Toward Good Abstractions for Lifelong Learning

David Abel, Dilip Arumugam, Lucas Lehnert, Michael L. Littman
Department of Computer Science
Brown University
Providence, RI 02906
{david_abel,dilip_arumugam,lucas_lehnert,michael_littman}@brown.edu

Abstract

Lifelong Reinforcement Learning presents a diversity of challenges. Agents must effectively transfer knowledge across tasks while simultaneously addressing exploration, credit assignment, and generalization. Abstraction can help overcome these hurdles by compressing the state space or empowering the action space of a learning agent, thereby reducing the computational and statistical burdens of learning. In this work, we offer several new results on the effect of lifelong learning with abstractions. First, we introduce a new class of value-preserving state abstractions whose optimal form can be computed efficiently, improving over existing NP-Hardness results. Second, we provide a generic sample bound for computing high confidence state abstractions of these kinds in the lifelong setting. Third, we show experimentally that only in the simplest of settings do state abstractions offer improvements to lifelong learning. In light of these limitations, we introduce a complementary action-abstraction algorithm that, when paired with a state abstraction, enables efficient learning across tasks. Further, the joint state-action abstraction defines a closed operator on MDPs, thereby yielding a simple recipe for constructing and analyzing hierarchies for reinforcement learning.

1 Introduction

Abstraction is a central representational operation for Reinforcement Learning (RL), enabling fast planning, deep exploration, and effective generalization. Previous work focuses on two forms of abstraction: (1) State abstraction, which groups together similar world-states to form compressed descriptions of the environment, and (2) Action abstraction, which yields compact models of temporally extended behavior by correlating sequences of actions. These two methods of abstraction provide powerful tools for simplifying complex problems, promising a principled method for scaling RL. Lifelong RL proves an especially challenging learning setting, as learners also have to form models that generalizes across tasks. The tools of abstraction are especially well suited to assist in lifelong RL, as abstractions capture relevant task structure to aid in information transfer.

State abstraction's core operation is *aggregation*, encompassing functions that group environmental states together based on some measure of state similarity. Prior work introduces state-abstraction types that are guaranteed to preserve properties about states they group together [8, 20, 11, 12, 10, 13, 1]. It is unknown how they generalize beyond a single task or, more generally, effect learning difficulty.

Action abstraction describes tools for defining *skills*, which denote long horizon sequences of action executions. The most canonical of these frameworks is that of Options [23]. The core principle is to augment the agent's model of the world with a collection of high level actions that probe more deeply into the search space, resulting in targeted exploration, deep planning, or transfer [18, 6, 5, 4].

Hierarchical Reinforcement Learning Workshop at the 31st Conference on Neural Information Processing Systems (HRL@NIPS 2017), Long Beach, CA, USA.

In this paper, we offer several new results on abstractions aimed at lifelong RL. First, we introduce a broad class of state-abstraction types whose optimal form can be computed efficiently, improving over existing NP-Hardness results. We provide a new state-abstraction type belonging to this class and prove that it preserves reasonable value loss bounds. Second, we provide a generic sample bound for computing high confidence state abstractions of these kinds in the lifelong setting. We then present a negative empirical result: we find empirically that learning is *not* accelerated by approximate state abstractions, even optimal ones. We provide a diagnosis for this failure, suggesting that only the combination of state *and* action abstraction can yield the expected reduction in learning difficulty. We then introduce a complementary algorithm for computing action abstractions, which, empirically, offers the most improvement in learning speed when combined with state abstractions. We close by showcasing a simple recipe for defining hierarchical abstractions from repeated recursion on applications of the introduced joint state–action abstraction. We suggest this paradigm offers routes to analyze hierarchical RL more generally.

