A Appendix

A.1 Algorithm

In the main text, we have discussed a naive option grounding algorithm:

- **Algorithm 1 (IRL-naive)**: naive option grounding algorithm which performs IRL over all starting states independently.

In this section, we present additional algorithms useful for option grounding:

- **Algorithm 2 (IRL module for IRL-naive)**: the IRL algorithm adapted from [Syed *et al.*, 2008], and used by the IRL-naive option grounding algorithm (Algorithm 1) to find the option policies starting from each starting state.
- **Algorithm 3 (IRL-batch):** an efficient option grounding algorithm which improves IRL-naive and performs batched learning using IRL.
- **Algorithm 4: (IRL module for IRL-batch)** the IRL module used by the IRL-batch option grounding algorithm (Algorithm 3) to find option policies starting from a batch of starting states.

IRL algorithm for the naive option grounding (Algorithm 2)

First, we introduce the IRL module adapted from [Syed *et al.*, 2008] and used by the IRL-naive option grounding algorithm:

Algorithm 2 IRL (used by Algorithm 1 IRL-naive)

Input: Augmented MDP $\mathcal{M} = \langle \mathcal{S}', \mathcal{A}', P', r, \gamma \rangle$, starting state s_{start}, state-action features $\boldsymbol{\theta}\colon \mathcal{S}' \times \mathcal{A}' \to \mathbb{R}^d$, abstract option $\boldsymbol{\psi}^{\bar{o}} \in \mathbb{R}^d$.

Output: ground and abstract option feature difference $\epsilon = \sum_{k=1}^{d} \epsilon_k$, option policy $\pi^o_{s_{\text{start}}}$

Solve LP for visitation frequencies $\mu_{s,a}$ and feature matching errors ϵ_k

$$\min_{\mu,\epsilon_k} \quad \sum_k \epsilon_k - \lambda_1 \sum_a \mu_{s_{\text{null}},a} + \lambda_2 \sum_s \mathbb{1}_{\mu_{s,a_{\text{T}}}>0} \tag{3}$$

$$\text{s.t.} \quad \sum_a \mu_{s,a} = \mathbb{1}_{s=s_{\text{start}}} + \gamma \sum_{s',a} P'(s|s',a)\mu_{s',a} \qquad \forall s \in \mathcal{S}' \tag{4}$$

$$\sum_{s,a} \mu_{s,a}\boldsymbol{\theta}_k(s,a) - \boldsymbol{\psi}^{\bar{o}}_k \le \epsilon_k \qquad \forall k \in 1,\dots,d \tag{5}$$

$$\sum_{s,a} \mu_{s,a}\boldsymbol{\theta}_k(s,a) - \boldsymbol{\psi}^{\bar{o}}_k \ge -\epsilon_k \qquad \forall k \in 1,\dots,d \tag{6}$$

$$\mu_{s,a} \ge 0 \quad \forall s \in \mathcal{S}', a \in \mathcal{A}' \tag{7}$$

Compute option policy $\pi^o_{s_{\text{start}}}(a|s) = \frac{\mu_{s,a}}{\sum_a \mu_{s,a}}$

The algorithm finds the policy and termination condition of the option $\bar{o}$ in the following two steps:

1. Compute the state-action visitation frequencies $\mu_{s,a}$ such that the corresponding expected feature vector (approximately) matches the abstract option feature $\boldsymbol{\psi}^{\bar{o}}$.
2. Compute the option policy (which includes the termination condition) from $\mu_{s,a}$.

Linear program. Adapting from prior work which uses linear programming approaches for solving MDPs and IRL [Syed *et al.*, 2008; Malek *et al.*, 2014; Manne, 1960], our LP aims to find the state-action visitation frequencies $\mu_{s,a}$ for all states and actions, which together (approximately) match the abstract option feature, i.e., $\boldsymbol{\psi}^o_{\text{start}} = \sum_{s,a} \mu_{s,a}\boldsymbol{\theta}(s,a) \approx \boldsymbol{\psi}^{\bar{o}}$. In particular, $\|\boldsymbol{\psi}^o_{\text{start}} - \boldsymbol{\psi}^{\bar{o}}\|_1 \le \sum_{k=1}^d \epsilon_k$.

