PAC-inspired Option Discovery in Lifelong Reinforcement Learning

Emma Brunskill EBRUNSKILL@CS.CMU.EDU
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213

Lihong Li LIHONGLI@MICROSOFT.COM
Microsoft Research, One Microsoft Way, Redmond, WA 98052

Abstract

A key goal of AI is to create lifelong learning agents that can leverage prior experience to improve performance on later tasks. In reinforcement learning problems, one way to summarize prior experience for future use is through options, which are behaviorally extended actions (subpolicies) for how to behave. Options can then be used to potentially accelerate learning in new reinforcement learning tasks. In this work we provide the first formal analysis of the sample complexity, a measure of learning speed, of reinforcement learning with options. This analysis helps shed light on some interesting prior empirical results on when and how options may accelerate learning. We then quantify the benefit of options in reducing sample complexity of a lifelong learning agent. Finally, the new theoretical insights inspire a novel option-discovery algorithm that aims at minimizing overall sample complexity in lifelong reinforcement learning.

1. Introduction

A key goal of AI is to create agents that can quickly learn to act well in new domains. It seems extremely likely that humans achieve this by leveraging a hierarchical structure for complex tasks: learning then involves selecting among temporally extended actions like "drive to the airport," "provide 4 weeks of chemotherapy," or "use robotic arm to pick up block," rather than primitive actions like "press gas for 1 second", "radiate skin voxel for 2 seconds" or "rotate robot arm joint 1 for 1 second." For people, such temporally extended actions often arise from chunking of prior experiences into subpieces that can be reused to improve learning speed and performance on future tasks.

Such observations have inspired prior heuristic algorithms for using temporally extended actions in reinforcement learning (RL) (Sutton et al., 1999; Barto & Mahadevan, 2003), as well as discovering such actions for use in future tasks (Thrun & Mitchell, 1995; Pickett & Barto, 2002; Mannor et al., 2004; Stolle & Precup, 2002; Konidaris & Barto, 2007). Past work has produced encouraging experimental evidence that leveraging similarity in policy substructure across tasks can lead to improved performance.

Despite these promising results, recent empirical investigations (Jong et al., 2008) suggests the need for a deeper understanding of when and how such temporally extended actions speed up learning. In that paper, backed by experimental evidence, the authors argued that augmenting primitive actions with options (a common practice in the literature on options) can only increase the amount of exploration required, thereby slowing learning.

Intrigued by the somewhat surprising findings above, and by the lack of theoretical analysis in the literature, this work aims to start to provide a theoretical foundation for understanding the conditions under which options can accelerate RL, and how such options might be obtained automatically during lifelong learning. Our objective differs from the interesting recent work of Mann & Mannor (2014), who prove that leveraging a given set of options can reduce *computation complexity of planning*. In contrast, we focus on analyzing the impact of options on *sample complexity*, which is a measure of learning efficiency that counts the number of steps on which an algorithm may select a non-near-optimal action (Kakade, 2003; Strehl et al., 2006).

We make several contributions in this work. First, we present the first formal analysis of how options can improve the sample complexity of RL. This analysis provides a solid theoretical foundation to explain of some of the prior empirical successes (and surprises). Second, we discuss the conditions under which we can discover a set of options

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

during lifelong learning across a series of RL tasks that is guaranteed to reduce or match the sample complexity of single-task learning in future RL tasks. Third, we present a two-phase lifelong learning algorithm that performs option discovery and then leverages the prior options. The key of the algorithm is a greedy option discovery method that directly leverages the results of our PAC analysis to construct options given experience in tasks in the first phase. Finally, this approach enables a significant performance improvement over RL without options on a prior benchmark simulation domain, as well as a strong option discovery baseline, and does almost as well as using a set of expert-constructed options. Under certain conditions, our discovered options are guaranteed not to lead to particular form of negative transfer: using the discovered options will not result in a worse sample complexity bound for later tasks, compared to that with primitive actions. This is important because negative transfer, where prior knowledge harms future performance, is a common challenge in lifelong learning.

2. Background and Setting

Much of existing RL research is for single-task learning in a Markov decision process (MDP). An MDP is a tuple $M = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are sets of state/actions, $P(s'|s,a)$ the transition probabilities, $R(s,a)$ the reward function, and $\gamma \in [0,1)$ the discount. A policy π maps states to actions. Its value in state s, $V^\pi(s)$, is the expected total discounted rewards of following π starting in s. The Bellman operator $V^\pi(s) = R(s,\pi(a)) + \gamma \sum_{s'} P(s'|s,\pi(a))V^\pi(s)$ can be used to compute the value of π, or find an optimal policy and value for a given MDP. In RL, the transition and/or reward function is initially unknown, and an algorithm typically aims to find a policy to maximize the expected total rewards.

In MDPs, the time to transition between states after taking an action is constant. Semi-Markov decision processes (SMDPs) allow the transition duration (*a.k.a.* waiting time, τ) to be non-constant and stochastic. An SMDP is a tuple $M = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, similar to a MDP, except $P(s', \tau|s,a)$ is the *joint* probability of transitioning to a state s' in τ steps, after taking action a in state s, and $R(s,a)$ is the expected discounted reward accumulated in a transition by taking action a in state s. We focus on SMDPs with finite state/actions and integer-valued waiting time: $S = |\mathcal{S}|$, $A = |\mathcal{A}|$, $\tau \in \mathbb{N}_+$. We denote by N_{sa} the number of next states reachable by taking a in s, and $N = \max_{s,a} N_{sa}$. A policy π in a SMDP is still a mapping from states to actions, whose value of a policy is defined similarly as in MDPs.

One important class of temporally extended actions are "Markov options" (Sutton et al., 1999) that are non-interruptible: when an agent chooses an option, it follows the option policy until it terminates. Formally, an option consists of three components, $o = \langle \mathcal{I}_o, \pi_o, \mathcal{T}_o \rangle$: $\mathcal{I}_o \subseteq \mathcal{S}$ is the set of initiation states where o can be started; π_o is the option policy, and $\mathcal{T}_o \subseteq \mathcal{S}$ is the set of termination states where the option ends.[1] Note that an analogous Bellman operator exists for MDPs with options (as it does for generic SMDPs), $Q(s,a) = r(s,a) + \sum_{s' \in S} p(s'|s,a) max_{a'} Q(s',a')$, where $p(s'|s,a) = \sum_{j=1}^{\infty} P(s', \tau = j|s,a)\gamma^j$ (Puterman, 1994; Sutton et al., 1999), and can be used to efficiently perform planning given a set of option models.

We are interested in the lifelong learning setting, where an agent acts in a series of RL tasks. An agent acts in a series of RL tasks where the t-th task, M_t, is drawn i.i.d. from a unknown distribution ν over an unknown and possibly infinite set $\mathcal{M}$ of MDPs. Similar to Wilson et al. (2007), these tasks share the same states and actions, but may differ in rewards and transition probabilities. We assume there exists a good (unknown) underlying set of options $\bar{\mathcal{O}}$ that is sufficient to achieve near-optimal performance for all MDPs in $\mathcal{M}$,[2]. We seek to discover this or a closely related set, and use that to improve the sample complexity of exploration as the agent continues to act in new tasks. Proofs, where omitted, are provided in the supplementary material.

3. Sample-Efficient Learning in SMDPs

Since we wish to discover options to improve learning efficiency in later tasks, it is necessary to first understand how options impact learning efficiency in single-task RL. As shown by Sutton et al. (1999), an MDP with options is an SMDP. Therefore, we study in this section the sample complexity of RL in SMDPs. We assume, without much loss of generality, that the per-step reward is in $[0,1]$, so the maximum state value, denoted $V_{\max}$, never exceeds $\frac{1}{1-\gamma}$.