1.1 Background

First, we clarify our learning setting and other relevant definitions. We assume the traditional RL formulation, wherein an agent interacts with a Markov Decision Process (MDP) to maximize long term expected reward. For background on RL, see Kaelbling et al. [15], and for background on MDPs, see Puterman [21]. Of growing interest in the RL literature is the *lifelong* or *multitask* setting, in which an agent must interact with and solve many related tasks over the course of a lifetime as in Brunskill and Li [5]. We offer the following instantiation of this setting:

Definition 1 (Lifelong RL): *In Lifelong RL, the agent receives $\mathcal{S}, \mathcal{A}, s_0 \in \mathcal{S}$, horizon H, a fixed but unknown transition model $\mathcal{T}$, and a fixed but unknown distribution over reward functions, D. The agent samples $\mathcal{R}_i \sim D$, and interacts with the MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}_i, \mathcal{T} \rangle$ for H timesteps, starting in state s_0. After H timesteps, the agent resamples from D and repeats.*

The goal of state abstraction is to reduce the size of the environmental state space by grouping together similar states in a way that doesn't change the underlying problem being solved. A state abstraction *type* is defined with respect to a two-argument predicate on state pairs:

Definition 2 (State Abstraction Type): *A state abstraction type is a collection of functions $\phi : \mathcal{S} \mapsto \mathcal{S}_\varphi$ associated with a fixed predicate on state pairs:*

$$p : \mathcal{S} \times \mathcal{S} \mapsto \{0, 1\}, \tag{1}$$

such that when ϕ clusters state pairs, the predicate must be true for that state pair:

$$\phi(s_1) = \phi(s_2) \to p(s_1, s_2). \tag{2}$$

Action abstraction has appeared under a diversity of names, including *skills* [17], *macro-operators* [19, 22], and *temporal abstraction*. Most recently, the literature has focused on Options [23]:

Definition 3 (Option): *An option is a triple $\langle \mathcal{I}, \beta, \pi \rangle$, where $\mathcal{I}$ is a predicate on states denoting when the option is available, β assigns a Bernoulli random variable to each state denoting the probability that the option terminates during execution in that state, and π is a policy.*

In the light of the generality of options, we consolidate action abstractions accordingly:

Definition 4 (Action Abstraction): *An action abstraction is a function $\omega : \mathcal{O} \times \mathcal{S} \longmapsto \mathcal{A}$ that projects an option to ground actions given a ground state, with $\mathcal{O}$, a non-empty set of options.*

An action abstraction *replaces* the primitive action space. Reasoning with options, agents only make a decision in states where no option is currently executing.

2 Results

We first present a general sample bound for computing state abstractions of arbitrary types in lifelong RL. All proofs are omitted due to space constraints, but we make them available in the appendix.

Theorem 2.1. *Let $\mathbb{P}$ denote the space of all predicates on state pairs. For a given $\delta \in (0, 1]$, any state abstraction of a type defined by a predicate in $\mathbb{P}$ can be computed such that two states are incorrectly aggregated (that is, grouped, but shouldn't be) only with probability $(1 - \delta)^2$ after $m \geq \frac{2\ln\left(\frac{2}{\delta}\right)}{\delta^2}$ samples $\mathcal{R} \sim D$ from the distribution.*

Thus, after m sampled and (exactly) solved MDPs, we can compute a high confidence state abstraction for the distribution D. In general, computing the optimal approximate state abstraction is NP-Hard [10]. However, we introduce a restricted class of approximate state abstractions that can be computed efficiently:

Theorem 2.2. *Consider any transitive predicate on state pairs, p that takes computational complexity c_p to evaluate for a given state pair. The state abstraction type ϕ_p that induces the smallest abstract state space can be computed*[1] *in $\mathcal{O}(|\mathcal{S}|^2 \cdot c_p)$.*

This is a dramatic reduction in computational complexity relative to the previous NP-Hardness result. The speed up comes from being able to determine a state's abstract state simply by evaluating each of its (at most $|\mathcal{S}|$) edges. To leverage this observation, we introduce a new class of approximate state abstractions that satisfy transitivity. The idea behind the class is to discretize the interval $[0, \text{VMAX}]$, where $\text{VMAX} = \frac{\text{RMAX}}{1-\gamma}$, and aggregate states if the Q values for each action fall into the same discrete bucket. Here, RMAX denotes the max possible reward received at a single time step, and VMAX an upper bound on the value function.

