Finding Options that Minimize Planning Time

Preprint · October 2018

CITATIONS
0

READS
11

4 authors, including:

David Abel
Brown University
11 PUBLICATIONS **49** CITATIONS
SEE PROFILE

George Konidaris
Brown University
94 PUBLICATIONS **1,784** CITATIONS
SEE PROFILE

Finding Options that Minimize Planning Time

Yuu Jinnai, David Abel, Michael Littman, George Konidaris
Brown University, Providence, RI, United States

Abstract

While adding temporally abstract actions, or options, to an agent's action repertoire can often accelerate learning and planning, existing approaches for determining which specific options to add are largely heuristic. We aim to formalize the problem of selecting the optimal set of options for planning, in two contexts: 1) finding the set of k options that minimize the number of value-iteration passes until convergence, and 2) computing the smallest set of options so that planning converges in less than a given maximum of ℓ value-iteration passes. We first show that both problems are NP-hard. We then provide a polynomial-time approximation algorithm for computing the optimal options for tasks with bounded return and goal states. We prove that the algorithm has bounded suboptimality for deterministic tasks. Finally, we empirically evaluate its performance against both the optimal options and a representative collection of heuristic approaches in simple grid-based domains including the classic four rooms problem.

1 Introduction

Markov Decision Processes (MDPs) (Puterman 1994) are an expressive yet simple model of sequential decision-making environments. However, MDPs are computationally expensive to solve. One approach to solving such problems is to add high-level, temporally extended actions—often formalized as options (Sutton, Precup, and Singh 1999)—to the action space. The right set of options allows planning to probe more deeply into the search space with a single computation. Thus, if options are chosen appropriately, planning algorithms can find good plans with less computation.

Indeed, previous work has offered substantial support that abstract actions can accelerate planning. However, little is known about how to find the right set of options. Prior work often seeks to codify an intuitive notion of what underlies an effective option, such as identifying relatively unusual states (Şimşek and Barto 2004), identifying bottleneck states or high-betweenness states (Şimşek, Wolfe, and Barto 2005; Şimşek and Barto 2009; Bacon 2013; Moradi et al. 2012), finding repeated policy fragments (Pickett and Barto 2002), or finding states that often occur on successful trajectories (McGovern and Barto 2001; Bakker and Schmidhuber 2004). While such intuitions often capture important aspects of the role of options in planning, the resulting algorithms are somewhat heuristic in that they are not based on optimizing any precise performance-related metric; consequently, their relative performance can only be evaluated empirically.

We aim to formalize what it means to find the set of options that is optimal for planning, and to use the resulting formalization to develop a practical algorithm with a principled theoretical foundation. Specifically, we consider two settings that describe the problem of finding the right set of options: 1) finding the set of k options that minimize the number of VI iterations until convergence, and 2) computing the smallest set of options so that planning converges in less than a given maximum of ℓ VI iterations. We show that both problems are NP-hard, even for a deterministic MDP, and therefore harder than directly solving the MDP, which takes polynomial time (Littman, Dean, and Kaelbling 1995). We then provide a polynomial-time approximation algorithm for a subclass of each problem, A-MIMO and A-MOMI, that computes approximately optimal options for MDPs with bounded return and goal states. We prove that both algorithms have bounded suboptimality for deterministic tasks. These algorithms are not practical for speeding up run-time performance, as they are computationally harder than solving the MDP itself. The purpose of the algorithm is to analyze and evaluate the utility of options generated by heuristic methods. Finally, we empirically evaluate the performance of two heuristic approaches for option discovery, betweenness options (Şimşek and Barto 2009) and eigenoptions (Machado, Bellemare, and Bowling 2018), against A-MIMO, A-MOMI and the optimal options in standard grid domains.

2 Background

We first provide background on Markov Decision Processes (MDPs), planning, and options.

2.1 Markov Decision Processes

An MDP is a five tuple: $\langle \mathcal{S}, \mathcal{A}, R, T, \gamma \rangle$, where $\mathcal{S}$ is a finite set of states; $\mathcal{A}$ is a finite set of actions; $R : \mathcal{S} \times \mathcal{A} \rightarrow [0, \text{RMAX}]$ is a reward function, $T : \mathcal{S} \times \mathcal{A} \rightarrow \Pr(\mathcal{S})$ is a transition function, denoting the probability of arriving in state $s' \in \mathcal{S}$ after executing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, and $\gamma \in [0, 1]$ is a discount factor, expressing the agent's preference for immediate over delayed rewards.

An action-selection strategy is modeled by a *policy*, $\pi : \mathcal{S} \rightarrow \Pr(\mathcal{A})$, mapping states to a distribution over actions.

Typically, the goal of planning in an MDP is to *solve* the MDP—that is, to compute an optimal policy. A policy π is evaluated according to the Bellman Equation, denoting the long term expected reward received by executing π:

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

We denote $\pi^*(s) = \arg\max_{\pi} V^{\pi}(s)$ and $V^*(s) = \max_{\pi} V^{\pi}(s)$ as the optimal policy and value function, respectively.

2.2 Planning

The core problem we study is *planning*, namely, computing near optimal policy for a given MDP. The main variant of the planning problem we study we denote the *value-planning problem*:

Definition 1 (Value-Planning Problem): **Given** *an MDP* $M = \langle \mathcal{S}, \mathcal{A}, R, T, \gamma \rangle$ *and a non-negative real-value* ϵ, **return** *a value function,* V *such that* $|V(s) - V^*(s)| < \epsilon$ *for all* $s \in \mathcal{S}$.

2.3 Options and Value Iteration

Temporally extended actions offer great potential for mitigating the difficulty of solving complex MDPs, either through planning or reinforcement learning (Sutton, Precup, and Singh 1999). Indeed, it is possible that options that are useful for learning are not necessarily useful for planning, and vice versa. Identifying techniques that produce good options in these scenarios is an important open problem in the literature.

We use the standard definition of options (Sutton, Precup, and Singh 1999):

Definition 2 (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,*
- $\pi : \mathcal{S} \to \Pr(\mathcal{A})$ *is a policy,*
- $\beta : \mathcal{S} \to [0, 1]$*, is a termination condition.*

We let $\mathcal{O}_{all}$ *denote the set containing all options.*

In planning, options have well defined transition and reward models for each state named the multi-time model, introduced by Precup and Sutton (1998):

$$T_{\gamma}(s, o, s') = \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s', \beta(s_t) \mid s, o). \quad (2)$$

$$R_{\gamma}(s, o) = \mathop{\mathbb{E}}_{o_{\pi}} \left[r_1 + \gamma r_2 + \ldots + \gamma^{k-1} r_k \;\middle|\; s, o \right]. \quad (3)$$

We use the multi-time model for value iteration. The algorithm computes a sequence of functions $V_0, V_1, ..., V_b$ using the Bellman optimality operator on the multi-time model:

$$V_{i+1}(s) = \max_{o \in A \cup \mathcal{O}} \left(R_{\gamma}(s, o) + \sum_{s' \in S} T_{\gamma}(s, o, s') V_i(s') \right). \quad (4)$$

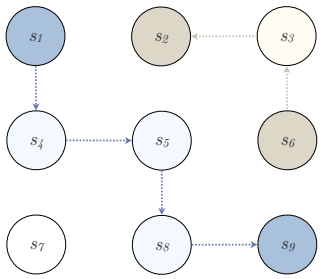


Figure 1: A single option can encode multiple unrelated behaviors. The dark circles indicate where the option can be initiated (s_1 & s_6) and terminated (s_2 & s_9), whereas the lighter circles denote the states visited by the option policy when applied in the respective initiating state.

