SKILL DISCOVERY WITH WELL-DEFINED OBJECTIVES

Yuu Jinnai, David Abel, Jee Won Park
Department of Computer Science
Brown University
Providence, USA

D Ellis Hershkowitz
Computer Science Department
Carnegie Mellon University
Pittsburgh, USA

Michael L. Littman, George Konidaris
Department of Computer Science
Brown University
Providence, USA

ABSTRACT

While many skill discovery methods have been proposed to accelerate learning and planning, most are based on heuristic methods without clear connections to how the skills impact the agent's objective. As such, the conditions under which the algorithms are effective is often unclear. We claim that we should pursue skill discovery algorithms with explicit relationships to the objective of the agent to understand in what scenarios skill discovery methods are useful. We analyze two scenarios, planning and reinforcement learning and show that we are able to give bounds to the performance of the option discovery algorithms. For planning, we show that the problem of finding a set of options which minimizes the planning time is NP-hard, and give a polynomial-time algorithm that is approximately optimal under certain conditions. For reinforcement learning, we target goal-based tasks with sparse reward where the agent has to navigate through the state-space to reach the goal state without any reward signals other than the goal state. We show that the difficulty of discovering a distant rewarding state in an MDP is bounded by the *expected cover time* of a random walk over the graph induced by the MDP's transition dynamics. We therefore propose an algorithm which finds an option which provably diminishes the expected cover time.

1 INTRODUCTION

An appropriate set of *skills*, or temporally extended actions, can significantly improve the performance of an agent in many scenarios (Sutton et al., 1999). Thus, many heuristic algorithms have proposed to discover skills based on intuitive descriptions of useful skills (Iba, 1989; McGovern & Barto, 2001; Menache et al., 2002; Stolle & Precup, 2002; Şimşek & Barto, 2004; Şimşek et al., 2005; Şimşek & Barto, 2009; Konidaris & Barto, 2009; Machado et al., 2017; Eysenbach et al., 2019). While empirical results show that these algorithms are useful in some scenarios, the conditions 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.

In order to discover options that are guaranteed to be useful, we claim that we should develop skill discovery algorithms which have an explicit connection to the objective of the agent. In this way, we can analytically evaluate the performance of the skill discovery algorithms instead of relying solely on empirical evaluations on benchmark tasks. For example, one can develop an approximate algorithm with a lower bound on performance improvement over an agent without options.

We show the analysis on two scenarios, planning and reinforcement learning by Jinnai et al. (2018; 2019). We show that by explicitly targeting the objective function of the agent, it is possible to give a guarantee on how much the algorithms improve the agent's objective. For planning, we show that the task of finding an option set which minimizes the planning time is NP-hard. Then, we provide

an approximate algorithm with performance guarantees under certain conditions. For reinforcement learning, we show that minimizing the *expected cover time*—the number of steps required for a random walk to visit every state Broder & Karlin (1989)—reduces the expected number of steps required to reach an unknown rewarding state. We introduce an option discovery method that explicitly aims to minimize the expected cover time and show that the algorithm provably diminishes it.

2 Objective Functions for Skill Discovery

We describe our approach to two objectives: planning and exploration in reinforcement learning. We show that by explicitly targeting the appropriate objective, our analyses result in theoretical guarantees on the utility of the options.

2.1 Finding Options that Minimize Planning Time

First, we consider a planning problem with the value iteration algorithm. We formalize what it means to find the set of options that is optimal for planning. More precisely, we consider the problem of finding a subset of options from a candidate set of options so that planning converges within a given iteration limit:

Definition 1 MOMI (MinOptionMaxIteration):
Given *an MDP $M = (\mathcal{S}, \mathcal{A}, R, T, \gamma)$, a non-negative real-value ϵ, a candidate option set $\mathcal{O}'$, and an integer ℓ,* **return** *$\mathcal{O}$ minimizing $|\mathcal{O}|$ subject to $L(\mathcal{O}) \leq \ell$ and $\mathcal{O} \subseteq \mathcal{O}'$, where $L(\mathcal{O})$ is the number of value iteration passes to solve the MDP using the option set $\mathcal{O}$.*

MOMI has the following complexity results:

Theorem 1.

1. *MOMI is $\Omega(\log n)$ hard to approximate even for deterministic MDPs unless $P = NP$.*
2. *MOMI is $2^{\log^{1-\epsilon} n}$-hard to approximate for any $\epsilon > 0$ even for deterministic MDP unless $NP \subseteq DTIME(n^{poly \log n})$.*

Here we describe the outline of the proof (see the Appendix for the full description). The proof is by reduction from the label cover and the set cover problem respectively to a special case of the problem where the set of options are constrained to be *point options*. A point option is a type of option which has exactly one state in initiation set and one state with termination probability set to one. Even for this limited setting, finding a set of point options is $2^{\log^{1-\epsilon} n}$-hard to approximate, and $\Omega(\log n)$-hard to approximate even for deterministic tasks. As we showed the inapproximability results for the special case, MOMI is also NP-hard to approximate. *Thus, Finding an optimal set of options for planning is NP-hard in general*.