Definition 5 ($\phi_{Q^*,d}$): *The $\phi_{Q^*,d}$ denotes a state-abstraction type with predicate:*

$$p_d(s_1, s_2) = \begin{cases} 1 & \forall_{a \in \mathcal{A}} : \lceil \frac{d \cdot Q^*(s_1,a)}{\text{VMAX}} \rceil = \lceil \frac{d \cdot Q^*(s_2,a)}{\text{VMAX}} \rceil \\ 0 & \textit{otherwise.} \end{cases} \tag{3}$$

Further, any function $\phi_{Q^*,d}$ induces an abstract model whose policy is near optimal in the true MDP:

Theorem 2.3. *The $\phi_{Q^*,d}$ abstraction type is a subclass of $\phi_{Q^*_\varepsilon}$, studied by Abel et al. [1] and Hutter [13], with $d = \varepsilon$, and therefore, for a single MDP:*

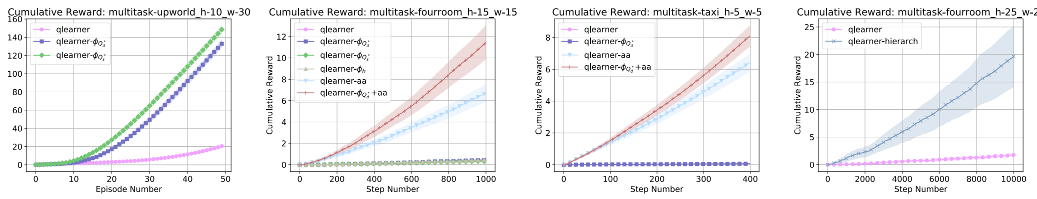
$$V^*(s_0) - V^{\pi_{\phi_{Q^*,d}}}(s_0) \leq \frac{2d\text{RMAX}}{(1-\gamma)^2}. \tag{4}$$

Thus, this class represents a reasonable candidate for state abstractions as it can be computed efficiently and has a value loss that scales according to a free parameter, d. We note that predicates defining other existing abstraction types such as ϕ_a^* [20] have natural translations to transitive predicates, too.

2.1 Experiments

We conduct three simple experiments to showcase the effects of learning with various abstraction combinations. In each experiment, we compute the state abstraction that induces the smallest abstract space of the $\phi_{Q^*,d}$ type with respect to the distribution, where we set $d = 0.05$. For comparison, we also compute a candidate $\phi_{Q^*,\varepsilon}$ state abstraction with $\varepsilon = 0.05$. We experiment with Q-Learning [24] with and without the new state-abstraction type, where Q-Learning uses an ϵ-greedy policy with exploration parameter $\epsilon = 0.1$ and learning rate $\alpha = 0.05$. When the agent reaches a goal, it receives $+1$ reward and is reset to the start position. We use a 10×30 lifelong, episodic variant of the Upworld domain from Abel et al. [1] where the goal can appear in any cell in the top row, and 15×15 lifelong variant of the Four Rooms domain from Sutton et al. [23], where the goal can appear in the corners of each of the three non-starting rooms, and the classical Taxi problem from Dietterich [9]. In Four Rooms, we also test with ϕ_h, a hand coded state abstraction that assigns an abstract state to each room.

[1]Notably, the complexity of c_p dictates the overall complexity of computing ϕ_p.



(a) Upworld (b) Four Rooms (c) Taxi (d) Big Four Rooms

Figure 1: Average cumulative reward over 500 (a), 1,000 (b), 200 (c), and 10,000 (d) task samples from each distribution comparing: state abstractions in (a) and (b), state-action abstraction combinations in (b) and (c), and a simple hierarchical abstraction in (d).

