Finding Options that Minimize Planning Time

Yuu Jinnai 1  David Abel 1  D Ellis Hershkowitz 2  Michael L. Littman 1  George Konidaris 1

Abstract

We formalize the problem of selecting the optimal set of options for planning as that of computing the smallest set of options so that planning converges in less than a given maximum of value-iteration passes. We first show that the problem is \( \text{NP}\)-hard, even if the task is constrained to be deterministic—the first such complexity result for option discovery. We then present the first polynomial-time boundedly suboptimal approximation algorithm for this setting, and empirically evaluate it against both the optimal options and a representative collection of heuristic approaches in simple grid-based domains.

1. Introduction

Markov Decision Processes or MDPs (Puterman, 1994) are a widely used expressive model of sequential decision-making. However, MDPs are computationally expensive to solve (Papadimitriou & Tsitsiklis, 1987; Littman, 1997; Goldsmith et al., 1997). One approach to solving such problems is to add high-level, temporally extended actions—often formalized as options (Sutton et al., 1999)—to the set of actions available to the agent. 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 (Mann & Mannor, 2014; Silver & Ciosek, 2012). However, little is known about how to find the right set of options for planning. Prior work often seeks to codify an intuitive notion of what underlies an effective option, such as identifying relatively novel states (Šimšek & Barto, 2004), identifying bottleneck states or high-betweenness states (Šimšek et al., 2005; Šimšek & Barto, 2009; Bacon, 2013; Moradi et al., 2012), finding repeated policy fragments (Pickett & Barto, 2002), or finding states that often occur on successful trajectories (McGovern & Barto, 2001; Bakker & 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 an algorithm with performance guarantees and a principled theoretical foundation. Specifically, we consider the problem of finding the smallest set of options so that planning converges in fewer than \( \ell \) value iterations (VI). Our main result is that this problem is \( \text{NP}\)-hard. More precisely, the problem:

1. is \( 2^{\log^{1-\varepsilon} n} \)-hard to approximate for any \( \varepsilon > 0 \) unless \( \text{NP} \subseteq \text{DTIME}(n^{\text{poly log } n}) \); where \( n \) is the input size;
2. is \( \Omega((\log n)) \)-hard to approximate even for deterministic MDPs unless \( P = \text{NP} \);
3. has an \( O(n) \)-approximation algorithm;
4. has an \( O((\log n)) \)-approximation algorithm for deterministic MDPs.

In Section 4, we introduce A-MOMI, a polynomial-time approximation algorithm that has \( O(n) \) suboptimality in general and \( O((\log n)) \) suboptimality for deterministic MDPs. The expression \( 2^{\log^{1-\varepsilon} n} \) is only slightly smaller than \( n \): if \( \varepsilon = 0 \) then \( \Omega(2^{\log n}) = \Omega(n) \). Thus, the inapproximability results claim that A-MOMI is close to the best possible approximation factor.

In addition, we consider the complementary problem of finding a set of \( k \) options that minimize the number of VI iterations until convergence. We show that this problem is also \( \text{NP}\)-hard, even for a deterministic MDP and introduce A-MIMO, a polynomial time approximation algorithm.

\(^1\)This is a standard complexity assumption: See, for example, Dinitz et al. (2012).
Finally, we empirically evaluate the performance of two heuristic approaches for option discovery, betweenness options (Šimsek & Barto, 2009) and Eigenoptions (Machado et al., 2017), against the proposed approximation algorithms 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 and Planning

An MDP is a five tuple: \((\mathcal{S}, \mathcal{A}, R, T, \gamma)\), 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 \(\pi\) is evaluated according to the Bellman equation, denoting the long term expected reward received by executing \(\pi\):

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

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.

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

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

The value-planning problem can be solved in time polynomial in the size of the state space (Littman et al., 1995).

2.2. 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 et al., 1999). However, it is possible that options that are useful for learning are not necessarily useful for planning, and vice versa. In fact, we don’t have an explicit metric for measuring the quality of an option set for planning. Therefore, 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 et al., 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,
- \(\pi : \mathcal{S} \rightarrow \Pr(\mathcal{A})\) is a policy,
- \(\beta : \mathcal{S} \rightarrow [0, 1]\) is a termination condition.

We let \(\mathcal{O}\) denote the set containing all options.

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

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

\[
R_\gamma(s, o) = \mathbb{E}_{a \sim \pi} \left[ r_1 + \gamma r_2 + \ldots + \gamma^{k-1} r_k \mid s, o \right].
\]

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

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

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

**Definition 3** (\(L_{\epsilon, V_0}(\mathcal{O})\)): The number of iterations \(L_{\epsilon, V_0}(\mathcal{O})\) of VI using the joint action 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 \(s \in \mathcal{S}, b' \geq b\).

2.2.1. 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 inde-
we instead introduce and study "point options", which only when applied in the respective initiating state.

To plan with a point option from state with termination probability 1 can be represented as tic MDPs. Third, any other options with a single termination can be calculated as a path-planning problem for determinis-

tics. First, a point option is a simple model for a tempo-

The dark circles indicate where the option can 4 emerge from "one" option. Consequently, it is difficult to address the question: which grows larger, a combinatorial number of different behaviors can emerge from “one” option. As MDPs grow large, one option can encode a large number of possible, independent behaviors. Thus, we instead introduce and study “point options”, which only allow for a single continuous stream of behavior:

For simplicity, we denote the initiation state as $I_o$ and the termination state as $\beta_o$ for a point option $o$.

To plan with a point option from state $s$, the agent runs value iteration using a model (Eq. 2, 3) in addition to the backup operations by primitive actions where $k$ is the duration of the option. 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 tempo-

rally extended action. Second, the policy of the point option can be calculated as a path-planning problem for determinis-
tic MDPs. Third, any other options with a single termination state with termination probability 1 can be represented as a collection of point options. Fourth, a point option has constant computational overhead per iteration.

3. Complexity Results

Our main results focus on two computational problems:

1. $\text{MINOPTIONMAXITER (MOMI)}$: Which set of options lets value iteration converge in at most $\ell$ iterations?
2. $\text{MINTERMAXOPTION (MIMO)}$: Which set of $k$ or fewer options minimizes the number of iterations to convergence?

More formally, MOMI is defined as follows.

**Definition 5** (MOMI): The $\text{MINOPTIONMAXITER}$ problem:

Given an MDP $M$, a non-negative real-value $\epsilon$, an initial value function $V_0$, and an integer $\ell$ return $\mathcal{O}$ that minimizes $|\mathcal{O}|$ subject to $\mathcal{O} \subseteq \mathcal{O}_p$ and $L_{\epsilon, V_0}(\mathcal{O}) \leq \ell$.

We then consider the complementary optimization problem $\text{MINTERMAXOPTION (MIMO)}$: compute a set of $k$ options which minimizes the number of iterations:

**Definition 6** (MIMO): The $\text{MINTERMAXOPTION}$ problem:

Given an MDP $M$, a non-negative real-value $\epsilon$, an initial value function $V_0$, and an integer $k$ return $\mathcal{O}$ that minimizes $L_{\epsilon, V_0}(\mathcal{O})$, subject to $\mathcal{O} \subseteq \mathcal{O}_p$ and $|\mathcal{O}| \leq k$.

