $(k+1)$-MP problem given $\mathcal{F}$, C, the cost function d, and penalties as defined in the previous step. Let F be the set of $k+1$ states chosen by the algorithm.
3. Select a state s in F and construct k options connecting the states in $F \setminus \{s\}$ to s.

Theorem 4 of Meyerson *et al.*. *There exists a polynomial-time $(4\alpha, 2)$-approximation algorithm for ASPDM. In particular, this algorithm gives at most $2k-1$ edges yielding cost at most 4α-times the optimum k-edge cost.*

In the resulting construction, the options are connected as a star and so any of the k states in F can be reached from another in at most 2 steps. Connecting states u and v with an option corresponds to paying the cost for u to be connected to a state in F, a fixed cost of 2, and the cost for v to be connected to a state in F. This corresponds to traveling from u to the nearest option, taking at most 2 options, and then traveling the remaining distance to v.

3.2 Approximation Ratio and Run-Time Analysis

Since we consider the special case of point options connecting pairs of states, we can equivalently think of them as "shortcut edges" that we add to the underlying graph. These edges will have length $\delta = 1$ because it takes 1 step to look ahead or back up values through such edges during planning. The weights w_{uv} for each pair of states $u, v \in S$ are also set to 1 because we care about the average distance between all states equally.[1] As given by theorem 4 of Meyerson and Tagiku (2009) we have that the construction given above is a polynomial time $(4\alpha, 2)$-approximation algorithm.

Computing the distances between all pairs of states in the graph G with n vertices and m edges can be done in $\mathcal{O}(n^2 \log(n))$ time (Kumar, R, and Iyer 2009). Solving the k-median problem exactly is known to be NP-Hard so we are forced to use an approximation instead (Kariv and Hakimi 1979). The k-median with penalties subroutine with $\alpha < 2$ can be done in $\mathcal{O}(n^3)$ time (Jain et al. 2002) although the α and run-time vary with choice of k-median algorithm. The total run-time of our algorithm is $\mathcal{O}(n^2 \log(n) + n^3)$.

3.3 Bound on Planning Time

We now demonstrate that $d(G)$ bounds the number of iterations necessary for convergence of VI, by considering the specific case of a deterministic MDP with a single goal state. This bound also extends to the multi-goal setting because the bound for convergence over all tasks is the maximum number of VI iterations for any single task. This bound does hold for both the discrete domains and discretized continuous domains used to evaluate option discovery methods in the empirical results section.

For a deterministic MDP we can rewrite the VI update as

$$V_{t+1}(s) = \max_{a \in A} \left(R(s') + \gamma V_t(s') \right) \qquad \text{with } s' = T(s, a). \tag{4}$$

We assume that the reward function R is 1 for a single state $g \in S$ and 0 everywhere else and that the state g is absorbing. With these assumptions we can consider running VI until convergence in a finite number of iterations and take $V(s)$ for some $s \in S$. By expanding the VI update equation above we have that $V(s)$ obtains its value from the state g along the shortest path from g to s. The value of $V(s)$ is consequently $\gamma^{d(g,s)}$. This also tells us that the $V(s)$ obtains this value after $d(g, s)$ iterations and by assumption never changes again. This means that the number of iterations of VI necessary to plan to reach a certain goal $g \in S$ can be written as $\max_{s \in S} d(s, g)$ and the average over all goal states is $\sum_{g \in S} \max_{s \in S} d(s, g)$. Notice that the average steps to convergence of VI over goals is upper bounded by the average graph distance:

$$\sum_{g \in S} \max_{s \in S} d(s, g) < d(G) = \sum_{s \in S} \sum_{s' \in S} d(s, s'). \tag{5}$$

[1] We could also deal with non-uniform task distributions with minimal changes to our approach.

Therefore, by minimizing the average distance for a given graph we minimize the upper bound for number iterations of VI before convergence.