The problem we consider is to find a set of options to add to the set of primitive actions that seek to minimize the number of iterations required for VI to converge:[1]

Definition 3 ($L(\mathcal{O})$): *The number of iterations* $L(\mathcal{O})$ *of a value-iteration algorithm using option set* $\mathcal{A} \cup \mathcal{O}$*, with* $\mathcal{O}$ *a non-empty set of options, is the smallest* b *at which* $|V_b(s) - V^*(s)| < \epsilon$ *for all* $s \in \mathcal{S}$.

Point options. The options formalism is immensely general. Due to its generality, a single option can actually encode several completely unrelated sets of different behaviors. Consider the nine-state example MDP pictured in Figure 1; a single option can in fact initiate, make decisions in, and terminate along entirely independent trajectories. As we consider more complex MDPs (which, as discussed earlier, is often a motivation for introducing options), the number of independent behaviors that can be encoded by a single option increases further still.

As a result, it can be difficult to reason about the impact of adding a single option, in the traditional sense. As the MDP grows larger, a combinatorial number of different behaviors can emerge from "one" option. Consequently, it is difficult to address the question: which *single* option helps planning? As MDPs grow large, one option can in fact encode a huge number of possible, independent behaviors. Thus, we instead introduce and study "point options", which only allow for a single continuous stream of behavior:

Definition 4 (Point option): *A* **point option** *is any option whose initiation set and termination set are each true for exactly one state each:*

$$|\{s \in \mathcal{S} : \mathcal{I}(s) = 1\}| = 1, \quad (5)$$

$$|\{s \in \mathcal{S} : \beta(s) > 0\}| = 1, \quad (6)$$

$$|\{s \in \mathcal{S} : \beta(s) = 1\}| = 1. \quad (7)$$

[1] We can ensure $|V^*(s) - V_i(s)| < \epsilon$ by running VI until $|V_{i+1}(s) - V_i(s)| < \epsilon(1 - \gamma)/2\gamma$ for all $s \in \mathcal{S}$ (Williams and Baird 1993).

We let $\mathcal{O}_p$ denote the set containing all point options.

For simplicity, we denote the initiation state as $\mathcal{I}_o$ and the termination state as β_o for a point option o.

To plan with a point option from state s, the agent runs value iteration using a model $Q(s,o) = R(s,o) + \gamma(o)V(s')$ in addition to the backup operations by primitive actions. We assume that the model of each option is given to the agent and ignore the computation cost for computing the model for the options.

Point options are a useful subclass to consider for several reasons. First, a point option is a simple model for a temporally extended action. Second, the policy of the point option can be calculated as a path-planning problem for deterministic MDPs. Third, any other options with a single termination state with termination probability 1 can be represented as a collection of point options. Forth, a point option has a fixed amount of computational overhead per iteration.

3 Complexity Results

Our main results focus on two computational problems:

1. MINITERMAXOPTION (MIMO): Which set of k or fewer options minimizes the number of iterations to convergence?
2. MINOPTIONMAXITER (MOMI): Which set of options let value iteration converge in at most ℓ iterations?

More formally, MIMO is defined as follows.

Definition 5 (MIMO): *The* MINITERMAXOPTION *problem:*
Given *an MDP M, a non-negative real-value ϵ, and an integer k,* **return** *$\mathcal{O}$ that minimizes $L(\mathcal{O})$, subject to $\mathcal{O} \subseteq \mathcal{O}_p$ and $|\mathcal{O}| \leq k$.*

We then consider the complementary optimization problem: compute the smallest set of point options such that planning time is guaranteed to fall below some threshold. Motivated by this scenario, the second problem we study is MINOPTIONMAXITER (MOMI).

Definition 6 (MOMI): *The* MINOPTIONMAXITER *problem:*
Given *an MDP M, a non-negative real-value ϵ, and an integer ℓ,* **return** *$\mathcal{O}$ that minimizes $|\mathcal{O}|$ subject to $\mathcal{O} \subseteq \mathcal{O}_p$ and $L(\mathcal{O}) \leq \ell$.*

We now introduce our main result, which shows that both MIMO and MOMI are NP-hard.

Theorem 1. *MIMO and MOMI are NP-hard.*

Proof. We consider a problem OI-DEC which is a decision version of MIMO and MOMI. The problem asks if we can solve the MDP within ℓ iterations using at most k point options.

Definition 7 (OI-DEC):
Given *an MDP M, a non-negative real-value ϵ, and integers k and ℓ,* **return** *'Yes' if the there exists an option set $\mathcal{O}$ such that $\mathcal{O} \subseteq \mathcal{O}_p$, $|\mathcal{O}| \leq k$ and $L(\mathcal{O}) \leq \ell$. 'No' otherwise.*

We prove the theorem by reduction from the decision version of the set-cover problem—known to be NP-complete—to OI-DEC. The set-cover problem is defined as follows.

Definition 8 (SetCover-DEC):
Given *a set of elements $\mathcal{U}$, a set of subsets $\mathcal{X} = \{X \subseteq \mathcal{U}\}$, and an integer k,* **return** *'Yes' if there exists a cover $\mathcal{C} \subseteq \mathcal{X}$ that $\bigcup_{X \in \mathcal{C}} X = \mathcal{U}$ and $|\mathcal{C}| \leq k$. 'No' otherwise.*

If there is some $u \in \mathcal{U}$ that is not included in at least one of the subsets X, then the answer is 'No'. Assuming otherwise, we construct an instance of a shortest path problem (a special case of an MDP problem) as follows (Figure 2). There are four types of states in the MDP: (1) $u_i \in \mathcal{U}$ represents one of the elements in $\mathcal{U}$, (2) $X_i \in \mathcal{X}$ represents one of the subsets in $\mathcal{X}$, (3) $X'_i \in \mathcal{X}'$: we make a copy for every state $X_i \in \mathcal{X}$ and call them X'_i, (4) a goal state g. Thus, the state set is $\mathcal{U} \cup \mathcal{X} \cup \mathcal{X}' \cup \{g\}$. We build edges between states as follows: (1) $e(u, X) \in E$ iff $u \in X$: For $u \in \mathcal{U}$ and $X \in \mathcal{X}$, there is an edge between u and X. (2) $\forall X_i \in \mathcal{X}$, $e(X_i, X'_i) \in E$: For every $X_i \in \mathcal{X}$, we have a edge from X_i to X'_i. (3) $\forall e(X', g) \in E$: for every $X' \in \mathcal{X}'_i$ we have a edge from X_i to the goal g. This construction can be done in polynomial time.