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

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

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

**Definition 7** (OI-DEC):

Given an MDP $M$, a non-negative real-value $\epsilon$, an initial value function $V_0$, and integers $k$ and $\ell$, return ‘Yes’ if there exists an option set $\mathcal{O}$ such that $\mathcal{O} \subseteq \mathcal{O}_p$, $|\mathcal{O}| \leq k$ and $L_{\epsilon, V_0}(\mathcal{O}) \leq \ell$. ‘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 $U$, a set of subsets $X = \{X \subseteq U\}$, and an integer $k$, return ‘Yes’ if there exists a
Finding Options that Minimize Planning Time

Let $M$ be the MDP constructed in this way. We show that

$$\text{SetCover}(\mathcal{U}, X, k) = \text{OI-DEC}(M, V_0 = 0, k; 2).$$

Note that by construction every state $X_i, X'_i$, 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 $I_o = X$, 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. Thus, it is sufficient to consider that every option is type (a). Let $C$ be a set of initial states of each option in $\mathcal{O}$ ($C = \{I_o | o \in \mathcal{O}\}$). This construction exactly matches the solution of the SetCover-DEC.

3.1. Generalizations of MOMI and MIMO

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. There is little sense in considering MOMI where one can choose any option since clearly the best option is the option whose policy is the optimal policy. Thus, using the space of all options $O_{alt}$ we generalize MOMI as follows (MIMO$_{gen}$ are defined analogously):

\begin{definition}[MOMI$_{gen}$]
Given an MDP $M$, a non-negative real-value $\epsilon$, an initial value function $V_0$, $O' \subseteq O_{alt}$, and an integer $\ell$, return $\mathcal{O}$ minimizing $|\mathcal{O}|$ subject to $L_{\epsilon, V_0}(\mathcal{O}) \leq \ell$ and $\mathcal{O} \subseteq O'$.
\end{definition}

\begin{theorem}
MOMI$_{gen}$ and MIMO$_{gen}$ are NP-hard.
\end{theorem}

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

We next consider the multi-task generalization, where we aim to find a smallest number of options which the expected number of iterations to solve a problem $M$ sampled from a distribution of MDPs, $D$, is bounded:

\begin{definition}[MOMI$_{multi}$]
Given a distribution of MDPs $D$, $O' \subseteq O_{alt}$, a non-negative real-value $\epsilon$, an initial value function $V_0$, and an integer $\ell$, return $\mathcal{O}$ that minimizes $|\mathcal{O}|$ such that $E_{M \sim D}[L_M(\mathcal{O})] \leq \ell$ and $\mathcal{O} \subseteq O'$.
\end{definition}

\begin{theorem}
MOMI$_{multi}$ and MIMO$_{multi}$ are NP-hard.
\end{theorem}

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

In light of the computational difficulty of both problems, the appropriate approach is to find tractable approximation algorithms. However, even approximately solving MOMI is hard. More precisely:

\begin{theorem}
1. MOMI is $\Omega(\log n)$ hard to approximate even for deterministic MDPs unless $P = NP$.
2. MOMI$_{gen}$ is $2^{\log^{1+\epsilon} n}$-hard to approximate for any $\epsilon > 0$ even for deterministic MDPs unless $NP \subseteq DTIME(n^{poly \log n})$.
\end{theorem}
Finding Options that Minimize Planning Time

3. MOMI is $2^{\log^{1-\epsilon} n}$-hard to approximate for any $\epsilon > 0$ unless $NP \subseteq DTIME(n^{\text{poly log } n})$.

Proof. See appendix. □

Note that an $O(n)$-approximation is achievable by the trivial algorithm that returns a set of all candidate options. Thus, Theorem 4 roughly states that there is no polynomial time approximation algorithms other than the trivial algorithm for MOMI.

In the next section we show that an $O(\log n)$-approximation is achievable for MOMI if the MDP is deterministic and the agent is given a set of all point options. Thus, together, these two results give a formal separation between the hardness of abstraction in MDPs with and without stochasticity.

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

4. Approximation Algorithms

We now provide polynomial-time approximation algorithms, A-MIMO and A-MOMI, to solve MOMI and MIMO, 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 $g \in S$ 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, $V_\pi^*(s) := \mathbb{E}[\sum_{i=0}^\infty \max\{0, R(s_i, a_i)\}] < \infty$ for all policies $\pi$. 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 these algorithms may be a useful foundation for future option discovery methods.

4.1. A-MOMI

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

1. Compute an asymmetric distance function $d_s(s, s') : S \times S \to \mathbb{N}$ representing the number of iterations for a state $s$ to reach its $\epsilon$-optimal value if we add a point option from a state $s'$ to a goal state $g$.
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 $\ell$ 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 $s_i \in S \setminus X_g^+$, where $X_g^+$ is a set of states that converges within $\ell$ 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 $S$ converge within $\ell$ steps.
5. Generate a set of point options with initiation states set to one of the center states in the solution of the set-cover, and termination states set to the goal.

We compute a distance function $d_s : S \times S \to \mathbb{N}$, defined as follows:

$$d_s(s, s') = \min \{d'_s(s, s'), d'_s(s, s') + d'(s, s')\} + 1.$$ 

More formally, let $d'_s(s_i)$ denote the number of iterations needed for the value of state $s_i$ to satisfy $|V(s_i) - V^*(s_i)| < \epsilon$, and let $d'_s(s_i, s_j)$ 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 $d_s(s_i, s_j) := \min \{d'_s(s_i, s_j), d'_s(s_i, s_j)\}$. For simplicity, we use $d$ to denote the function $d_s$. Consider the following example.

Example. Table 3b is a distance function for the MDP shown in Figure 3a. 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 $d$. $d_s(s, s')$ can be computed once you solved the MDP without any options and store all value functions $V_i$ ($i = 1, \ldots b$) until convergence as a function of $V_i$: $V_i(s) = f(V_i(s_0), V_i(s_1), \ldots).$ If we add a point option $\hat{\phi}$, $d_{\hat{\phi}}(s, s')$ satisfies the triangle inequality, but does not satisfy the symmetry and the indiscernibles.
Finding Options that Minimize Planning Time

Theorem 5. The set of elements for the set cover is the set of states is a set of two point options from set-cover optimization problem (algorithm finds the optimal solution if the subsets are given as: ample, included in a subset the example, set of nodes that can be solved within ` generate an instance of a set-cover optimization problem. Example. Consider the problem of finding a set of options V1(s) = V*(s'). Thus, d(s, s') is the smallest i where V(s) reaches c-optimal if we replace V1(s') with V*(s') when computing V(s) as a function of V1.