Results are presented in Figure 1. Across all of our experiments (even those not reported in this version), the Upworld task distribution is the *only* domain that consistently yields improvements for the state-abstraction learner relative to baselines. Results for the Four Rooms experiment are presented in Figure 1b. In Four Rooms, we find that learners with approximate state abstractions do not improve over Q-Learning (at all) for the given sample budget. We suggest that the domain is indicative of the limitations of state abstractions: During most time steps, any agent using a state abstraction remains in the same abstract state. As a result, state abstraction fails to mitigate the difficulties of exploration (and possibly makes it more challenging), and makes credit assignment even harder by spreading out any received feedback across a variety of aggregated states. Thus, the *learning* aspects of the problem are still challenging (possibly more so), even if the planning horizon is shortened.

We suggest that the combination of state *and* action abstraction can overcome these difficulties by changing the exploration problem and directly addressing credit assignment through temporally extended actions, while retaining the generalizability of useful state abstractions. To this end, we experiment with an algorithm for computing action abstractions for the lifelong setting; the algorithm computes options that move the agent between each abstract cluster. We compute both the state and action abstractions prior to beginning any learning as opposed to learning them online, which we leave as future work. Due to space constraints, we defer the presentation of the algorithm to the full version of the paper. Its details, ultimately, are unimportant for our present investigation; here, we focus on the impact of various combinations of abstractions on learning. Algorithms run with action abstraction are labeled with 'aa' in Figures 1b and 1c. Note that in both Four Rooms and Taxi, just adding the action abstraction improves learning, but the combination of state *and* action abstraction outperforms all other learners.

As a final note, we observe that the introduced combination of state and action abstractions defines a closed operator on MDPs, and so suggests a natural method for computing hierarchies for RL:

Theorem 2.4. *A given state-action abstraction pair* (ϕ, ω) *induces a closed operation on MDPs.*

From the above result, we find a natural method for learning, computing, and making decisions with hierarchies. We suggest the simplicity of construction of these hierarchies offers rich directions for future analysis, both in terms of quality of the learning and planning speedups afforded by the hierarchy, and in terms of efficiently computing or learning the representational structure itself. Further, these computed hierarchies can be paired with any arbitrary RL algorithm. As a proof of concept, we compute a simple hierarchy using this methodology for a $25{\times}25$ lifelong Four Room task distribution and present results comparing a Q-Learning agent with and without the hierarchy in Figure 1d. The results support the (well established) conclusion that effective hierarchies reduce learning difficulty [9, 14, 3, 2, 7, 16]. Implementation details of the hierarchy along with code for reproducing all results are publicly available.[2]

In summary, we introduce a new, efficiently computable, value-preserving state-abstraction class. We offer a simple empirical demonstration that state abstractions do not necessarily simplify learning while joint state–action abstractions do improve learning. We end highlighting a natural method of deriving abstraction hierarchies, suggesting directions for future analysis and experiments.

[2]`https://github.com/david-abel/rl_abstraction`

References

[1] David Abel, D Ellis Hershkowitz, and Michael L. Littman. Near optimal behavior via approximate state abstraction. In *ICML*, 2016.

[2] Bram Bakker and Jürgen Schmidhuber. Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In *Proc. of the 8-th Conf. on Intelligent Autonomous Systems*, pages 438–445, 2004.

[3] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379, 2003.

[4] Andrew G Barto, Satinder Singh, and Nuttapong Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *Proc. 3rd Int. Conf. Development Learn*, pages 112–119, 2004.

[5] Emma Brunskill and Lihong Li. PAC-inspired option discovery in lifelong reinforcement learning. In *ICML*, pages 316–324, 2014.

[6] Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1281–1288, 2004.

[7] Peter Dayan and Geoffrey Hinton. Feudal Reinforcement Learning. *Advances in neural information processing systems*, pages 271–278, 1993.

[8] Thomas Dean, Robert Givan, and Sonia Leach. Model reduction techniques for computing approximately optimal solutions for Markov decision processes. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 124–131. Morgan Kaufmann Publishers Inc., 1997.

[9] Thomas G Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[10] Eyal Even-Dar and Yishay Mansour. Approximate equivalence of Markov decision processes. In *Learning Theory and Kernel Machines*, pages 581–594. Springer, 2003.