Let M be the MDP constructed in this way. We show that SetCover($\mathcal{U}, \mathcal{X}, k$) = OI-DEC($M, k, 2$). Note that by construction every state s_i, s'_i, and g converges to its optimal value within 2 iterations as it reaches the goal state g within 2 steps. A state $u \in \mathcal{U}$ converges within 2 steps if and only if there exists a point option (a) from X to g where $u \in X$, (b) from u to X' where $u \in X$, or (c) from u to g. For options of type (b) and (c), we can find an option of type (a) that makes u converge within 2 steps by setting the initial state of the option to $\mathcal{I}_o = X$, where $u \in X$, and the termination state to $\beta_o = g$. Let $\mathcal{O}$ be the solution of OI-DEC($M, k, 2$). If there exists an option of type (b) or (c), we can swap them with an option of type (a) and still maintain a solution. Let $\mathcal{C}$ be a set of initial states of each option in $\mathcal{O}$ ($\mathcal{C} = \{\mathcal{I}_o | o \in \mathcal{O}\}$). This construction exactly matches the solution of the SetCover-DEC.

□

In light of the computational difficulty of both problems, the appropriate approach is to find tractable approximation algorithms. However, from the reduction from the set-cover optimization problem, we can show that MOMI cannot be approximated to a constant factor. More precisely:

Theorem 2. *MOMI cannot be approximated in polynomial time within a factor of $c \cdot \ln n$, where c is a constant and n is the number of states, unless P = NP.*

Proof. The optimization version of the set-cover problem cannot be approximated within a factor of $c \cdot \ln n$ by a polynomial-time algorithm unless P = NP (Raz and Safra 1997). The set-cover optimization problem can be reduced to MOMI with a similar construction for a reduction from

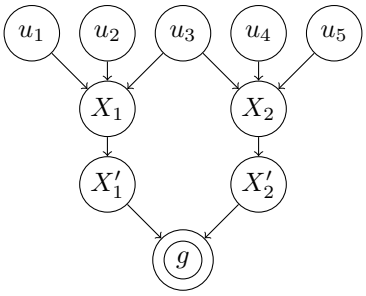


Figure 2: Reduction from SetCover-DEC to OI-DEC. The example shows the reduction from an instance of SetCover-DEC which asks if we can pick two subsets from $\mathcal{X} = \{X_1, X_2\}$ where $X_1 = \{1, 2, 3\}, X_2 = \{3, 4, 5\}$ to cover all elements $\mathcal{U} = \{1, 2, 3, 4, 5\}$. The SetCover-DEC can be reduced to an instance of OI-DEC where the question is whether the MDP can be solved with 2 iterations of VI by adding at most two point options. The answer of OI-DEC is 'Yes' (adding point options from X_1 and X_2 to g will solve the problem), thus the answer of the SetCover-DEC is 'Yes'. Here the set of initial states corresponds to the cover for the SetCover-DEC.

SetCover-DEC to OI-DEC. Here, the targeted minimization values of the two problems are equal: $P(\mathcal{C}) = |\mathcal{O}|$, and the number of states in OI-DEC is equal to the number of elements in the set cover on transformation. Assume there is a polynomial-time algorithm within a factor of $c \cdot \ln n$ approximation for MOMI where n is the number of states in the MDP. Let SetCover$(\mathcal{U}, \mathcal{X})$ be an instance of the set-cover problem. We can convert the instance into an instance of MOMI$(M, 0, 2)$. Using the approximation algorithm, we get a solution $\mathcal{O}$ where $|\mathcal{O}| \leq c \ln n |\mathcal{O}^*|$, where $\mathcal{O}^*$ is the optimal solution. We construct a solution for the set cover $\mathcal{C}$ from the solution to the MOMI $\mathcal{O}$ (see the construction in the proof of Theorem 1). Because $|\mathcal{C}| = |\mathcal{O}|$ and $|\mathcal{C}^*| = |\mathcal{O}^*|$, where $\mathcal{C}^*$ is the optimal solution for the set cover, we get $|\mathcal{C}| = |\mathcal{O}| \leq c \ln n |\mathcal{O}^*| = c \ln n |\mathcal{C}^*|$. Thus, we acquire a $c \cdot \ln n$ approximation solution for the set-cover problem within polynomial time, something only possible if P=NP. Thus, there is no polynomial-time algorithm with a factor of $c \cdot \ln n$ approximation for MOMI, unless P=NP. □

3.1 Generalizations of MIMO and MOMI

A natural question is whether Theorem 1 extends to more general option-construction settings. We consider two possible extensions, which we believe offer significant coverage of finding optimal options for planning in general.

We first consider the case where the options are not necessarily point options. Using the space of all options $\mathcal{O}_{all}$ we generalize MIMO as follows:

Definition 9 (MIMO$_{gen}$):
Given *an MDP M, a non-negative real-value ϵ, $\mathcal{O}' \subseteq \mathcal{O}_{all}$, and an integer ℓ,* **return** *$\mathcal{O}$ minimizing $L(\mathcal{O})$ such that $\mathcal{O} \subseteq \mathcal{O}'$ and $|\mathcal{O}| \leq k$.*

Theorem 3. *MIMO$_{gen}$ and MOMI$_{gen}$ are NP-hard.*

The proof follows from the fact that MIMO$_{gen}$ is a superset of MIMO MOMI$_{gen}$ is a superset of MOMI.

We next consider the multi-task generalization, where we aim to minimize the average number of iterations to solve a problem M sampled from a distribution of MDPs, D:

Definition 10 (MIMO$_{multi}$):
Given *A distribution of MDPs D, $\mathcal{O}' \subseteq \mathcal{O}_{all}$, a non-negative real-value ϵ, and an integer ℓ,* **return** *$\mathcal{O}$ that minimizes $E_{M \sim D}[L_M(\mathcal{O})]$ such that $\mathcal{O} \subseteq \mathcal{O}'$ and $|\mathcal{O}| \leq k$.*

Theorem 4. *MIMO$_{multi}$ and MOMI$_{multi}$ are NP-hard.*

The proof follows from the fact that MIMO$_{multi}$ is a superset of MIMO$_{gen}$ and MOMI$_{multi}$ is a superset of MOMI$_{gen}$.

In summary, *the problem of computing optimal behavioral abstractions for planning is intractable*.

4 Approximation Algorithms

We now provide polynomial-time approximation algorithms, A-MIMO and A-MOMI, to solve MIMO and MOMI, respectively. Both algorithms have bounded suboptimality slightly worse than a constant factor for deterministic MDPs. We assume that (1) there is exactly one absorbing state $g \in \mathcal{S}$ with $T(g, a, g) = 1$ and $R(g, a) = 0$, and every optimal policy eventually reaches g with probability 1, (2) there is no cycle with a positive reward involved in the optimal policy's trajectory. That is, $V^{\pi}_{+}(s) := \mathbb{E}[\sum_{t=0}^{\infty} \max\{0, R(s, a)\}] < \infty$ for all policies π. Note that we can convert a problem with multiple goals to a problem with a single goal by adding a new absorbing state g to the MDP and adding a transition from each of the original goals to g.

Unfortunately, these algorithms are computationally harder than solving the MDP itself, and are thus not practical for planning. Instead, they are useful for analyzing and evaluating heuristically generated options. If the option set generated by the heuristic methods outperforms the option set found by the following algorithms, then one can claim that the option set found by the heuristic is close to the optimal option set (for that MDP). Our algorithms have a formal guarantee on bounded suboptimality if the MDP is deterministic, so any heuristic method that provably exceeds our algorithm's performance will also guarantee bounded suboptimality. We also believe that the following algorithms may be a useful foundation for future option discovery methods.

4.1 A-MIMO

The outline of the approximation algorithm for MIMO, A-MIMO, is as follows.

