**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 con- verges in less than a given maximum of value- iteration passes. We first show that the problem is *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 approxi- mation 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 including the clas- sic four-rooms problem.

arXiv:1810.07311v3 [cs.AI] 16 Mar 2019

# 1. Introduction

Markov Decision Processes or MDPs (Puterman, 1994) are an expressive yet simple model of sequential decision- making environments. However, MDPs are computation- ally expensive to solve (Papadimitriou & Tsitsiklis, 1987; Littman, 1997; Goldsmith et al., 1997). One approach to solving such problems is to add high-level, temporally ex- tended actions—often formalized as options (Sutton et al., 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 appro- priately, planning algorithms can find good plans with less computation.

Indeed, previous work has offered substantial support that abstract actions can accelerate planning (Mann & Man- nor, 2014). However, little is known about how to find the right set of options to use for planning. Prior work of- ten seeks to codify an intuitive notion of what underlies an effective option, such as identifying relatively novel states (S¸ ims¸ek & Barto, 2004), identifying bottleneck states or high-betweenness states (S¸ ims¸ek et al., 2005; S¸ ims¸ek & Barto, 2009; Bacon, 2013; Moradi et al., 2012), finding re- peated policy fragments (Pickett & Barto, 2002), or finding

1Brown University, Providence, RI, United States 2Carnegie

states that often occur on successful trajectories (McGov- ern & 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 pre- cise performance-related metric*; consequently, their relative performance can only be evaluated empirically.

We aim to formalize what it means to find the set of op- tions that is optimal for planning, and to use the resulting formalization to develop an approximation algorithm with a principled theoretical foundation. Specifically, we consider the problem of finding the smallest set of options so that planning converges in fewer than a given maximum of *A* iterations of the planning algorithm, value iteration (VI). Our main result shows that this problem is *NP*-hard. More precisely, the problem:

1. is 2log1*−g n*-hard to approximate for any *s >* 0 unless

*NP ⊆ DTIME*(*n*poly log *n*)1, where *n* is the input size;

1. is Ω(log *n*)-hard to approximate even for deterministic MDPs unless *P* = *NP*;
2. has a *O*(*n*)-approximation algorithm;
3. has a *O*(log *n*)-approximation algorithm for determin- istic MDPs.

In Section 4, we show A-MOMI, a polynomial-time approx- imation algorithm that has *O*(*n*) suboptimality in general and *O*(log *n*) suboptimality for deterministic MDPs. The expression 2log1*−g n* is only slightly smaller than *n*: if *s* = 0

then Ω(2log *n*) = Ω(*n*). Thus, the inapproximability results

claim that A-MOMI is close to the best possible approxi-

mation 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 *NP*-hard, even for a deterministic MDP.

Finally, we empirically evaluate the performance of two heuristic approaches for option discovery, betweenness op- tions (S¸ ims¸ek & Barto, 2009) and Eigenoptions (Machado

Mellon University, Pittsburgh, PA, United States. Correspondence

to: Yuu Jinnai *<*yuu [jinnai@brown.edu](mailto:jinnai@brown.edu)*>*.

1This is a standard complexity assumption: See, for example, Dinitz et al. (2012)

et al., 2017), against the proposed approximation algorithms and the optimal options in standard grid domains.

# Background

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

## Markov Decision Processes and Planning

An MDP is a five tuple: *, , R, T, γ* , where is a finite set of states; is a finite set of actions; *R* :

*A S × A →*

*(S A ) S*

[0*,* RMAX] is a reward function; *T* : Pr( ) is a transition function, denoting the probability of arriving in

*S × A → S*

state *sj* after executing action *a* in state *s* ; and *γ* [0*,* 1] is a discount factor, expressing the agent’s

*∈*

*∈ S ∈ A ∈ S*

preference for immediate over delayed rewards.

An action-selection strategy is modeled by a *policy*, *π* :

**Definition 2** (option): *An option o is defined by a triple:* (*I, π, β*) *where:*

* *is a set of states where the option can initiate,*

*I ⊆ S*

* *π* : *S →* Pr(*A*) *is a policy,*
* *β* : *S →* [0*,* 1]*, is a termination condition.*

*We let Oall denote the set containing all options.*

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

*∞*

Σ *t*

*Tγ*(*s, o, sj*) = *γ* Pr(*st* = *sj, β*(*st*) *| s, o*)*.* (2)

*t*=0

Σ

Pr( ), mapping states to a distribution over actions. Typically, the goal of planning in an MDP is to *solve* the

*S → A*

*Rγ*(*s, o*) = E *r*1

*oπ*

+ *γr*2

+ *. . .* + *γk−*1*rk*

. *s, o*Σ *.* (3)

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*:

We use the multi-time model for value iteration. The algo- rithm computes a sequence of functions *V*0*, V*1*, ..., Vb* using the Bellman optimality operator on the multi-time model:

*V π*(*s*) = *R*(*s, π*(*s*)) + *γ T* (*s, π*(*s*)*, sj*)*V π*(*sj*)*.* (1)

Σ

*st∈S*

*Vi*+1(*s*) = max

*o∈A∪O*

.*Rγ*(*s, o*) +

*s*Σ*t∈S*

*Tγ*(*s, o, sj*)*Vi*(*sj*)Σ *.*

We denote *π∗*(*s*) = arg max*π V π*(*s*) and *V ∗*(*s*) = max*π V π*(*s*) as the optimal policy and value function, re-

spectively.

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 we denote the *value- planning problem*:

*value s,* **return** *a value function, V such that |V* (*s*) *−*

*V ∗*(*s*)*| < s for all s ∈ S.*

**Definition 1** (Value-Planning Problem): **Given** *an MDP M* = *(S, A, R, T, γ) and a non-negative real-*

(4)

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:2

*|Vbj*(*s*) *− V ∗*(*s*)*| < s for all s ∈ S, bj ≥ b.*

*O a non-empty set of options, is the smallest b at which*

*Ls,V*0 (*O*) *of VI using the joint action set A ∪ O, with*

**Definition 3** (*Ls,V*0 (*O*)): *The number of iterations*

The value-planning problem can be solved in time polyno- mial in the size of the state space.

## Options and Value Iteration

Temporally extended actions offer great potential for mitigat- ing the difficulty of solving complex MDPs, either through planning or reinforcement learning (Sutton et al., 1999). Indeed, it is possible that options that are useful for learn- ing 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 et al., 1999):

2.2.1. POINT OPTIONS.

The options formalism is immensely general. Due to its generality, a single option can actually encode several com- pletely unrelated sets of different behaviors. Consider the nine-state example MDP pictured in Figure 1; a single op- tion 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- pendent 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

2We can ensure *V ∗*(*s*) *Vi*(*s*) *< g* by running VI until

*| − | − ∈ S*

*| − |*

*Vi*+1(*s*) *Vi*(*s*) *< g*(1 *γ*)*/*2*γ* for all *s* (Williams &

Baird, 1993).

*s1*

*s2*

*s3*

*s4*

*s5*

*s6*

*s8*

*s9*

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.

*s7*

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 the most? 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:

**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 ∈ S* : *I*(*s*) = 1*}|* = 1*,* (5)

*|{s ∈ S* : *β*(*s*) *>* 0*}|* = 1*,* (6)

*|{s ∈ S* : *β*(*s*) = 1*}|* = 1*.* (7)

*We let Op denote the set containing all point options.*

For simplicity, we denote the initiation state as *o* and the termination state as *βo* for a point option *o*.

*I*

To plan with a point option from state *s*, the agent runs value iteration using a model *Q*(*s, o*) = *R*(*s, o*) + *γkV* (*sj*) 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 a fixed amount of computational overhead per iteration.

# Complexity Results

Our main results focus on two computational problems:

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

More formally, MOMI is defined as follows.