[11] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 162–169. AUAI Press, 2004.

[12] Norm Ferns, Pablo Samuel Castro, Doina Precup, and Prakash Panangaden. Methods for computing state similarity in Markov decision processes. *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, 2006.

[13] Marcus Hutter. Extreme state aggregation beyond markov decision processes. *Theoretical Computer Science*, 650:73–91, 2016.

[14] Nicholas K. Jong and Peter Stone. Hierarchical model-based reinforcement learning. *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 432–439, 2008.

[15] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, pages 237–285, 1996.

[16] George Konidaris. Constructing abstraction hierarchies using a skill-symbol loop. In *IJCAI: proceedings of the conference*, volume 2016, page 1648. NIH Public Access, 2016.

[17] George Konidaris and Andre S Barreto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems*, pages 1015–1023, 2009.

[18] George Konidaris and Andrew G Barto. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, volume 7, pages 895–900, 2007.

[19] Richard E Korf. Macro-operators: A weak method for learning. *Artificial intelligence*, 26(1): 35–77, 1985.

[20] Lihong Li, Thomas J Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for MDPs. In *ISAIM*, 2006.

[21] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[22] Satinder P Singh. Scaling reinforcement learning algorithms by learning variable temporal resolution models. In *Proceedings of the Ninth International Machine Learning Conference*, pages 406–415, 1992.

[23] 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.

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

Appendix

We here present the proofs of theorems and the pseudocode for the algorithm to compute options.

Proofs

Theorem 2.1. *Let $\mathbb{P}$ denote the space of all predicates on state pairs. For a given $\delta \in (0, 1]$, any state abstraction of a type defined by a predicate in $\mathbb{P}$ can be computed such that states are incorrectly aggregated only with probability $(1-\delta)^2$ after $m \geq \frac{2\ln\left(\frac{2}{\delta}\right)}{\delta^2}$ samples $\mathcal{R} \sim D$ from the distribution.*

Proof. For each sampled reward function $\mathcal{R}_i \sim D$, consider a graph G_i where each state s_i of the MDP is mapped to a vertex v_i. Let $e_i(v_1, v_2)$, denoting whether there is an edge between v_1 and v_2, be true if and only if under reward function $\mathcal{R}_i$ the predicate $p(s_1, s_2)$ is true.

Our goal is to compute a state abstraction of type ϕ_p that prescribes as many state-pair equivalences as possible while only making costly mistakes with low probability. A *costly* mistake is one such that, for the sampled reward function, there is a pair of states such that $\phi(s_1) = \phi(s_2)$ but $\neg e_i(v_1, v_2)$.

After m sampled reward functions, we compute the empirical mean of each edge:

$$\hat{e}(v_1, v_2) = \frac{1}{m}\sum_{i=1}^{m} e_i(v_1, v_2). \tag{5}$$

By the Hoeffding inequality, we bound the deviation of $\hat{e}$, letting $\varepsilon = \frac{\delta}{2}$:

$$\Pr\left(|\hat{e}(a,b) - \mathbb{E}\left[e(a,b)\right]| \geq \frac{\delta}{2}\right) \leq 2e^{\left(-m\delta^2/2\right)}.$$

Let $\delta = 2\exp\left(-m\delta^2/2\right)$. By algebra, $m = \frac{2\ln\left(\frac{2}{\delta}\right)}{\delta^2}$.

Now, consider the graph, $G_{\hat{e}}$, formed by including those edges for which $\hat{e}(v_1, v_2) - \frac{\delta}{2} \geq 1 - \delta$:

$$\forall_{v_1, v_2} : \hat{e}_G(v_1, v_2) = \mathbb{1}\left\{\hat{e}_i(v_1, v_2) - \frac{\delta}{2} \geq 1 - \delta\right\}.$$

The edges in $G_{\hat{e}}$ are those that occur with probability at least $1 - \delta$ according to $\hat{e}_i$. Further, the approximate edge weights $\hat{e}_i$ are *also* accurate (within $\frac{\delta}{2}$) with probability $1 - \delta$. Thus, with probability $(1-\delta)^2$, our edge approximation is accurate *and* the edge is a high probability edge. Thus, we do *not* make a costly mistake. Further, we label all other edges that are known with high probability.