1. Compute an asymmetric distance function $d_\epsilon(s, s') : S \times S \to \mathbb{N}$ representing the number of iterations for a state s to reach its ϵ-optimal value if we add a point option from a state s' to a goal state g.
2. Using this distance function, solve an asymmetric k-center problem, which finds a set of center states that minimizes the maximum number of iterations for every state to converge.

3. Generate point options with initiation states set to the center states in the solution of the asymmetric k-center, and termination states set to the goal.

(1) First, we compute a distance function $d_\epsilon : \mathcal{S} \times \mathcal{S} \to \mathbb{N}$, defined as follows:

Definition 11 (Distance $d_\epsilon(s_i, s_j)$): *$d_\epsilon(s_i, s_j)$ is the number of iterations for s_i to reach ϵ-optimal if we add a point option from s_j to g minus one.*

More formally, let $d'_\epsilon(s_i)$ denote the number of iterations needed for the value of state s_i to satisfy $|V(s_i) - V^*(s_i)| < \epsilon$, and let $d'_\epsilon(s_i, s_j)$ be an upper bound of the number of iterations needed for the value of s_i to satisfy $|V(s_i) - V^*(s_i)| < \epsilon$, if the value of s_j is initialized such that $|V(s_j) - V^*(s_j)| < \epsilon$. We define $d_\epsilon(s_i, s_j) := \min(d'_\epsilon(s_i) - 1, d'_\epsilon(s_i, s_j))$. For simplicity, we use d to denote the function d_ϵ. Consider the following example.

Example. Table 1 is a distance function for the MDP shown in Figure 3. For a deterministic MDP, $d_0(s)$ corresponds to the number of edge traversals from state s to g, where we have edges only for those that corresponds to the state transition by the optimal actions. The quantity $d_0(s, s') - 1$ is the minimum of $d_0(s)$ and one plus the number of edge traversals from s to s'. □

Note that we only need to solve the MDP once to compute d. $d(s, s')$ can be computed once you solved the MDP without any options and store all value functions V_i ($i = 1, ...b$) until convergence as a function of V_1: $V_i(s) = f(V_1(s_0), V_1(s_1), ...)$. If we add a point option from s' to g, then $V_1(s') = V^*(s')$. Thus, $d(s, s')$ is the smallest i where $V_i(s)$ reaches ϵ-optimal if we replace $V_1(s')$ with $V^*(s')$ when computing $V_i(s)$ as a function of V_1.

(2) We exploit this characteristic of d and solve the asymmetric k-center problem (Panigrahy and Vishwanathan 1998) on $(\mathcal{U}, d, k)$ to get a set of centers, which we use as initiation states for point options. The asymmetric k-center problem is a generalization of the metric k-center problem where the function d obeys the triangle inequality, but is not necessarily symmetric:

Definition 12 (AsymKCenter):
Given *a set of elements $\mathcal{U}$, a function $d : \mathcal{U} \times \mathcal{U} \to \mathbb{N}$, and an integer k,* **return** *$\mathcal{C}$ that minimizes $P(\mathcal{C}) = \max_{s \in U} \min_{c \in \mathcal{C}} d(s, c)$ subject to $|\mathcal{C}| \leq k$.*

We solve the problem using a polynomial-time approximation algorithm proposed by Archer (2001). The algorithm has a suboptimality bound of $O(\log^* k)$ where $k < |\mathcal{U}|$ and it is proven that the problem cannot be solved within a factor of $\log^* |\mathcal{U}| - \theta(1)$ unless P=NP (Chuzhoy et al. 2005). As the procedure by Archer (2001) often finds a set of options smaller than k, we generate the rest of the options by greedily adding $\log k$ options at once. See the supplementary material for details.

(3) We generate a set of point options with initiation-states set to one of the centers and the termination state set to the goal state of the MDP. That is, for every c in $\mathcal{C}$, we generate a point option starting from c to the goal state g.

Example. Consider an MDP shown in Figure 3. The distance d_0 for the MDP is shown in Table 1. Note that $d(s, s') \leq d(s, g)$ holds for every s, s' pair. Let us first consider finding one option ($k = 1$). This process corresponds to finding a column with the smallest maximum value in the Table 1. The optimal point option is from s_5 to g as it has the smallest maximum value in the column. If $k = 2$, an optimal set of options is from s_2 and s_4 to g. Note that the optimal option for $k = 1$ is not in the optimal option set of size 2. This example shows that the strategy of greedily adding options does not find the optimal set. In fact, the improvement $L(\emptyset) - L(\mathcal{O})$ on by the greedy algorithm can be arbitrary small (i.e. 0) compared to the optimal option (see Proposition 1 in the supplementary material for proof). □

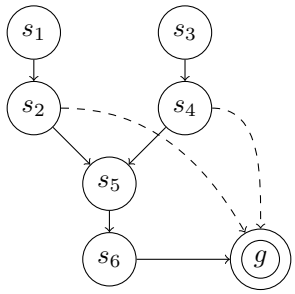


Figure 3: Example for A-MIMO with $k = 2$. Discovered options are denoted by the dashed lines.

$s \setminus s'$	s_1	s_2	s_3	s_4	s_5	s_6
s_1	0	1	3	3	2	3
s_2	2	0	2	2	1	2
s_3	3	3	0	1	2	3
s_4	2	2	2	0	1	2
s_5	1	1	1	1	0	1
s_6	0	0	0	0	0	0

Table 1: $d_0(s, s')$ for Figure 3.

Theorem 5. A-MIMO *has the following properties:*

1. A-MIMO *runs in polynomial time.*
2. *If the MDP is deterministic, it has a bounded suboptimality of $\log^* k$.*
3. *The number of iterations to solve the MDP using the option set acquired is upper bounded by $P(\mathcal{C})$.*

Proof. See the supplementary material. □

4.2 A-MOMI

We now describe a polynomial-time approximation algorithm, A-MIMO, based on using set cover to solve MOMI. The overview of the procedure is as follows.

1. Compute d for every state pair.

2. For every state s_i, compute a set of states X_{s_i} within $\ell - 1$ distance of reaching s_i. The set X_{s_i} represents the states that converge within ℓ steps if we add a point option from s_i to g.
3. Let $\mathcal{X}$ be a set of X_{s_i} for every $s_i \in \mathcal{S} \setminus X_g^+$, where X_g^+ is a set of states that converges within ℓ without any options (thus can be ignored).
4. Solve the set-cover optimization problem to find a set of subsets that covers the entire state set using the approximation algorithm by Chvatal (1979). This process corresponds to finding a minimum set of subsets $\{X_{s_i}\}$ that makes every state in $\mathcal{S}$ converge within ℓ steps.
5. Generate a set of point options with initiation states set to one of the center states in the solution of the asymmetric k-center, and termination states set to the goal.