Example. We use the MDP shown in Figure 3a 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 elements 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 S \ X^+. Here, we denote a set of nodes that can be solved within ℓ steps by X^+_ℓ. In the example, U = S \ X^+_2 = {s1, s2, s3, s4}. A state s is included in a subset X_s \ X^+_2 iff d(s, s') ≤ ℓ − 1. For example, X_s1 = {s1}, X_s2 = {s1, s2}. Thus, the set of subsets are given as: X_s1 = {s1}, X_s2 = {s1, s2}, X_s3 = {s3}, X_s4 = {s3, s4}. In this case, the approximation algorithm finds the optimal solution C = {X_s2, X_s4} for the set-cover optimization problem (U, C). We generate a point option for each state in C. Thus, the output of the algorithm is a set of two point options from s2 and s4 to g.

Theorem 5. 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 O.
3. If the MDP is deterministic, the option set is at most O(log n) times larger than the smallest option set pos-

Proof. See the supplementary material.

Note that the approximation bound for a deterministic MDP will inherit any improvements to the approximation algorithm for set cover. Set cover is known to be NP-hard to approximate up to a factor of (1 − o(1)) log n (Dinur & 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-MOMI.

4.2. A-MIMO

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

1. Compute d_c(s, s') : S × S → N for each pair of states.
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.

As in A-MOMI, we first compute the function d. Then, we exploit this characteristic of d and solve the asymmetric k-center problem (Panigrahy & Vishwanathan, 1998) on (U, d, k) to get a set of center states, 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 U, a function d : U × U → N, and an integer k, return C that minimizes P(C) = max_{c ∈ C} d(s, c) subject to |C| ≤ 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)^4 where k < |U|. It is known that the problem cannot be solved within a factor of log^* |U| − Θ(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. Finally, 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 C, we generate a point option starting from c to the goal state g.

[^3]: The notation O(log n) is the number of times the logarithm function must be iteratively applied before the result is less than or equal to 1.
We ran the experiments on an domain. We evaluate the performance of the value-iteration algorithm using options generated by the 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. As an optimal set of options is not unique, we picked one arbitrarily. We are only able to find optimal solutions up to four options within 10 minutes, not unique, we picked one arbitrarily. 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. For eigenoptions, we ignored the eigenvector corresponding to the smallest eigenvalue ($\lambda_0 = 0$) in the graph Laplacian because it has a constant value for every state. Both betweenness options and eigenoptions are polynomial time algorithm, thus run in a few minutes. Figure 4 shows the optimal and bounded suboptimal set of options computed by A-MIMO. See the supplementary material for visualizations for the $9 \times 9$ grid domain.

**Theorem 6.** 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 $O(\log^* k)$.
3. The number of iterations to solve the MDP using the option set acquired is upper bounded by $P(\mathcal{O})$.

**Proof.** See the supplementary material. □

**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 $11 \times 11$ four-room domain and a $9 \times 9$ 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. As an optimal set of options is not unique, we picked one arbitrarily. 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. For eigenoptions, we ignored the eigenvector corresponding to the smallest eigenvalue ($\lambda_0 = 0$) in the graph Laplacian because it has a constant value for every state. Both betweenness options and eigenoptions are polynomial time algorithm, thus run in a few minutes. Figure 4 shows the optimal and bounded suboptimal set of options computed by A-MIMO. See the supplementary material for visualizations for the $9 \times 9$ grid domain.

Figure 4e shows the four bottleneck states with highest shortest-path betweenness centrality in the state-transition graph (Şimşek & 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.

**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 $9 \times 9$ grids using $k$ options. 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 is 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 (Iba, 1989; McGovern & Barto, 2001; Menache et al., 2002; Stolle & Precup, 2002; Şimşek & Barto, 2004; Şimşek & Barto, 2009; Konidaris & Barto, 2009; Machado et al., 2017; Eysenbach et al., 2019). For example, some investigate the use of bottleneck states (Stolle & Precup, 2002; Şimşek & Barto, 2009; Menache et al., 2002; Lehnert et al., 2018). Stolle & Precup (2002) proposed to set states with high visitation counts as subgoal states, which identifies bottleneck states in the four-room domain. Şimşek & Barto (2009) generalized the concept of a bottleneck to (shortest-path) betweenness of the graph to capture how pivotal the state is. Menache et al. (2002) used a learned model of the environment to run a Max-Flow/Min-Cut algorithm to the state-space graph to identify bottleneck states. These methods generate options that leverage the idea that subgoals are states visited most frequently. On the other hand, Şimşek & Barto (2004) proposed to generate options to encourage exploration by generating options to relatively novel states. Eysenbach et al. (2019) 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.

While empirical results show that these algorithms are useful in some scenarios, the conditions under which the methods are effective is often unclear because the relationship between the objective of the skill discovery algorithm and that of the agent is often not established. In fact, Jong et al. (2008) sought to investigate the utility of skills empirically and pointed out that introducing skills might worsen the learning performance. Harb et al. (2017) proposed to formulate good options to be options which minimize the deliberation costs in the bounded rationality framework (Simon, 1957). Brunskill & Li (2014) targeted the lifelong reinforcement learning setting and proposed an option generation method for lifelong reinforcement learning. They analyzed the sample complexity of RMAX using options and proposed an option discovery targeting to minimize the sample complexity. Solway et al. (2014) formalized an optimal behavioral hierarchy as a model which fits the behavior of the agent in tasks the best.

For planning, several works have shown empirically that adding a particular set of options or macro-operators can speed up planning algorithms (Francis & Ram, 1993; Sutton & Barto, 1998; Silver & Ciosek, 2012; Konidaris, 2016). Mann et al. (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 the value function is initialized pessimistically.

7. Conclusions

We considered two fundamental theoretical questions concerning the use of behavioral abstractions to solve MDPs: (1) minimize the size of option set given a maximum number of iterations (MOMI) and (2) minimize the number of iterations given a maximum size of option set (MIMO). We showed that both problems are computationally intractable, even for deterministic MDPs. For each problem, we produced a polynomial-time algorithm for MDPs with bounded reward and goal states, and with bounded optimality for deterministic MDPs. Although these algorithms are not practical for a single-task planning, we believe they may be a useful foundation for future option discovery methods. 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 to improve the inapproximability result for MOMI. We would like to thank Maehara Takanori and Kazuki Yoshizoe for their advice on formulating the problems. This work is supported in part by an AFOSR Young Investigator Grant to George Konidaris, under agreement number FA9550-17-1-0124. D Ellis Hershkowitz is supported in part by NSF grants CCF-1618280, CCF-1814603, CCF-1527110 and NSF CAREER award CCF-1750808.

References


Finding Options that Minimize Planning Time


Finding Options that Minimize Planning Time (Appendix)

Yuu Jinna 1  David Abel 1  D Ellis Hershkowitz 2  Michael L. Littman 1  George Konidaris 1

A. Appendix: Inapproximability of MOMI

In this section we prove Theorem 4:

Theorem 4.