We first extend the notion of sample complexity to SMDPs. An RL algorithm $\mathbf{A}$ is interpreted as a *non-stationary* policy mapping histories to actions. Starting in state s_1, at the t-th decision epoch ($t = 1, 2, \ldots$), based on the history $h_t = (s_1, a_1, \tau_1, r_1, s_2, a_2, \tau_2, r_2, \ldots, s_t)$, $\mathbf{A}$ has to choose a_t, and then transitions to a new state s_{t+1} with waiting time τ_t and immediate reward r_t. Here, we use "step" for the actual time, and "epoch" for steps where an action is chosen. In an MDP, $\tau_t \equiv 1$, so every step is an epoch.

Definition 1 *Given history h_t at epoch t, an RL algorithm is viewed as a non-stationary policy, denoted $\mathbf{A}_t$. For any*

[1] Note that in general an option can terminate stochastically (the β function defined by Sutton et al. (1999)). It suffices for our purposes to focus on options that end deterministically, since restricting to deterministic options does not sacrifice optimality.

[2] The set of primitive actions always satisfies this condition.

fixed ϵ, the sample complexity of exploration *(or "sample complexity") of* $\mathbf{A}$ *is* $\sum_t \tau_t \cdot \mathbb{I}\left(V^{\mathbf{A}_t}(s_t) \leq V^*(s_t) - \epsilon\right)$, *where* $\mathbb{I}(C)$ *is the set-indicator function that evaluates to* 1 *if event* C *occurs and* 0 *otherwise.*

In other words, the sample complexity of an algorithm in an SMDP is the number of epochs where it takes non-ϵ-optimal actions, weighted by the (random) weighting time. Clearly, if τ is always 1, the definition is consistent with the counterpart for MDPs, as desired.

Definition 2 *An RL algorithm* $\mathbf{A}$ *is said to be PAC-SMDP (Probably Approximately Correct in SMDPs) if, for any SMDP,* $\epsilon > 0$, *and* $0 < \delta < 1$, *the sample complexity of* $\mathbf{A}$ *is bounded by some function polynomial in* S, A, $1/\epsilon$, $1/\delta$, *and* $1/(1-\gamma)$, *with probability at least* $1-\delta$.

3.1. A Generic PAC-SMDP Theorem

Despite the similarity to MDPs, further conditions are required on the transition probabilities for *any* algorithm to be PAC-SMDP. To see this, consider a non-ϵ-optimal action a in some state s. If the waiting time τ of taking a in state s is infinite, and an algorithm chooses a in s, then its sample complexity is unbounded, according to Definition 1. Intuitively, an action that makes the algorithm wait forever prevents the algorithm from recovering from a sub-optimal decision, thus the infinite sample complexity.

To prevent this pathological case, we assume there exist known constants L and C so that for any (s, a), (i) $\mathbf{E}[\tau] < L$; (ii) the distribution of τ is sub-Gaussian with parameter C, namely, the waiting time τ satisfies $\Pr\left(|\tau - \mathbf{E}[\tau|s,a]| \geq \lambda\right) \leq 2\exp\left(-C\lambda^2\right)$. This assumption is rather mild, covering many common distributions. For instance, if the support of τ is bounded, meaning that $\Pr(\tau < \tau_0) = 1$ for some fixed τ_0, then τ is also sub-Gaussian. Sub-Gaussianness implies the following lemma:

Lemma 1 *If* τ *is sub-Gaussian with parameter* C, *then with probability at least* $1-\delta$, $\tau < L + \frac{1}{\sqrt{C}} \ln \frac{2}{\delta}$.

Prior work on RL with primitive actions (Kearns & Singh, 2002; Brafman & Tennenholtz, 2002; Strehl et al., 2006) proves efficient exploration can be achieved by tracking a set of known state–actions whose rewards and transitions have been estimated to sufficient accuracy, and using this information to guide exploration. We take a similar approach for SMDPs, and define $\mathcal{K}$, a known set of state-action pairs. $\mathcal{K}$ are state–action pairs for which we have a good estimate of their reward and transition distributions, quantified by absolute error for rewards and total variation for transition probabilities. Given a set $\mathcal{K}$ we define a known state–action SMDP, $M_{\mathcal{K}_t}$, whose state–action model parameters are identical to the true SMDP on state–action pairs in $\mathcal{K}$, and have a reward of R_{max} and self-loop dynamics for unknown state–action pairs.

We can now provide a SMDP-analogue of the PAC-MDP result of Strehl et al. (2006).

Theorem 1 *Let* $\mathbf{A}$ *be an algorithm that chooses actions greedily according to its estimated state–action value function, denoted* Q_t *at step t. Define* $V_t = \max_{a\in\mathcal{A}} Q_t(s,a)$ *and* $\pi_t = \arg\max_{a\in\mathcal{A}} Q_t(s,a)$. *In every step t, there is a set of known state–action pairs* $\mathcal{K}_t \subseteq \mathcal{S} \times \mathcal{A}$ *that depends only on the algorithm's history up to t. We assume* $\mathcal{K}_t = \mathcal{K}_{t+1}$, *unless either one of the following happens: (i)* $Q_{t+1} \neq Q_t$, *or (ii)* $(s_t, a_t) \notin \mathcal{K}_t$. *Let* $M_{\mathcal{K}_t}$ *be the known state–action SMDP and* π_t *be the greedy policy with respect to* Q_t. *Suppose that for any* ϵ *and* δ *(both known to the algorithm), the following conditions hold for all steps t with probability at least* $1-\delta/2$:

1. *(Optimism)* $V_t(s_t) \geq V^*(s_t) - \epsilon/4$;
2. *(Accuracy)* $V_t(s_t) - V^{\pi_t}_{M_{\mathcal{K}_t}}(s_t) \leq \epsilon/4$; *and*
3. *(Bounded Surprises) The total number of steps where either an update to the Q-function occurs or the agent visits an unknown state–action pair is bounded by a function* $\zeta = \zeta(\epsilon, \delta)$.

Then, with probability $1-\delta$, *the sample complexity of* $\mathbf{A}$ *is*

$$\mathbb{O}\left(\frac{V_{\max}}{\epsilon}\left(\zeta + \ln\frac{1}{\delta}\right)\left(\frac{\ln\frac{1}{\epsilon(1-\gamma)}}{1-\gamma} + L + \frac{1}{\sqrt{C}}\ln\frac{V_{\max}\zeta}{\epsilon\delta(1-\gamma)}\right)\right).$$

The E^3 (Kearns & Singh, 2002) and RMAX algorithms (Brafman & Tennenholtz, 2002; Kakade, 2003) can be extended to SMDPs and analyzed similarly. For concreteness, we describe SMDP-RMAX, a straightforward extension of RMAX to SMDPs, a variant of which was used in our experiments. It takes as input the states and actions of an SMDP M. At each step, the agent maintains an SMDP M_t with the same states and actions, but where the known state–action pairs' transitions and rewards are maximum-likelihood estimates given the agent's experience, and the unknown state–action pairs' dynamics are set to a self-loop with maximum reward. SMDP-RMAX then uses value iteration to compute an optimal policy π_t for M_t, which can be done efficiently (Puterman, 1994).