Example. We use the MDP shown in Figure 3 as an example. Consider the problem of finding a set of options so that the MDP can be solved within 2 iterations. We generate an instance of a set-cover optimization problem. The set of element for the set cover is the set of states of the MDP that do not reach their optimal value within ℓ steps without any options $\mathcal{S} \setminus X_g^+$. Here, we denote a set of nodes that can be solved within ℓ steps by X_g^+. In the example, $\mathcal{U} = S \setminus X_g^+ = \{s_1, s_2, s_3, s_4\}$. A state s is included in a subset $X_{s'}$ iff $d(s, s') \leq \ell - 1$. For example, $X_{s_1} = \{s_1\}, X_{s_2} = \{s_1, s_2\}$. Thus, the set of subsets are given as: $X_{s_1} = \{s_1\}, X_{s_2} = \{s_1, s_2\}, X_{s_3} = \{s_3\}, X_{s_4} = \{s_3, s_4\}$. In this case, the approximation algorithm finds the optimal solution $\mathcal{C} = \{X_{s_2}, X_{s_4}\}$ for the set-cover optimization problem $(\mathcal{U}, \mathcal{X})$. We generate a point option for each state in $\mathcal{C}$. Thus, the output of the algorithm is a set of two point options from s_2 and s_4 to g. □

Theorem 6. A-MOMI *has the following properties:*

1. A-MOMI *runs in polynomial time.*
2. *It guarantees that the MDP is solved within ℓ iterations using the option set acquired by* A-MOMI $\mathcal{O}$.
3. *If the MDP is deterministic, the option set is at most $O(\log k)$ times larger than the smallest option set possible to solve the MDP within ℓ iterations.*

Proof. See the supplementary material. □

Note that the approximation bound for deterministic MDP may be improved by improving the approximation algorithm for the set cover. It is proven to be NP-hard to approximate up to a factor of $(1 - o(1)) \log n$ (Dinur and Steurer 2014), thus there may be an improvement on the approximation ratio for the set cover problem, which will also improve the approximation ratio of A-MIMO.

5 Experiments

We evaluate the performance of the value-iteration algorithm using options generated by the approximation algorithms on several grid-based simple domains.

We ran the experiments on an 11×11 four-room domain and a 9×9 grid world with no walls. In both domains, the agent's goal is to reach a specific square. The agent can move in the usual four directions but cannot cross walls.

Visualizations First, we visualize a variety of option types, including the optimal point options, those found by our approximation algorithms, and several option types proposed in the literature. We computed the optimal set of point options by enumerating every possible set of point options and picking the best. We are only able to find optimal solutions up to four options within 10 minutes, while the approximation algorithm could find any number of options within a few minutes. Both betweenness options and eigenoptions are discovered by a polynomial time algorithm, thus able to discover within a few minutes. Figure 8 shows the optimal and bounded suboptimal set of options computed by A-MIMO. See the supplementary material for visualizations for the 9×9 grid domain.

Figure 4e shows the four bottleneck states with highest shortest-path betweenness centrality in the state-transition graph (Şimşek and Barto 2009). Interestingly, the optimal options are quite close to the bottleneck states in the four-room domain, suggesting that bottleneck states are also useful for planning as a heuristic to find important subgoals.

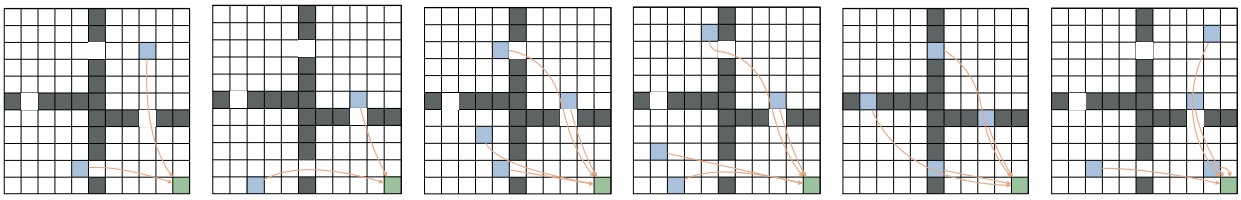
Figure 4f shows the set of subgoals discovered by graph Laplacian analysis following the method of Machado, Bellemare, and Bowling (2018). While they proposed to generate options to travel between subgoals for reinforcement learning, we generate a set of point options from each subgoal to the goal state as that is a better use of the subgoals for planning setting.

Quantitative Evaluation Next, we run value iteration using the set of options generated by A-MIMO and A-MOMI. Figures 5a and 5b show the number of iterations on the four-room and the 9×9 grids using a set of options of size k. The experimental results suggest that the suboptimal algorithm finds set of options similar to, but not quite as good as, the optimal ones. For betweenness options and eigenoptions, we evaluated every subset of options among the four and present results for the best subset found. Because betweenness options are placed close to the optimal options, the performance is close to optimal especially when the number of options are small.

In addition, we used A-MOMI to find a minimum option set to solve the MDP within the given number of iterations. Figures 5c and 5d show the number of options generated by A-MOMI compared to the minimum number of options.

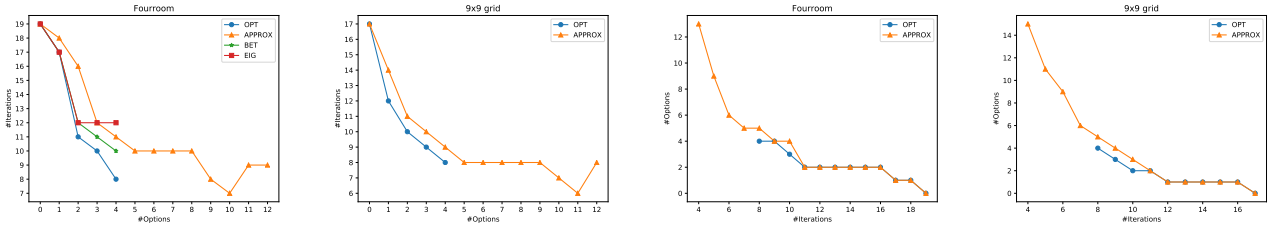
6 Related Work

Many heuristic algorithms have proposed to discover options useful for some purposes (Iba 1989; McGovern and Barto 2001; Menache, Mannor, and Shimkin 2002; Stolle and Precup 2002; Şimşek and Barto 2004; Şimşek and Barto 2009; Konidaris and Barto 2009; Machado, Bellemare, and Bowling 2018; Eysenbach et al. 2018). These algorithms seek to capture varying intuitions about what makes behavioral abstraction useful. Jong, Hester, and Stone (2008) sought to investigate the utility of options empirically and pointed out that introducing options might worsen learning performance.



(a) optimal $k = 2$ (b) approx. $k = 2$ (c) optimal $k = 4$ (d) approx. $k = 4$ (e) Betweenness (f) Eigenoptions

Figure 4: Comparison of the optimal point options with options generated by the approximation algorithm A-MIMO. The green square represents the termination state and the blue squares the initiation states. Observe that the approximation algorithm is similar to that of optimal options. Note that the optimal option set is not unique: there can be multiple optimal option sets, and we are visualizing just one returned by the solver.



(a) Four Room (MIMO) (b) 9×9 grid (MIMO) (c) Four Room (MOMI) (d) 9×9 grid (MOMI)

Figure 5: MIMO and MOMI evaluations. Parts (a)–(c) show the number of iterations for VI using options generated by A-MIMO. Parts (d)–(f) show the number of options generated by A-MOMI to ensure the MDP is solved within a given number of iterations. OPT: optimal set of options. APPROX: a bounded suboptimal set of options generated by A-MIMO an A-MOMI. BET: betweenness options. EIG: eigenoptions.