1. MOMI is Ω(log n) hard to approximate even for deterministic MDPs unless P = NP.

2. MOMI gen is 2O(1−ε) n-hard to approximate for any ε > 0 even for deterministic MDPs unless NP ⊆ DTIME(n poly log n).

3. MOMI is 2O(1−ε) n-hard to approximate for any ε > 0 unless NP ⊆ DTIME(n poly log n).

For Theorems 4.2 and 4.3 we reduce our problem to the Min-Rep problem, originally defined by (Kortsarz, 2001). Min-Rep is a variant of the better studied label cover problem (Dinur & Safra, 2004) and has been integral to recent hardness of approximation results in network design problems (Dinitz et al., 2012; Bhattacharyya et al., 2012). Roughly, Min-Rep asks how to assign as few labels as possible to nodes in a bipartite graph such that every edge is “satisfied.”

Definition 1 (Min-Rep):

Given a bipartite graph G = (A ∪ B, E) and alphabets ΣA and ΣB for the left and right sides of G respectively. Each e ∈ E has associated with it a set of pairs πe ⊆ ΣA × ΣB which satisfy it. Return a pair of assignments γA : A → P(ΣA) and γB : B → P(ΣB) such that for every e = (Ai, Bj) ∈ E there exists an (a, b) ∈ πe such that a ∈ γA(Ai) and b ∈ γB(Bj). The objective is to minimize ∑a∈A |γA(Ai)| + ∑b∈B |γB(Bj)|.

We illustrate a feasible solution to an instance of Min-Rep in Figure 1.

The crucial property of Min-Rep we use is that no polynomial-time algorithm can approximate Min-Rep well. Let n = |A| + |B|.

Lemma 1 (Kortsarz 2001). Unless NP ⊆ DTIME(n poly log n), Min-Rep admits no 2O(1−ε) n polynomial-time approximation algorithm for any ε > 0.

As a technical note, we emphasize that all relevant quantities in Min-Rep are polynomially-bounded. In Min-Rep we have |ΣA||ΣB| ≤ nε for constant ε. It immediately follows that ∑e |πe| ≤ nε for constant ε.

A.1. Hardness of Approximation of MOMI with Deterministic MDP

Theorem 4.1 Proof. The optimization version of the set-cover problem cannot be approximated within a factor of c · ln n by a polynomial-time algorithm unless P = NP (Raz & Safra, 1997). The set-cover optimization problem can be reduced to MOMI with a similar construction for a reduction from SetCover-DEC to OI-DEC. Here, the targeted minimization values of the two problems are equal: P(⌜C⌟) = |⌜C⌟|, 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 · ln n approximation for MOMI where n is the number of states in the MDP. Let SetCover(U, 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 O where |⌜O⌟| ≤ c ln n |⌜O*⌟|, where ⌜O*⌟ is the optimal solution. We construct a solution for the set cover C from the solution to the MOMI O (see the construction in the proof of Theorem 1). Because |⌜C⌟| = |⌜O⌟| and |⌜C*⌟| = |⌜O*⌟|, where ⌜C*⌟ is the optimal solution for the set cover, we get |⌜C⌟| = |⌜O⌟| ≤ c ln n |⌜O*⌟| = c ln n |⌜C*⌟|. Thus, we acquire a c · 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 · ln n approximation for MOMI, unless P=NP.

A.2. Hardness of Approximation of MOMI gen

We now show our hardness of approximation of 2O(1−ε) n for MOMI gen, Theorem 4.2.1

We start by describing our reduction from an instance of Min-Rep to an instance of MOMI gen. The intuition behind our reduction is that we can encode choosing a label for a

---

1We assume that ⌜O*⌟ is a “good” set of options in the sense that there exists some set ⌜O⌟ ⊆ ⌜O*⌟ such that Lκ,Y (⌜O⌟) ≤ ℓ. We also assume, without loss of generality, that ε < 1 throughout this section; other values of ε can be handled by re-scaling rewards in our reduction.
vertex in Min-Rep as choosing an option in our \( \text{MOMI}_{gen} \) instance. In particular, we will have a state for each edge in our Min-Rep instance and reward will propagate quickly to that state when value iteration is run only if the options corresponding to a satisfying assignment for that edge are chosen.

More formally, our reduction is as follows. Consider an instance of Min-Rep, \( MR \), given by \( G = (A \cup B, E), \Sigma_A, \Sigma_B \) and \( \{ \pi_e \} \). Our instance of \( \text{MOMI}_{gen} \) is as follows where \( \gamma = 1 \) and \( l = 3 \):

- **State space** We have a single goal state \( S_g \) along with states \( S_e^g \) and \( S_e^{g'} \). For each edge \( e \) we create a state \( S_e \). Let \( \text{Sat}_A(e) \) consist of all \( a \in \Sigma_A \) such that \( a \) is in some assignment in \( \pi_e \). Define \( \text{Sat}_B(e) \) symmetrically. For each edge \( e \in E \) we create a set of \( 2 \cdot |\text{Sat}_A(e)| \) states, namely \( S_{ea} \) and \( S_{ea}' \) for every \( a \in \text{Sat}_A(e) \). We do the same for \( b \in \text{Sat}_B(e) \).

- **Actions and Transitions** We have a single action from \( S_e^g \) to \( S_e \), a single action from \( S_e^{g'} \) to \( S_{e'}^g \). For each edge \( e \) we have the following deterministic actions: Every \( S_{ea} \) has a single outgoing action to \( S_{ea} \) for \( a \in \text{Sat}_A(e) \); Every \( S_{eb} \) has a single outgoing action to \( S_{eb'} \) for \( b \in \text{Sat}_B(e) \); Every \( S_{ea} \) has an outgoing action to \( S_{eb} \) if \( (a, b) \in \pi_e \) and every \( S_{eb}' \) has a single outgoing action to \( S_{g'} \). Lastly, we have a single action from \( S_{ea}^{g'} \) to \( S_{g''} \) for every \( a \in \text{Sat}_A(e) \).

- **Reward** The reward of arriving in \( S_g \) is 1. The reward of arriving in every other state is 0.

- **Option Set** Our option set \( O' \) is as follows. For each vertex \( A_i \in A \) and each \( a \in \Sigma_A \) we have an option \( O(A_i, a) \): The initiation set of this option is \( S_e \) where \( e \) is incident to \( A_i \); The termination set of this option is \( S_{e'} \).

---

Footnote 2: It is easy to generalize these results to \( l \geq 4 \) by replacing certain edges with paths.

---

Figure 2: Our \( \text{MOMI}_{gen} \) reduction applied to the Min-Rep problem in Figure 1. \( e_1 = (A_1, B_1), e_2 = (A_1, B_2), e_3 = (A_2, B_2) \). Actions given in solid lines and each option in \( O' \) represented in its own color as a dashed line from initiation to termination states. Notice that a single option goes from \( S_{eb1} \) and \( S_{eb1} \) to \( S_g \).

One should think of choosing option \( O(v, x) \) as corresponding to choosing label \( x \) for vertex \( v \) in the input Min-Rep instance. Let \( \text{MOMI}_{gen} \) be the MDP output given instance MR of Min-Rep and see Figure 3 for an illustration of our reduction.

Let \( \text{OPT}_{\text{MOMI}_{gen}} \) be the value of the optimal solution to \( \text{MOMI}_{gen} \) (MR) and let \( \text{OPT}_{MR} \) be the value of the optimal Min-Rep solution to MR. The following lemmas demonstrates the correspondence between a \( \text{MOMI}_{gen} \) and Min-Rep solution.