3.4 Stochastic MDPs

We now generalize our algorithm to the case of stochastic *communicating* MDPs (actions are not necessarily invertible). An MDP is said to be communicating, if for any pair of states $s, s' \in S$ there exists a sequence of actions $a_0, a_1, \ldots a_n$ with non-zero probability of reaching s' from s, $P(s_n = s' | s_0 = s, a_0, a_1, \ldots a_n) \neq 0$. If the MDP is communicating then the directed graph representing the MDP is strongly connected. We redefine the distance $d(s, s')$ for the stochastic communicating MDP M to be the expected number of steps to reach s' using the optimal policy $\pi^*_{s'}$ starting from state s. Note that $d(s, s')$ is not necessarily equal to $d(s', s)$. We now define $D(s, s') = d(s, s') + d(s', s)$ and show that it is a distance metric satisfying symmetry and the triangle inequality.

$$\begin{aligned} D(s, s') &= d(s, s') + d(s', s) = \\ &\quad d(s', s) + d(s, s') = D(s', s) \\ D(s, s'') &= d(s, s'') + d(s'', s) < \\ &d(s, s') + d(s', s'') + d(s'', s') + d(s', s) \\ &\quad = D(s, s') + D(s', s''), \text{ for states } s, s', s'' \in S. \end{aligned}$$

As in the discrete case, we define the average distance D for a graph as $D(G) = \sum_{s,s' \in S} D(s, s') = 2 * \sum_{s,s' \in S} d(s, s')$. Importantly, minimizing $D(G)$ is equivalent to minimizing $\sum_{s,s' \in S} d(s, s')$. Because D is symmetric and satisfies the triangle inequality, we know that the k-MP algorithm is still admissible and so the proposed algorithm can be extended to stochastic communicating MDPs using this new distance function. Furthermore, the reasoning for the approximation ratio remains the same: we can connect the endpoints of the optimal set of options O^* as a star to get $\hat{O}$ with $D(G + \hat{O}) < 2D(G + O^*)$ using at most twice as many options. The k-MP subroutine pays at most $2D(u, v)$ for each pair $u, v \in S$, so it has cost less than $2\alpha D(G + \hat{O})$. For a solution O to the k-MP subroutine we have that:

$$D(G + O) \leq 2\alpha D(G + \hat{O}) < 4\alpha D(G + O^*). \tag{6}$$

Therefore, our algorithm still finds a $(4\alpha, 2)$-approximation for the optimal set of options O^*.

3.5 Fast Average Options

Average options minimizes the average distance between states with bounded suboptimality, but it is resource intensive and impractical in large domains. The time and space complexity of Average Options can be significantly reduced by replacing the k-MP subroutine with a related problem, k-medians[2]. We call this amended algorithm Fast Average

[2] For a graph G, k-medians aims to find a set of states F, $|F| = k$, such that the sum of the distances from any state to the nearest state in F is minimized. For $k = 1$ this is equivalent to finding the centroid of G (Pozanco et al. 2019; Karpas 2022; Pozanco, Torralba, and Borrajo 2024).