This inapproximability results suggest that efficient option discovery algorithms only exist in a more restricted setting that the above cases. We now present a polynomial-time algorithm A-MOMI for approximately computing the optimal set of point options for tasks with bounded return and goal states. The overview of the procedure is as follows.

1. Compute $d : \mathcal{S} \times \mathcal{S} \to \mathbb{Z}_{\geq 0}$ for every state pair where d is the number of iterations for s_i to reach ϵ-optimal if we add a point option from s_j to g, minus one.
2. For every state s_i, compute a set of states X_{s_i} within $\ell - 1$ distance of reaching s_i. The set X_{s_i} represents the states that converge within ℓ steps if we add a point option from s_i to g.
3. Let $\mathcal{X}$ be a set of X_{s_i} for every $s_i \in \mathcal{S} \setminus X_g^+$, where X_g^+ is a set of states that converges within ℓ without any options (thus can be ignored).
4. Solve the set-cover optimization problem to find a set of subsets that covers the entire state set using the approximate algorithm by Chvatal (1979). This process corresponds to finding a minimum set of subsets $\{X_{s_i}\}$ that makes every state in $\mathcal{S}$ converge within ℓ steps.
5. Generate a set of point options with initiation states set to one of the center states in the solution of the asymmetric k-center, and termination states set to the goal.

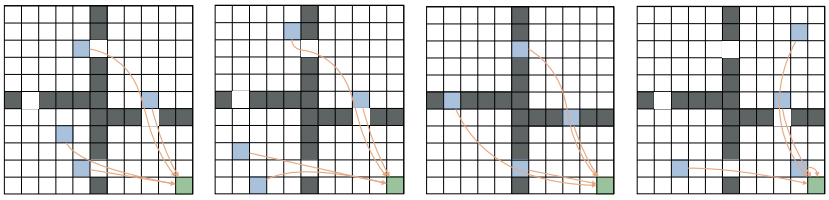
The algorithm has the following properties:

Theorem 2.

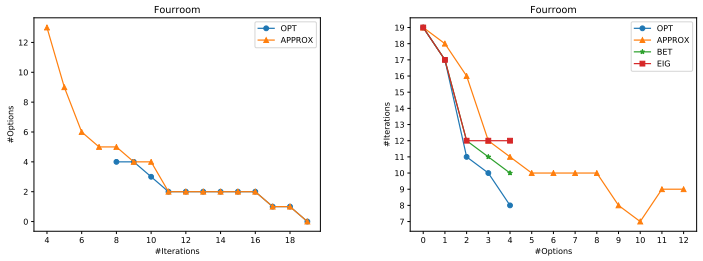
1. A-MOMI *runs in polynomial time.*
2. *The MDP will be solved within ℓ iterations using the option set acquired by* A-MOMI.
3. *If the MDP is deterministic, the option set is at most $O(\log k)$ times larger than the smallest option set possible to solve the MDP within ℓ iterations.*

See the Appendix for the proof. To our knowledge, our method is the first option discovery algorithm with a performance guarantee.

We also consider MIMO, the complementary problem of finding a set of k options that minimize the number of iterations until convergence. The problem is also NP-hard and exists a polynomial-time approximate algorithm, A-MIMO. See the Appendix for the proof.



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



(e) Four Room (MOMI) (f) Four Room (MIMO)

Figure 1: (a)–(d) Comparison of the optimal point options with options generated by the approximation algorithm. 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. (e)–(f) Figures show the number of options generated by A-MOMI and A-MIMO. OPT: an optimal set of options. APPROX: a bounded suboptimal set of options generated by A-MIMO an A-MOMI. BET: betweenness options. EIG: eigenoptions.

We empirically evaluated the performance of the approximate algorithms against the optimal option set and two heuristic approaches for option discovery, betweenness options (Şimşek & Barto, 2009) and eigenoptions (Machado et al., 2017) in simple grid-world tasks. The results indicate that the approximation algorithm is on par with other heuristic algorithms. While heuristic algorithms have no theoretical performance guarantees (e.g. betweenness options are not necessarily helpful when the task has no bottleneck states), our algorithm offers a performance guarantee in any domain.

2.2 Finding Options that Minimize Learning Time for Hard Exploration Tasks

We now consider reinforcement learning tasks, where the environment model is not available. In particular, we consider how options can improve exploration in goal-based tasks with sparse reward. We model the initial exploratory behavior of a reinforcement learning agent in a sparse reward task by a random walk induced by a fixed stationary distribution. This is because (1) it is a reasonable model for an agent with no prior knowledge of the task and (2) it serves as a worst-case analysis: it is reasonable to assume that efficient exploration algorithms explore faster than the random policy.

We aim to minimize the time required by an agent to explore the task. More precisely, we aim to minimize the *expected cover time*: the expected number of steps required for a random walk to visit all the vertices in a graph (Broder & Karlin, 1989). The expected cover time quantifies how quickly a random walk reaches to a rewarding state.

Theorem 3. *Assume a stochastic shortest path problem to reach a goal state $g \in \mathcal{S}$ where a non-positive reward $r_c \leq 0$ is given for non-goal states and $\gamma = 1$. Let P be a random walk transition matrix: $P(s, s') = \sum_{a \in A} \pi(s) T(s, a, s')$:*

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

where $C(G)$ is the expected cover time of the graph G.