*minimizes |O| subject to O ⊆ Op and Ls,V*0 (*O*) *≤ A.*

*tial value function V*0*, and an integer A* **return** *O that*

**Definition 5** (MOMI): *The* MINOPTIONMAXITER

*problem:*

**Given** *an MDP M, a non-negative real-value s, an ini-*

We then consider the complementary optimization problem: compute a set of *k* options which minimizes the number of iterations. Motivated by this scenario, the second problem we study is MINITERMAXOPTION (MIMO).

**Definition 6** (MIMO): *The* MINITERMAXOPTION

*problem:*

**Given** *an MDP M, a non-negative real-value s, an initial value function V*0*, and an integer k* **return** *O*

*that minimizes Ls,V*0 (*O*)*, subject to O ⊆ Op and*

*|O| ≤ 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 *A* iterations using at most *k* point options.

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

**Given** *an MDP M, a non-negative real-value s, an initial value function V*0*, and integers k and A,* **return**

*‘Yes’ if the there exists an option set O such that O ⊆*

*Op, |O| ≤ k and L*(*O*) *≤ A. ‘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 ⊆*

*cover C ⊆ X such that X∈C X* = *U and |C| ≤ k.*

*U}, and an integer k,* **return** *‘Yes’ if there exists a*

S

*‘No’ otherwise.*

If there is some *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) *ui* represents one of the elements in , (2) *Xi* represents one of the

*∈ U*

*U ∈ X*

*∈ U*

subsets in , (3) *Xij j*: we make a copy for every state *Xi* and call them *Xij*, (4) a goal state *g*. Thus, the state set is *j g* . We build edges between states

*U ∪ X ∪ X ∪ { }*

*∈ X*

*X ∈ X*

as follows: (1) *e*(*u, X*) *E* iff *u X*: For *u* and *X* , there is an edge between *u* and *X*. (2) *Xi* , *e*(*Xi, Xij*) *E*: For every *Xi* , we have an edge from

*∈ ∈ X*

*∈ X ∀ ∈ X*

*∈ ∈ ∈ U*

*Xi* to *Xij*. (3) *e*(*Xj, g*) *E*: for every *Xj ij* we have an edge from *Xi* to the goal *g*. This construction can be

*∀ ∈ ∈ X*

done in polynomial time.

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

SetCover( *, , k*) = OI-DEC(*M, V*0 = 0*, k,* 2). Note that by construction every state *Xi*, *Xij*, and *g* converges to its optimal value within 2 iterations as it reaches the goal state

*U X*

*g* within 2 steps. A state *u* converges within 2 steps if and only if there exists a point option (a) from *X* to *g* where *u X*, (b) from *u* to *Xj* where *u X*, or (c) from *u* to

*∈ U*

*∈ ∈*

*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 *o* = *X*, where *u X*, and the termination state to *βo* = *g*. Let 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 be a set of initial states of each option in ( = *o o* ). This construction exactly matches the solution of the SetCover-DEC.

*O*

*I ∈*

*O C {I | ∈ O}*

*C*

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 = *X*1*, X*2 where *X*1 = 1*,* 2*,* 3 *, X*2 = 3*,* 4*,* 5 to cover



*u*1 *u*2 *u*3 *u*4 *u*5

*X*1

*X*2

*X*1*j*

*X*2*j*

*g*

all elements = 1*,* 2*,* 3*,* 4*,* 5 . The SetCover-DEC can

*U { }*

*{ } { } { }*

*X*

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.

**Theorem 2.** *MOMIgen and MIMOgen are NP-hard.*

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:

*EM∼D*[*LM* (*O*)] *≤ A and O ⊆ Oj.*

*an integer A,* **return** *O that minimizes |O| such that*

*negative real-value s, an initial value function V*0*, and*

**Given** *A distribution of MDPs D, Oj ⊆ Oall, a non-*

**Definition 10** (MOMI*multi*):

## Generalizations of MOMI and MIMO

A natural question is whether Theorem 1 extends to more general option-construction settings. We consider two possi- ble 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 nec- essarily 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 pol- icy. Thus, using the space of all options *all* we generalize MOMI as follows:

*O*

**Definition 9** (MOMI*gen*):

**Given** *an MDP M, a non-negative real-value s, an*

**Theorem 3.** *MOMImulti and MIMOmulti are NP-hard.*

The proof follows from the fact that MOMI*multi* is a super- set 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:

## Theorem 4.

1. *MOMI is* Ω(log *n*) *hard to approximate even for deter- ministic MDPs unless P* = *NP.*

*initial value function V*0*, Oj ⊆ Oall, and an integer*

*A,* **return** *O minimizing |O| subject to Ls,V* (*O*) *≤ A*

1. *MOMI*

*gen*

*is* 2log1*−g n-hard to approximate for any*

*and j.*

0 *s >* 0 *even for deterministic MDPs unless NP ⊆*

*O ⊆ O*

*DTIME*(*npoly* log *n*)*.*

1. *MOMI is* 2log1*−g n-hard to approximate for any s >* 0

*unless NP ⊆ DTIME*(*npoly* 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 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*.

# Approximation Algorithms

We now provide polynomial-time approximation algorithms, A-MIMO and A-MOMI, to solve MOMI and MIMO, re- spectively. 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* with *T* (*g, a, g*) = 1 and *R*(*g, a*) = 0, and ev- ery 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 π*(*s*) :=

*s-optimal if we add a point option from sj to g, minus one.*

*smallest number b such that for all bj ≥ b, Vbj*(*si*) *is*

**Definition 11** (Distance *ds*(*si, sj*)): *ds*(*si, sj*) *is the*

*∈ S*

The overview of the procedure is as follows.

1. Compute an asymmetric distance function *ds*(*s, sj*) :

*× →*

*S S* N representing the number of iterations for

a state *s* to reach its *s*-optimal value if we add a point option from a state *sj* to a goal state *g*.

1. For every state *si*, compute a set of states *Xsi* within *A* 1 distance of reaching *si*. The set *Xsi* represents the states that converge within *A* steps if we add a point option from *si* to *g*.

*−*

1. Let be a set of *Xsi* for every *si X* , where *X*+ is a set of states that converges within *A* without any options (thus can be ignored).

*g*

*g*

+

*X ∈ S \*

1. Solve the set-cover optimization problem to find a set of subsets that covers the entire state set using the ap-

proximation algorithm by Chvatal (1979). This process corresponds to finding a minimum set of subsets *{Xsi }*

that makes every state in *S* converge within *A* steps.

1. 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 *ds* : N3, defined as follows:

*S × S →*

Σ*∞* +

E[

*t*=0 max*{*0*, R*(*st, at*)*}*] *< ∞* for all policies *π*. Note

that we can convert a problem with multiple goals to a prob- lem 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 eval- uating 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.

## A-MOMI

We now describe a polynomial-time approximation algo- rithm, A-MOMI, based on using set cover to solve MOMI.

More formally, let *djs*(*si*) denote the number of itera- tions needed for the value of state *si* to satisfy *V* (*si*)

*V ∗*(*si*) *< s*, and let *djs*(*si, sj*) be an upper bound of the number of iterations needed for the value of *si* to satisfy *V* (*si*) *V ∗*(*si*) *< s*, if the value of *sj* is ini-

*|*

*| −*

*| − |*

*| − |*

tialized such that *V* (*sj*) *V ∗*(*sj*) *< s*. We define

*ds*(*si, sj*) := min(*djs*(*si*) 1*, dsj* (*si, sj*)). For simplicity, we use *d* to denote the function *ds*. 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, sj*) 1

*−*

is the minimum of *d*0(*s*) and one plus the number of edge

traversals from *s* to *sj*. *¤*

3Formally, *d* satisfies triangle inequality, but does not satisfy symmetry and indiscernibles.

* 1. *It guarantees that the MDP is solved within A iterations using the option set acquired by* A-MOMI *O.*



*s*1

*s*3

*s*2

*s*4

*s*5

*s*6

*g*

* 1. *If the MDP is deterministic, the option set is at most O*(log *n*) *times larger than the smallest option set pos- sible to solve the MDP within A iterations.*

*Proof.* See the supplementary material.