They argued that options can potentially improve the learning performance by encouraging exploitation or exploration. For example, Stolle and Precup (2002) proposed to set states with high visitation count as subgoal states, resulting in identifying bottleneck states in the four-room domain. Şimşek and Barto (2009) generalized the concept of the bottleneck to (shortest-path) betweenness of the graph to capture how pivotal the state is. Menache, Mannor, and Shimkin (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. These methods generate options to leverage the idea that subgoals are states visited most frequently. On the other hand, Şimşek and Barto (2004) proposed to generate options to encourage exploration by generating options to relatively novel states, encouraging exploration. Eysenbach et al. (2018) instead proposed learning a policy for each option so that the diversity of the trajectories by the set of options are maximized. These methods generate options to explore infrequently visited states. That being said, the problem of discovering efficient behavioral abstraction in reinforcement learning is still an open question.

For planning, several works have shown empirically that adding a particular set of options or macro-operators can speed up planning algorithms (Francis and Ram 1993; Sutton and Barto 1998; Silver and Ciosek 2012; Konidaris 2016). In terms of theoretical analysis, Mann, Mannor, and Precup (2015) analyzed the convergence rate of approximate value iteration with and without options and showed that options lead to faster convergence if their durations are longer and a value function is initialized pessimistically. As in reinforcement learning, how to find efficient temporal abstractions for planning automatically remains an open question.

7 Conclusions

We considered a fundamental theoretical question concerning the use of behavioral abstractions to solve MDPs. We considered two problem formulations for finding options: (1) minimize the number of iterations given a maximum size of option set (MIMO) and (2) minimize the size of option set given a maximum number of iterations (MOMI). We showed that the two problems are both computationally intractable, even for deterministic MDPs. For each problem, we produced a polynomial-time algorithm for MDPs with bounded reward and goal states, with bounded optimality for deterministic MDPs. In the future, we are interested in using the insights established here to develop principled option-discovery algorithms for model-based reinforcement learning. Since we now know which options minimize planning time, we can better guide model-based agents toward learning them and potentially reduce sample complexity considerably.

Acknowledgments

We would like to thank the anonymous reviewer for their advice and suggestions on Theorem 2 to improve the inapproximability result for MOMI.

References

[2001] Archer, A. 2001. Two O(log* k)-approximation algorithms for the asymmetric k-center problem. In *International Conference on Integer Programming and Combinatorial Optimization*, 1–14.

[2013] Bacon, P.-L. 2013. On the bottleneck concept for options discovery. *Masters thesis, McGill University*.

[2004] Bakker, B., and Schmidhuber, J. 2004. Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In *Proceedings of the 8th Conference on Intelligent Autonomous Systems*, 438–445.

[2005] Chuzhoy, J.; Guha, S.; Halperin, E.; Khanna, S.; Kortsarz, G.; Krauthgamer, R.; and Naor, J. S. 2005. Asymmetric k-center is log* n-hard to approximate. *Journal of the ACM* 52(4):538–551.

[1979] Chvatal, V. 1979. A greedy heuristic for the set-covering problem. *Mathematics of operations research* 4(3):233–235.

[2014] Dinur, I., and Steurer, D. 2014. Analytical approach to parallel repetition. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, 624–633. ACM.

[2018] Eysenbach, B.; Gupta, A.; Ibarz, J.; and Levine, S. 2018. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*.

[1993] Francis, A. G., and Ram, A. 1993. The utility problem in case-based reasoning. In *Case-Based Reasoning: Papers from the 1993 Workshop*, 160–161.

[1982] Hochbaum, D. S. 1982. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing* 11(3):555–556.

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

[2008] Jong, N. K.; Hester, T.; and Stone, P. 2008. The utility of temporal abstraction in reinforcement learning. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, 299–306.

[2009] 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.

[2016] Konidaris, G. 2016. Constructing abstraction hierarchies using a skill-symbol loop. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, volume 2016, 1648.

[1995] Littman, M. L.; Dean, T. L.; and Kaelbling, L. P. 1995. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 394–402.

[2018] Machado, M. C.; Bellemare, M. G.; and Bowling, M. 2018. A Laplacian framework for option discovery in reinforcement learning. In *Proceedings of the Thirty-fourth International Conference on Machine Learning*.

[2015] Mann, T. A.; Mannor, S.; and Precup, D. 2015. Approximate value iteration with temporally extended actions. *Journal of Artificial Intelligence Research* 53:375–438.

[2001] 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*, 361–368.

[2002] 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.

[2012] Moradi, P.; Shiri, M. E.; Rad, A. A.; Khadivi, A.; and Hasler, M. 2012. Automatic skill acquisition in reinforcement learning using graph centrality measures. *Intelligent Data Analysis* 16(1):113–135.

[1998] Panigrahy, R., and Vishwanathan, S. 1998. An O(log* n) approximation algorithm for the asymmetric p-center problem. *Journal of Algorithms* 27(2):259–268.

[2002] Pickett, M., and Barto, A. 2002. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 506–513.

[1998] Precup, D., and Sutton, R. S. 1998. Multi-time models for temporally abstract planning. In *Advances in neural information processing systems*, 1050–1056.

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

[1997] Raz, R., and Safra, S. 1997. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 475–484. ACM.

[2012] Silver, D., and Ciosek, K. 2012. Compositional planning using optimal option models. In *Proceedings of the 29th International Conference on Machine Learning*, 1063–1070.

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

[2009] Şimşek, Ö., and Barto, A. G. 2009. Skill characterization based on betweenness. In *Advances in Neural Information Processing Systems*, 1497–1504.

[2005] Şimşek, Ö.; Wolfe, A.; and Barto, A. 2005. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the Twenty Second International Conference on Machine Learning*, 816–823.

[2002] Stolle, M., and Precup, D. 2002. Learning options in reinforcement learning. In *International Symposium on Abstraction, Reformulation, and Approximation*, 212–223.

[1998] Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

[1999] Sutton, R.; 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.

[1993] Williams, R. J., and Baird, L. C. 1993. Tight performance bounds on greedy policies based on imperfect value functions. Technical report, College of Computer Science, Northeastern University.

Appendix: Proofs

7.1 A-MIMO

In this subsection we show the follwing theorem:

Theorem 5. A-MIMO *has following properties:*

1. A-MIMO *runs in polynomial time.*
2. *If the MDP is deterministic, it has a bounded suboptimality of* $\log^* k - O(1)$.
3. *The number of iterations to solve the MDP using the acquired options is upper bounded by* $P(\mathcal{C})$.

Theorem 5.1. A-MIMO *runs in polynomial time.*

Proof. Each step of the procedure runs in polynomial time.

(1) Solving an MDP takes polynomial time. To compute d we need to solve MDPs at most $|\mathcal{S}|$ times. Thus, it runs in polynomial time.

(2) The approximation algorithm we deploy for solving the asymmetric-k center which runs in polynomial time (Archer 2001). Because the procedure by Archer (2001) terminates immediately after finding a set of options which guarantees the suboptimality bounds, it tends to find a set of options smaller than k. In order to use the rest of the options effectively within polynomial time, we use a procedure Expand to greedily add a few options at once until it finds all k options. We enumerate all possible set of options of size $r = \lceil \log k \rceil$ (if $|\mathcal{O}| + \log k > k$ then we set $r = k - |\mathcal{O}|$) and add a set of options which minimizes ℓ (breaking ties randomly) to the option set $\mathcal{O}$. We repeat this procedure until $|\mathcal{O}| = k$. This procedure runs in polynomial time. The number of possible option set of size r is ${}_rC_n = O(n^r) = O(k)$. We repeat this procedure at most $\lceil k/\log k \rceil$ times, thus the total computation time is bounded by $O(k^2/\log k)$.