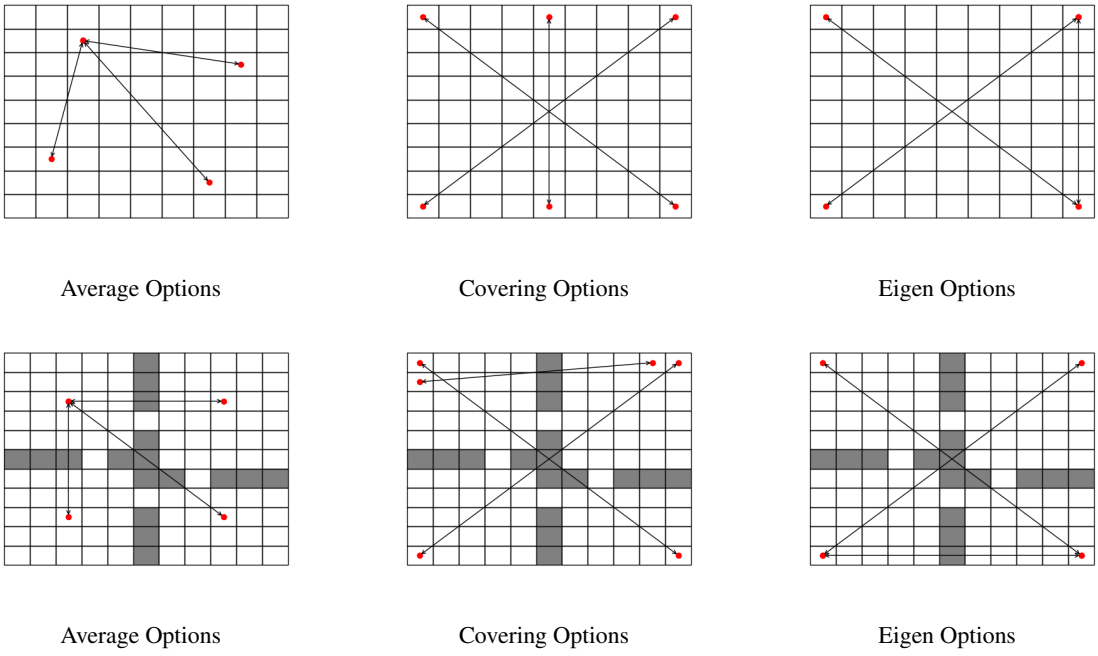


Figure 1: Visualization of options generated in discrete domains. Shaded grid squares denote walls, lines are the generated options, and the red points are option initiation and termination states. On small domains like these, Fast Average Options generates very similar options to Average Options and is omitted.

Options. Fast Average Options no longer has the same bound on suboptimality but performs similarly empirically and provides a significant improvement on run time.

3.6 Average Options in Continuous Domains

To use our method in continuous domains, we first discretize the state-space using a set covering (Lobel et al. 2022) generated by iterating over all the states and adding states to the set cover when they are a distance greater than ρ to all other states in the set cover. Then we build a graph on the set covering by joining states within distance ϵ of each other or if they are k-nearest neighbors (Tenenbaum, de Silva, and Langford 2000). Finally, we run our algorithm on the resulting discretized graph of the domain. The radius of the ϵ-balls used for the set cover is the *discretization factor*, ρ.

In continuous domains, we can no longer use point options because the probability of the agent being in a single state approaches zero. We therefore extend the initiation set of a point option to be a small ϵ-ball around the initiation state.

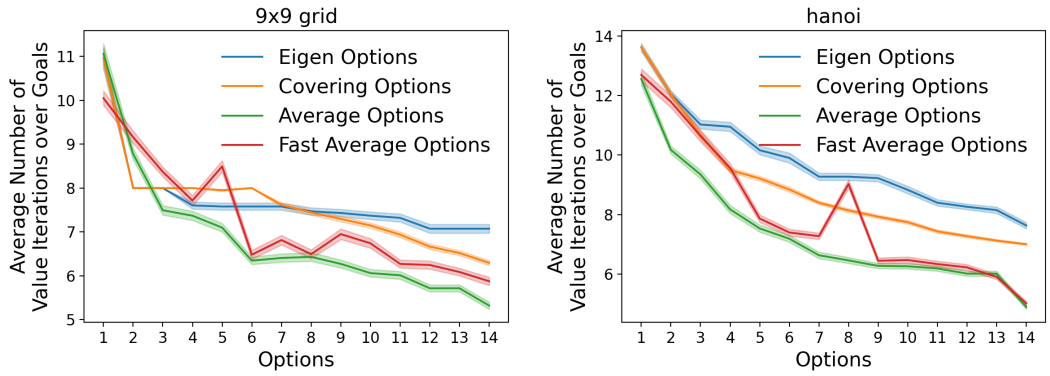
4 Empirical Results

Discrete Domains. First, we qualitatively evaluate our algorithms on two discrete grid-world domains: 9x9 grid and Four-Rooms (Sutton, Precup, and Singh 1999). Figure 1 shows a qualitative comparison between Average Options, Covering Options (Jinnai et al. 2019b) and Eigen Options (Machado et al. 2018). Covering Options and Eigen Options connected the corners of the 9×9 grid (top row) and the Four Rooms (bottom row). On the other hand, Average Options connects more central states in 9×9 grid and the centers of the four rooms—by connecting the centers of the different rooms, the agent learns options that minimize graph distance on average over all goal states.