Theorem 2 SMDP-RMAX *is PAC-SMDP with a sample complexity of, ignoring logarithmic terms,*

$$\tilde{\mathbb{O}}\left(\frac{V^3_{\max}}{\epsilon^3}\sum_{s,a}\frac{N_{sa}}{(1-\bar{\gamma}_s)^2}\left(\frac{1}{1-\gamma} + L + \frac{1}{\sqrt{C}}\right)\right). \quad (1)$$

where N_{sa} *is the number of reachable next states of* (s,a), $\bar{\gamma}_s = \max_a \sum_\tau \gamma^\tau P(\tau|s,a)$, *and* $P(\tau|s,a) = \sum_{s'} P(s',\tau|s,a)$ *is the marginal waiting-time distribution.*

Note that Equation 1 uses slightly less standard notation of summing over all state–action pairs rather than the SA dependence typical in MDPs. We do so because different states may have access to different numbers of options.

3.2. Applications to MDPs with Options

The general results just derived for SMDPs also apply to MDPs with options, as a MDP M with options $\mathcal{O}$ defines a SMDP $M' = \langle \mathcal{S}', \mathcal{O}, P', R', \gamma \rangle$ (Sutton et al., 1999). The resulting SMDP only must include states where at least one option may be initiated, $\mathcal{S}' = \cup_{o \in \mathcal{O}} \mathcal{I}_o$.

Note the sample complexity bound for using options can be a significant improvement over that with primitive actions. To show this, we compare the sample complexity of RMAX in MDPs (Kakade, 2003), $\tilde{\mathbb{O}}\left(\frac{V_{\max}^3}{\epsilon^3}\frac{|\mathcal{S}||\mathcal{A}|N}{(1-\bar{\gamma}_s)^3}\right)$, to the bound in Theorem 2 (with $S = |\mathcal{S}'|$ and $A = |\mathcal{A}'|$). The bound with options is lower when the effective discount factors $\bar{\gamma}_s$ are smaller than γ and the waiting time is not much larger than $\frac{1}{1-\gamma}$, and/or when the number of state–options pairs is smaller than the number of state–action pairs.

The above observation provides useful insights to understand empirical results with options, and to guide options design in practice. For example, Jong et al. (2008) reported several surprising empirical results that showed adding options as well as primitive actions may worsen learning speed, compared to agents with primitive actions. This is as we would expect given our sample complexity bound Equation 1, which predicts an increase if more state–action tuples are added. In contrast, if the agent can only take options in some states and primitive actions in others, Jong et al. observed a substantial learning speedup. We calculated the sample complexity bound in Theorem 2 using the same options as in their domain. Our numbers are consistent with their empirical results: the bound is 9.6×10^6 for primitive actions everywhere, but only 1.8×10^6 for the restricting some states to options and others to primitive actions (*c.f.*, Figure 6 of Jong et al. (2008)), highlighting the potential benefit of introducing a smaller set of good options to replace some state–primitive–action pairs.

4. Lifelong RL with Option Transfer

4.1. Setup

The prior section suggests that a good set of options can substantially improve the sample complexity of RL. Intuitively, if the agent is acting in a series of RL problems with related structure, the agent may be able to capture this structure through option discovery, and then leverage the resulting options to learn more efficiently in later RL tasks.

We now describe more formally how options may be discovered from previously solved tasks to potentially reduce the sample complexity of learning in future tasks. It is assumed a set $\mathcal{O}^*$ of candidate options are given, but with no assumption on how $\mathcal{O}^*$ is designed. For example, it can be options that take the same action until a certain termination state is reached, or options that last up to a certain steps. As will be made precise later, one needs to control the cardinality of $\mathcal{O}^*$ for nontrivial analysis to be possible.

We denote by $O^* = |\mathcal{O}^*|$ the number of candidate options. Note that $\mathcal{O}^*$ is always finite since the underlying MDPs are finite. Given an MDP M and an option set $\mathcal{O}$, an SMDP M' is induced, as described in Section 3.2. We also need to quantify the quality of an option set: $\mathcal{O}$ is called ε-optimal for M if ε-optimal policies can be found with these options in M; that is, we have $V_M^*(s) - V_{M'}^*(s) \le \varepsilon$ for all $s \in \mathcal{S}'$.

4.2. Lifelong Sample Complexity

The primary objective of this section is to show: (1) under some restrictions, if there exists a small set of options $\bar{\mathcal{O}}$ sufficient to achieve ε-optimal performance in all MDPs $\mathcal{M}$ the agent will encounter, then the agent can discover this set; and (2) to bound the total sample complexity incurred during this lifelong learning with option discovery. Whether option discovery results in a reduction in the sample complexity depends on the properties of $\bar{\mathcal{O}}$, which appears directly in the lifelong sample complexity bound. In the experiments of Section 6, we will provide an example where the sample complexity is significantly improved with discovered options. Our focus here is to formally quantify the sample complexity, so the computational complexity is ignored at the moment. However, the theory to be developed here will inspire a computationally more tractable option discovery algorithm in the next section.

Our analysis makes these assumptions: (1) Tasks are drawn i.i.d. from an unknown distribution ν over $\mathcal{M}$; (2) There is a set $\bar{\mathcal{O}} \subset \mathcal{O}^*$ that is ε-optimal for all MDPs in $\mathcal{M}$, and $\left|\bar{\mathcal{O}}\right| \le \bar{O}$ for some known constant $\bar{O}$; (3) There exists a known threshold $p_{\min} \in (0,1)$ such that: for any $\mathcal{O}' \subset \bar{\mathcal{O}}$ with $|\mathcal{O}'| \le \bar{O}$, if $\mathcal{O}'$ is not ϵ-optimal for $\mathcal{M}$, then, with probability at least $p_{\min}$ under ν, it fails to represent an ϵ-optimal policy of MDPs drawn from $\mathcal{M}$; (4) There exists a known diameter D, such that for every M in $\mathcal{M}$, any state s' is reachable from any s in at most D steps on average;

The first assumption is fairly mild and is common in lifelong learning (*e.g.*, Wilson et al. (2007)). The second formalizes the intuition that option transfer is useful in lifelong learning only if there is a small number of them that can represent near-optimal policies.[3] The third requires that each option in $\bar{\mathcal{O}}$ be needed with some minimum probabil-

[3] It is always possible to find ε-optimal option sets for all MDPs in $\mathcal{M}$, for any $\varepsilon \ge 0$. One can simply include all primitive actions in $\bar{\mathcal{O}}$ to achieve that, although $\bar{\mathcal{O}}$ will be large. In practice, domain knowledge is needed to design a reasonably small $\bar{\mathcal{O}}$.

ity. This assumption essentially rules out options that are needed to represent ε-optimal policies for extremely rare MDPs. One can avoid this assumption and state the results below in a slightly different way, but exposition would be more complicated. The fourth is one of the key assumptions needed in the analysis. The diameter, first defined by Jaksch et al. (2010), measures how many steps it takes, on average, to navigate from a state to another following the shortest path between them. The assumption allows the agent to explore an MDP quickly, if it intends to do so.

To simplify the analysis, a two-phase algorithm is used. Briefly, for tasks encountered in phase 1, the algorithm performs single-task learning, while trying to construct accurate models of these tasks at the same time. Then, based on information collected from phase 1, the algorithm discovers a set of options $\hat{\mathcal{O}} \subset \mathcal{O}^*$ that ϵ-covers the MDPs in phase 1. In phase 2, the algorithm uses the discovered options to solve tasks, treating them as SMDPs induced by underlying MDPs and $\hat{\mathcal{O}}$. More details are given below.