Inputs: Recall that to enable the modelling of options and their termination conditions, the input augmented MDP $\mathcal{M} = \langle \mathcal{S}', \mathcal{A}', P', r, \gamma \rangle$ was constructed in Algorithm 1 by adding a null state s_{null} and termination action a_{T} such that a_{T} leads from any regular state to s_{null}.

Variables: 1. ϵ_k: Upper bounds for the absolute difference between the (learner) learned ground option successor feature and the abstract option (expert) feature in the k-th dimension.

2. $\mu_{s,a}$: Expected cumulative state-action visitation of state-action pair s, a. Additionally, denote μ_s as the expected cumulative state visitation of s, i.e., $\forall s \in \mathcal{S}, a \in \mathcal{A}$,

$$\mu_{s,a} = \mathbb{E}_{\mathcal{M},\pi}[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{S_t=s,A_t=a} | s_0 \sim P_{\text{start}}], \quad \text{and} \quad \mu_s = \mathbb{E}_{\mathcal{M},\pi}[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{S_t=s} | s_0 \sim P_{\text{start}}] \tag{8}$$

where P_{start} is the distribution over starting states. In this naive option grounding algorithm the start state is a single state s_{start}. Observe that μ_s and $\mu_{s,a}$ are related by the policy $\pi\colon \mathcal{S} \to \mathcal{A}$ as

$$\mu_{s,a} = \pi(a|s)\mu_s, \quad \text{and} \quad \mu_s = \sum_a \pi(a|s)\mu_s = \sum_a \mu_{s,a}. \tag{9}$$

And the Bellman flow constraint is given by either μ_s or $\mu_{s,a}$:

$$\mu_s = \mathbb{1}_{s=s_{\text{start}}} + \gamma \sum_{s',a} P(s|s',a)\pi(a|s')\mu_{s'} \iff \sum_a \mu_{s,a} = \mathbb{1}_{s=s_{\text{start}}} + \gamma \sum_{s',a} P(s|s',a)\mu_{s',a} \tag{10}$$

By Equation (9), the policy π corresponding to the state-action visitation frequencies is computed as

$$\pi^o_{s_{\text{start}}} = \frac{\mu_{s,a}}{\mu_s} = \frac{\mu_{s,a}}{\sum_a \mu_{s,a}}. \tag{11}$$

Objective function: $\sum_k \epsilon_k - \lambda_1 \sum_a \mu_{s_{\text{null, a}}} + \lambda_2 \sum_s \mathbb{1}_{\mu_{s,a_{\text{T}}}>0}$

1. $\sum_k \epsilon_k$: the first term is the feature difference ϵ between the abstract and computed ground option.

2. $-\lambda_1 \sum_a \mu_{s_{\text{null, a}}}$: The second term is a small penalty on the option length to encourage short options. This is achieved by putting a small bonus (e.g., $\lambda_1 = 0.01$) on the expected cumulative visitation of the null state s_{null}, which the agent reaches after using the terminate action.

3. $\lambda_2 \sum_s \mathbb{1}_{\mu_{s,a_{\text{T}}}>0}$: The last term is a regularisation on the number of terminating states. This helps guide the LP to avoid finding mixtures of option policies which together match the feature expectation.

In this way, the linear program can find an option policy which terminates automatically.

Constraints: Equation (4) is the Bellman flow constraint [Syed *et al.*, 2008; Malek *et al.*, 2014] which specifies how the state-action visitation frequencies are related by the transition dynamics, see Equation (10). Equations (5) and (6) define the feature difference ϵ of the ground option $\boldsymbol{\psi}^o_{s_{\text{start}}} = \sum_{s,a} \mu_{s,a}\boldsymbol{\theta}(s,a)$ and abstract option $\boldsymbol{\psi}^{\bar{o}}$ for the k-th dimension.

Step 2 derives the policy from the state-action visitation frequencies $\mu_{s,a}$, see Equation (11).