We quantitatively compare all methods on four discrete domains: 9x9 grid, Two-Rooms, Four-Rooms and Towers of Hanoi; these domains are taken without modification from Jinnai et al. (2019c). For Average Options we use a k-median with penalties subroutine with an approximation ratio of $\alpha = 5$ for improved run-time (Arya et al. 2001).

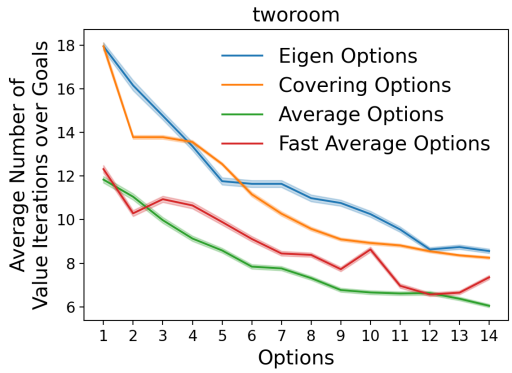
For each method, we generate a set of n options O and then run VI for every goal $g \in S$ in the graph $G + O$. For each run of VI, we make g an absorbing state with reward 1 and give all other states a reward of 0. We run VI until convergence and record the number of iterations taken. We then average the number of iterations over all goals. Average options consistently performs better across the tested domains (Figure 2). Fast Average Options has variable performance but tends to outperform Covering Options and Eigen Options as the environment gets larger.

Continuous Domains. We compare Fast Average Options to Covering Options and Eigen Options on Ant U-maze (Fu et al. 2020) and Fetch-Reach (Plappert et al. 2018). In Ant U-maze, we have to control a quadrupedal robot in a U-shaped maze (Fu et al. 2020) and in Fetch-Reach, we have to manipulate a robotic arm to reach specific points in 3D space (Plappert et al. 2018). Domain parameters are in Appendix A.4.

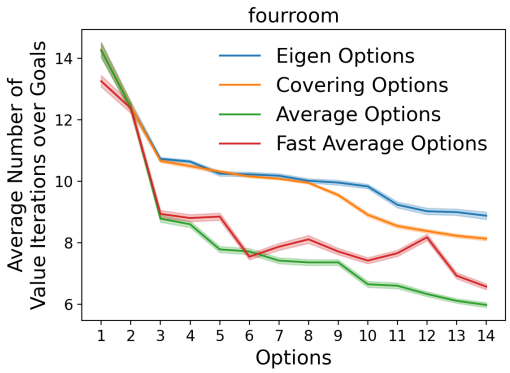


(a) 9x9 grid

(b) Towers of Hanoi



(c) Two Room



(d) four-room

Figure 2: Comparison of the average number of steps for convergence of VI on 9x9 gird, fourroom, Hanoi, and tworoom environments; error bands denote standard error over all possible goals; lower is better.

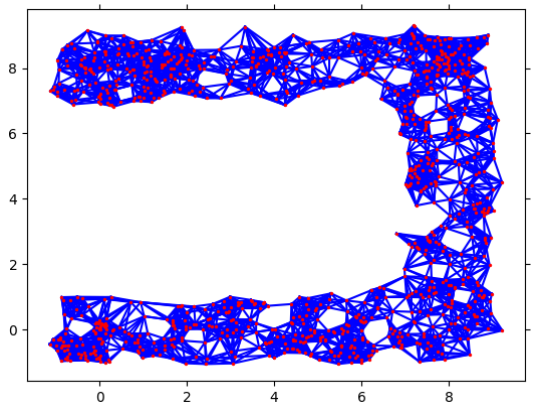


Figure 3: Discretized graph on Ant U-Maze with $\rho = 6.5$, $\epsilon = 0.5$, $k = 10$. The state-space has 29 dimensions, but only the (x, y) location of the Ant is visualized here.