See the Appendix for the proof. The theorem suggests that *the smaller the expected cover time, the easier exploration tends to be*. Now the question is how to reduce the expected cover time of the random walk without prior information about the task.

We now present Covering option, an algorithm which discovers options that minimize the expected cover time. The algorithm is approximate since the problem of finding such a set of options is computationally intractable; even a good solution is hard to find due to the Braess's paradox (Braess, 1968; Braess et al., 2005), which states that the expected cover time does not monotonically decrease as edges are added to the graph. Thus, expected cover time is often minimized indirectly via maximizing algebraic connectivity (Fiedler, 1973; Chung, 1996). The expected cover time is upper bounded by a quantity involving the algebraic connectivity, and by maximizing it the bound can be minimized (Broder & Karlin, 1989). As adding a set of edges to maximize the algebraic connectivity is still NP-hard (Mosk-Aoyama, 2008), we use the approximation method by Ghosh & Boyd (2006). The algorithm is as follows:

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

The algorithm is guaranteed to reduce the upper bound of the expected cover time:

Theorem 4. *Assume that a random walk induced by a policy π is a uniform random walk and the multiplicity of the second smallest eigenvalue of $\mathcal{L}$ is one. Adding the two options identified by the algorithm improves the upper bound of the cover time:*

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

where $\mathbb{E}[C(G')]$ is the expected cover time of the resulting random walk, $F = \frac{(v_i - v_j)^2}{6/(\lambda_3 - \lambda_2) + 3/2}$, v_i, v_j are the maximum and minimum values of the Fiedler vector, and λ_2 is the second smallest eigenvalue of $\mathcal{L}$, and n is the number of states. If the multiplicity of the second smallest eigenvalue is greater than one, then adding any single option cannot improve the bound.

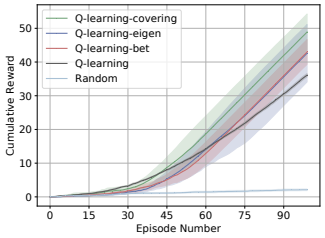
See the Appendix for the proof. Note that the procedure is similar to eigenoptions, proposed by Machado et al. (2017). Both algorithms use the eigenvectors of the Laplacian matrix to generate options. While eigenoptions have no performance guarantees, by explicitly targeting an objective we are able to derive a lower bound on improving the expected cover time and also achieve better empirical performance (Fig 2). Table 2a shows our preliminary results on comparing the expected cover time on simple tabular domains. Our algorithm successfully generates a set of options which reduce the cover time more than the eigenoptions (Machado et al., 2017). In addition, Covering option is fast to compute as it only needs to compute the Fiedler vector. Although computing the whole graph spectrum is a computationally complex matrix operation, the Fiedler vector can be computed efficiently even for very large graphs (Koren et al., 2002).

We now evaluate the utility of each type of discovered options when learning. We used Q-learning (Watkins & Dayan, 1992) ($\alpha = 0.1, \gamma = 0.95$) for 100 episodes of 100 timesteps each and generated 8 options with each algorithms using the adjacency matrix representing the state-transition of the MDP. Figure 2 shows the comparison of accumulated rewards averaged over 5 runs. In all experiments, Covering option outperformed or was on par with eigenoptions.

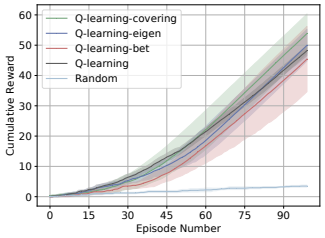
fourroom	λ_2	Cover Time
Covering option	**0.065**	**672.0**
Eigenoptions	0.054	695.9
No options	0.023	1094.8

9x9 grid	λ_2	Cover Time
Covering option	**0.24**	**258.6**
Eigenoptions	0.19	261.5
No options	0.12	460.5

(a) λ_2 and Expected Cover Time



(b) four-room



(c) 9x9 grid

Figure 2: (a) Comparison of the algebraic connectivity and the expected cover time. For Covering option and eigenoptions we add 8 options. (b)–(c) Comparison of performance with different option generation methods. Options are generated offline from the adjacency matrix for four-room and 9x9grid. Reward information is not used for generating options.

3 Related Work

While many option discovery algorithms are relying on a heuristic, several works have proposed methods with well-defined objectives.

Several works have proposed learning the policy and the termination condition of the option by gradient descent using the observed rewards (Mankowitz et al., 2016; Bacon et al., 2017; Harb et al., 2018). Bacon et al. (2017) proposed the option-critic framework and generated options which directly minimize the expected accumulative reward (i.e. the objective of the agent). Harb et al. (2018) proposed to generate options which minimize the sum of expected accumulative reward and the deliberation cost (Simon, 1957) using the option-critic framework (Bacon et al., 2017). The method successfully sped up the learning time by taking into account of the deliberation cost to prefer options with long duration. As they require the reward information, options discovered are task-dependent. Levy et al. (2019) proposed an architecture to learn goal-conditioned policies (i.e. options) to reach certain goal states. They showed that the method can speed up the learning even in long-horizon problems by discovering short horizon subtasks automatically. 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.