**Lemma 2.** \( \text{OPT}_{\text{MOMI}_{gen}} \leq \text{OPT}_{MR} \)

*Proof.* Given a solution \( (\gamma_A, \gamma_B) \) to MR, define \( O_{\gamma_A, \gamma_B} := \{ O(v, x) : v \in V(G) \land (\gamma_A(v) = x \lor \gamma_B(v) = x) \} \) as the corresponding set of options. Let \( \gamma_A' \) and \( \gamma_B' \) be the optimal solutions to MR which is of cost \( \text{OPT}_{MR} \).

We now argue that \( O_{\gamma_A', \gamma_B'} \) is a feasible solution to \( \text{MOMI}_{gen} \) (MR) of cost \( \text{OPT}_{MR} \), demonstrating that the op-
Finding Options that Minimize Planning Time (Appendix)

We now show that a solution to $\text{MOMI}_{\text{gen}}(\text{MR})$ has cost at most $\text{OPT}_{\text{MR}}$. To see this notice that by construction the $\text{MOMI}_{\text{gen}}$ cost of $O_{\gamma_A^* \gamma_B^*}$ is exactly the Min-Rep cost of $(\gamma_A^* , \gamma_B^*)$.

We need only argue, then, that $O_{\gamma_A^* \gamma_B^*}$ is feasible for $\text{MOMI}_{\text{gen}}(\text{MR})$ and do so now. The value of every state in $\text{MOMI}_{\text{gen}}(\text{MR})$ is 1. Thus, we must guarantee that after 3 iterations of value iteration, every state has value 1. However, without any options every state except each $S_e$ has value 1 after 3 iterations of value iteration. Thus, it suffices to argue that $O_{\gamma_A^* \gamma_B^*}$ guarantees that every $S_e$ will have value 1 after 3 iterations of value iteration. Since $(\gamma_A^* , \gamma_B^*)$ is a feasible solution to MR we know that for every $e = (A_i, B_j)$ there exists an $\bar{a} \in \gamma_A^*(A_i)$ and $\bar{b} \in \gamma_B^*(B_j)$ such that $(\bar{a}, \bar{b}) \in \pi_c$; correspondingly there are options $O(A_i, \bar{a}), O(B_j, \bar{b}) \in O_{\gamma_A^* \gamma_B^*}$. It follows that, given options $O_{\gamma_A^* \gamma_B^*}$ from, $S_e$ one can take option $O(A_i, \bar{a})$ then the action from $S_a$ to $S_{eb}$ and then option $O(B_j, \bar{b})$ to arrive in $S_g$; thus, after 3 iterations of value iteration the value of $S_e$ is 1. Thus, we conclude that after 3 iterations of value iteration every state has converged on its value.

We now show that a solution to $\text{MOMI}_{\text{gen}}(\text{MR})$ corresponds to a solution to MR. For the remainder of this section $\gamma_A^*(A_i) := \{a: O(A_i, a) \in O\}$ and $\gamma_B^*(B_j) := \{b: O(B_j, b) \in O\}$ is the Min-Rep solution corresponding to option set $O$.

**Lemma 3.** For a feasible solution to $\text{MOMI}_{\text{gen}}(\text{MR})$, $O$, we have $(\gamma_A^O, \gamma_B^O)$ is a feasible solution to MR of cost $|O|$.

**Proof.** Notice that by construction the Min-Rep cost of $(\gamma_A^O, \gamma_B^O)$ is exactly $|O|$. Thus, we need only prove that $(\gamma_A^O, \gamma_B^O)$ is a feasible solution for MR.

We do so now. Consider an arbitrary edge $e = (A_i, B_j) \in E$; we wish to show that $(\gamma_A^O, \gamma_B^O)$ satisfies $e$. Since $O$ is a feasible solution to $\text{MOMI}_{\text{gen}}(\text{MR})$ we know that after 3 iterations of value iteration every state must converge on its value. Moreover, notice that the value of every state in $\text{MOMI}_{\text{gen}}(\text{MR})$ is 1. Thus, it must be the case that for every $S_e$, there exists a path of length 3 from $S_e$ to $S_g$ using either options or actions. The only such paths are those that take an option $O(A_i, a)$, then an action from $S_{ea}$ to $S_{eb}$, then option $O(B_j, b)$ where $(a, b) \in \pi_c$. It follows that $a \in \gamma_A^O(A_i)$ and $b \in \gamma_B^O(B_j)$. But since $(a, b) \in \pi_c$, we then know that $e$ is satisfied. Thus, every edge is satisfied and so $(\gamma_A^O, \gamma_B^O)$ is a feasible solution to MR.

**Theorem 4.2 Proof.** Assume $\text{NP} \not\subseteq \text{DTIME}(n^{\text{poly} \log n})$ and for the sake of contradiction that there exists an $\varepsilon > 0$ for which polynomial-time algorithm $A_{\text{MOMI}_{\text{gen}}}$ can $2^{\log^{1-\varepsilon} n}$ approximate $\text{MOMI}_{\text{gen}}$. We use $A_{\text{MOMI}_{\text{gen}}}$ to $2^{\log^{1-\varepsilon} n}$ approximate Min-Rep for a fixed constant $\varepsilon' > 0$ in polynomial-time, thereby contradicting Lemma 1. Again, $n$ is the number of vertices in the graph of the Min-Rep instance.

We begin by noting that the relevant quantities in $\text{MOMI}_{\text{gen}}(\text{MR})$ are polynomially-bounded. Notice that the number of states $n$ in the MDP in $\text{MOMI}_{\text{gen}}(\text{MR})$ is at most $O(n^2|\Sigma_A||\Sigma_B|) = \tilde{n}$ for some fixed constant $c$ by the aforementioned assumption that $\Sigma_A$ and $\Sigma_B$ are polynomially-bounded in $n$.

Our polynomial-time approximation algorithm to approximate instance MR of Min-Rep is as follows: Run $A_{\text{MOMI}_{\text{gen}}}$ on $\text{MOMI}_{\text{gen}}(\text{MR})$ to get back option set $O$. Return $(\gamma_A^O, \gamma_B^O)$ as defined above as our solution to MR.