In phase 1 that lasts for T_1 tasks, a variant of the E^3 algorithm is used for each task encountered, as in Brunskill & Li (2013). For each task, the algorithm keeps track of "known" states that have been visited sufficiently often, as controlled by a threshold parameter, so that their transition and rewards have been estimated accurately. The algorithm then tries to explore unknown states until all of them become known. As argued by Brunskill & Li (2013), the diameter assumption leads to the following fact:

Lemma 2 *If every task in phase 1 is run for H steps, then, with probability at least $1-\delta$, the number of visits to each state–action pair is at least $\Omega\left(\frac{H}{DSA}\ln\frac{T}{\delta}\right)$.*

At the end of phase 1, the algorithm uses the observed data to discover options. We desire to find a subset $\hat{\mathcal{O}} \subset \mathcal{O}$ that can represent ϵ-optimal policies for tasks to be solved in phase 2. Note that this is only possible if the best possible option set $\mathcal{O}^*$ is capable of representing ε-optimal policies[4] for all $\mathcal{M}$ where $\varepsilon < \epsilon$. We assume that we have access to an option-discovery algorithm that can find a set of options $\hat{\mathcal{O}}$ that achieves the following two conditions: (i) $\left|\hat{\mathcal{O}}\right| \leq \bar{O}$; and (ii) for every task M encountered in phase 1,

$$V^*_{\hat{M}}(s) - V^*_{\hat{M}'}(s) \leq (\epsilon + \varepsilon)/2 \tag{2}$$

where $\hat{M}$ is the model estimate of M (by averaging observed rewards and state transitions), and $\hat{M}'$ is the SMDP induced by $\hat{M}$ and $\hat{\mathcal{O}}$. Ties are broken arbitrarily if there are multiple option subsets that satisfy these conditions.

The core issue in the option discovery step above is to ensure the found option set $\hat{\mathcal{O}}$ will not include unnecessary options and is ε-optimal for all future tasks. Lemma 3 below quantifies the length of phase 1 so that every option in $\bar{\mathcal{O}}$ will be useful at least once in phase 1 tasks.

Lemma 3 *If we set $T_1 = p_{\min}^{-1}\ln\frac{F}{\delta}$, where $F = \mathbb{O}\left((O^*)^{\bar{O}}\right)$, then, with probability at least $1-\delta$, every non-ϵ-optimal subset of $\bar{\mathcal{O}}$ will fail to represent an ϵ-optimal policy in at least one task in phase 1.*

The next lemma considers how large H should be in order for the discovered options to be ε-optimal for all $\mathcal{M}$.

Lemma 4 *If the horizon H of tasks in phase 1 is $\Omega\left(\frac{SANDV_{\max}^2\ln(F/\delta)}{(\epsilon-\varepsilon)^2(1-\gamma)^2}\right)$, where $N = \max_{s,a} N_{s,a}$, then, with probability at least $1-\delta$, some option set $\hat{\mathcal{O}}$ will satisfy the criterion of Equation 2, and is ε-optimal.*

In phase 2, equipped with the option set $\hat{\mathcal{O}}$, every task encountered becomes an SMDP. We may use the single-task SMDP algorithms like variants of E^3 and RMAX. Invoking Theorem 1, we reach the main result of the section:

Theorem 3 *(Lifelong sample complexity with option transfer) Under the conditions in Lemmas 2–4, if the two-phase algorithm is run for T tasks, then, with probability $1-\delta$, the total sample complexity across all T tasks is*

$$\tilde{\mathbb{O}}\left(\frac{T_1 SANV_{\max}^3}{\epsilon(\epsilon-\varepsilon)^2(1-\gamma)^3} + \frac{(T-T_1)V_{\max}^3}{\epsilon^3} \times \right.$$
$$\left. \sum_{s\in\hat{\mathcal{S}},o\in\hat{\mathcal{O}}_s} \frac{N_{so}}{(1-\bar{\gamma}_s)^2}\left(\frac{1}{1-\gamma} + L + \frac{1}{\sqrt{C}}\right)\right), \tag{3}$$

where $\hat{\mathcal{S}} = \cup_{o\in\hat{\mathcal{O}}}\mathcal{I}_o$, N_{so} is the number of next states after taking option o in state s, for any $\epsilon > \varepsilon$ and $\delta \in (0,1)$.

This theorem provides a mechanism for avoiding negative transfer using options, as measured by the sample-complexity upper bound, since we can always reject the discovered option set if it increases the sample complexity bound beyond the bound with primitive actions. As sample complexity bounds are conservative, empirically the discovered options may not always outperform primitive actions. However, Section 6 provides promising evidence that the discovered options do enable improved performance.

Options will be beneficial when the optimal policies of the set of MDPS $\mathcal{M}$ share a significant amount of substructure. This does not require that the MDPs themselves be identical, merely that parts of their policies overlap.

5. Option Discovery

It is shown in previous sections that leveraging shared structure in task policies through options has the potential

[4] Again, we can always ensure this for any ε, even 0, by selecting the primitive action set.

to significantly reduce the sample complexity of later RL tasks. We now propose an option-discovery algorithm that is guided by attempting to reduce the sample complexity of future RL tasks, while preserving high-quality performance. We require that given an input set of MDPs $\mathcal{M}$, the discovered option set should be ϵ-optimal for each MDP in the set $\mathcal{M}$.[5] If the input set of MDPs is sufficiently representative, under the assumptions discussed in the previous section, the resulting set of options will also be able to be used to achieve high performance in future tasks.

There are many such option sets that ensure near-optimal rewards can be achieved in future tasks (e.g. the set of primitive state–action pairs suffices), and we wish to select one that minimizes the sample complexity of performing reinforcement learning with these options. However, this represents a computationally challenging task. If we were provided with a set of possible options to choose among, then the problem would involve selecting a subset with the minimal sample complexity that covers at least one ϵ-optimal action for each (s, m) state–MDP pair (actions that are ϵ-optimal for MDP m in state s). We have the additional condition that any terminal state of one option must also be an initiation state of another option, in order to guarantee the agent always has at least one available option to execute. Therefore, finding an optimal set of options is at least as hard as the set cover problem, whose complexity is exponential in $S\,|\mathcal{M}|$.

Instead, we propose a scalable, greedy approach to option discovery that avoids the exponential blowup required by exhaustive enumeration to ensure global optimality.[6] Though options can have more than one initiation state, it does not make a difference from a sample-complexity perspective to require them to have a single initiation state, because it is necessary to separately learn a reward and/or transition model for each initiation state of an option. Therefore, without loss of generality, we restrict our attention to discovering options with a single initiation state.

As outlined in Algorithm 1, our algorithm involves repeatedly selecting an initial state, and then trying to grow an option starting from it to cover the ϵ-optimal policies of a subset of state–MDP pairs. Expanding an option involves adding in states reachable by following the existing option's policy from its initial states, and specifying the option policy for each of these subsequent states. We consider any potential assignment of actions to these successor states, and select the first evaluated action assignment to improve an upper bound on the overall sample complexity of the full set of options, as discussed further below.