Several works have shown empirically that adding a particular set of options or macro-operators can speed up planning algorithms (Sutton & Barto, 1998; Hauskrecht et al., 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. As in reinforcement learning, how to find efficient temporal abstractions for planning automatically remains an open question.

4 Conclusions

In this paper, we analyzed two scenarios, planning and reinforcement learning. For planning, we considered the problem of minimizing the size of the option set given a maximum number of iterations (MOMI) and showed that the problem is computationally intractable. We described a polynomial-time approximate algorithm for solving MOMI under certain conditions. For reinforcement learning, we proposed Covering option and showed that it has a guarantee on how much it improves the expected cover time of a random walk. These theoretical guarantees are available because the skill discovery algorithms are directly tailored to the objective of the agent.

There are multiple directions to pursue. First, developing a theory and an algorithm for multitask planning is future work. While the NP-hardness for generating optimal options in Sec. 2.1 implies that it is not useful to generate options for single-task planning since it takes more computational

effort to find the options than to solve the task, it may be useful for multitask planning where the agent can use the acquired skills for future tasks. Second, we aim to scale up the framework of automatically discovering skills for reinforcement learning to continuous state-space tasks. Recent work by Wu et al. (2019) showed that the spectral analysis can be applied to continuous state-space MDPs. Combining their method with option discovery algorithms may enable to discover skills in continuous state-space MDPs. Third, our analysis for reinforcement learning assumes that the state-transition graph is an unweighted undirected graph. This is a severe simplification for MDPs and we plan to extend our method to weighted directed graphs.

REFERENCES

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 1726–1734, 2017.

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

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

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

Emma Brunskill and Lihong Li. PAC-inspired option discovery in lifelong reinforcement learning. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 316–324, 2014.

Fan RK Chung. *Spectral graph theory*. American Mathematical Society, 1996.

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

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *Proceedings of the Seventh International Conference on Learning Representations*, 2019.

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

Arpita Ghosh and Stephen Boyd. Growing well-connected graphs. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 6605–6611. IEEE, 2006.

Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option: Learning options with a deliberation cost. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Hierarchical solution of Markov decision processes using macro-actions. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 220–229. Morgan Kaufmann Publishers Inc., 1998.

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

Yuu Jinnai, David Abel, D Ellis Hershkowitz, Michael Littman, and George Konidaris. Finding options that minimize planning time. *arXiv preprint arXiv:1810.07311*, 2018.

Yuu Jinnai, Jee Won Park, David Abel, and George Konidaris. Discovering options for exploration by minimizing cover time. *arXiv preprint arXiv:1903.00606*, 2019.

Nicholas K Jong, Todd Hester, and Peter Stone. The utility of temporal abstraction in reinforcement learning. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*, pp. 299–306. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

George Konidaris. Constructing abstraction hierarchies using a skill-symbol loop. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 1648–1654, 2016.

George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Proceedings of the 22nd Conference on Advances in Neural Information Processing Systems*, pp. 1015–1023, 2009.

Yehuda Koren, Liran Carmel, and David Harel. Ace: A fast multiscale eigenvectors computation for drawing huge graphs. In *Proceedings of the IEEE Symposium on Information*, pp. 137–144, 2002.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. In *Proceedings of the Seventh International Conference on Learning Representations*, 2019.

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

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

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

Amy McGovern and Andrew G Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

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

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

David Silver and Kamil Ciosek. Compositional planning using optimal option models. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

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

Özgür Şimşek and Andrew G Barto. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 95. ACM, 2004.

Özgür Şimşek and Andrew G Barto. Skill characterization based on betweenness. In *Proceedings of the Advances in neural information processing systems*, pp. 1497–1504, 2009.

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

Alec Solway, Carlos Diuk, Natalia Córdova, Debbie Yee, Andrew G Barto, Yael Niv, and Matthew M Botvinick. Optimal behavioral hierarchy. *PLoS computational biology*, 10(8):e1003779, 2014.

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

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

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

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Yifan Wu, George Tucker, and Ofir Nachum. The Laplacian in RL: Learning representations with efficient approximations. In *Proceedings of the Seventh International Conference on Learning Representations*, 2019.

SKILL DISCOVERY WITH WELL-DEFINED OBJECTIVES (APPENDIX)

Yuu Jinnai, David Abel, Jee Won Park
Department of Computer Science
Brown University
Providence, USA

D Ellis Hershkowitz
Computer Science Department
Carnegie Mellon University
Pittsburgh, USA

Michael Littman, George Konidaris
Department of Computer Science
Brown University
Providence, USA

1 FINDING OPTIONS THAT MINIMIZE PLANNING TIME

Theorem 1.

1. *MOMI is $\Omega(\log n)$ hard to approximate even for deterministic MDPs unless $P = NP$.*
2. *MOMI is $2^{\log^{1-\epsilon} n}$-hard to approximate for any $\epsilon > 0$ even for deterministic MDP unless NP $\subseteq DTIME(n^{poly \log n})$.*

Proof. First, we show Theorem 4.1 by a reduction from the set cover problem to MOMI with deterministic MDP. We consider two computational problems:

1. MINOPTIONMAXITER (MOMI): Which set of options let value iteration converge in at most ℓ 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.