We first argue that our algorithm is a $2^{\log^{1-\varepsilon} \tilde{n}}$ approximation for Min-Rep for some $\varepsilon' > 0$. Applying Lemma 3, the approximation of $A_{\text{MOMI}_{\text{gen}}}$ and then Lemma 2, we have that $(\gamma_A^O, \gamma_B^O)$ is a feasible solution for MR with cost

$$\text{COST}_{\text{Min-Rep}}(\gamma_A^O, \gamma_B^O) = |O| \leq 2^{\log^{1-\varepsilon} \tilde{n}} \text{OPT}_{\text{MOMI}_{\text{gen}}} \leq 2^{\log^{1-\varepsilon} \tilde{n}} \text{OPT}_{\text{MR}}$$

Thus, $(\gamma_A^O, \gamma_B^O)$ is a $2^{\log^{1-\varepsilon} n}$ approximation for the optimal Min-Rep solution where $n$ is the number of states in the MDP of $\text{MOMI}_{\text{gen}}(\text{MR})$. Now recalling that $n \leq \tilde{n}$ for fixed constant $c$. We therefore have that $(\gamma_A^O, \gamma_B^O)$ is a $2^{\log^{1-\varepsilon} \tilde{n}} = 2^{\varepsilon' \log^{1-\varepsilon} \tilde{n}} \leq \varepsilon' \cdot 2^{\log^{1-\varepsilon} \tilde{n}} \approx \varepsilon' \cdot 2^{\log^{1-\varepsilon} \tilde{n}}$ approximation for a constant $\varepsilon'$. Choosing $\varepsilon$ sufficiently small, we have that $\varepsilon' \cdot 2^{\log^{1-\varepsilon} \tilde{n}} \approx 2^{\log^{1-\varepsilon} \tilde{n}}$ for sufficiently large $\tilde{n}$.

Thus, our polynomial-time algorithm is a $2^{\log^{1-\varepsilon} \tilde{n}}$ approximation for Min-Rep for $\varepsilon' > 0$, thereby contradicting Lemma 1. We conclude that $\text{MOMI}_{\text{gen}}$ cannot be $2^{\log^{1-\varepsilon} n}$-approximated.

**A.3. Hardness of Approximation of MOMI with Stochastic MDP**

We now show our hardness of approximation of $2^{\log^{1-\varepsilon} n}$ for MOMI. **Theorem 4.3.** We will notably use the stochasticity of $\text{MOMI}_{\text{gen}}$.
We will exploit the fact that the value of a state in an MDP
We begin by describing our reduction from an instance of
we will use the fact that in a stochastic MDP the value of a
we will have vertex for each edge in our Min-Rep instance
To solve this issue we critically use stochasticity. Whether
every
this section; rewards in our reduction can be re-scaled to handle
More formally, our reduction is as follows. Consider in-
and
or not a given edge in a Min-Rep is satisfied is an or of
that we can no longer constrain a solution to choose options
exactly corresponding to a feasible Min-Rep solution.

To solve this issue we critically use stochasticity. Whether
or not a given edge in a Min-Rep is satisfied is an or of
ands: A fixed edge is satisfied when “or” in Min-Rep and
we will use the fact that in a stochastic MDP the value of a
(state, action) pair is the sum over states to encode the “and”
in Min-Rep.

More formally, our reduction is as follows. Consider in-
stance MR of Min-Rep given by \( G = (A \cup B, E), \Sigma_A, \Sigma_B \)
and \{\pi_e\}. Our instance of MOMI is as follows where \( \gamma = 1 \)
and \( l = 2 \).

- **State space** We have a goal state \( S_i \) for each \( A_i \in A \).
  Again, let \( \text{Sat}_A(e) \) consist of all \( a \in \Sigma_A \) such that \( a \)
is in some assignment in \( \pi_e \). For each \( A_i \in A \) and
  \( a \in \text{Sat}_A(e) \) we will add to our MDP states \( S_{ia} \)
and \( S_{ia}' \). We symmetrically do the same for all states
in \( \Sigma_B \). For each \( e \in E \) we will also add a state \( S_e \).

- **Actions and Transitions** Every \( S_{ia} \) state has a single
  action to \( S_{ia}' \) and every \( S_{ia}' \) state has a single action
  to \( S_i \). The same symmetrically holds for states
  from a \( B_j \in B \). Every \( S_e \) for \( e = (A_i, B_j) \) has
  \(|\pi_{(A_i, B_j)}|\) actions associated with it, namely \{\( \alpha_{(a, b)} \)\}
  where \( (a, b) \in \pi_{(A_i, B_j)} \). Action \( \alpha_{(a, b)} \) has a probability
  .5 of transitioning to state \( S_{ia} \) and a probability .5
  of transitioning to state \( S_{ib} \).

- **Reward** The reward of arriving in any \( S_i \) or \( S_j \) for
  \( A_i \in A \) or \( B_j \in B \) is 1 and 0 for every other state.

\footnote{We may assume without loss of generality \( \varepsilon < .5 \) throughout
this section; rewards in our reduction can be re-scaled to handle
larger \( \varepsilon \).}

\footnote{It is easy to generalize these results to \( l \geq 3 \) by replacing
edges with paths.}

\footnote{It is not hard to see that this construction can be modified so
that we have only a single goal state if need be; we need only set
every \( S_i \) and \( S_j \) to be the same state. We assume multiple goal
states for ease of exposition.}

Notice that no point options have \( S_e \) as an initialization
state since any such option would have a .5 probability
of never terminating (and we assume our options always
terminate). See Figure 3 for an illustration of our reduction.
One should think of choosing a point option from \( S_{ia} \)
to \( S_i \) as corresponding to choosing label \( a \) for \( A_i \) in the input
Min-Rep instance. The same holds for label \( b \) for \( B_j \)
and choosing a point option from \( S_{ib} \) to \( S_j \). Let \( \text{MOMI}(MR) \)
be the MOMI instance output by our reduction given instance
MR of Min-Rep.

We now demonstrate that our reduction allows us to show
that MOMI cannot be \( 2\log \varepsilon^{-\varepsilon} \varepsilon \)-approximated for any \( \varepsilon > 0 \).
Let \( \text{OPT}_{\text{MOMI}} \) be the value of the optimal solution to
MOMI(MR) and let \( \text{OPT}_{\text{MR}} \) be the value of the optimal
Min-Rep solution to MR. The following lemmas demonstrates the correspondence between a MOMI and Min-Rep
solution.

**Lemma 4.** \( \text{OPT}_{\text{MOMI}} \leq \text{OPT}_{\text{MR}} \)

**Proof.** Our proof translates between point options in our
reduction and assignments in the input Min-Rep instance in
the natural way. Given a solution \((\gamma_A, \gamma_B)\) to MR, define
\( O_{\gamma_A, \gamma_B} \) as consisting of all point options from \( S_{ia} \)
to \( S_i \) if \( a \in \gamma_A(A_i) \) and all points options from \( S_{ib} \)
to \( S_j \) if \( b \in \gamma_B(B_j) \). Let \( \gamma_A^* \) and \( \gamma_B^* \)
be the optimal solutions to MR which is of cost \( \text{OPT}_{\text{MR}} \).

We claim that \( O_{\gamma_A^*, \gamma_B^*} \) is a feasible solution to MOMI(MR)
of cost \( \text{OPT}_{\text{MR}} \), demonstrating that the optimal solution to
MOMI(MR) has cost at most \( \text{OPT}_{\text{MR}} \). To see this notice
that by construction the MOMI cost of \( O_{\gamma_A^*, \gamma_B^*} \) is exactly

\[ \text{OPT}_{\text{MOMI}} \leq \text{OPT}_{\text{MR}} \]
the Min-Rep cost of $\gamma_A \cdot \gamma_B$.