To generate options for continuous domains, we construct a discretized graph as described in Section 3.6. For each method, $4, 8,$ and 16 options are constructed using the discretized graph and planning time is averaged over a collection of start-goal pairs. Details on start and goal state selection are in Appendix A.4.

Figure 4 shows the options discovered by Average Options in Ant U-Maze; qualitatively speaking, these options effectively navigate the ant to different regions of the maze. To quantitatively evaluating all methods in continuous domains, we use "Random Shooting" (Tedrake 2022) for planning, a simple trajectory optimization algorithm, where the planner samples 400 trajectories composed of primitive actions and learned options, then executes the trajectory that minimizes the Euclidean distance to the goal. Trajectory optimization is repeated until the goal state is reached or $10,000$ actions are sampled. The number of sampled actions is averaged over start-goal pairs to compute average planning time. Detailed description of the planning algorithm and experimental setup are in Appendix A.4.

In Figures 5 and 6, we see that Fast Average Options outperforms both Covering Options and Eigen Options. We also note that Average Options provides a greater improvement

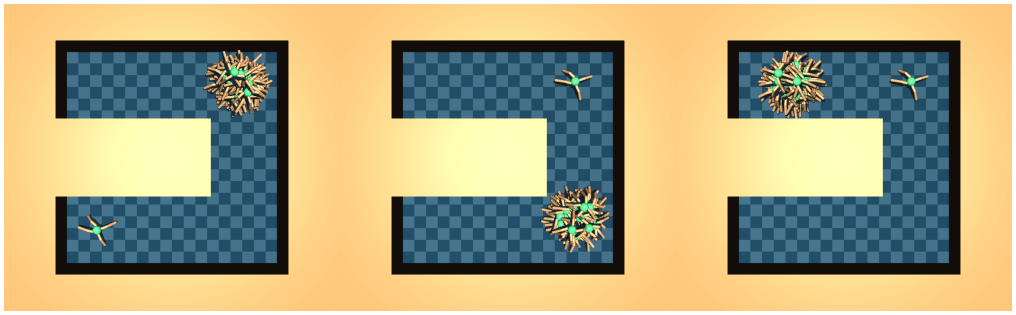
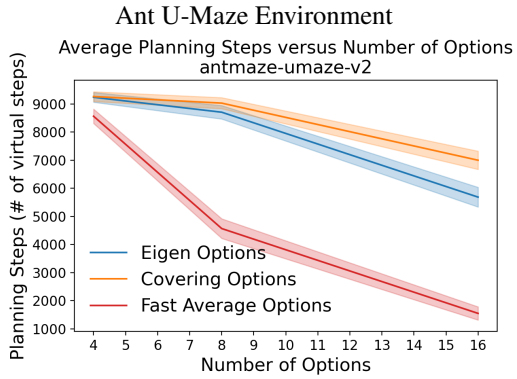
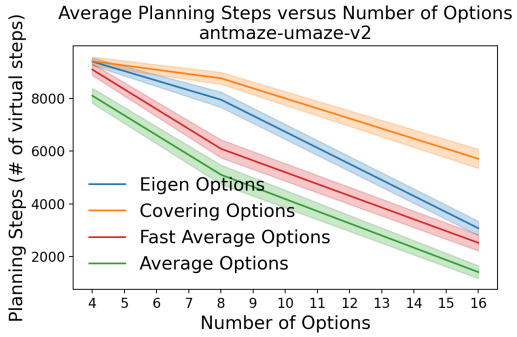


Figure 4: Options discovered in Ant U-Maze. Initiation sets are shown as sampled ant poses; the termination state is shown as one ant.