Definition 1 MOMI: *The* MINOPTIONMAXITER *problem:*
Given *an MDP M, a non-negative real-value ϵ, and an integer ℓ,* **return** *$\mathcal{O}$ that minimizes $|\mathcal{O}|$ subject to $\mathcal{O} \subseteq \mathcal{O}_p$ and $L(\mathcal{O}) \leq \ell$.*

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

Definition 2 OI-DEC:
Given *an MDP M, a non-negative real-value ϵ, and integers k and ℓ,* **return** *'Yes' if the there exists an option set $\mathcal{O}$ such that $\mathcal{O} \subseteq \mathcal{O}_p$, $|\mathcal{O}| \leq k$ and $L(\mathcal{O}) \leq \ell$. 'No' otherwise.*

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

Definition 3 SetCover-DEC:
Given *a set of elements $\mathcal{U}$, a set of subsets $\mathcal{X} = \{X \subseteq \mathcal{U}\}$, and an integer k,* **return** *'Yes' if there exists a cover $\mathcal{C} \subseteq \mathcal{X}$ that $\bigcup_{X \in \mathcal{C}} X = \mathcal{U}$ and $|\mathcal{C}| \leq k$. 'No' otherwise.*

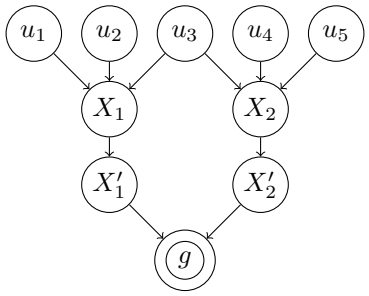


Figure 1: Reduction from SetCover-DEC to OI-DEC. The example shows the reduction from an instance of SetCover-DEC which asks if we can pick two subsets from $\mathcal{X} = \{X_1, X_2\}$ where $X_1 = \{1, 2, 3\}, X_2 = \{3, 4, 5\}$ to cover all elements $\mathcal{U} = \{1, 2, 3, 4, 5\}$. The SetCover-DEC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

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

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

□

For Theorems 4.2 and 4.3 we reduce our problem to the Min-Rep, problem, originally defined by **?**. Min-Rep is a variant of the better studied label cover problem **?** and has been integral to recent hardness of approximation results in network design problems **??**. 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 4 Min-Rep:
Given *a bipartite graph $G = (A \cup B, E)$ and alphabets Σ_A and Σ_B for the left and right sides of G respectively. Each $e \in E$ has associated with it a set of pairs $\pi_e \subseteq \Sigma_A \times \Sigma_B$ which satisfy it.* **Return** *a pair of assignments $\gamma_A : A \to \mathcal{P}(\Sigma_A)$ and $\gamma_B : B \to \mathcal{P}(\Sigma_B)$ such that for every $e = (A_i, B_j) \in E$ there exists an $(a, b) \in \pi_e$ such that $a \in \gamma_A(A_i)$ and $b \in \gamma_B(B_j)$. The objective is to minimize $\sum_{A_i \in A} |\gamma_A(A_i)| + \sum_{B_j \in B} |\gamma_B(B_j)|$.*

We illustrate a feasible solution to an instance of Min-Rep in Figure 2.

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

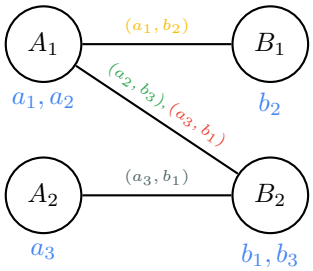


Figure 2: An instance of Min-Rep with $\Sigma_A = \{a_1, a_2, a_3\}$ and $\Sigma_B = \{b_1, b_2, b_3\}$. Edge e is labeled with pairs in π_e. Feasible solution (γ_A, γ_b) illustrated where $\gamma_A(A_i)$ and $\gamma_B(B_j)$ below A_i and B_j in blue. Constraints colored to coincide with stochastic action colors in Figure 3.

Lemma 1 (? ?). *Unless NP $\subseteq$ DTIME($n^{poly \log n}$), Min-Rep admits no $2^{\log^{1-\varepsilon} \tilde{n}}$ polynomial-time approximation algorithm for any $\varepsilon > 0$.*

As a technical note, we emphasize that all relevant quantities in Min-Rep are polynomially-bounded. In Min-Rep we have $|\Sigma_A|, |\Sigma_B| \leq \tilde{n}^{c'}$ for constant c'. It immediately follows that $\sum_e |\pi_e| \leq n^c$ for constant c.