We need only argue, then, that $O_{\gamma_A \cdot \gamma_B}$ is feasible for MOMI(MR) and do so now. Notice that the value of every state in MOMI is 1. Thus, we must guarantee that after 2 iterations of value iteration, every state has value 1. However, without any options every state except for $S_e$ where $e \in E$ has value 1 after 2 iterations of value iteration. Thus, it suffices to argue that $O_{\gamma_A \cdot \gamma_B}$ guarantees that every $S_e$ will have value 1 after 2 iterations of value iteration. Since $(\gamma_A \cdot \gamma_B)$ is a feasible solution to MR we know that for every $e = (A_i, B_j)$ there exists $\bar{a} \in \gamma_A(A_i)$ and $\bar{b} \in \gamma_B(B_j)$ such that $(\bar{a}, \bar{b}) \in \pi_e$; correspondingly there is some action from $S_e$ with a $\frac{1}{2}$ probability of resulting in state $S_{\bar{a}}$ and a $\frac{1}{2}$ probability of resulting in state $S_{\bar{b}}$ where $O_{\gamma_A \cdot \gamma_B}$ has a point option from $S_{\bar{a}}$ to $S_i$ and a point option from $S_{\bar{b}}$ to $S_j$. That is, $V_1(S_{\bar{a}}) = 1$ and $V_1(S_{\bar{b}}) = 1$. Thus, after one iteration of value iteration the values of $S_{\bar{a}}$ and $S_{\bar{b}}$ are both 1 and so after two iterations of value iteration the value of $S_e$ is

$$V_2(S_e) = \max_{\alpha(a,b)} .5 \cdot (V_1(S_{\bar{a}})) + .5 \cdot (V_1(S_{\bar{b}})) \geq .5 \cdot (V_1(S_{\bar{a}})) + .5 \cdot (V_1(S_{\bar{b}})) = 1.$$

Thus, $V_2(S_e) = 1$ for every $S_e$ and so we conclude that after two iterations of value iteration every state has converged on its value.

We now show that a solution to MOMI(MR) corresponds to a solution to MR. For this reason of this section let $\gamma_A(A_i) = \{a : O(S_{ja}, S_j) \in O\}$ and $\gamma_B(B_j) = \{b : O(S_{jb}, S_j) \in O\}$ where for the remainder of this section $O(S, S')$ stands for a point option with initiation state $S$ and termination state $S'$.

**Lemma 5.** For any feasible solution $O$ to MOMI(MR) we have $(\gamma_A \cdot \gamma_B)$ is a feasible solution to MR of cost $|O|$.

**Proof.** Notice that by construction the Min-Rep cost of $(\gamma_A \cdot \gamma_B)$ is exactly $|O|$. Thus, we need only prove that $(\gamma_A \cdot \gamma_B)$ is a feasible solution for MR.

We do so now. Consider an arbitrary edge $e = (A_i, B_j) \in E$; we wish to show that $(\gamma_A \cdot \gamma_B)$ satisfies $e$. Since $O$ is a feasible solution we know that after two iterations of value iteration every state must converge on its value (up to an $\epsilon$ factor which we can ignore by our above assumption that $\epsilon < .5$). Moreover, notice that the value of every state in MOMI(MR) is 1. Thus, we must be the case that for every $S_e$ we have $V_2(S_e) = 1$ for $e = (A_i, B_j)$. It follows, then, that there is some action $\alpha(a,b)$ where $(a, b) \in \pi_e$ such that

$$1 = V_2(S_e) = .5 \cdot (V_1(S_{\bar{a}})) + .5 \cdot (V_1(S_{\bar{b}})).$$

Since the value of every state is at most 1, it follows that $V_1(S_{\bar{a}}) = V_1(S_{\bar{b}}) = 1$. However, since $V_1(S_{\bar{a}})$ and $V_1(S_{\bar{b}})$ are both two hops from the only goal reachable from them ($S_i$ and $S_j$ respectively) it must be the case that there is some point option from $S_{\bar{a}}$ to $S_i$ and $S_{\bar{b}}$ to $S_j$. Thus, by definition of $(\gamma_A \cdot \gamma_B)$ we then have $\bar{a} \in \gamma_A$ and $\bar{b} \in \gamma_B$. Since $(\bar{a}, \bar{b}) \in \pi_e$ it follows that arbitrary edge $e = (A_i, B_j)$ is satisfied. Thus, every edge in $E$ is satisfied and so $(\gamma_A \cdot \gamma_B)$ is a feasible solution for MR.

Finally, we conclude the hardness of approximation of MOMI.

**Theorem 4.3 Proof.** Assume NP $\nsubseteq$ DTIME$(\mu n \log n)$ and for the sake of contradiction that there exists an $\epsilon > 0$ for which a polynomial-time algorithm $A_{MOMI}$ can $2^{\log^{-\epsilon} n}$-approximate MOMI. We use $A_{MOMI}$ to $2^{\log^{-\epsilon'} n}$-approximate Min-Rep for a fixed constant $\epsilon' > 0$ in polynomial-time in $n$, thereby contradicting Lemma 1. Again, $n$ is the number of vertices in the graph of the Min-Rep instance.

We begin by noting that the relevant quantities in MOMI(MR) are polynomially-bounded. Let $\bar{n} := |A| + |B|$ be the number of vertices in our MR instance. Notice that the number of states in the MDP, $n$, in our MOMI(MR) instance is at most $O(\bar{n}^2)|\Sigma_A| + |B| |\Sigma_B| + |E|) \leq \bar{n}^2$ for some fixed constant $c$ by the aforementioned assumption that $\Sigma_A$ and $\Sigma_B$ are polynomially-bounded in $\bar{n}$.

Our polynomial-time approximation algorithm to approximate instance MR of Min-Rep is as follows: Run $A_{MOMI}$ on MOMI(MR) to get back option set $O$. Return $(\gamma_A \cdot \gamma_B)$ as defined above as our solution to MR.

We first argue that our algorithm is polynomial time in $\bar{n}$. For each vertex in MR, we create a polynomial number of states and actions. Thus, the number of states in MOMI(MR) is polynomially-bounded in $\bar{n}$ and so $A_{MOMI}$ runs in time polynomial in $\bar{n}$. A polynomial runtime of our algorithm immediately follows.

We now argue that our algorithm is a $2^{\log^{-\epsilon} \bar{n}}$-approximation for Min-Rep for some $\epsilon' > 0$. Applying Lemma 5, the approximation of $A_{MOMI}$ and then Lemma 4, we have that the Min-Rep cost of $(\gamma_A \cdot \gamma_B)$ is

$$\text{cost}_{\text{Min-Rep}}(\gamma_A \cdot \gamma_B) = |O| \leq 2^{\log^{-\epsilon} \bar{n}} \text{OPT}_{\text{MOMI}} \leq 2^{\log^{-\epsilon} \bar{n}} \text{OPT}_{\text{MR}}.$$

Thus, $(\gamma_A \cdot \gamma_B)$ is a $2^{\log^{-\epsilon} \bar{n}}$-approximation for the optimal.\footnote{It is worth noting, also, that since we create at most $\sum |\pi_e|$ actions for any state, the number of total actions in our MDP is at most polynomial in $\bar{n}$.}
We need to solve MDPs at most $(2)$.