Algorithm 1 PAC-Inspired Option-Discovery

```
Input: $\mathcal{M}$, $\epsilon$
while uncovered or remaining terminal $(s, m)$ pairs do
    Initialize new option $o$ from one such state $s$
    Assign action to $\pi_o(s)$ which covers most MDPs
    while new option improves sample complexity do
        Add successors $s'$ reachable under current option
        state(s) and policy
        $l$ is an assignment of $\epsilon$-optimal actions to $s'$ set
        $FoundImprovement = 0$
        while $FoundImprovement == 0$ and haven't
        tried all $l$ do
            $sc(l) =$ sample complexity bound given $l$
            if $scl(l)$ improves over $sc$ without option then
                $FoundImprovement = 1$
                Add $s'$ as leaves of $o$
            else
                Create a new assignment $l$
            end if
        end while
    end while
    Mark $s'$ as terminal of $o$, and add $o$ to set of options
    Update lists of uncovered $(s, m)$ pairs and terminals
end while
```

If no action assignment to the added successor states yields an improvement in the sample complexity, then all successor states are set as terminal states of the option, and construction of this option completes. The option is then added to the full option set. The algorithm then removes all state–MDP pairs whose ϵ-optimal action is covered by the new option from the set of $Uncovered$ state–MDP pairs. Note that the same subset $\mathcal{M}'$ of $\mathcal{M}$ are covered by all of the states inside an option: this is required so that when an agent initiates an option in the root state, executing that option results in an ϵ-optimal policy for any states visited until the option terminates, under any MDP covered by that option.[7] The algorithm also adds: each (s_{root}, m') where m' is a MDP covered by the option to the list of initiation state–MDP pairs, $InitStateMDP$, and all (s_{term}, m') terminal-state–MDP pairs of the option to the set of $UncoveredTerminal$ pairs, except for pairs that are already initiation states of other options.

We now describe how to evaluate the change in sample complexity of adding a particular option. First, as discussed in the prior section, learning multi-step options themselves may reduce the sample complexity, since the number of mistakes they may incur is bounded by $\tilde{\mathbb{O}}\left(\frac{1}{(1-\bar{\gamma})^2}\left(\frac{1}{1-\gamma}+L+\frac{1}{\sqrt{C}}\right)\right)$, compared to $\tilde{\mathbb{O}}\left(\frac{1}{(1-\gamma)^3}\right)$

[5] We assume the algorithm is provided as input a list of each of the ϵ-optimal actions for each state s, for each MDP $m \in \mathcal{M}$.

[6] It remains an interesting open issue to find approximate algorithms with strong bounds.

[7] If only a subset of an option's states were covered for a MDP, then executing that option could yield non-ϵ-optimal performance for that MDP. So this MDP is not covered by such options.

for primitive actions, as can be seen from Equation 1. Second, creating an option that covers multiple MDPs may reduce the number of primitive actions required to cover remaining MDPs. Therefore, we compute a bound on the sample complexity due to learning the non-terminal option states both with and without an option, and evaluate if the option improves the resulting bound. See Appendix 11 for an example of this calculation.

To further reduce computation, we bound the number of considered successors s' at each stage of option construction. Our experiments in Section 6 used 4 as the threshold.

6. Experiments

We consider a non-episodic variant of the four-room maze of Sutton et al. (1999). Actions are to go in the 8 directions and succeed with probability 0.8; otherwise the agent goes in one of the other orthogonal directions with equal probability. The transition models in all MDPs are the same. Potential high reward states lie on the outer edges of the upper left and lower right rooms. These states also result in an immediate transition back to the upper right corner.

In phase 1, the agent acts in a series of 40 MDPs with this underlying structure. Each MDP in phase 1 is generated by randomly sampling one of the 17 potential high reward states and assigning it a mean reward of 1.0. The other potential high reward states are allocated a mean reward of 0.1. All other states have a mean reward of 0. Rewards are sampled from the mean reward of the state plus Gaussian noise with a standard deviation of 0.1. The discount factor γ was set to 0.9 and in each task the agent acts for 200,000 steps. During phase 1 the agent executes the E^3 PAC-MDP algorithm in each task, and constructs an estimate of its optimal state–acton values. After phase 1, the agent uses the resulting policies to perform option discovery.

In phase 2, the agent uses the discovered options to do RL in new tasks. One of the benefits of options should be to generalize to new MDPs with different models but similar near-optimal policies that can be composed of parts of near-optimal policies of prior tasks. To explore this, in phase 2 each task involved two high reward states, one in the upper left room, and the other randomly sampled from the possible high reward states. In phase 2 the agent ran SMDP-RMAX with the discovered options, which generally has better performance than the SMDP version of E^3, although it can be replaced by any PAC-SMDP RL algorithm.

For phase 2, our algorithm was compared to three baselines. In the PrimitiveOnly baseline, we compare to an agent using only the primitive actions. In HandCoded, we define a set of options that we expect to be the minimum number needed to obtain good performance: for the upper left and lower right rooms each state has a single option that takes it to the upper or right hallway respectively, the other two rooms have two options per state to take them to the outgoing hallway, and the upper and right hallways have options to go to each of the possible goal states. In our third baseline, we construct a set of options using the PolicyBlocks algorithm (Pickett & Barto, 2002). PolicyBlocks constructs options by forming subpolicies that cover subsets of the optimal policies of an input set of MDP policies, and greedily selecting subpolicies with the largest coverage, and is similar in performance and spirit to the SKILLS algorithm (Thrun & Schwartz, 1995). We selected PolicyBlocks since it has previously demonstrated good performance, requires no outside expert knowledge, and like our algorithm extracts options given an input set of policies.

As there are 8 actions and 104 states, baseline PrimitiveOnly has 832 state-option pairs. In contrast, our algorithm discovered 550 pairs (144 multi-step, the rest primitive actions) to cover the 40 MDPs, suggesting a significant improvement is possible. Constructing options by hand yielded 189 options: note that this is the minimal set of options we expect to be needed to achieve an optimal policy for all possible MDPs, and provides an upper bound on the potential performance. Interestingly, PolicyBlocks resulted in 985 options, more than the set of primitive actions. This is because PolicyBlocks extracts subpolicies, subsets of states with matching policies across subsets of the input tasks. These subpolicies have no defined initiation set, so we allowed each state member within a subpolicy to be an initiation state. Though this yields a large number of possible options, since a 6-state subpolicy adds an option to each of these states, it is unclear how else to ensure the original task policies could be covered given the subpolicies.

We now examined the sample-complexity bound for the different discovered option sets. For our approach, this yielded a bound of 511,605 for the discovered set compared to 832,000 if primitive state–action pairs were used, suggesting a potential advantage if the options were useful in the task in phase 2. The bound for hand-defined options was 85,765 and for the PolicyBlocks was 942,450.

To evaluate the discovered options, we first examined if the discovered options were capable of representing the optimal value function of MDPs in phase 2, if the MDP models were known. To do so, we computed the optimal value of MDP models with the just-described structure. We then computed the discovered option models given the true transition and reward probabilities, and then computed the best value function using the discovered options. The loss from using options was only approximately 2-3%. Therefore the discovered options are capable of representing close-to-optimal policies for the tasks in phase 2.

We then evaluated whether the discovered options could be used with SMDP-RMAX to improve performance in phase

Table 1. Average Reward in Phase 2 with Different Option Sets

PAC-inspired	PrimitiveOnly	HandCoded	PolicyBlocks
13145	10470	14178	11229

2. We sampled 100 tasks with the structure just described, and report in Table 1 the average of the total rewards in these tasks for each of the algorithms. As expected, the hand-coded options did the best. But our approach was the best that requires no human input, and obtained 93% of the reward of the hand-coded options designed with substantial domain knowledge. Our approach achieved over 25% more reward than using primitive actions alone, and this difference was statistically significant (t-test, $p < 10^{-18}$). Our discovered options also did about 16% better than those found by PolicyBlocks (t-test, $p < 10^{-16}$). PolicyBlocks did slightly better than PrimitiveOnly, likely because, due to the starting state and our assumed deterministic completion of options (they only finish when a terminal state is reached), many states are never reached when options from earlier states are chosen, and therefore do not need to be explored.