Grounding Abstract Options for Starting States in Batches (Algorithm 3)

Algorithm 3 Grounding Abstract Options (IRL-Batch)

```
    Input: MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, abstract option $\boldsymbol{\psi}^{\bar{o}}$, $\epsilon_{\text{threshold}}$
    Output: initiation set $I^{\bar{o}}$, dictionary of ground option policies $\Pi^{\bar{o}}$, termination probabilities $\Xi^{\bar{o}}$
 1: // Construct augmented MDP
 2: $\mathcal{S}' \leftarrow \mathcal{S} \cup \{s_{\text{null}}\}, \mathcal{A}' \leftarrow \mathcal{A} \cup \{a_{\text{T}}\}, P' \leftarrow P$
 3: $\forall s \in \mathcal{S}' \colon P'(s_{\text{null}} \mid s, a_{\text{T}}) = 1; \forall a \in \mathcal{A}' \colon P'(s_{\text{null}} \mid s_{\text{null}}, a) = 1$

 4: // Find ground options for all starting states
 5: $I^{\bar{o}}, \Pi^{\bar{o}}, \Xi^{\bar{o}}$ = MATCH-AND-DIVIDE($\mathcal{S}' \setminus \{s_{\text{null}}\}$)

 6: // Recursive Function
 7: function MATCH-AND-DIVIDE($c_{\text{start}}$)
 8:     $\epsilon_{c_{\text{start}}}, \Pi^{o}_{c_{\text{start}}} \leftarrow$ IRL($\mathcal{S}', \mathcal{A}', P', \gamma, c_{\text{start}}, \boldsymbol{\psi}^{\bar{o}}$),        ▷ for IRL see Algorithm 4
 9:     $c_{\text{match}}, c_{\text{no-match}}, C_{\text{ambiguous}}$ = CLASSIFY($c_{\text{start}}, \Pi^{o}_{c_{\text{start}}}, \epsilon_{c_{\text{start}}}, \epsilon_{\text{threshold}}$)
10:     for all $s_{\text{start}} \in c_{\text{match}}$ do
11:         $\pi^{o}_{s_{\text{start}}} \leftarrow \Pi^{o}_{c_{\text{start}}}(s_{\text{start}})$
12:         $I^{\bar{o}} \leftarrow I^{\bar{o}} \cup \{s_{\text{start}}\}, \Pi^{\bar{o}}(s_{\text{start}}) = \pi^{o}_{s_{\text{start}}}, \Xi^{\bar{o}}(s_{\text{start}}) = \beta^{o},$ where $\beta^{o}(s) = \pi^{o}_{s_{\text{start}}}$
13:     end for
14:     if $C_{\text{ambiguous}} \neq \emptyset$ then
15:         for all $c_i \in C_{\text{ambiguous}}$ do
16:             MATCH-AND-DIVIDE($c_i$)        ▷ Note: $C_{\text{ambiguous}}$ is a set of clusters
17:         end for
18:     end if
19:     return $I^{\bar{o}}, \Pi^{\bar{o}}, \Xi^{\bar{o}}$
20: end function

21: // Classify start states according to the policy found by IRL
22: function CLASSIFY($c_{\text{start}}, \Pi^{o}_{c_{\text{start}}}, \epsilon_{c_{\text{start}}}, \epsilon_{\text{threshold}}$)
23:     $c_{\text{match}}, c_{\text{no-match}}, c_{\text{ambiguous}} \leftarrow \emptyset$
24:     for all $s_{\text{start}} \in c_{\text{start}}$ do
25:         Execute $o$ from $s_{\text{start}}$ to get successor feature $\boldsymbol{\psi}^{o}_{s_{\text{start}}}$ and terminating distribution $P^{o}_{s_{\text{start}},s'}$
26:         Compute feature difference $\epsilon^{o}_{s_{\text{start}}} = |\boldsymbol{\psi}^{\bar{o}} - \boldsymbol{\psi}^{o}_{s_{\text{start}}}|$
27:     end for
28:     $c_{\text{no-match}} \leftarrow c_{\text{start}}$ if $\min_{s_{\text{start}} \in c_{\text{start}}} \epsilon^{o}_{s_{\text{start}}} > \epsilon_{\text{threshold}}$ and $\epsilon_{c_{\text{start}}} > \epsilon_{\text{threshold}}$
29:     $c_{\text{match}} \leftarrow \{s_{\text{start}} \in c_{\text{start}} | \epsilon^{o}_{s_{\text{start}}} \leq \epsilon_{\text{threshold}}\}$
30:     $c_{\text{ambiguous}} \leftarrow c_{\text{start}} \setminus c_{\text{match}}, c_{\text{no-match}}$
31:     if $c_{\text{ambiguous}} \neq \emptyset$ then        ▷ Cluster by termination distributions or successor features$^{(1)}$
32:         $C_{\text{ambiguous}} \leftarrow$ CLUSTER($c_{\text{ambiguous}}, P^{o}_{s_{\text{start}},s'}$)
33:     end if
34:     return $c_{\text{match}}, c_{\text{no-match}}, C_{\text{ambiguous}}$
35: end function
```

(1) In practice both approaches work well and address the challenge that the solution found could be a mixture of different option policies which individually yield different SF but together approximate the target SF. Clustering by SF approaches this issue by enforcing the SF of option policies from all s_{start} in a cluster to be similar and match the target. Clustering by termination distribution is an effective heuristic which groups together nearby s_{start}, also resulting in options with similar SF.