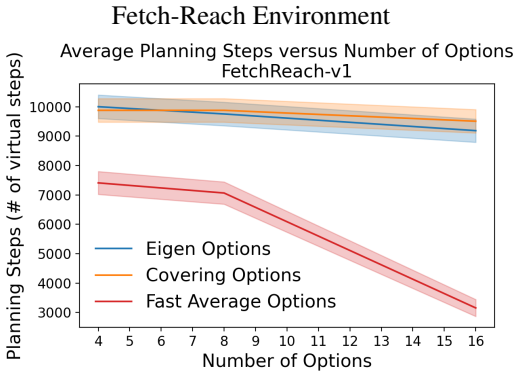


(a) $\rho = 6.5$

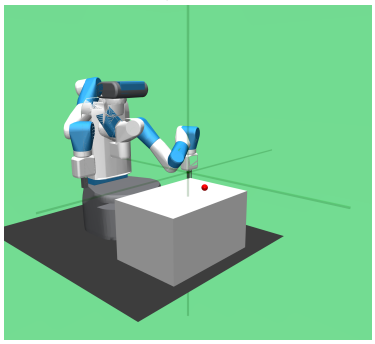


(b) $\rho = 8$

Figure 5: Mean planning time as a function of the number of discovered options in Ant U-Maze; error bands are standard error over 156 start-goal pairs; lower is better.



(a) $\rho = 0.1$



(b) Option generated in Fetch-Reach Environment

Figure 6: Mean planning time vs the number of discovered options in Fetch-Reach (top); error bands are standard error over 81 start-goal pairs; lower is better. Visualization of an option discovered by Fast Average Options in Fetch-Reach (bottom).

to planning time than Fast Average Options, although this comes at the cost of greater computation time.

5 Conclusion

We presented a formal objective for discovering options that are useful for planning in a multi-goal setting. We derived an exact algorithm, Average Options, that provably minimizes planning time and bounded the algorithm's sub-optimality. In addition to this exact algorithm, we also presented Fast Average Options, with improved run-time and comparable empirical results to Average Options. Empirically, we demonstrated that our algorithms consistently outperform competing methods in challenging control problems.

Acknowledgments

This work was supported in part by NSF grant #1955361 and ONR grant #N00014-22-12592.

References

Agostinelli, A.; Arulkumaran, K.; Sarrico, M.; Richemond, P.; and Bharath, A. A. 2019. Memory-Efficient Episodic Control Reinforcement Learning with Dynamic Online k-means.

An, H.-C.; and Svensson, O. 2017. *Recent Developments in Approximation Algorithms for Facility Location and Clustering Problems*, 1–19. Singapore: Springer Singapore. ISBN 978-981-10-6147-9.

Arya, V.; Garg, N.; Khandekar, R.; Munagala, K.; and Pandit, V. 2001. Local search heuristic for k-median and facility location problems. 21–29.

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

Bagaria, A.; and Konidaris, G. 2020. Option Discovery using Deep Skill Chaining. In *International Conference on Learning Representations*.

Bagaria, A.; Senthil, J.; Slivinski, M.; and Konidaris, G. 2021. Robustly Learning Composable Options in Deep Reinforcement Learning. In *30th International Joint Conference on Artificial Intelligence*.

Bagaria, A.; Senthil, J. K.; and Konidaris, G. 2021. Skill discovery for exploration and planning using deep skill graphs. In *International Conference on Machine Learning*, 521–531. PMLR.

Bar, A.; Talmon, R.; and Meir, R. 2020. Option discovery in the absence of rewards with manifold analysis. In *International Conference on Machine Learning*, 664–674. PMLR.

Barto, A. G.; and Mahadevan, S. 2003. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2): 41–77.

Bellemare, M. G.; Candido, S.; Castro, P. S.; Gong, J.; Machado, M. C.; Moitra, S.; Ponda, S. S.; and Wang, Z. 2020. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588: 77 – 82.

Brunskill, E.; and Li, L. 2014. PAC-inspired Option Discovery in Lifelong Reinforcement Learning. In Xing, E. P.; and Jebara, T., eds., *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, 316–324. Bejing, China: PMLR.