The reader may be curious about the relatively poor performance of RMAX with primitive actions, since the algorithm is able to learn the optimal policies if we set c, the threshold for making a state–action known, sufficiently large. However, a too large c value results in suboptimal performance for finite-horizon problems; for instance, the average reward with $c = 75$ is much worse than that in Table 1, although optimal policies are always identified. We thus manually searched, for each baseline algorithm, over c values, and report returns for the best possible settings.

These encouraging results show our option discovery algorithm is capable of discovering options that can allow good performance in new related tasks, almost as high as if given expert-tuned options for these tasks, and outperform primitive actions and another option-discovery baseline.

7. Related Work

There has been limited work on theoretical justifications for lifelong RL, and most has considered transfer between a source and target task (e.g. Sorg & Singh (2009); Lazaric & Restelli (2012); Mann & Choe (2012)). In lifelong RL, every current task is a target task, and will be one of the source tasks in the future. Perhaps the closest work is the sample-complexity analysis of sequential multi-task RL by Brunskill & Li (2013). However, they assume the agent repeatedly acts in a finite set of MDPs, whereas here the set of possible MDPs may be infinite, yet option discovery can still help if the MDP policies share substructure.

Options have been studied by many authors as a form of knowledge transfer between tasks. The SKILLS algorithm (Thrun & Schwartz, 1995) uses a description length device to discover common pieces of policies in multiple related tasks. In our experiments, we compared to the PolicyBlocks algorithm of Pickett & Barto (2002) which is similar to SKILLS. Stolle & Precup (2002) and Mannor et al. (2004) discover sub-goals, and create options towards those sub-goals. Homomorphisms (Soni & Singh, 2006) and shared features (Konidaris & Barto, 2007) can be used to discover useful options across multiple tasks with different state/action spaces. In contrast to all previous work we are aware of, this paper proposes the first discovery algorithm that is theoretically sound, justifying how and when option transfer improves sample complexity in lifelong RL, compared to single-task RL.

Finally, our results were developed with typical PAC analysis in the literature (*e.g.*, Strehl et al. (2009)), which can be improved with a tighter analysis. For instance, an interesting observation in Lattimore & Hutter (2012) is that one can use variance information in empirical Bernstein inequalities to get a sharper bound on the speed of model parameter concentration. Their bound has a better dependency on $1/\epsilon$ and $1/(1-\gamma)$, attained by a more complicated algorithm UCRLγ. Applying the same idea to SMDPs and lifelong RL appears to be highly nontrivial, and is an interesting and worthwhile avenue for future work.

8. Future Work and Conclusions

In addition to the direction just described for a tighter analysis, there remain a few interesting areas for future work. It could be helpful to employ intra-option learning, and move to incremental discovery instead of a two-phase approach. Also, algorithms like RMAX tend to explore all states, but in some cases prior experience, encoded by options, may suggest it is suboptimal to visit certain states (driving on the highway at rush hours is bad on almost all weekdays), and we would like to create RL algorithms capable of better leveraging such knowledge. We could also use richer types of temporally extended actions, like hierarchical or nested policies, which have proven beneficial in planning.

To conclude, we presented the first, to our knowledge, formal analysis of how temporally extended actions can affect and benefit the sample complexity of RL, providing theoretical support for some of the surprising results in the literature. We showed how an agent performing lifelong learning across a series of tasks can learn a set of options that is ε-optimal for all tasks the agent may encounter, and analyzed the sample complexity of lifelong learning with the discovered option. We also presented a PAC-inspired option discovery heuristic algorithm, and showed that it enables us to achieve significantly better performance in a benchmark domain. Our results support the substantial benefit that temporally extended actions can have in RL.

References

Barto, Andrew and Mahadevan, Sridhar. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, 13(1–2): 41–77, 2003.

Brafman, Ronen I. and Tennenholtz, Moshe. R-max—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, October 2002.

Brunskill, Emma and Li, Lihong. Sample complexity of multi-task reinforcement learning. In *Proc. of the Twenty-Ninth Conf. on Uncertainty in Artificial Intelligence*, pp. 122–131, 2013.

Jaksch, Thomas, Ortner, Ronald, and Auer, Peter. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

Jong, Nicholas K., Hester, Todd, and Stone, Peter. The utility of temporal abstraction in reinforcement learning. In *Proc. of the Seventh Int'l Conf. on Autonomous Agents and Multiagent Systems (AAMAS-08)*, pp. 299–306, 2008.

Kakade, Sham. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, UK, 2003.

Kearns, Michael J. and Singh, Satinder P. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2–3):209–232, 2002.

Konidaris, George and Barto, Andrew G. Building portable options: Skill transfer in reinforcement learning. In *Proc. of the Twentieth Int'l Joint Conf. on Artificial Intelligence*, pp. 895–900, 2007.

Lattimore, Tor and Hutter, Marcus. PAC bounds for discounted MDPs. In *Proc. of the Twenty-Third Int'l Conf. on Algorithmic Learning Theory (ALT-12)*, volume 7568 of *Lecture Notes in Computer Science*, pp. 320–334, 2012.

Lazaric, Alessandro and Restelli, Marcello. Transfer from multiple MDPs. In *Advances in Neural Information Processing Systems*, pp. 1746–1754, 2012.

Mann, Timothy A. and Choe, Yoonsuck. Directed exploration in reinforcement learning with transferred knowledge. In *Proc. of the Tenth European Workshop on Reinforcement Learning*, 2012.

Mann, Timothy A. and Mannor, Shie. Scaling up approximate value iteration with options: Better policies with fewer iterations. In *Proc. of the Int'l Conf. on Machine Learning*, 2014.

Mannor, Shie, Menache, Ishai, Hoze, Amit, and Klein, Uri. Dynamic abstraction in reinforcement learning via clustering. In *Proc. of the Twenty-First Int'l Conf. on Machine Learning*, pp. 560–567, 2004.

Pickett, Marc and Barto, Andrew G. PolicyBlocks: An algorithm for creating useful macro-actions in reinforcement learning. In *Proc. of the Nineteenth Int'l Conf. on Machine Learning*, pp. 506–513, 2002.

Puterman, Martin L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, New York, 1994. ISBN 0-471-61977-9.

Soni, Vishal and Singh, Satinder P. Using homomorphisms to transfer options across continuous reinforcement learning domains. In *Proc. of the Twenty-First National Conf. on Artificial Intelligence*, pp. 494–499, 2006.

Sorg, Jonathan and Singh, Satinder P. Transfer via soft homomorphisms. In *Proc. of the Eighth Int'l Conf. on Autonomous Agents and Multiagent Systems*, pp. 741–748, 2009.

Stolle, Martin and Precup, Doina. Learning options in reinforcement learning. In *Proc. of the Fifth Int'l Symp. on Abstraction, Reformulation and Approximation*, volume 2371 of *Lecture Notes in Computer Science*, pp. 212–223, 2002.

Strehl, Alexander L., Li, Lihong, and Littman, Michael L. Incremental model-based learners with formal learning-time guarantees. In *Proc. of the Twenty-Second Conf. on Uncertainty in Artificial Intelligence*, pp. 485–493, 2006.

Strehl, Alexander L., Li, Lihong, and Littman, Michael L. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, 10:2413–2444, 2009.

Sutton, Richard S., Precup, Doina, and Singh, Satinder P. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999.

Thrun, Sebastian and Mitchell, Tom M. Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1–2): 25–46, 1995.

Thrun, Sebastian and Schwartz, Anton. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7*, pp. 385–392, 1995.

Wilson, Aaron, Fern, Alan, Ray, Soumya, and Tadepalli, Prasad. Multi-task reinforcement learning: A hierarchical Bayesian approach. In *Proc. of the Twenty-Fourth Int'l Conf. on Machine Learning*, pp. 1015–1022, 2007.