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

1.2 Hardness of Approximation of MOMI

We now show our hardness of approximation of $2^{\log^{1-\epsilon} n}$ for MOMI, Theorem 4.2.[1]

We start by describing our reduction from an instance of Min-Rep to an instance of MOMI. 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 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 $G = (A \cup B, E)$, Σ_A, Σ_B and $\{\pi_e\}$. Our instance of MOMI is as follows where $\gamma = 1$ and $l = 3$.[2]

- **State space** We have a single goal state S_g along with states S'_g and S''_g. For each edge e we create a state S_e. Let Sat$_A(e)$ consist of all $a \in \Sigma_A$ such that a is in some assignment

[1] We assume that $\mathcal{O}'$ is a "good" set of options in the sense that there exists some set $\mathcal{O}^* \subseteq \mathcal{O}'$ such that $L(\mathcal{O}^*) \leq \ell$. We also assume, without loss of generality, that $\varepsilon < 1$ throughout this section; other values of ε can be handled by re-scaling rewards in our reduction.

[2] It is easy to generalize these results to $l \geq 4$ by replacing certain edges with paths.

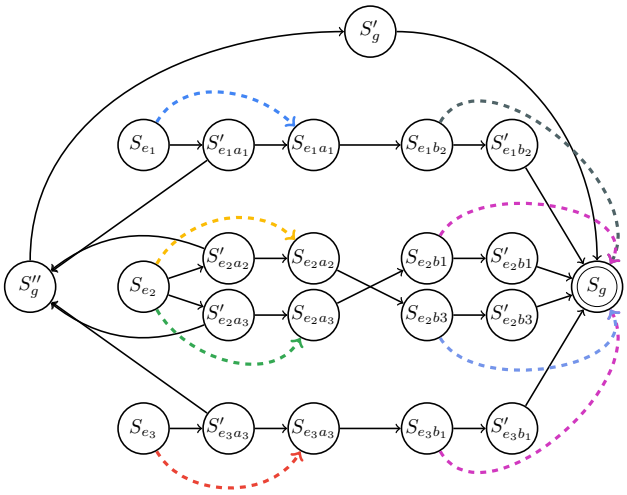


Figure 3: Our MOMI reduction applied to the Min-Rep problem in Figure 2. $e_1 = (A_1, B_1)$, $e_2 = (A_1, B_2)$, $e_3 = (A_2, B_2)$. Actions given in solid lines and each option in $\mathcal{O}'$ represented in its own color as a dashed line from initiation to termination states. Notice that a single option goes from S_{e3b1} and $S_{e_2b_1}$ to S_g.

in π_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'_g to S_g, a single action from S''_g to S'_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} 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 $\mathcal{O}'$ 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 every S_e where e is incident to A_i; The termination set of this option is every S_{ea} where A_i is incident to e; The policy of this option takes the action from S'_{ea} to S_{ea} when in S'_{ea} and the action from S_e to S'_{ea} when in S_e.

  Symmetrically, for every vertex $B_j \in B$ and each $b \in \Sigma_B$ we have an option $O(B_j, b)$: The initiation set of this option is every S_{eb} where e is incident to B_j; The termination set of this option is S_g; The policy of this option takes the action from S_{eb} to S'_{eb} when in S_{eb} and from S'_{eb} to S_g when in S'_{eb}.

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 $MOMI(\text{MR})$ be the MDP output given instance MR of Min-Rep and see Figure 3 for an illustration of our reduction.

Let OPT_{MOMI} be the value of the optimal solution to $MOMI(\text{MR})$ and let OPT_{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 2. *$OPT_{MOMI} \leq OPT_{MR}$*

Proof. Given a solution (γ_A, γ_B) to MR, define $\mathcal{O}_{\gamma_A,\gamma_B} := \{O(v, x) : v \in V(G) \wedge (\gamma_A(v) = x \vee \gamma_B(v) = x)\}$ as the corresponding set of options. Let γ^*_A and γ^*_B be the optimal solutions to MR which is of cost OPT_{MR}.

We now argue that $\mathcal{O}_{\gamma_A^*,\gamma_B^*}$ is a feasible solution to $MOMI(\text{MR})$ of cost OPT_{MR}, demonstrating that the optimal solution to $MOMI(\text{MR})$ has cost at most OPT_{MR}. To see this notice that by construction the MOMI cost of $\mathcal{O}_{\gamma_A^*,\gamma_B^*}$ is exactly the Min-Rep cost of (γ_A^*, γ_B^*).

We need only argue, then, that $\mathcal{O}_{\gamma_A^*,\gamma_B^*}$ is feasible for $MOMI(\text{MR})$ and do so now. The value of every state in $MOMI(\text{MR})$ 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 $\mathcal{O}_{\gamma_A^*,\gamma_B^*}$ guarantees that every S_e will have value 1 after 3 iterations of value iteration. Since (γ_A^*, γ_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 $(\bar{a}, \bar{b}) \in \pi_e$; correspondingly there are options $O(A_i, \bar{a}), O(B_j, \bar{b}) \in \mathcal{O}_{\gamma_A^*,\gamma_B^*}$. It follows that, given options $\mathcal{O}_{\gamma_A^*,\gamma_B^*}$ from, S_e one can take option $O(A_i, \bar{a})$ then the action from $S_{e\bar{a}}$ to $S_{e\bar{b}}$ 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 $MOMI(\text{MR})$ corresponds to a solution to MR. For the remainder of this section $\gamma_A^{\mathcal{O}}(A_i) := \{a : O(A_i, a) \in \mathcal{O}\}$ and $\gamma_B^{\mathcal{O}}(B_j) := \{b : O(B_j, b) \in \mathcal{O}\}$ is the Min-Rep solution corresponding to option set $\mathcal{O}$.