Campos, V.; Trott, A.; Xiong, C.; Socher, R.; Giró-i Nieto, X.; and Torres, J. 2020. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, 1317–1327. PMLR.

Chaganty, A. T.; Gaur, P.; and Ravindran, B. 2012. Learning in a small world. In *AAMAS*.

Chiu, C.-C.; and Soo, V.-W. 2010. Automatic complexity reduction in reinforcement learning. *Computational Intelligence*, 26(1): 1–25.

Şimşek, O.; and Barto, A. 2008. Skill Characterization Based on Betweenness. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc.

Dohan, D.; Karp, S.; and Matejek, B. 2015. K-median Algorithms: Theory in Practice.

Evans, J. B.; and Özgür Şimşek. 2024. Creating Multi-Level Skill Hierarchies in Reinforcement Learning. arXiv:2306.09980.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4RL: Datasets for Deep Data-Driven Reinforcement Learning.

Jain, K.; Mahdian, M.; Markakis, E.; Saberi, A.; and Vazirani, V. V. 2002. Greedy Facility Location Algorithms Analyzed using Dual Fitting with Factor-Revealing LP. arXiv:cs/0207028.

Jinnai, Y.; Abel, D.; Hershkowitz, D.; Littman, M.; and Konidaris, G. 2019a. Finding options that minimize planning time. In *International Conference on Machine Learning*, 3120–3129. PMLR.

Jinnai, Y.; Park, J. W.; Abel, D.; and Konidaris, G. 2019b. Discovering Options for Exploration by Minimizing Cover Time. In *Proceedings of the 36th International Conference on Machine learning*.

Jinnai, Y.; Park, J. W.; Machado, M. C.; and Konidaris, G. 2019c. Exploration in reinforcement learning with deep covering options. In *International Conference on Learning Representations*.

Kaelbling, L. P. 1993. Learning to achieve goals. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, 1094–1099. Citeseer.

Kariv, O.; and Hakimi, S. L. 1979. An Algorithmic Approach to Network Location Problems. II: The p-Medians. *Siam Journal on Applied Mathematics*, 37: 539–560.

Karpas, E. 2022. A Compilation Based Approach to Finding Centroids and Minimum Covering States in Planning. *Proceedings of the International Conference on Automated Planning and Scheduling*, 32(1): 174–178.

Klissarov, M.; and Machado, M. C. 2023. Deep laplacian-based options for temporally-extended exploration. *arXiv preprint arXiv:2301.11181*.

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

Konidaris, G. D.; Kaelbling, L. P.; and Lozano-Pérez, T. 2018. From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning. *J. Artif. Intell. Res.*, 61: 215–289.

Kumar, U.; R, R.; and Iyer, K. 2009. All-pairs shortest-paths problem for unweighted graphs in O(n2 log n) time. *World Academy of Science, Engineering and Technology*, 38.

Levy, A.; Konidaris, G.; Platt, R.; and Saenko, K. 2019. Hierarchical Reinforcement Learning with Hindsight. In *International Conference on Learning Representations*.

Li, L.; Walsh, T. J.; and Littman, M. L. 2006. Towards a Unified Theory of State Abstraction for MDPs. *ISAIM*, 4(5): 9.

Lobel, S.; Bagaria, A.; Allen, C.; Gottesman, O.; and Konidaris, G. 2022. Optimistic Initialization for Exploration in Continuous Control. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36.

Machado, M. C.; Rosenbaum, C.; Guo, X.; Liu, M.; Tesauro, G.; and Campbell, M. 2018. Eigenoption Discovery through the Deep Successor Representation. arXiv:1710.11089.

Mahadevan, S.; and Maggioni, M. 2007. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, 8(Oct): 2169–2231.

McGovern, A.; and Barto, A. G. 2001. Automatic Discovery of Subgoals in Reinforcement Learning Using Diverse Density. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, 361–368. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN 1558607781.

Mcgovern, E. A. 2002. *Autonomous discovery of temporal abstractions from interaction with an environment*. Ph.D. thesis, University of Massachusetts at Amherst.