9. Further Details for Section 3

9.1. Proof Sketch of Theorem 1

The proof parallels the proof of Proposition 1 by Strehl et al. (2006) for MDPs, except the horizon (denoted by T in their paper) needs to be redefined:

$$H = \frac{1}{1-\gamma} \ln \frac{4}{\epsilon(1-\gamma)} + L + \frac{1}{\sqrt{C}} \ln \frac{2}{\delta'}.$$

This choice of H ensures that, for any epoch t, the non-stationary policy $\mathbf{A}_t$ in state s_t is either $\Theta(\epsilon)$-optimal, or will reach an unknown state in H steps with probability at least $\epsilon(1-\gamma)$. In either case, the algorithm will reach a next state between step $\frac{1}{1-\gamma} \ln \frac{4}{\epsilon(1-\gamma)}$ and H, since with probability at least $1-\delta'$, the waiting time of taking action a_t in state s_t is $L + \frac{1}{\sqrt{C}} \ln \frac{1}{\delta'}$ (Lemma 1). Taking a union bound over all possible non-ϵ-optimal steps (which is polynomial in ζ, $1/\epsilon$, $1/\delta$, and $1/(1-\gamma)$), that is, setting δ' to $\delta/\mathrm{poly}(\zeta, 1/\epsilon, 1/\delta, 1/(1-\gamma))$, we can prove the theorem as done in Strehl et al. (2006). Note that we need not take a union over all epochs, but only those where the decision is potentially non-ϵ-optimal; if $\mathbf{A}_t$ is ϵ-optimal in epoch t, it does not count towards the sample complexity anyway.

9.2. Definition of Known-state SMDP

Definition 3 *Let $M = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$ be an SMDP, Q is a state–action value function, and $\mathcal{K} \subseteq \mathcal{S} \times \mathcal{A}$ a set of "known" state–actions. Define the known state–action SMDP (with respect to $\mathcal{K}$) as $M_{\mathcal{K}} = \langle \mathcal{S}, \mathcal{A}, P_{\mathcal{K}}, R_{\mathcal{K}}, \gamma \rangle$, where*

$$\begin{aligned} P_{\mathcal{K}}(s', \tau | s, a) &= \begin{cases} P(s', \tau | s, a), & \textit{if } (s,a) \in \mathcal{K} \\ \mathbb{I}\left(s = s', \tau = 1\right), & \textit{otherwise} \end{cases} \\ R_{\mathcal{K}}(s,a) &= \begin{cases} R(s,a), & \textit{if } (s,a) \in \mathcal{K} \\ (1-\gamma) Q(s,a), & \textit{otherwise.} \end{cases} \end{aligned}$$

In other words, the known state–action SMDP $M_{\mathcal{K}}$ has identical dynamics to M except in unknown state–actions where (i) the transitions are all self-loops with unit waiting time, and (ii) the Q-values are exact.

9.3. Proof of Theorem 2

Clearly, the construction leads to optimistic value functions, so the first condition of Theorem 1 holds.

We now consider when a state–action pair (s,a) becomes known. Define the *effective* transition probabilities by

$$P^S(s'|s,a) = \sum_\tau P(s', \tau | s, a) \gamma^\tau,$$

and the marginal distribution of waiting time by

$$P^T(\tau | s, a) = \sum_{s'} P(s', \tau | s, a).$$

We first generalize the simulation lemma (see, *e.g.*, Kearns & Singh (2002); Strehl et al. (2009)) for MDPs to SMDPs, giving a bound on the value function differences in terms of model estimation errors:

Lemma 5 *Let $M_i = \langle \mathcal{S}, \mathcal{A}, P_i, R_i, \gamma \rangle$ $(i = 1, 2)$ be two SMDPs that differ only in reward and transition functions, and V_i^* and Q_i^* the respective optimal value functions. Let $\bar{\gamma}_{s,a}$ be the effective discount factor for (s,a) under M_2:*

$$\bar{\gamma}_{s,a} = \sum_\tau \gamma^\tau P_2^T(\tau | s, a).$$

and define the discount-adjusted model estimation error by

$$\begin{aligned} \varepsilon_{s,a} \quad = \quad & \frac{1}{1-\bar{\gamma}_{s,a}} \big(|R_1(s,a) - R_2(s,a)| \\ & + V_{\max} \left\| P_1^S(\cdot|s,a) - P_2^S(\cdot|s,a) \right\|_1 \big). \end{aligned}$$

Then, for any s and a,

$$\begin{aligned} |Q_1^*(s,a) - Q_2^*(s,a)| &\leq \max_{s,a} \varepsilon_{s,a} \\ |V_1^*(s) - V_2^*(s)| &\leq \max_{s,a} \varepsilon_{s,a} \end{aligned}$$

Proof Let (s,a) be the state–action pair that achieves maximum difference of $|Q_1^*(\cdot,\cdot) - Q_2^*(\cdot,\cdot)|$. To simplify notation, define

$$\begin{aligned} \varepsilon_R &= |R_1(s,a) - R_2(s,a)| \\ \varepsilon_P &= \left\| P_1^S(\cdot|s,a) - P_2^S(\cdot|s,a) \right\|_1 \\ \Delta &= |Q_1^*(s,a) - Q_2^*(s,a)| \end{aligned}$$

Then,

$$\begin{aligned}
\Delta &= |Q_1^*(s,a) - Q_2^*(s,a)| \\
&= \left| \left(R_1(s,a) + \sum_{s',\tau} \gamma^\tau P_1(s',\tau|s,a) V_1^*(s') \right) \right. \\
&\qquad \left. - \left(R_2(s,a) + \sum_{s',\tau} \gamma^\tau P_2(s',\tau|s,a) V_2^*(s') \right) \right| \\
&\leq |R_1(s,a) - R_2(s,a)| \\
&\qquad + \left| \sum_{s',\tau} \gamma^\tau \left(P_1(s',\tau|s,a) - P_2(s',\tau|s,a) \right) V_1^*(s') \right| \\
&\qquad + \left| \sum_{s',\tau} \gamma^\tau P_2(s',\tau|s,a) (V_1^*(s') - V_2^*(s') \right| \\
&\leq \varepsilon_R + V_{\max}\varepsilon_P + \Delta \left| \sum_{s',\tau} \gamma^\tau P_2(s',\tau|s,a) \right| \\
&= (\varepsilon_R + V_{\max}\varepsilon_P) + \bar{\gamma}_{s,a}\Delta \\
&= (1 - \bar{\gamma}_{s,a})\varepsilon_{s,a} + \bar{\gamma}_{s,a}\Delta.
\end{aligned}$$

Rearranging terms, we have

$$\Delta \leq \varepsilon_{s,a} \leq \max_{s',a'} \varepsilon_{s',a'}.$$

The case for V^* follows immediately from the following observation: for any state s,

$$\begin{aligned}
|V_1^*(s) - V_2^*(s)| &= \left| \max_a Q_1^*(s,a) - \max_a Q_2^*(s,a) \right| \\
&\leq \max_a |Q_1(s,a) - Q_2(s,a)| \;\leq\; \Delta.
\end{aligned}$$

□

Clearly, $R(s,a) \in [0, \frac{1}{1-\gamma}]$. Using a concentration argument based on Hoeffding's inequality, one can establish that $\mathbb{O}\left(1/(\varepsilon^2(1-\gamma)^2)\right)$ samples suffice to ensure ε accuracy in the reward estimate. Similarly, the effective transition probabilities $P(s'|s,a)$ can also be estimated within ε total variation with $\mathbb{O}\left(N_{sa}/\varepsilon^2\right)$ samples. Therefore, by setting ε appropriately, the accuracy condition in Theorem 1 can be satisfied.