Figure 3: Example. Options discovered by A-MIMO with

*k* = 2 are denoted by the dashed lines.

*s \ sj 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, sj*) for Figure 3.

Note that we only need to solve the MDP once to com- pute *d*. *d*(*s, sj*) can be computed once you solved the MDP without any options and store all value functions

*Vi* (*i* = 1*, ...b*) until convergence as a function of *V*1: *Vi*(*s*) = *f* (*V*1(*s*0)*, V*1(*s*1)*, ...*). If we add a point option from *sj* to *g*, then *V*1(*sj*) = *V ∗*(*sj*). Thus, *d*(*s, sj*) is the smallest *i* where *Vi*(*s*) reaches *s*-optimal if we replace *V*1(*sj*) with *V ∗*(*sj*) when computing *Vi*(*s*) as a function of *V*1.

*Example.* We use the MDP shown in Figure 3 as an ex- ample. 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 *A* steps without any options *X*+. Here, we denote a set of nodes that can be solved within *A* steps by *X*+. In

*g*

*g*

*S \*

the example, *U* = *S \ X*+ = *{s*1*, s*2*, s*3*, s*4*}*. A state *s* is included in a subset *Xst* iff *d*(*s, sj*) *≤ A −* 1. For ex- ample, *Xs*1 = *{s*1*}, Xs*2 = *{s*1*, s*2*}*. Thus, the set of subsets are given as: *Xs*1 = *{s*1*}, Xs*2 = *{s*1*, s*2*}, Xs*3 =

*g*

*{s*3*}, Xs*4 = *{s*3*, s*4*}*. In this case, the approximation al- gorithm finds the optimal solution *C* = *{Xs*2 *, Xs*4 *}* for the set-cover optimization problem (*U, X* ). We generate a point option for each state in *C*. Thus, the output of the algorithm is a set of two point options from *s*2 and *s*4 to *g*. *¤*

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

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

Note that the approximation bound for a deterministic MDP will inherent any improvements to the approximation al- gorithm 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.

## A-MIMO

*−*

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

1. Compute *ds*(*s, sj*) : *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 distance function. Then, we exploit this characteristic of *d* and solve the asym- metric *k*-center problem (Panigrahy & Vishwanathan, 1998) on ( *, d, k*) to get a set of centers, which we use as ini- tiation 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:

*U*

**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*s∈U* min*c∈C d*(*s, c*) *subject to |C| ≤ k.*

We solve the problem using a polynomial-time approxima- tion algorithm proposed by Archer (2001). The algorithm has a suboptimality bound of *O*(log*∗ k*)4 where *k <* . It

*|U|*

is known that the problem cannot be solved within a factor

of log*∗ |U| − θ*(1) unless *P* = *NP* (Chuzhoy et al., 2005).

4log*∗* is the number of times the logarithm function must be iteratively applied before the result is less than or equal to 1.

As the procedure by Archer (2001) often finds a set of op- tions smaller than *k*, we generate the rest of the options by greedily adding log *k* options at once. See the supplemen- tary 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 , we generate a point option starting from *c* to the goal state *g*.

*C*

*Example.* Consider an MDP shown in Figure 3. The dis- tance *d*0 for the MDP is shown in Table 1. Note that

*d*(*s, sj*) *d*(*s, g*) holds for every *s, sj* pair. Let us first consider finding one option (*k* = 1). This process corre-

*≤*

sponds 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 *Ls,V*0 ( ) *Ls,V*0 ( ) on by the greedy algorithm can be arbitrary small (i.e. 0) compared to

*∅ − O*

the optimal option (see Proposition 1 in the supplementary material for a proof). *¤*

**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 subopti- mality of O*(log*∗ k*)*.*
  3. *The number of iterations to solve the MDP using the option set acquired is upper bounded by P* (*C*)*.*

*Proof.* See the supplementary material.

# 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 ap- proximation algorithms, and several option types proposed in the literature. We computed the optimal set of point op- tions by enumerating every possible set of point options and picking the best. As an optimal set of options is not unique, we picked one arbitrary. 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 (=0) in the graph Laplacian because it has a constant value for every state. Both betweenness options and eigenoptions are discovered by a polynomial time algorithm, thus able to discover within a few minutes. Figure 11 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 (S¸ ims¸ek & 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 et al. (2017). While they proposed to generate options to travel between subgoals for reinforcement learning, we gen- erate 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 us- ing 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 between- ness 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.

# Related Work

Many heuristic algorithms have proposed to discover op- tions useful for some purposes (Iba, 1989; McGovern & Barto, 2001; Menache et al., 2002; Stolle & Precup, 2002; S¸ ims¸ek & Barto, 2004; S¸ ims¸ek & Barto, 2009; Konidaris & Barto, 2009; Machado et al., 2017; Eysenbach et al., 2018). These algorithms seek to capture varying intuitions about what makes behavioral abstraction useful. Jong et al. (2008) sought to investigate the utility of options empirically and pointed out that introducing options might worsen learning performance. They argued that options can potentially im- prove the learning performance by encouraging exploitation



* 1. 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.

Fourroom

9x9 grid

Fourroom

9x9 grid

19 17

OPT

APPROX

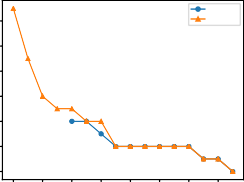
BET

EIG

OPT

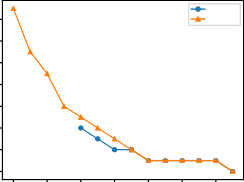
APPROX

EIG



OPT

APPROX



OPT

APPROX

18 16 12 14

17 15 12

10

16 14

15 10

13 8

#Iterations

#Iterations

#Options

#Options

14

12 8

13

11 6

12 6

10

11 4

10 9 4

9 8 2

2

8 7

7 6 0 0

0 1 2 3 4 5 6 7 8 9 10 11 12

#Options

1. Four Room (MIMO)

0 1 2 3 4 5 6 7 8 9 10 11 12

#Options

1. 9 *×* 9 grid (MIMO)

4 6 8 10 12 14 16 18

#Iterations

1. Four Room (MOMI)

4 6 8 10 12 14 16

#Iterations

1. 9 *×* 9 grid (MOMI)

Figure 5: MIMO and MOMI evaluations. Parts (a)–(b) show the number of iterations for VI using options generated by A-MIMO. Parts (c)–(d) 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.

or exploration. For example, some works investigate the use of bottleneck states (Stolle & Precup, 2002; S¸ ims¸ek & Barto, 2009; Menache et al., 2002; Lehnert et al., 2018). Stolle & Precup (2002) proposed to set states with high visitation count as subgoal states, resulting in identifying bottleneck states in the four-room domain. S¸ ims¸ek & Barto (2009) generalized the concept of the 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 en- vironment to run a Max-Flow/Min-Cut algorithm to the state-space graph to identify bottleneck states. These meth- ods generate options to leverage the idea that subgoals are states visited most frequently. On the other hand, S¸ ims¸ek & 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 diver- sity of the trajectories by the set of options are maximized. These methods generate options to explore infrequently visited states. Harb et al. (2017) proposed to formulate good options to be options which minimize the delibera- tion costs in the bounded rationality framework (Simon, 1957). The problem of finding minimum state-abstraction with bounded performance loss was studied by Even-Dar & Mansour (2003). They showed it is *NP*-hard and proposed

a polynomial time bicriteria approximation algorithm.

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 approx- imate value iteration with and without options and showed that options lead to faster convergence if their duration 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.

# Conclusions

We considered a fundamental theoretical question concern- ing the use of behavioral abstractions to solve MDPs. We considered two problem formulations for finding options:

1. minimize the size of option set given a maximum num- ber of iterations (MOMI) and (2) minimize the number of iterations given a maximum size of option set (MIMO). We showed that the two problems are both computationally in- tractable, 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. Although these algorithms are not practical for a single-task planning, we believe these algorithms 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 learn- ing. Since we now know which options minimize plan- ning 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.