Menache, I.; Mannor, S.; and Shimkin, N. 2002. Q-cut—dynamic discovery of sub-goals in reinforcement learning. In *European conference on machine learning*, 295–306. Springer.

Meyerson, A.; and Tagiku, B. 2009. Minimizing average shortest path distances via shortcut edge addition. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 272–285. Springer.

Nachum, O.; Gu, S. S.; Lee, H.; and Levine, S. 2018. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, 3303–3313.

Pateria, S.; Subagdja, B.; Tan, A.-h.; and Quek, C. 2021. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5): 1–35.

Pickett, M.; and Barto, A. 2002. PolicyBlocks: An Algorithm for Creating Useful Macro-Actions in Reinforcement Learning.

Plappert, M.; Andrychowicz, M.; Ray, A.; McGrew, B.; Baker, B.; Powell, G.; Schneider, J.; Tobin, J.; Chociej, M.; Welinder, P.; Kumar, V.; and Zaremba, W. 2018. Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research.

Pozanco, A.; E-Martín, Y.; Fernández, S.; and Borrajo, D. 2019. Finding Centroids and Minimum Covering States in Planning. *Proceedings of the International Conference on Automated Planning and Scheduling*, 29(1): 348–352.

Pozanco, A.; Torralba, ; and Borrajo, D. 2024. Computing Planning Centroids and Minimum Covering States Using Symbolic Bidirectional Search. *Proceedings of the International Conference on Automated Planning and Scheduling*, 34(1): 455–463.

Ramesh, R.; Tomar, M.; and Ravindran, B. 2019. Successor Options: An Option Discovery Framework for Reinforcement Learning.

Sharma, A.; Gu, S.; Levine, S.; Kumar, V.; and Hausman, K. 2020. Dynamics-Aware Unsupervised Discovery of Skills. In *International Conference on Learning Representations (ICLR)*.

Silver, D.; and Ciosek, K. 2012. Compositional Planning Using Optimal Option Models. In *ICML*.

Şimşek, Ö.; and Barto, A. 2004. Using Relative Novelty to Identify Useful Temporal Abstractions in Reinforcement Learning.

Şimşek, Ö.; Wolfe, A. P.; and Barto, A. G. 2005. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd international conference on Machine learning*, 816–823.

Solway, A.; Diuk, C.; Córdova, N.; Yee, D.; Barto, A. G.; Niv, Y.; and Botvinick, M. M. 2014. Optimal behavioral hierarchy. *PLOS Computational Biology*, 10(8): 1–10.

Srinivas, A.; Krishnamurthy, R.; Kumar, P.; and Ravindran, B. 2016. Option Discovery in Hierarchical Reinforcement Learning using Spatio-Temporal Clustering.

Stolle, M.; and Precup, D. 2002. Learning options in reinforcement learning. 212–223.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

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

Tedrake, R. 2022. *Underactuated Robotics*.

Tenenbaum, J. B.; de Silva, V.; and Langford, J. C. 2000. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500): 2319–2323.

Tiwari, S.; and Thomas, P. S. 2019. Natural option critic. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 5175–5182.

Wan, Y.; and Sutton, R. S. 2022. Toward Discovering Options that Achieve Faster Planning. *CoRR*, abs/2205.12515.

Wang, K.; Zhou, K.; Zhang, Q.; Shao, J.; Hooi, B.; and Feng, J. 2021. Towards Better Laplacian Representation in Reinforcement Learning with Generalized Graph Drawing. In *International Conference on Machine Learning*, 11003–11012. PMLR.

Wu, Y.; Tucker, G.; and Nachum, O. 2018. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*.

Xu, G.; and Xu, J. 2005. An LP rounding algorithm for approximating uncapacitated facility location problem with penalties. *Inf. Process. Lett.*, 94: 119–123.

Young, K.; and Sutton, R. S. 2023. Iterative Option Discovery for Planning, by Planning. arXiv:2310.01569.