(3) Immediate.

Therefore, A-MIMO runs in polynomial time. □

Before we show that it is sufficient to consider a set of options with its terminal state set to the goal state of the MDP.

Lemma 1. *There exists an optimal option set for MIMO and MOMI with all terminal state set to the goal state.*

Proof. Assume there exists an option with terminal state set to a state other than the goal state in the optimal option set $\mathcal{O}$. By triangle inequality, swapping the terminal state to the goal state will monotonically decrease $d(s, g)$ for every state. By swapping every such option we can construct an option set $\mathcal{O}'$ with $L(\mathcal{O}') \leq L(\mathcal{O})$. □

Lemma imply that discovering the best option set among option sets with their terminal state fixed to the goal state is sufficient to find the best option set in general. Therefore, our algorithms seek to discover options with termination state fixed to the goal state.

Using the option set acquired, the number of iterations to solve the MDP is bounded by $P(\mathcal{C})$. To prove this we first generalize the definition of the distance function to take a state and a set of states as arguments $d_\epsilon : \mathcal{S} \times 2^{\mathcal{S}} \to \mathbb{N}$. Let $d_\epsilon(s, \mathcal{C})$ the number of iterations for s to converge ϵ-optimal if every state $s' \in \mathcal{C}$ has converged to ϵ-optimal: $d_\epsilon(s, \mathcal{C}) := \min(d'_\epsilon(s), 1 + d'_\epsilon(s, \mathcal{C})) - 1$. As adding an option will never make the number of iterations larger,

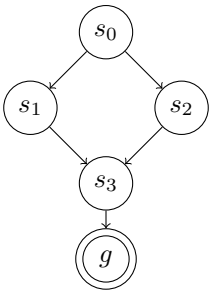


Figure 6: An example of an MDP where $d(s, \mathcal{C}) < \min_{s' \in \mathcal{C}} d(s, s')$. Here the transition induced by the optimal policy is stochastic, thus from s_0 one may go to s_1 and s_2 by probability 0.5 each. Either adding an option from s_1 or s_2 to g does not make the convergence faster, but adding both makes it faster.

Lemma 2.

$$d(s, \mathcal{C}) \leq \min_{s' \in \mathcal{C}} d(s, s'). \tag{8}$$

Using this, we show the following proposition.

Theorem 5.2. *The number of iterations to solve the MDP using the acquired options is upper bounded by* $P(\mathcal{C})$.

Proof. $P(\mathcal{C}) = \max_{s \in S} \min_{c \in \mathcal{C}} d(s, c) \geq \max_{s \in S} d(s, \mathcal{C}) = L(\mathcal{O})$ (using Equation 8). Thus $P(\mathcal{C})$ is an upper bound for $L(\mathcal{O})$. □

The reason why $P(\mathcal{C})$ does not always give us the exact number of iterations is because adding two options starting from s_1, s_2 may make the convergence of s_0 faster than $d(s_0, s_1)$ or $d(s_0, s_2)$. Example: Figure 6 is an example of such an MDP. From s_0 it may transit to s_1 and s_2 with probability 0.5 each. Without any options, the value function converges to exactly optimal value for every state with 3 steps. Adding an option either from s_1 or s_2 to g does not shorten the iteration for s_0 to converge. However, if we add two options from s_1 and s_2 to g, s_0 converges within 2 steps, thus the MDP is solved with 2 steps.

The equality of the statement 8 holds if the MDP is deterministic. That is, $d(s, \mathcal{C}) = \min_{s' \in \mathcal{C}} d(s, s')$ for deterministic MDP.

Theorem 5.3.

If the MDP is deterministic, it has a bounded suboptimality of $\log^* k$.

Proof. First we show $P(\mathcal{C}^*) = L(\mathcal{O}^*)$ for deterministic MDP. From $d(s, \mathcal{C}) = \min_{s' \in \mathcal{C}} d(s, s')$, $P(\mathcal{C}^*) = \max_{s \in \mathcal{S}} \min_{c \in \mathcal{C}^*} d(s, c) = \max_{s \in \mathcal{S}} d(s, \mathcal{C}^*) = L(\mathcal{O}^*)$.

The asymmetric k-center solver guarantees that the output $\mathcal{C}$ satisfies $P(\mathcal{C}) \leq c(\log^* k + O(1))P(\mathcal{C}^*)$ where n is the number of nodes (Archer 2001). Let MIMO(M, ϵ, k) be an instance of MIMO. We convert this instance to an instance of asymmetric k-center AsymKCenter$(\mathcal{U}, d, k)$, where $|\mathcal{U}| = |\mathcal{S}|$. By solving the asymmetric k-center with the

approximation algorithm, we get a solution $\mathcal{C}$ which satisfies $P(\mathcal{C}) \leq c(\log^* k + O(1))P(\mathcal{C}^*)$. Thus, the output of the algorithm $\mathcal{O}$ satisfies $L(\mathcal{O}) = P(\mathcal{C}) \leq c(\log^* k + O(1))P(\mathcal{C}^*) = c(\log^* k + O(1))L(\mathcal{O}^*)$. Thus, $L(\mathcal{O}) \leq c(\log^* k + O(1))L(\mathcal{O}^*)$ is derived. □

Proposition 1 (Greedy Strategy)**.** *Let an option set $\mathcal{O}$ be a set of point option constructed by greedily adding one point option which minimizes the number of iterations. An improvement $L(\emptyset) - L(\mathcal{O})$ by the greedy algorithm can be arbitrary small (i.e. 0) compared to the optimal option set.*

Proof. We show by the example in a shortest-path problem in Figure 7. The MDP can be solved within 4 iterations without options: $L(\emptyset) = 4$. With an optimal option set of size $k = 2$ the MDP can be solved within 2 iterations: $L(\mathcal{O}^*) = 2$ (an initiation state of each option in optimal option set is denoted by $*$ in the Figure). On the other hand, a greedy strategy may not improve L at all. No single point option does not improve L. Let's say we picked a point option from s_1 to g. Then, there is no single point option we can add to that option to improve L in the second iteration. Therefore, the greedy procedure returns $\mathcal{O}$ which has $L(\emptyset) - L(\mathcal{O}) = 0$. Therefore, $(L(\emptyset) - L(\mathcal{O}))/(L(\emptyset) - L(\mathcal{O}^*))$ can be arbitrary small non-negative value (i.e. 0).

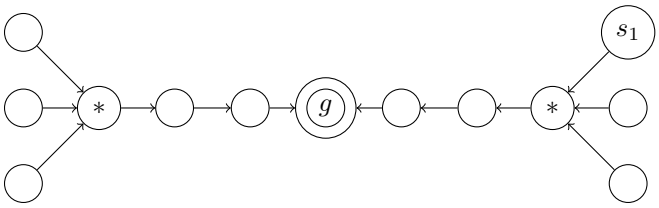


Figure 7: Example of MIMO where the improvement of a greedy strategy can be arbitrary small compared to the optimal option set.