# References

Archer, A. Two O(log\* k)-approximation algorithms for the asymmetric k-center problem. In *International Confer- ence on Integer Programming and Combinatorial Opti- mization*, pp. 1–14, 2001.

Bacon, P.-L. On the bottleneck concept for options discov- ery. *Masters thesis, McGill University*, 2013.

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

Bhattacharyya, A., Grigorescu, E., Jung, K., Raskhodnikova, S., and Woodruff, D. P. Transitive-closure spanners. *SIAM Journal on Computing*, 41(6):1380–1425, 2012.

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

Chvatal, V. A greedy heuristic for the set-covering problem.

*Mathematics of operations research*, 4(3):233–235, 1979.

Dinitz, M., Kortsarz, G., and Raz, R. Label cover instances with large girth and the hardness of approximating basic k-spanner. In *International Colloquium on Automata, Languages, and Programming*, pp. 290–301. Springer, 2012.

Dinur, I. and Safra, S. On the hardness of approximating label-cover. *Information Processing Letters*, 89(5):247– 254, 2004.

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

Even-Dar, E. and Mansour, Y. Approximate equivalence of Markov decision processes. In *Learning Theory and Kernel Machines*, pp. 581–594. Springer, 2003.

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

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

Goldsmith, J., Littman, M. L., and Mundhenk, M. The com- plexity of plan existence and evaluation in probabilistic domains. In *Proceedings of the Thirteenth Annual Con- ference on Uncertainty in Artificial Intelligence (UAI–97)*, pp. 182–189, San Francisco, CA, 1997. Morgan Kauf- mann Publishers.

Harb, J., Bacon, P.-L., Klissarov, M., and Precup, D. When waiting is not an option: Learning options with a deliber- ation cost. *arXiv preprint arXiv:1709.04571*, 2017.

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

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

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

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

Kortsarz, G. On the hardness of approximating spanners.

*Algorithmica*, 30(3):432–450, 2001.

Lehnert, L., Laroche, R., and van Seijen, H. On value function representation of long horizon problems. In *AAAI*, 2018.

Littman, M. L. Probabilistic propositional planning: Rep- resentations and complexity. In *Proceedings of the Four- teenth National Conference on Artificial Intelligence*, pp. 748–754. AAAI Press/The MIT Press, 1997.

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

Machado, M. C., Bellemare, M. G., and Bowling, M. A Laplacian framework for option discovery in reinforce- ment learning. In *Proceedings of the Thirty-fourth Inter- national Conference on Machine Learning*, 2017.

Mann, T. and Mannor, S. Scaling up approximate value iter- ation with options: Better policies with fewer iterations. In *International Conference on Machine Learning*, pp. 127–135, 2014.

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

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

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

Moradi, P., Shiri, M. E., Rad, A. A., Khadivi, A., and Hasler,

M. Automatic skill acquisition in reinforcement learn- ing using graph centrality measures. *Intelligent Data Analysis*, 16(1):113–135, 2012.

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

Papadimitriou, C. H. and Tsitsiklis, J. N. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, August 1987.

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

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

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

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

Silver, D. and Ciosek, K. Compositional planning using op- timal option models. In *Proceedings of the 29th Interna- tional Conference on Machine Learning*, pp. 1063–1070, 2012.

Simon, H. A. *Models of man; social and rational.* Wiley, 1957.

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

S¸ ims¸ek, O¨ . and Barto, A. G. Skill characterization based on betweenness. In *Advances in Neural Information Processing Systems*, pp. 1497–1504, 2009.

S¸ ims¸ek, O¨ ., Wolfe, A., and Barto, A. Identifying useful sub- goals in reinforcement learning by local graph partition- ing. In *Proceedings of the Twenty Second International Conference on Machine Learning*, pp. 816–823, 2005.

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

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

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

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

# A. Appendix: Inapproximability of MOMI

In this section we prove Theorem 4:

polynomial-time algorithm can approximate Min-Rep well. Let *n*˜ = *|A|* + *|B|*.

## Theorem 4.

**Lemma 1** (Kortsarz 2001)**.** *Unless NP DTIME*(*npoly* log *n*)*, Min-Rep admits no* 2log

1*−ε⊆n*˜

* 1. *MOMI is* Ω(log *n*) *hard to approximate even for deter- ministic MDPs unless P* = *NP.*
  2. *MOMI is* 2log1*−g n-hard to approximate for any s >* 0 *even for deterministic MDPs unless NP DTIME*(*npoly* log *n*)*.*

*⊆*

*gen*

*3. MOMI is* 2log1*−g n-hard to approximate for any s >* 0

*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 *ct* for constant *cj*. It immediately follows

that

*e |πe| ≤ nc* for constant *c*.

*|*Σ*A|, |*Σ*B| ≤ n*˜

Σ

*unless NP ⊆ DTIME*(*npoly* 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 hard- ness 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 13** (Min-Rep):

*phabets* Σ*A and* Σ*B for the left and right sides of*

**Given** *a bipartite graph tt* = (*A ∪ B, E*) *and al-*

*tt respectively. Each e ∈ E has associated with*

*it a set of pairs πe ⊆* Σ*A ×* Σ*B which satisfy it.*

**Return** *a pair of assignments γA* : *A → P*(Σ*A*)

(*Ai, Bj*) *∈ E there exists an* (*a, b*) *∈ πe such that*

*and γB* : *B → P*(Σ*B*) *such that for every e* =

*a ∈ γ* (*A* ) *and b ∈ γB*(*Bj*)*. The objective is to*

*A i*

*minimize*

Σ

*A ∈A*

*i*

*|γ* (*A* )*|* +

*A i*

Σ

*B ∈B*

*j*

*|γ* (*B* )*|.*

*B j*

We illustrate a feasible solution to an instance of Min-Rep in Figure 6.

## 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 mini- mization values of the two problems are equal: *P* ( ) = , 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( *,* ) be an instance of the set- cover problem. We can convert the instance into an instance

*·*

*·*

*U X*

*C |O|*

of MOMI(*M,* 0*,* 2). Using the approximation algorithm, we get a solution where *c* ln *n ∗* , where *∗* is the optimal solution. We construct a solution for the set

*O |O| ≤ |O | O*

cover from the solution to the MOMI (see the construc- tion in the proof of Theorem 1). Because = and

*|C| |O|*

*C O*

*∗* = *∗* , where *∗* is the optimal solution for the set cover, we get = *c* ln *n ∗* = *c* ln *n ∗* . Thus, we acquire a *c* ln *n* approximation solution for the set-cover

*·*

*|C| |O| ≤ |O | |C |*

*|C | |O | C*

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.

## Hardness of Approximation of MOMI*gen*

We now show our hardness of approximation of 2log1*−g n*

for MOMI

(*a*1*, b*2)

*A*1

*a*1*, a*2

*B*1

*b*2

(*a*3*, b*1)

*A*2 *B*2

*gen*

, Theorem 4.2.5

*a*3 *b*1*, b*3

Figure 6: An instance of Min-Rep with Σ*A* = *a*1*, a*2*, a*3 and Σ*B* = *b*1*, b*2*, b*3 . Edge *e* is labeled with pairs in *πe*. Feasible solution (*γA, γb*) illustrated where *γA*(*Ai*) and *γB*(*Bj*) below *Ai* and *Bj* in blue. Constraints colored to coincide with stochastic action colors in Figure 8.