Based on the naive option grounding algorithm (Algorithm 1), we now introduce an efficient algorithm for grounding abstract options, which performs IRL over the start states in batches (Algorithm 3).

Challenges. The main challenges regarding performing batched IRL for grounding the options are: 1. By naively putting a uniform distribution over all possible starting states, the IRL LP cannot typically find the ground options which match the abstract option. Moreover, a closely related problem as well as one of the reasons for the first problem is 2. Since there are many different starting states, the state-action visitation frequencies found by the LP for matching the option feature may be a mixture of different option policies from the different starting states, and the induced ground option policies individually cannot achieve the successor feature of the abstract option.

Solutions. For the first challenge, we introduce the batched IRL module (Algorithm 4) to be used by IRL-batch. It flexibly learns a starting state distribution with entropy regularisation.

For the second challenge, Algorithm 3 is a recursive approach where each recursion performs batched-IRL on a set of starting states. Then, by executing the options from each starting state using the transition dynamics, we prune the starting states which successfully match the abstract successor option's feature, and those where matching the abstract option is impossible. And cluster the remaining ambiguous states based on their option termination distribution (or their achieved successor features) and go to the next recursion. Intuitively, we form clusters of similar states (e.g., which are nearby and belong to a same community according to the option termination distribution). And running batched-IRL over a cluster of similar starting states typically returns a single ground option policy which applies to all these starting states.

Algorithm 3 (IRL-batch: Grounding Abstract Options in Batches). The algorithm first constructs the augmented MDP with the null states and terminate actions in the same way as the naive algorithm. Then it uses a recursive function *Match-And-Divide(c_{start})*, which first computes the ground option policies corresponding to the set of starting states c_{start} through batched IRL (Algorithm 4) over c_{start}, then *Classify* the starting states by their corresponding ground options' termination distributions or successor feature, into the following 3 categories: 1. c_{match}: the start states where an abstract option can be initiated (i.e., there exists a ground option whose successor feature matches the abstract option); 2. $c_{\text{no-match}}$: the start states where the abstract option cannot be initiated; and 3. $C_{\text{ambiguous}}$: a set of clusters dividing the remaining ambiguous states. If $C_{\text{ambiguous}}$ is not empty, then each cluster goes through the next recursion of *Match-And-Divide*. Otherwise the algorithm terminates and outputs the initiation set, option policy and termination conditions.

Algorithm 4 IRL (used by Algorithm 3 IRL-Batch)

```
Input: Augmented MDP $\mathcal{M} = \langle \mathcal{S}', \mathcal{A}', P', r, \gamma \rangle$,, state-action features $\boldsymbol{\theta} \colon \mathcal{S}' \times \mathcal{A}' \to \mathbb{R}^d$, abstract option $\boldsymbol{\psi}^{\bar{o}} \in \mathbb{R}^d$, a set of starting states $c_{\text{start}} \subseteq \mathcal{S}' \setminus \{s_{\text{null}}\}$.
Output: (joint over $c_{\text{start}}$) ground and abstract option feature difference $\epsilon = \sum_{k=1}^{d} \epsilon_k$, dictionary of ground option policy $\Pi^o(s_{\text{start}})$ for all start states in $c_{\text{start}}$
    // Step 1: Solve LP for state-action visitation frequencies $\mu_{s,a}$
```