□

7.2 A-MOMI

In this subsection we show the following theorem:

Theorem 6. A-MOMI *has the following properties:*

1. A-MOMI *runs in polynomial time.*
2. *It guarantees that the MDP is solved within ℓ iterations using the option set acquired by* A-MOMI *$\mathcal{O}$.*
3. *If the MDP is deterministic, the option set is at most $\max_{s \in \mathcal{S}} X_s$ times larger than the smallest option set possible to solve the MDP within ℓ iterations.*

Theorem 6.1. A-MOMI *runs in polynomial time.*

Proof. Each step of the procedure runs in polynomial time.

(1) Solving an MDP takes polynomial time (Littman, Dean, and Kaelbling 1995). To compute d we need to solve MDPs at most $|\mathcal{S}|$ times. Thus, it runs in polynomial time.

(4) We solve the set cover using a polynomial time approximation algorithm (Hochbaum 1982) which runs in $O(|\mathcal{S}|^3)$, thus run in polynomial time.

(2), (3), and (5) Immediate. □

Theorem 6.2. A-MOMI *guarantees that the MDP is solved within ℓ iterations using the option set $\mathcal{O}$.*

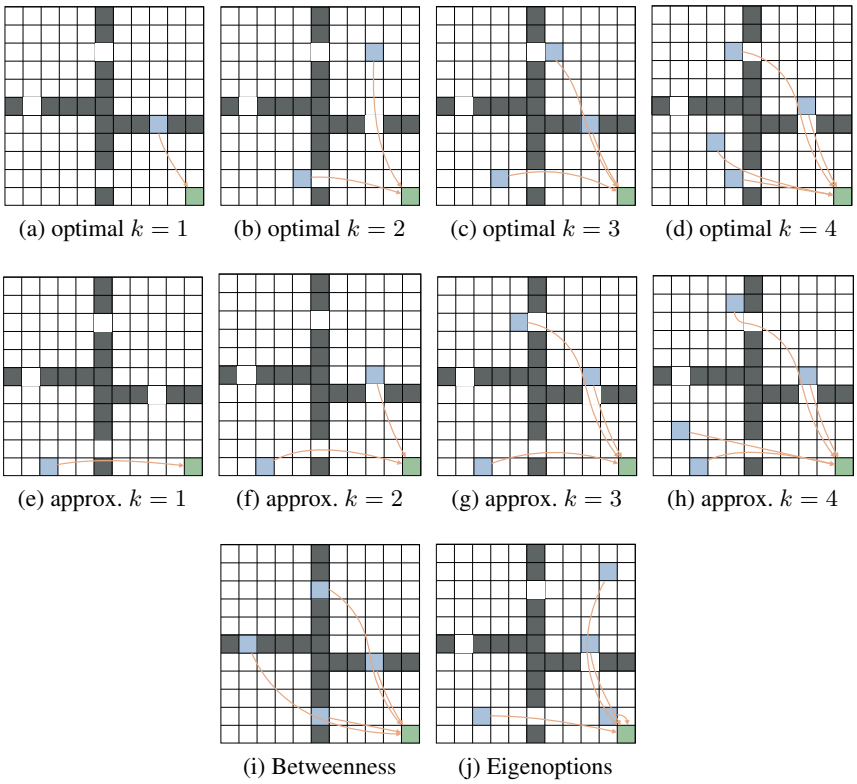
Proof. A state $s \in X_g^+$ reaches optimal within ℓ steps by definition. For every state $s \in \mathcal{S} \setminus X_g^+$, the set cover guarantees that we have $X_{s'} \in \mathcal{C}$ such that $d(s, s') < \ell$. As we generate an option from s' to g, s' reaches to optimal value with 1 step. Thus, s reaches to ϵ-optimal value within $d(s, s') + 1 \leq \ell$. Therefore, every state reaches ϵ-optimal value within ℓ steps. □

Theorem 6.3. *If the MDP is deterministic, the option set is at most $\max_{s \in \mathcal{S}} X_s$ times larger than the smallest option set possible to solve the MDP within ℓ iterations.*

Proof. Using a suboptimal algorithm by Hochbaum (1982) we get $\mathcal{C}$ such that $|\mathcal{C}| \leq \Delta|\mathcal{C}|$ where Δ is the maximum size of subsets in $\mathcal{X}$. Thus, $|\mathcal{O}| = |\mathcal{C}| \leq \Delta|\mathcal{C}^*| = \Delta|\mathcal{O}^*|$. □

Appendix: Experiments

We show the figures for experiments. Figure 8 shows the options found by solving MIMO optimally/suboptimally in four room domain. Figure 9 shows the options in 9x9 grid domain.

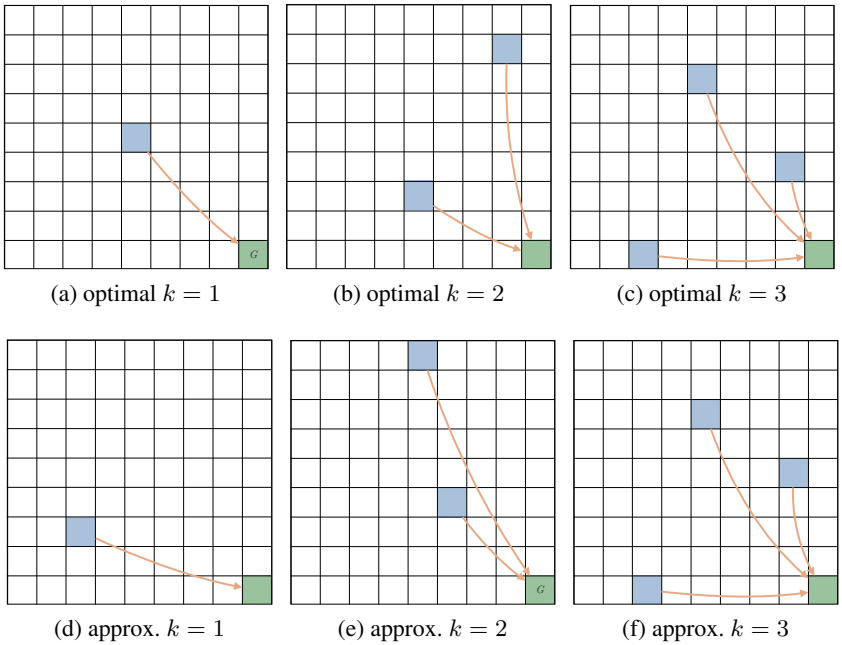


(a) optimal $k = 1$ (b) optimal $k = 2$ (c) optimal $k = 3$ (d) optimal $k = 4$

(e) approx. $k = 1$ (f) approx. $k = 2$ (g) approx. $k = 3$ (h) approx. $k = 4$

(i) Betweenness (j) Eigenoptions

Figure 8: Comparison of the optimal point options vs. options generated by the approximation algorithm A-MIMO. We observed that the approximation algorithm is similar to that of optimal options. Note that optimal option set is not unique: there can be multiple optimal option set, and we are visualize one of them returned by the solver.



(a) optimal $k = 1$ (b) optimal $k = 2$ (c) optimal $k = 3$

(d) approx. $k = 1$ (e) approx. $k = 2$ (f) approx. $k = 3$

Figure 9: Comparison of the optimal point options for planning vs. bottleneck options proposed for reinforcement learning in the four room domain. Initating conditions are shown in blue, the goal in green.