*{ }*

*{ }*

The crucial property of Min-Rep we use is that no

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 vertex in Min-Rep as choosing an option in our MOMI*gen* instance. In particular, we will have a state for each edge

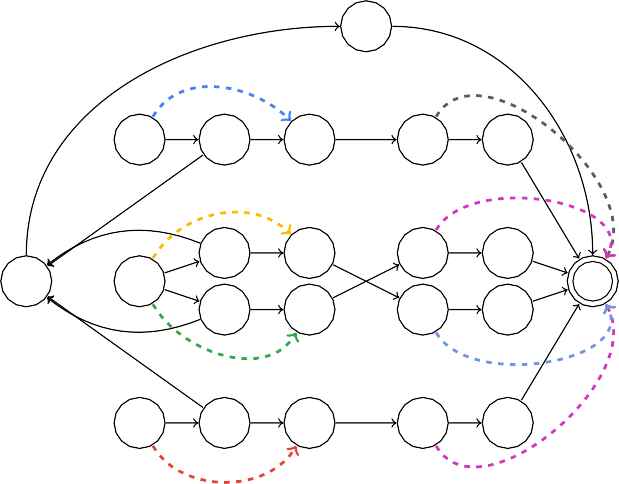
5We assume that *j* is a “good” set of options in the sense that there exists some set *∗ j* such that *Ls,V*0 ( *∗*) *A*. We also assume, without loss of generality, that *ε <* 1 throughout this

*O ⊆ O O ≤*

*O*

section; other values of *ε* can be handled by re-scaling rewards in our reduction.

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.



*Sgj*

*Se*1 *Sej* 1 *a*1 *Se*1 *a*1

*Se*1 *b*2

*Sej*

1 *b*2

*Sej a*

2 2

*Se*2 *a*2

*Se*2 *b*1

*Sej b*1

2

*Sgjj*

*Se*2

*Sg*

*Sej* 2 *a*3 *Se*2 *a*3 *Se*2 *b*3

*Sej*

2 *b*3

*Se*3 *Sej* 3 *a*3 *Se*3 *a*3

*Se*3 *b*1

*Sej*

3 *b*1

More formally, our reduction is as follows. Consider an instance of Min-Rep, MR, given by *tt* = (*A B, E*), Σ*A*, Σ*B* and *πe* . Our instance of MOMI*gen* is as follows where *γ* = 1 and *l* = 3.6

*{ }*

*∪*

* + - **State space** We have a single goal state *Sg* along with states *Sgj* and *Sgjj*. For each edge *e* we create a state *Se*. Let Sat*A*(*e*) consist of all *a* Σ*A* such that *a* is in

*∈*

some assignment in *πe*. Define Sat*B*(*e*) symmetrically. For each edge *e ∈ E* we create a set of 2 *· |*Sat*A*(*e*)*|*

states, namely *Sea* and *Sej a* for every *a ∈* Sat*A*(*e*). We

do the same for *b ∈* Sat*B*(*e*).

* + - **Actions and Transitions** We have a single action from *Sgj* to *Sg*, a single action from *Sgjj* to *Sgj* . For each edge *e* we have the following deterministic actions:

Every *Sej a* has a single outgoing action to *Sea* for *a ∈*

Figure 7: Our MOMI*gen* reduction applied to the Min- Rep problem in Figure 6. *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 *j* represented in its own color as a dashed line from initiation to termination states. Notice that a single option

Sat*A*(*e*); Every *Seb* has a single outgoing action to *Sebt*

for *b ∈* Sat*B*(*e*); Every *Sea* has an outgoing action to

*Seb* if (*a, b*) *∈ πe* and every *Sej b* has a single outgoing

*O*

goes from *S*

*e*3*b*1

and *S*

*e*2*b*1

to *Sg*.

action to *Sg*; Lastly, we have a single action from *Sej a*

to *Sgjj* for every *a ∈* Sat*A*(*e*).

* + - **Reward** The reward of arriving in *Sg* is 1. The reward of arriving in every other state is 0.

Min-Rep solution to MR. The following lemmas demon- strates the correspondence between a MOMI*gen* and Min- Rep solution.

**Lemma 2.** *OPT*MOMI*gen ≤ OPTMR*

* + - **Option Set** Our option set *Oj* is as follows. For each

vertex *Ai ∈ A* and each *a ∈* Σ*A* we have an option

*{ ∈ ∧ ∨ }*

*Proof.* Given a solution (*γA, γB*) to MR, define *OγA,γB* :=

*O*(*Ai, a*): The initiation set of this option is every *Se*

*O*(*v, x*) : *v V* (*tt*) (*γA*(*v*) = *x γB*(*v*) = *x*) as the

corresponding set of options. Let *γ∗* and *γ∗* be the optimal

where *e* is incident to *Ai*; The termination set of this option is every *Sea* where *Ai* is incident to *e*; The

*A B*

solutions to MR which is of cost OPTMR.

policy of this option takes the action from *Sej a* to *Sea*

We now argue that *OγA∗ ,γB∗* is a feasible solution to

when in *Sej a* and the action from *Se* to *Sej a* when in *Se*.

Symmetrically, for every vertex *Bj B* and each *b* Σ*B* we have an option *O*(*Bj, b*): The initiation set of this option is every *Seb* where *e* is incident to *Bj*;

*∈*

*∈*

The termination set of this option is *Sg*; The policy of this option takes the action from *Seb* to *Sej b* when in *Seb* and from *Sej b* to *Sg* when in *Sej b*.

One should think of choosing option *O*(*v, x*) as correspond- ing to choosing label *x* for vertex *v* in the input Min-Rep instance. Let MOMI*gen*(MR) be the MDP output given instance MR of Min-Rep and see Figure 8 for an illustration of our reduction.

MOMI*gen*(MR) of cost OPTMR, demonstrating that the op- timal solution to MOMI*gen*(MR) has cost at most OPTMR. To see this notice that by construction the MOMI*gen* cost

of *OγA∗ ,γB∗* is exactly the Min-Rep cost of (*γA∗ , γB∗* ).

We need only argue, then, that *γA∗ ,γB∗* is feasible for MOMI*gen*(MR) and do so now. The value of every state in MOMI*gen*(MR) is 1. Thus, we must guarantee that after 3 iterations of value iteration, every state has value

*O*

1. However, without any options every state except each

*Se* has value 1 after 3 iterations of value iteration. Thus, it suffices to argue that *γA∗ ,γB∗* guarantees that every *Se* will have value 1 after 3 iterations of value iteration. Since

*O*

(*γA∗ , γB∗* ) is a feasible solution to MR we know that for every

Let OPT MOMI

MOMI*gen*

be the value of the optimal solution to

*e* = (*Ai, Bj*) there exists an *a*¯ *∈ γA∗* (*Ai*) and ¯*b ∈ γB∗* (*Bj*)

such that (*a*¯*,* ¯*b*) *∈ πe*; correspondingly there are options

*gen*(MR) and let OPTMR be the value of the optimal

*O*(*Ai, a*¯)*, O*(*Bj,* ¯*b*) *∈ Oγ∗ ,γ∗* . It follows that, given op-

6It is easy to generalize these results to *l ≥* 4 by replacing

*A*

*B*

tions *OγA∗ ,γB∗* from, *Se* one can take option *O*(*Ai, a*¯) then

certain edges with paths.

the action from *Sea*¯ to *Se*¯*b* and then option *O*(*Bj,* ¯*b*) to ar-

rive in *Sg*; thus, after 3 iterations of value iteration the value of *Se* is 1. Thus, we conclude that after 3 iterations of value iteration every state has converged on its value.

We first argue that our algorithm is polynomial-time in *n*˜. However, notice that for each vertex, we create a polynomial number of states. Thus, the number of states in MOMI*gen*(MR) is polynomially-bounded in *n*˜ and so

We now show that a solution to MOMI

*gen*

(MR) corre-

MOMI*gen* runs in time polynomial in *n*˜. A polynomial

runtime of our algorithm immediately follows.

*A*

sponds to a solution to MR. For the remainder of this sec- tion *γO*(*A* ) := *{a* : *O*(*A , a*) *∈ O}* and *γO*(*B* ) := *{b* :

*A*

*i*

*i*

*B*

*j*

*O*(*Bj, b*) *∈ O}* is the Min-Rep solution corresponding to

option set *O*.