Finally, there are at most SA many state–actions, each becoming known when it is visited sufficiently often. The bounded-surprises condition in Theorem 1 thus holds.

Therefore all three conditions of Theorem 1 hold, and the result follows.

10. Further Details for Section 4

10.1. Proof Sketch of Lemma 3

Fix a non-ϵ-optimal option set $\mathcal{O}' \subset \mathcal{O}^*$ with $|\mathcal{O}'| \leq \bar{O}$. By assumption, it fails to represent a near-optimal policy for MDPs drawn i.i.d. from ν over $\mathcal{M}$ with probability at least p_{min}. Following the same argument for Lemma 1 of Brunskill & Li (2013), $p_{\min}^{-1} \ln \frac{1}{\delta'}$ many tasks suffices to reveal the non-ϵ-optimality of $\mathcal{O}'$, with probability at least $1 - \delta'$. Taking a union bound over all subsets of $\mathcal{O}^*$ up to size $\bar{O}$, which number $F = \mathbb{O}\left((O^*)^{\bar{O}}\right)$, one finishes the proof of the lemma.

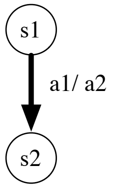
10.2. Proof Sketch of Lemma 4

For convenience, define $\epsilon_1 = (\epsilon - \varepsilon)/4$. The proof relies on three major steps, each holding with probability at least $1 - \delta$.

- *The MDP models are all estimated to sufficient accuracy:* The condition together with Lemma 2 implies every state–action will be visited at least $\Omega(NV_{\max}^2 \epsilon_1^{-2}(1-\gamma)^{-2} \ln 1/\delta)$ times. Applying Hoeffding's inequality together with Lemma 8.5.5 of Kakade (2003), the reward and transition probabilities of every state–action pair are estimated with $\epsilon_1(1-\gamma)/V_{\max}$ accuracy. By the simulation lemma (*c.f.*, Kearns & Singh (2002); Strehl et al. (2009)), $\left|V_M^*(s) - V_{\hat{M}}^*(s)\right| < \epsilon_1$, and similarly, by Lemma 5, $\left|V_{M'}^*(s) - V_{\hat{M}'}^*(s)\right| < \epsilon_1$, where M and $\hat{M}$ are the underlying/estimated MDPs, and M' and $\hat{M}'$ the corresponding SMDPs induced by the discovered option set $\hat{\mathcal{O}}$.
- *The discovered option set $\hat{\mathcal{O}}$ is ϵ-optimal for all MDPs in $\mathcal{M}$:* Using the triangle inequality together with the two inequalities established in the previous step, we have

  $$\begin{aligned}
  &V_M^*(s) - V_{M'}^*(s) \\
  &\quad\leq \left|V_M^*(s) - V_{\hat{M}}^*(s)\right| + \left|V_{M'}^*(s) - V_{\hat{M}'}^*(s)\right| \\
  &\qquad + \left|V_{\hat{M}}^*(s) - V_{\hat{M}'}^*(s)\right| \\
  &\quad\leq 2\epsilon_1 + \left|V_{\hat{M}}^*(s) - V_{\hat{M}'}^*(s)\right|.
  \end{aligned}$$

  In the option-discovery step, $\hat{\mathcal{O}}$ must satisfy $V_{\hat{M}}^*(s) - V_{\hat{M}'}^*(s) \leq (\epsilon + \varepsilon)/2$. Therefore, $V_M^*(s) - V_{M'}^*(s) \leq 2\epsilon_1 + (\epsilon + \varepsilon)/2 = \epsilon$; that is, the option set $\hat{\mathcal{O}}$ is ϵ-optimal for all MDPs *encountered in phase 1*. According to Lemma 3, $\hat{\mathcal{O}}$ must also be ϵ-optimal for all MDPs in $\mathcal{M}$; otherwise, it will fail to represent ϵ-optimal policies in at least one MDP in phase 1.
- *There exists at least one option set that satisfies the criterion of Equation 2:* According to the assumption,



	m1	m2	m3	m4
s1	a1	a1	a2	a2
s2	a4	a4	a5	a5

Figure 1. Example for sample complexity calculation illustration. The table shows the ϵ-optimal actions for each MDP. There are 5 actions but a3 is never optimal for any MDP.

there exists some option set $\bar{\mathcal{O}}$ that is ε-optimal for $\mathcal{M}$: for any M and any s, $V^*_M(s) - V^*_{M'}(s) < \varepsilon$, where M' is the SMDP induced by M and $\bar{\mathcal{O}}$. Using the triangle inequality as well as the accuracy guarantee established in step 1, one gets

$$\begin{array}{rcl} V^*_{\hat{M}}(s) - V^*_{\hat{M}'}(s) & < & V^*_M(s) + \epsilon_1 - V^*_{M'}(s) + \epsilon_1 \\ & < & \varepsilon + 2\epsilon_1 \\ & = & (\epsilon + \varepsilon)/2. \end{array}$$

In other words, $\bar{\mathcal{O}}$ will satisfy Equation 2.

The overall failure probability is at most δ: All three steps above hold with high probability. The first two steps require a union bound over all possible subsets of $\mathcal{O}^*$ with size up to $\bar{O}$. There are $F = \mathbb{O}\left((O^*)^{\bar{O}}\right)$ many such subsets. It suffices to set $\delta \leftarrow \delta/C$ for the union bound to complete the whole proof.

10.3. Proof of Theorem 3

The sample complexity can be divided into two terms: the tasks in phase 1 and the tasks in phase 2. The sample complexity of the MDP tasks in phase 1 is simply the number of tasks in phase 1, T_1, multiplied by the sample complexity of the E^3 algorithm.

11. Further Details for Section 5

We now illustrate the process of evaluting the bound on the sample complexity benefit with the small example shown in Figure 1. In this example there are 2 states and 4 MDPs, and each MDP has a single ϵ-optimal action in each state, shown in the Figure's table. Assume that state s_1 deterministically transitions to s_2. Before introducing an option, there were 4 state–action combinations $(s_1, a_1), (s_1, a_2), (s_2, a_4), (s_2, a_5)$ needed to cover the ϵ-optimal policies of each MDP, resulting in a sample complexity bound of $\mathbb{O}\left(\frac{4}{(1-\gamma)^6}\right)$. Now consider adding the option whose initiation state is s_1 and that takes action a_2 in state s_1 and action a_5 in state s_2. The length of this option is always 2, so from the prior section the option's contribution to the sample complexity is $\mathbb{O}\left(\frac{1}{(1-\gamma^2)^2(1-\gamma)^3}\left(2 + \frac{1}{1-\gamma}\right)\right)$. This option covers MDPs m_3 and m_4. To cover s_1 and s_2 for the remaining uncovered MDPs requires just 2 primitive state–action pairs, with a resulting $\mathbb{O}\left(\frac{2}{(1-\gamma)^6}\right)$ contribution to the sample complexity bound. Therefore, introducing the option will reduce this upper bound on the sample complexity if

$$\begin{array}{rcl} \frac{1}{(1-\gamma^2)^2(1-\gamma)^3}(2 + \frac{1}{1-\gamma}) + \frac{2}{(1-\gamma)^6} & < & \frac{4}{(1-\gamma)^6} \\ \Leftrightarrow \qquad 5 & < & 6\gamma + \gamma^2 \end{array}$$

which holds for large γ, such as $\gamma = 0.9$. The algorithm evaluates this expression for the input γ, and keeps the option if the expression holds.