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

Proof. Notice that by construction the Min-Rep cost of $(\gamma_A^{\mathcal{O}}, \gamma_B^{\mathcal{O}})$ is exactly $|\mathcal{O}|$. Thus, we need only prove that $(\gamma_A^{\mathcal{O}}, \gamma_B^{\mathcal{O}})$ 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 $(\gamma_A^{\mathcal{O}}, \gamma_B^{\mathcal{O}})$ satisfies e. Since $\mathcal{O}$ is a feasible solution to $MOMI(\text{MR})$ 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 $MOMI(\text{MR})$ 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 $(a, b) \in \pi_e$. It follows that $a \in \gamma_A^{\mathcal{O}}(A_i)$ and $b \in \gamma_B^{\mathcal{O}}(B_j)$. But since $(a, b) \in \pi_e$, we then know that e is satisfied. Thus, every edge is satisfied and so $(\gamma_A^{\mathcal{O}}, \gamma_B^{\mathcal{O}})$ is a feasible solution to MR. □

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

We begin by noting that the relevant quantities in $MOMI(\text{MR})$ are polynomially-bounded. Notice that the number of states n in the MDP in $MOMI(\text{MR})$ is at most $O(\tilde{n}^2|\Sigma_A||\Sigma_B|) = \tilde{n}^c$ for some fixed constant c by the aforementioned assumption that Σ_A and Σ_B are polynomially-bounded in $\tilde{n}$.[3]

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

We first argue that our algorithm is polynomial-time in $\tilde{n}$. However, notice that for each vertex, we create a polynomial number of states. Thus, the number of states in $MOMI(\text{MR})$ is polynomially-bounded in $\tilde{n}$ and so $\mathcal{A}_{MOMI}$ runs in time polynomial in $\tilde{n}$. A polynomial runtime of our algorithm immediately follows.

We now 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 $\mathcal{A}_{MOMI}$ and then Lemma 2, we have that $(\gamma_A^{\mathcal{O}}, \gamma_B^{\mathcal{O}})$ is a

[3]It is also worth noticing that since we create at most $O(\tilde{n}|\Sigma_A| + \tilde{n}|\Sigma_B|)$ options, the total number of options in $\mathcal{O}'$ is at most polynomial in $\tilde{n}$.

feasible solution for MR with cost

$$\begin{aligned}\text{cost}_{\text{Min-Rep}}(\gamma_A^{\mathcal{O}}, \gamma_B^{\mathcal{O}}) &= |\mathcal{O}| \\ &\leq 2^{\log^{1-\varepsilon} n} \text{OPT}_{MOMI} \\ &\leq 2^{\log^{1-\varepsilon} n} \text{OPT}_{\text{MR}}\end{aligned}$$

Thus, $(\gamma_A^{\mathcal{O}}, \gamma_B^{\mathcal{O}})$ 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 $MOMI$(MR). Now recalling that $n \leq \tilde{n}^c$ for fixed constant c. We therefore have that $(\gamma_A^{\mathcal{O}}, \gamma_B^{\mathcal{O}})$ is a $2^{\log^{1-\varepsilon} \tilde{n}^c} = 2^{c^{1-\varepsilon} \log^{1-\varepsilon} \tilde{n}} \leq c' \cdot 2^{\log^{1-\varepsilon} \tilde{n}}$ approximation for a constant c'. Choosing ε sufficiently small, we have that $c' \cdot 2^{\log^{1-\varepsilon} \tilde{n}} \leq 2^{\log^{1-\varepsilon'} \tilde{n}}$ 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 MOMI cannot be $2^{\log^{1-\epsilon} n}$-approximated. □

Theorem 2. A-MOMI *has the following properties:*

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

Theorem 2.1. A-MOMI *runs in polynomial time.*

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

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

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

(2), (3), and (5) Immediate. □

Theorem 2.2. A-MOMI *guarantees that the MDP is solved within ℓ iterations using the option set $\mathcal{O}$.*

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

Theorem 2.3. *If the MDP is deterministic, the option set is at most $\max_{s \in \mathcal{S}} X_s$ times larger than the smallest option set possible to solve the MDP within ℓ iterations.*

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

1.3 A-MIMO

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

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

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

Theorem 2.4. A-MIMO *runs in polynomial time.*

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

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

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

(3) Immediate.

Therefore, A-MIMO runs in polynomial time. □

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

Lemma 4. *There exists an optimal option set for MIMO and MOMI with all terminal state set to the goal state.*

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

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

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

Lemma 5.

$$d(s, \mathcal{C}) \leq \min_{s' \in \mathcal{C}} d(s, s'). \tag{1}$$

Using this, we show the following proposition.