**Lemma 3.** *For a feasible solution to* MOMI*gen*(*MR*)*, O, we have* (*γAO, γBO* ) *is a feasible solution to MR of cost |O|.*

*Proof.* Notice that by construction the Min-Rep cost of

(*γO, γO*) is exactly *|O|*. Thus, we need only prove that

*A*

*B*

We now argue that our algorithm is a 2log1*−εt n*˜- approximation for Min-Rep for some *εj >* 0. Apply- ing Lemma 3, the approximation of *A*MOMI*gen* and then Lemma 2, we have that (*γAO, γBO* ) is a feasible solution for

MR with cost

costMin-Rep(*γAO, γBO* ) = *|O|*

log1*−ε n*

*≤* 2 OPT

MOMI*gen*

log1*−ε n*

(*γAO, γBO* ) is a feasible solution for MR.

We do so now. Consider an arbitrary edge *e* = (*Ai, Bj*) *E*; we wish to show that (*γO, γO*) satisfies *e*. Since is a feasible solution to MOMI*gen*(MR) we know that after

*A*

*B*

*O*

*∈*

3 iterations of value iteration every state must converge on its value. Moreover, notice that the value of every state in MOMI*gen*(MR) is 1. Thus, it must be the case that for every *Se* there exists a path of length 3 from *Se* to *Sg* using either options or actions. The only such paths are those that take an option *O*(*Ai, a*), then an action from *Sea* to

*Seb* then option *O*(*Bj, b*) where (*a, b*) *∈ πe*. It follows that

*a γAO* (*Ai*) and *b γBO* (*Bj*). But since (*a, b*) *πe*, we

*∈ ∈ ∈*

then know that *e* is satisfied. Thus, every edge is satisfied and so (*γAO, γBO* ) is a feasible solution to MR.

*Theorem 4.2 Proof.* Assume NP DTIME(*n*poly log *n*) and for the sake of contradiction that there exists an *ε >* 0 for

*ƒ⊆*

*≤* 2 OPTMR

Thus, (*γO, γO*) is a 2log1*−ε n* approximation for the opti- mal Min-Rep solution where *n* is the number of states in the MDP of MOMI*gen*(MR). Now recalling that *n ≤ n*˜*c*

*A*

*B*

for fixed constant *c*. We therefore have that (*γAO, γBO* ) is a

2log *n*˜ *c* log *n*˜ *j* log *n*˜ approximation for a constant *cj*. Choosing *ε* sufficiently small, we have that *cj ·* 2log *n*˜ *≤* 2log *n*˜ for sufficiently large *n*˜.

*≤ ·*

1*−εt*

1*−ε* 1*−εt*

1*−ε c* = 2 1*−ε* 1*−ε c* 2 1*−ε*

Thus, our polynomial-time algorithm is a 2log *n*˜- approximation for Min-Rep for *εj >* 0, thereby contra- dicting Lemma 1. We conclude that MOMI*gen* cannot be

2log *n*-approximated.

1*−g*

## Hardness of Approximation of MOMI with Stochastic MDP

which polynomial-time algorithm *A*

MOMI*gen*

can 2log1*−ε n*-

We now show our hardness of approximation of 2log

1*−g*

*n* for

approximate MOMI

*gen*

. We use *A*

MOMI*gen*

to 2log1*−εt n*˜

MOMI, Theorem 4.3. We will notably use the stochasticity

approximate Min-Rep for a fixed constant *εj >* 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 MOMI*gen*(MR) are polynomially-bounded. Notice that the number of states *n* in the MDP in MOMI*gen*(MR) is at most *O*(*n*˜2 Σ*A* Σ*B* ) = *n*˜*c* for some fixed constant *c* by the aforementioned assumption that Σ*A* and Σ*B* are polynomially-bounded in *n*˜.7

*| || |*

Our polynomial-time approximation algorithm to approxi-

mate instance MR of Min-Rep is as follows: Run *A*MOMI*gen* on MOMI*gen*(MR) to get back option set *O*. Return (*γAO, γBO* ) as defined above as our solution to MR.

7It is also worth noticing that since we create at most

*O*(*n*˜ Σ*A* + *n*˜ Σ*B* ) options, the total number of options in *j*

*| | | | O*

is at most polynomial in *n*˜.

of the input MDP to show this result.8

We begin by describing our reduction from an instance of Min-Rep to an instance of MOMI. The intuition behind our reduction is as follows. As in our reduction for MOMI*gen* we will have vertex for each edge in our Min-Rep instance and reward will propagate quickly to that vertex when value iteration is run only if the options corresponding to a satis- fying assignment for that edge are chosen. The challenge, however, is that since our options are now only point options (whereas in MOMI*gen* they were arbitrary options) it seems 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

8We may assume without loss of generality *ε < .*5 throughout this section; rewards in our reduction can be re-scaled to handle larger *ε*.

ands: A fixed edge is satisfied when *one* of its satisfying assignments is met (an or) and a given satisfying assignment is met when both endpoints have the right labels (an and). We will exploit the fact that the value of a state in an MDP is a max over actions to encode the “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 *tt* = (*A B, E*), Σ*A*, Σ*B* and *πe* . Our instance of MOMI is as follows where *γ* = 1 and *l* = 2.9

*{ }*

*∪*

* + - **State space** We have a goal state *Si* for each *Ai A*. Again, let Sat*A*(*e*) consist of all *a* Σ*A* such that *a* is in some assignment in *πe*. For each *Ai A* and *a* Sat*A*(*e*) we will we add to our MDP states *Sia*

*∈*

*∈*

*∈*

*∈*

and *Sija*. We symmetrically do the same for all states in Σ*B*. For each *e ∈ E* we will also add a state *Se*.10

* + - **Actions and Transitions** Every *Sia* state has a sin- gle action to *Sija* and every *Sia* state has a single ac- tion to *Si*. The same symmetrically holds for states

from a *Bj B*. Every *Se* for *e* = (*Ai, Bj*) has *π*(*Ai,Bj* ) actions associated with it, namely *α*(*a,b*) where (*a, b*) *π*(*Ai,Bj* ). Action *α*(*a,b*) has a probabil- ity *.*5 of transitioning to state *Sia* and a probability *.*5 of transitioning to state *Sjb*.

*∈*

*| | { }*

*∈*

* + - **Reward** The reward of arriving in any *Si* or *Sj* for

*Ai ∈ A* or *Bj ∈ B* is 1 and 0 for every other state.

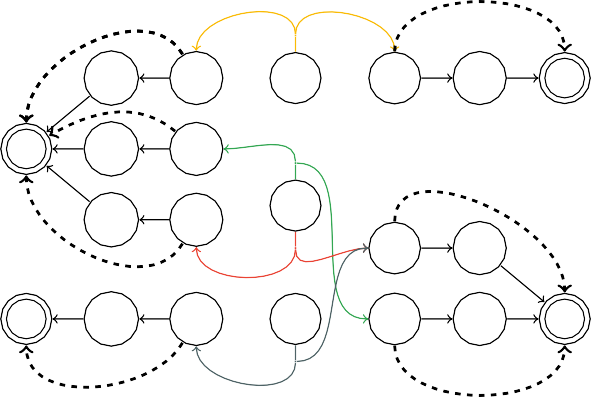
Notice that no point options have *Se* as an initialization state since any such option would have a *.*5 probability of never terminating (and we assume our options always terminate). See Figure 8 for an illustration of our reduction. One should think of choosing a point option from *Sia* to *Si* as corresponding to choosing label *a* for *Ai* in the input Min-Rep instance. The same holds for label *b* for *Bj* and choosing a point option from *Sjb* to *Sj*. Let 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 2log *n*-approximated for any *ε >* 0. Let OPTMOMI be the value of the optimal solution to MOMI(MR) and let OPTMR be the value of the optimal Min-Rep solution to MR. The following lemmas demon-

*≥*

1*−ε*

9It is easy to generalize these results to *l* 3 by replacing edges with paths.