The approximation algorithm we deploy for solving MDPs takes polynomial time. To compute the optimal policy, we enumerate all possible options with its terminal state set to the goal state of the MDP.

Theorem 6. 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 $O(\log^* k)$.
3. The number of iterations to solve the MDP using the acquired options is upper bounded by $P(C)$.

Theorem 6.1. 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 $|S|$ 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 sets of options of size $r = \lceil \log k \rceil$ (if $|O| + \log k > k$ then we set $r = k - |O|$) and add a set of options which minimizes $\ell$ (breaking ties randomly) to the option set $O$. We repeat this procedure until $|O| = k$. This procedure runs in polynomial time. The number of possible option set of size $r$ is $C_n = O(n^r) = O(k)$. 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.

A.4. A-MIMO

In this subsection we show the following theorem (we show Theorem 5 later):

Theorem 6. 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 $O(\log^* k)$.
3. The number of iterations to solve the MDP using the acquired options is upper bounded by $P(C)$.

Theorem 6.1. 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 $|S|$ 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 sets of options of size $r = \lceil \log k \rceil$ (if $|O| + \log k > k$ then we set $r = k - |O|$) and add a set of options which minimizes $\ell$ (breaking ties randomly) to the option set $O$. We repeat this procedure until $|O| = k$. This procedure runs in polynomial time. The number of possible option set of size $r$ is $C_n = O(n^r) = O(k)$. 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 6. 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 $O$. 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 $O'$ with $L_{L, V_0}(O') \leq L_{L, V_0}(O)$.

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

Lemma 7. $d(s, C) \leq \min_{s' \in C} d(s, s')$.

Using this, we show the following proposition.

Theorem 6.2. The number of iterations to solve the MDP using the acquired options is upper bounded by $P(C)$.

Proof. $P(C) = \max_{s \in S} \min_{s' \in C} d(s, s) \geq \max_{s \in S} d(s, C) = L_{L, V_0}(O)$ (using Equation 1). Thus $P(C)$ is an upper bound for $L_{L, V_0}(O)$.

The reason why $P(C)$ 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 4 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 1 holds if the MDP is deterministic. That is, $d(s, C) = \min_{s' \in C} d(s, s')$ for deterministic MDP.

Theorem 6.3. If the MDP is deterministic, it has a bounded suboptimality of $O(\log^* k)$. 
we picked a point option from without options: which satisfies with the approximation algorithm, we get a solution where Thus, the asymmetric instance of asymmetric MIMO. We convert this instance to an asymmetric MDP. From Archer, (2001). Let MIMO put\( L \leq \) all. No single point option does not improve in optimal option set is denoted by \( tions: \)

\[
\begin{align*}
L_{\epsilon,V_0}(\emptyset) &= \min_{\epsilon > 0} L_{\epsilon,V_0}(\emptyset) \\
\end{align*}
\]

Proof. First we show \( P(C^*) = L_{\epsilon,V_0}(O^*) \) for deterministic MDP. From \( d(s,C) = \min_{s' \in C} d(s,s') \), \( P(C^*) = \max_{s \in S} \min_{c \in C} d(s,c) = \max_{s \in S} d(s,C^*) = L_{\epsilon,V_0}(O^*) \).

The asymmetric \( k \)-center solver guarantees that the output \( C \) satisfies \( P(C) \leq c(\log^* k + O(1))P(C^*) \) where \( n \) is the number of nodes (Archer, 2001). Let MIMO \((M, \epsilon, k) \) be an instance of MIMO. We convert this instance to an instance of asymmetric \( k \)-center AsymKCenter(\( U, d, k \)), where \( |U| = |S| \). By solving the asymmetric \( k \)-center with the approximation algorithm, we get a solution \( C \) which satisfies \( P(C) \leq c(\log^* k + O(1))P(C^*) \). Thus, the output of the algorithm \( O \) satisfies \( L_{\epsilon,V_0}(O) = P(C) \leq c(\log^* k + O(1))P(C^*) = c(\log^* k + O(1))L_{\epsilon,V_0}(O^*) \).

Thus, \( L_{\epsilon,V_0}(O) \leq c(\log^* k + O(1))L_{\epsilon,V_0}(O^*) \) is derived.

**Proposition 1 (Greedy Strategy).** Let an option set \( O \) be a set of point option constructed by greedily adding one point option which minimizes the number of iterations. An improvement \( L_{\epsilon,V_0}(\emptyset) - L_{\epsilon,V_0}(O) \) 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 5. The MDP can be solved within 4 iterations without options: \( L_{\epsilon,V_0}(\emptyset) = 4 \). With an optimal option set of size \( k = 2 \) the MDP can be solved within 2 iterations: \( L_{\epsilon,V_0}(O^*) = 2 \) (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 \( O \) which has \( L_{\epsilon,V_0}(\emptyset) - L_{\epsilon,V_0}(O) = 0 \). Therefore, \( (L_{\epsilon,V_0}(\emptyset) - L_{\epsilon,V_0}(O^*) \) can be arbitrary small non-negative value (i.e. 0).

**A.5. A-MOMI**

In this subsection we show the following theorem:

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

1. **A-MOMI runs in polynomial time.**

2. It guarantees that the MDP is solved within \( \ell \) iterations using the option set acquired by A-MOMI \( O \).

3. If the MDP is deterministic, the option set is at most \( O(\log n) \) times larger than the smallest option set possible to solve the MDP within \( \ell \) iterations.

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

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

1. Solving an MDP takes polynomial time (Littman et al., 1995). To compute \( d \) we need to solve MDPs at most \( |S| \) times. Thus, it runs in polynomial time.

2. (2), (3), and (5) Immediate.

**Theorem 5.2.** A-MOMI guarantees that the MDP is solved within \( \ell \) iterations using the option set \( O \).

Proof. A state \( s \in X_g^+ \) reaches optimal within \( \ell \) steps by definition. For every state \( s \in S \setminus X_g^+ \), the set cover guarantees that we have \( X_g^+ \) 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 \( \epsilon \)-optimal value within \( d(s,s') + 1 \leq \ell \). Therefore, every state reaches \( \epsilon \)-optimal value within \( \ell \) steps.

**Theorem 5.3.** If the MDP is deterministic, the option set is at most \( O(\log n) \) times larger than the smallest option set possible to solve the MDP within \( \ell \) iterations.
Proof. Using a suboptimal algorithm by Chvatal (1979) we get $C$ such that $|C| \leq O(\log n)|C^*|$. Thus, $|O| = |C| \leq O(\log n)|C^*| = O(\log n)|O^*|$. 

Appendix: Experiments

We show the figures for experiments. Figure 6 shows the options found by solving MIMO optimally/suboptimally in four room domain. Figure 7 shows the options in 9x9 grid domain.

References


Figure 6: 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.

Figure 7: Comparison of the optimal point options for planning vs. bottleneck options proposed for reinforcement learning in the four room domain. Initiating conditions are shown in blue, the goal in green.