Thus, we conclude that for $m \geq \frac{2\ln\left(\frac{2}{\delta}\right)}{\delta^2}$ sampled and solved MDPs, we can compute an abstraction ϕ_p that only makes costly mistakes with low probability. $\square$

....

Theorem 2.2. *Consider any transitive predicate on state pairs, p that takes computational complexity c_p to evaluate for a given state pair. The state abstraction type ϕ_p that induces the smallest abstract state space can be computed[3] in $\mathcal{O}(|\mathcal{S}|^2 \cdot c_p)$.*

Proof. Let c_p denote the computational complexity associated with computing the predicate p for a given state pair. Consider the algorithm consisting of the following four rules for constructing abstract clusters (which define the abstract states) using queries to each of the $|\mathcal{S}|^2$ state pairs. Let (s_i, s_j) denote the current state pair:

[3]Notably, the complexity of c_p dictates the overall complexity of computing ϕ_p.

1. If $p(s_i, s_j)$ is true, and neither state is in an abstract cluster yet, make a new cluster consisting of these two states.
2. If $p(s_i, s_j)$ is true and only one of the states is already in a cluster, add the other state to the existing cluster.
3. If $p(s_i, s_j)$ is true and both s_i and s_j are in different cluster, merge the clusters.
4. If $p(s_i, s_j)$ is false, add each state not yet in a cluster to its own cluster.

Running this algorithm makes one query per state pair, of which there are $|\mathcal{S}|^2$. Thus the complexity os $O\left(|\mathcal{S}|^2 \cdot c_p\right)$.

From steps 1-3, after iterating through the possible state pairs, there cannot exist a state pair (s_x, s_y) such that $p(s_x, s_y)$ is true but s_x and s_y are in different clusters. Further, by transitivity, when we apply the cluster merge in step 3, we are guaranteed that every state pair in the resulting cluster necessarily satisfies the predicate. Thus, we compute the smallest clustering definable by p. □

....

Theorem 3.3. *The $\phi_{Q^*,d}$ abstraction type is a subclass of $\phi_{Q^*_\varepsilon}$, with $d = \varepsilon$, and therefore, for a single MDP:*

$$V^*(s_0) - V^{\pi_{\phi_{Q^*,d}}}(s_0) \leq \frac{2d\text{RMAX}}{(1-\gamma)^2}. \tag{6}$$

Proof. For any two state-action pairs that satisfy the predicate ϕ^*_d, we know by definition of the predicate that for each action a, there exists a Q_{lower} such that:

$$Q_{lower} \leq Q(s_1, a) \leq Q_{lower} + d,$$
$$Q_{lower} \leq Q(s_2, a) \leq Q_{lower} + d.$$

Therefore, for each action a:

$$|Q(s_1, a) - Q(s_2, a)| \leq d. \tag{7}$$

Therefore, $\phi^*_{Q,d}$ is a subclass of $\phi^*_{Q,\varepsilon}$. □

....

Theorem 3.4. *A given state-action abstraction pair (ϕ, ω) induces a closed operation on MDPs.*

Proof. Let $\mathcal{R}^a_s$ denote $\mathcal{R}(s,a)$, $\mathcal{T}^a_{s,s'}$ denote $\mathcal{T}(s,a,s')$. Then, by the multi-time model introduced by Sutton et al. [23]:

$$\mathcal{R}_\omega(s,o) = \mathcal{R}^{\pi_o(s)}_s + \sum_{s' \in \mathcal{S}} (1 - \beta_o(s'))\mathcal{T}^{\pi_o(s)}_{s,s'} \mathcal{R}_\omega(s', o)$$
$$\mathcal{T}^o_{s,x} = \sum_{s' \in \mathcal{S}} \left[(1 - \beta_0(s'))\mathcal{T}^o_{s',x} + \beta(s')\mathbb{1}_{s'=x}\right].$$