*SAj a*

1 1

*SA*1 *a*1

*Se*1

*SB*1 *b*2

*SBj b*

1 2

*SB*1

*SA*1 *SAj a*

1 2

*SA*1 *a*2

*SAj a*

*Se*2

1 3

*SA*1 *a*3

*SB*2 *b*1

*SBj b*

2 1

*SA*2 *SAj a*

2 3

*SA*2 *a*3

*Se*3

*SB*2 *b*3

*SBj b*

2 3

*SB*2

Figure 8: Our MOMI reduction applied to the Min-Rep problem in Figure 6. *e*1 = (*A*1*, B*1), *e*2 = (*A*1*, B*2), *e*3 = (*A*2*, B*2). Stochastic options colored according to the pair in *πe* to which they correspond, branching into the two states in which they arrive with equal probability. Deterministic action given as solid black arcs. Possible point options given as dashed arcs.

strates the correspondence between a MOMI and Min-Rep solution.

**Lemma 4.** *OPT*MOMI *≤ OPTMR*

*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 (*γA, γB*) to MR, define

*OγA,γB* as consisting of all point options from *Sia* to *Si* if *a ∈ γA*(*Ai*) and all points options from *Sjb* to *Sj* if *b γB*(*Bj*). Let *γA∗* and *γB∗* be the optimal solutions to MR

*∈*

which is of cost OPTMR.

We claim that *γA∗ ,γB∗* is a feasible solution to MOMI(MR) of cost OPTMR, demonstrating that the optimal solution to MOMI(MR) has cost at most OPTMR. To see this notice

*O*

that by construction the MOMI cost of *OγA∗ ,γB∗* is exactly the Min-Rep cost of *γA∗ , γB∗* .

We need only argue, then, that *γA∗ ,γB∗* 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. How- ever, without any options every state except for *Se* where

*O*

*e ∈ E* has value 1 after 2 iterations of value iteration. Thus, it suffices to argue that *γA∗ ,γB∗* guarantees that every *Se* will have value 1 after 2 iterations of value iteration. Since (*γA∗ , γB∗* ) is a feasible solution to MR we know that for ev- ery *e* = (*Ai, Bj*) there exists *a*¯ *∈ γA∗* (*Ai*) and ¯*b ∈ γB∗* (*Bj*)

*O*

such that (*a*¯*,* ¯*b*) *∈ πe*; correspondingly there is some action

10It 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

from *Se*

with a *.*5

probability of resulting in state

*Sia*¯

and

every *Si* and *Sj* to be the same state. We assume multiple goal states for ease of exposition.

a *.*5 probability of resulting in state *Sj*¯*b* where *γA∗ ,γB∗* has

a point option from *Sia*¯ to *Si* and a point options from *Sj*¯*b*

*O*

to *S* . That is, *V* (*S*

*j*

1

) = 1 and *V* (*S*

) = 1. Thus, after

which a polynomial-time algorithm *A*

can 2log1*−ε n*-

one iteration of value iteration the values of *Sia*¯ and *Sj*¯*b* are

*ia*¯

1

*j*¯*b*

approximate MOMI. We use *A*

MOMI

MOMI

to 2log1*−εt n*˜ approx-

both 1 and so after two iterations of value iteration the value of *Se* is

*V*2(*Se*) = max *.*5 (*V*1(*Sia*)) + *.*5 (*V*1(*Sjb*))

*· ·*

*α*(*a,b*)

*≥ .*5 *·* (*V*1(*Sia*¯)) + *.*5 *·* (*V*1(*Sj*¯*b*))

= 1*.*

Thus, *V*2(*Se*) = 1 for every *Se* 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 the remainder of this section let

*γO*(*A* ) := *{a* : *O*(*S , S* ) *∈ O}* and *γO*(*B* ) := *{b* :

*A*

*i*

*ja*

*j*

*B*

*j*

imate Min-Rep for a fixed constant *εj >* 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 *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*(*n*˜ + 2 *A* Σ*A* + *B* Σ*B* + *E* ) *n*˜*c* for some fixed constant *c* by the aforementioned assumption that Σ*A* and Σ*B* are polynomially-bounded in *n*˜.11

*| |*

*| |*

*| || | | || | | | ≤*

Our polynomial-time approximation algorithm to approxi- mate instance MR of Min-Rep is as follows: Run *A*MOMI

on MOMI(MR) to get back option set *O*. Return (*γAO, γBO* )

as defined above as our solution to MR.

*O*(*Sjb, Sj*) where for the remainder of this section

*∈ O}*

*O*(*S, Sj*) stands for a point option with initiation state *S* and termination state *Sj*.

**Lemma 5.** *For any feasible solution O to* MOMI(*MR*) *we have* (*γAO, γBO* ) *is a feasible solution to MR of cost |O|.*

*Proof.* Notice that by construction the Min-Rep cost of

(*γAO, γBO* ) is exactly *|O|*. Thus, we need only prove that

(*γAO, γBO* ) is a feasible solution for MR.

We do so now. Consider an arbitrary edge *e* = (*Ai, Bj*) *∈*

*E*; we wish to show that (*γAO, γBO* ) satisfies *e*. Since is a

*O*

feasible solution we know that after two iterations of value

iteration every state must converge on its value (up to an *s* factor which we can ignore by our above assumption that *ε < .*5). Moreover, notice that the value of every state in MOMI(MR) is 1. Thus, it must be the case that for every *Se* we have *V*2(*Se*) = 1 for *e* = (*Ai, Bj*). It follows, then,

We first argue that our algorithm is polynomial time in *n*˜. For each vertex in MR, we create a polynomial num- ber of states and actions. Thus, the number of states in MOMI(MR) is polynomially-bounded in *n*˜ and so MOMI runs in time polynomial in *n*˜. A polynomial runtime of our algorithm immediately follows.

We now argue that our algorithm is a 2log1*−εt n*˜- approximation for Min-Rep for some *εj >* 0. Applying Lemma 5, the approximation of *A*MOMI and then Lemma 4, we have that the Min-Rep cost of (*γAO, γBO* ) is

*A*

costMin-Rep(*γAO, γBO* ) = *|O|*

log1*−ε n*

*≤* 2 OPT

MOMI

log1*−ε n*

*≤* 2 OPT

MR

Thus, (*γO, γO*) is a 2log1*−ε n* approximation for the opti-

that there is some action *α*(*a*¯*,*¯*b*) where (*a*¯*,* ¯*b*) *∈ π*(*Ai,Bj* ) such *A B*

that

1 = *V*2(*Se*) = *.*5 *·* (*V*1(*Sia*¯)) + *.*5 *·* (*V*1(*Sj*¯*b*))*.*

Since the value of every state is at most 1, it follows that *V*1(*Sia*¯) = *V*1(*Sj*¯*b*) = 1. However, since *V*1(*Sia*¯) and *V*1(*Sj*¯*b*) are both two hops from the only goal reachable

from them (*Si* and *Sj* respectively) it must be the case that

there is some point option from *Sia*¯ to *Si* and *Sj*¯*b* to *Sj*. Thus, by definition of (*γAO, γBO* ) we then have *a*¯ *∈ γAO* and

¯*b ∈ γBO* . Since (*a*¯*,* ¯*b*) *∈ π*(*A ,B* ) it follows that arbitrary

*i*

*j*

mal Min-Rep solution where *n* is the number of states in the MDP of MOMI(MR). Now recalling that *n ≤ n*˜*c*

for fixed constant *c*. We therefore have that (*γAO, γBO* ) is a

2log *n*˜ *c* log *n*˜ *j* log *n*˜ approximation for a constant *cj*. Choosing *ε* sufficiently small, we have that *cj ·* 2log *n*˜ *≤* 2log *n*˜ for sufficiently large *n*˜.

*≤ ·*

1*−εt*

1*−ε* 1*−εt*

1*−ε c* = 2 1*−ε* 1*−ε c* 2 1*−ε*

Thus, our polynomial-time algorithm is a 2log *n*˜- approximation for Min-Rep for *εj >* 0, thereby contra- dicting Lemma 1. We conclude that MOMI cannot be

2log *n*-approximated.