Theorem 2.5. *The number of iterations to solve the MDP using the acquired options is upper bounded by $P(\mathcal{C})$.*

Proof. $P(\mathcal{C}) = \max_{s \in S} \min_{c \in \mathcal{C}} d(s, c) \geq \max_{s \in S} d(s, \mathcal{C}) = L(\mathcal{O})$ (using Equation 2). Thus $P(\mathcal{C})$ is an upper bound for $L(\mathcal{O})$. □

The reason why $P(\mathcal{C})$ does not always give us the exact number of iterations is because adding two options starting from s_1, s_2 may make the convergence of s_0 faster than $d(s_0, s_1)$ or $d(s_0, s_2)$. Example: Figure 5 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

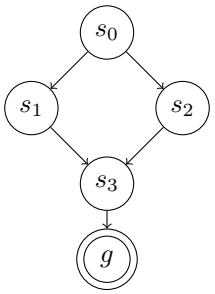


Figure 4: An example of an MDP where $d(s, \mathcal{C}) < \min_{s' \in \mathcal{C}} d(s, s')$. Here the transition induced by the optimal policy is stochastic, thus from s_0 one may go to s_1 and s_2 by probability 0.5 each. Either adding an option from s_1 or s_2 to g does not make the convergence faster, but adding both makes it faster.

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

Theorem 2.6.

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

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

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

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

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

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

Lemma 7.

$$d(s, \mathcal{C}) \leq \min_{s' \in \mathcal{C}} d(s, s'). \tag{2}$$

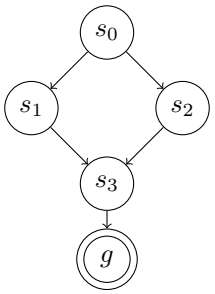


Figure 5: An example of an MDP where $d(s, \mathcal{C}) < \min_{s' \in \mathcal{C}} d(s, s')$. Here the transition induced by the optimal policy is stochastic, thus from s_0 one may go to s_1 and s_2 by probability 0.5 each. Either adding an option from s_1 or s_2 to g does not make the convergence faster, but adding both makes it faster.

Using this, we show the following proposition.

Theorem 2.7. *The number of iterations to solve the MDP using the acquired options is upper bounded by $P(\mathcal{C})$.*

Proof. $P(\mathcal{C}) = \max_{s \in S} \min_{c \in \mathcal{C}} d(s, c) \geq \max_{s \in S} d(s, \mathcal{C}) = L(\mathcal{O})$ (using Equation 2). Thus $P(\mathcal{C})$ is an upper bound for $L(\mathcal{O})$. □

The reason why $P(\mathcal{C})$ does not always give us the exact number of iterations is because adding two options starting from s_1, s_2 may make the convergence of s_0 faster than $d(s_0, s_1)$ or $d(s_0, s_2)$. Example: Figure 5 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 2 holds if the MDP is deterministic. That is, $d(s, \mathcal{C}) = \min_{s' \in \mathcal{C}} d(s, s')$ for deterministic MDP.

Theorem 2.8.

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

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

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

2 Finding Options that Minimize Learning Time for Hard-Exploration Tasks

Theorem 3. *Assume a stochastic shortest path problem to reach a goal g where a non-positive reward $r_c \leq 0$ is given for non-goal states and $\gamma = 1$. Let P be a random walk transition matrix: $P(s, s') = \sum_{a \in A} \pi(s) T(s, a, s')$:*

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

where $C(G) = \max_{s \in S} C_s(G)$ and $C_s(G)$ is a cover time of a transition matrix P starting from state s.

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

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

□

Theorem 4. *Assume that a random walk induced by a policy π is a uniform random walk. Adding two options by the algorithm improves the upper bound of the cover time if the multiplicity of the second smallest eigenvalue is one:*

$$\mathbb{E}[C(G')] \leq \frac{n^2 \ln n}{\lambda_2(\mathcal{L}) + F}(1 + o(1)), \tag{3}$$

where $\mathbb{E}[C(G')]$ is the expected cover time of the augmented graph, $F = \frac{(v_i - v_j)^2}{6/(\lambda_3 - \lambda_2 + 3/2}$, and v_i, v_j are the maximum and minimum values of the Fiedler vector. If the multiplicity of the second smallest eigenvalue is more than one, then adding any single option cannot improve the algebraic connectivity.

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

$$\lambda_2 \geq \lambda_2 + \frac{(v_i - v_j)^2}{6/(\lambda_3 - \lambda_2) + 3/2}, \tag{4}$$

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

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

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

$$\frac{n^2 \ln n}{\lambda_2 + \frac{(v_i - v_j)^2}{6/(\lambda_3 - \lambda_2) + 3/2}}(1 + o(1)) < \frac{n^2 \ln n}{\lambda_2}(1 + o(1)), \tag{5}$$

thus the upper bound is guaranteed to decrease.

Assume the second smallest eigenvalue is more than one. Then, $\lambda_2(\mathcal{L}) = \lambda_3(\mathcal{L})$. From eigenvalue interlacing **?**, for any edge insertion, $\lambda_2(\mathcal{L}) \leq \lambda_2(\mathcal{L}') \leq \lambda_3(\mathcal{L})$. Thus, $\lambda_2(\mathcal{L}') = \lambda_2(\mathcal{L})$. □