For some weighting function $w(s)$, we define M':

- $\mathcal{S}_\phi = \{\phi(s) : \forall_{s \in \mathcal{S}}\}$
- $\mathcal{A}_\omega = \mathcal{O}$
- $\mathcal{R}^o_{s_\phi} = \sum_{s \in \phi^{-1}(s_\phi)} \mathcal{R}_\omega(s,o) w(s)$
- $\mathcal{T}^o_{s_\phi, s'_\phi} = \sum_{s \in \phi^{-1}(s_\phi)} \sum_{x \in \phi^{-1}(s'_\phi)} \mathcal{T}^o_{s,x} w(s)$

Thus, the primary components of the MDP are well defined. Choice of γ will induce different value differences between M and M' depending on the MDP structure as well as choice of ϕ and ω, and is therefore left unspecified. Notably, the reward and transition models are Markov with the respect to the provided options and abstract states, thereby forming a well defined MDP. □

Algorithm

We here also include the pseudocode for the algorithm used to compute options in experiments:

Algorithm 1 Directed Options

```
INPUT: $D$, $\phi$, $S_0$
OUTPUT: $\mathcal{O}$

$\mathcal{O} = \{\ \}$
for $(s_{\phi,1}, s_{\phi,2}) \in \mathcal{S}_\phi \times \mathcal{S}_\phi$ do                ▷ Compute clique.
    $\mathcal{I}(s) \triangleq \mathbb{1}\{\phi(s) = s_{\phi,1}\}$
    $\beta(s) \triangleq \mathbb{1}\{\phi(s) = s_{\phi,2}\} \cdot (1-\gamma)$
    $\mathcal{R}_{1,2}(s) \triangleq \mathbb{1}\{\phi(s) = s_{\phi,2}\}$
    $s_0 = S_0$.get_start_of_cluster($s_{\phi,1}$)
    $\pi^*_{1,2} = \arg\max_\pi V^\pi_{\mathcal{R}_{1,2}}(s_0)$
    $o_{1,2} = Option(\mathcal{I}, \beta, \pi)$
    $\mathcal{O}.add(o_{1,2})$
end for
for $o \in \mathcal{O}$ do                                   ▷ Remove redundant options.
    if $\neg\exists_{s \in o.\mathcal{I}} : \Pr(o.\mathcal{I}(s'_t) \vee o.\beta(s'_t) \mid s, o.\pi) = 1$ then
        $\mathcal{O}.remove(o)$
    end if
end for                                                     ▷ Add optimal self loops.
for $\mathcal{R} \in D$ do
    if $\exists_{s \in \phi^{-1}(s_\phi), a \in \mathcal{A}} : \mathcal{R}(s,a) > 0$ then
        $\mathcal{I}(s) \triangleq \mathbb{1}\{\phi(s) = s_\phi\}$
        $\beta(s) \triangleq \mathbb{1}\{\phi(s) \neq s_\phi\} \cdot (1-\gamma)$
        $s_0 = S_0$.get_start_of_cluster($s_\phi$)
        $\pi^*_{self} = \arg\max_\pi V^\pi_{\mathcal{R}}(s_0)$
        $o_{self} = Option(\mathcal{I}, \beta, \pi^*_{self})$
        $\mathcal{O}.add(o_{self})$
    end if
end for
return $\mathcal{O}$
```

The algorithm takes as input the distribution over reward functions, D, a state abstraction ϕ, and a set of ground start states S_0, which contains a start state for each cluster. The algorithm proceeds in two stages:

1. Using the given state abstraction, compute the set of options that most expediently connects the abstract states. To complete this step, there are two subprocesses:
    (a) Compute every option that connects a pair of abstract states. That is, we form a clique over the graph with nodes $\mathcal{S}_\phi$, and options for edges.
    (b) Remove all options that are redundant. A redundant option is one such that, for every ground state in the initiation set, the option's policy transitions through some other abstract state before arriving in the option's terminal state.
2. Compute an option representing an optimal self-loop for each cluster.