1*−g*

edge *e* = (*Ai, Bj*) is satisfied. Thus, every edge in *E* is satisfied and so (*γAO, γBO* ) is a feasible solution for MR.

Finally, we conclude the hardness of approximation of MOMI.

## A-MIMO

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

*Theorem 4.3 Proof.* Assume NP *ƒ⊆* DTIME(*n*poly log *n*) and

11It is worth noting, also, that since we create at most Σ *|πe|*

for the sake of contradiction that there exists an *ε >* 0 for

*e*

actions for any state, the number of total actions in our MDP is at most polynomial in *n*˜.

**Theorem 6.** A-MIMO *has following properties:*

1. A-MIMO *runs in polynomial time.*
2. *If the MDP is deterministic, it has a bounded subopti- mality 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 times. Thus, it runs in polynomial time.

*|S|*

1. 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



*s*0

*s*1

*s*2

*s*3

*g*

Figure 9: An example of an MDP where *d*(*s, C*) *<* min*st∈C d*(*s, sj*). Here the transition induced by the op- timal 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.

Using the option set acquired, the number of iterations to solve the MDP is bounded by *P* ( ). To prove this we first generalize the definition of the distance function to take a

*C*

guarantees the suboptimality bounds, it tends to find a set of

options smaller than *k*. In order to use the rest of the options

state and a set of states as arguments *ds*

Let

*∈ C*

*C*

: *S ×* 2*S →* N.

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* = log *k* (if +log *k > k* then we set *r* = *k* ) and add a set of options which minimizes *A* (breaking ties randomly) to the option set . We repeat this procedure until = *k*. This procedure runs in polynomial time. The number of possible option set of size *r* is *rCn* = *O*(*nr*) = *O*(*k*). We repeat this procedure at most *k/* log *k* times, thus the total computation time is bounded by *O*(*k*2*/* log *k*).

*| |*

*|O|*

*O*

*| | |O| −|O|*

1. 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

. By triangle inequality, swapping the terminal state to the goal state will monotonically decrease *d*(*s, g*) for every

*O*

state. By swapping every such option we can construct an option set *Oj* with *Ls,V*0 (*Oj*) *≤ Ls,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.

*ds*(*s,* ) the number of iterations for *s* to converge *s*- optimal if every state *sj* has converged to *s*-optimal:

*ds*(*s,* ) := min(*djs*(*s*)*,* 1 + *djs*(*s,* )) 1. As adding an

*C C −*

option will never make the number of iterations larger,

## Lemma 7.

*d*(*s,* ) min *d*(*s, sj*)*.* (8)

*C ≤*

*st∈C*

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∈S* min*c∈C d*(*s, c*) *≥* max*s∈S d*(*s, C*) = *Ls,V*0 (*O*) (using Equation 8). Thus *P* (*C*) is an upper bound for *Ls,V*0 (*O*).

The reason why *P* ( ) 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 9 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.

*C*

The equality of the statement 8 holds if the MDP is determin- istic. That is, *d*(*s,* ) = min*st∈C d*(*s, sj*) for deterministic MDP.

*C*

## Theorem 6.3.

*If the MDP is deterministic, it has a bounded suboptimality of O*(log*∗ k*)*.*

*Proof.* First we show *P* (*C∗*) = *Ls,V*0 (*O∗*) for de- terministic MDP. From *d*(*s, C*) = min*st∈C d*(*s, sj*), *P* (*C∗*) = max*s∈S* min*c∈C∗ d*(*s, c*) = max*s∈S d*(*s, C∗*) = *Ls,V*0 (*O∗*).

The asymmetric *k*-center solver guarantees that the out- put satisfies *P* ( ) *c*(log*∗ k* + *O*(1))*P* ( *∗*) where *n* is the number of nodes (Archer, 2001). Let MIMO(*M, s, k*)

*C C ≤ C*

be an instance of MIMO. We convert this instance to an instance of asymmetric *k*-center AsymKCenter( *, d, k*), where = . By solving the asymmetric *k*-center

*|U| |S|*

*U*

with the approximation algorithm, we get a solution which satisfies *P* ( ) *c*(log*∗ k* + *O*(1))*P* ( *∗*). Thus, the output of the algorithm satisfies *Ls,V*0 ( ) = *P* ( ) *c*(log*∗ k* + *O*(1))*P* ( *∗*) = *c*(log*∗ k* + *O*(1))*Ls,V*0 ( *∗*).

*O ≤ O*

*C O*

*O O C ≤*

*C ≤ C*

*C*

Thus, *Ls,V*0 ( ) *c*(log*∗ k* + *O*(1))*Ls,V*0 ( *∗*) is de-

rived.

**Proposition 1** (Greedy Strategy)**.** *Let an option set be a set of point option constructed by greedily adding one point option which minimizes the number of iterations. An improvement Ls,V*0 ( ) *Ls,V*0 ( ) *by the greedy algorithm can be arbitrary small (i.e. 0) compared to the optimal option set.*

*O*

*∅ − O*

*Proof.* We show by the example in a shortest-path problem in Figure 10. The MDP can be solved within 4 iterations without options: *Ls,V*0 ( ) = 4. With an optimal option set of size *k* = 2 the MDP can be solved within 2 itera-

*∅*

tions: *Ls,V*0 ( *∗*) = 2 (an initiation state of each option in optimal option set is denoted by in the Figure). On

*∗*

*O*

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

## 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 A iterations using the option set acquired by* A-MOMI *O.*
3. *If the MDP is deterministic, the option set is at most O*(log *k*) *times larger than the smallest option set pos- sible to solve the MDP within A 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 times. Thus, it runs in polynomial time.

*|S|*

(4) We solve the set cover using a polynomial time approxi- mation algorithm (Chvatal, 1979) which runs in *O*(*n*3), thus run in polynomial time.

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

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

*Proof.* A state *s X*+ reaches optimal within *A* steps by definition. For every state *s X*+, the set cover guar-

*g*

*g*

*∈ S \*

*∈*

antees that we have *Xst* such that *d*(*s, sj*) *< A*. As we generate an option from *sj* to *g*, *sj* reaches to optimal value with 1 step. Thus, *s* reaches to *s*-optimal value within

*∈ C*

*d*(*s, sj*) + 1 *A*. Therefore, every state reaches *s*-optimal value within *A* steps.

*≤*

**Theorem 5.3.** *If the MDP is deterministic, the option set*

single point option we can add to that option to improve

*L* in the second iteration. Therefore, the greedy procedure

*is at most* max

*s∈S Xs*

*times larger than the smallest option*

returns which has *Ls,V* ( ) *Ls,V* ( ) = 0. Therefore, (*Ls,V*0 ( ) *Ls,V*0 ( ))*/*(*Ls,V*0 ( ) *Ls,V*0 ( *∗*)) can be ar- bitrary small non-negative value (i.e. 0).

*∅ − O ∅ − O*

0

0

*O ∅ − O*

*s*1



*∗*

*g ∗*



Figure 10: Example of MIMO where the improvement of a greedy strategy can be arbitrary small compared to the optimal option set.

*set possible to solve the MDP within A iterations.*

*Proof.* Using a suboptimal algorithm by Chvatal (1979)

we get *C* such that *|C| ≤ O*(log *n*)*|C∗|* where ∆ is the maximum size of subsets in *X* . Thus, *|O|* = *|C| ≤ O*(log *n*)*|C∗|* = *O*(log *n*)*|O∗|*.



# Appendix: Experiments

We show the figures for experiments. Figure 11 shows the options found by solving MIMO optimally/suboptimally in four room domain. Figure 12 shows the options in 9x9 grid domain.



* 1. optimal *k* = 1 (b) optimal *k* = 2 (c) optimal *k* = 3 (d) optimal *k* = 4



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



* 1. Betweenness (j) Eigenoptions

Figure 11: 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.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | *G* |



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

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

*G*

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

Figure 12: 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.