$$
\begin{aligned}
\min_{\mu,\epsilon_k,p^{\text{start}}} \quad & \sum_k \epsilon_k - \lambda_1 \sum_a \mu_{s_{\text{null}},a} + \lambda_2 \sum_s p_s^{\text{start}} \log p_s^{\text{start}} + \lambda_3 \sum_s \mathbb{1}_{\mu_{s,a_{\text{T}}}>0} \\
\text{s.t.} \quad & \sum_a \mu_{s,a} = p_s^{\text{start}} + \gamma \sum_{s',a} P'(s|s',a)\mu_{s',a} && \forall s \in \mathcal{S}' \\
& \sum_{s,a} \mu_{s,a}\boldsymbol{\theta}_k(s,a) - \boldsymbol{\psi}_k^{\bar{o}} \le \epsilon_k && \forall k \in 1,\dots,d \\
& \sum_{s,a} \mu_{s,a}\boldsymbol{\theta}_k(s,a) - \boldsymbol{\psi}_k^{\bar{o}} \ge -\epsilon_k && \forall k \in 1,\dots,d \\
& \mu_{s,a} \ge 0 && \forall s \in P', a \in \mathcal{A}' \\
& \sum_{s \in c_{\text{start}}} p_s^{\text{start}} = 1 \\
& p_s^{\text{start}} \ge 0 && \forall s \in c_{\text{start}} \\
& p_s^{\text{start}} = 0 && \forall s \notin c_{\text{start}}
\end{aligned}
$$

```
    // Step 2: Compute dictionary of option policies $\Pi^o$, $\forall s_{\text{start}} \in c_{\text{start}}$:
```

$$
\forall s \in \mathcal{S}, a \in \mathcal{A}', \pi^o_{s_{\text{start}}}(a|s) = \frac{\mu_{s,a}}{\sum_a \mu_{s,a}}, \qquad \text{then add to dictionary} \quad \Pi^o(s_{\text{start}}) \leftarrow \pi^o_{s_{\text{start}}}
$$

Algorithm 4 (IRL module for IRL-batch). Compared with the IRL algorithm presented in Algorithm 1, this batched algorithm performs IRL over a batch of starting states. As we discussed in the above challenges, naively putting a uniform distribution over all possible starting states would typically fail. Therefore, we enable the LP to *learn a starting state distribution* p_s^{start} which best matches the abstract option feature. However, the LP would again reduce to a single starting state which best matches the abstract option. To counteract this effect, we add a small entropy regularisation to the objective to increase the entropy of the starting state distribution, i.e., we add $\lambda_2 \sum_s p_s^{\text{start}} \log p_s^{\text{start}}$. Clearly, entropy is not a linear function. In practise, the objective is implemented using a piecewise-linear approximation method provided by the Gurobi optimisation package.[4]

[4]https://www.gurobi.com/

A.2 Feature-based (Variable-reward) SMDP Abstraction

In this section, we provide a general framework for feature-based abstraction using the variable-reward SMDPs. The state mapping and action mapping functions $f(s), g_s(o)$ can be defined to instantiate a new abstraction method.

Definition A.1 (Abstract $\boldsymbol{\psi}$-SMDP)**.** *Let $\mathcal{M} = \langle \mathcal{S}, \mathcal{O}, P, \boldsymbol{\psi}, \gamma \rangle$ be a ground $\boldsymbol{\psi}$-SMDP. We say that $\bar{\mathcal{M}} = \langle \bar{\mathcal{S}}, \bar{\mathcal{O}}, \bar{P}, \bar{\boldsymbol{\psi}}, \gamma \rangle$ is an* abstract $\boldsymbol{\psi}$-SMDP of $\mathcal{M}$ *if there exists (1) a state abstraction mapping $f\colon \mathcal{S} \to \bar{\mathcal{S}}$ which maps each ground state to an abstract state, (2) a weight function $w\colon \mathcal{S} \to [0,1]$ over the ground states such that $\forall \bar{s} \in \bar{\mathcal{S}}, \sum_{s \in f^{-1}(\bar{s})} w_s = 1$, (3) a state-dependent option abstraction mapping $g_s\colon \mathcal{O} \to \bar{\mathcal{O}}$, and (4) the abstract transition dynamics and features are*

$$\bar{P}^{\bar{o}}_{\bar{s},\bar{s}'} = \sum_{s \in f^{-1}(\bar{s})} w_s \sum_{s' \in f^{-1}(\bar{s}')} P^{g_s^{-1}(\bar{o})}_{s,s'}, \text{ and } \quad \bar{\boldsymbol{\psi}}^{\bar{o}}_{\bar{s}} = \sum_{s \in f^{-1}(\bar{s})} w_s \boldsymbol{\psi}^{g_s^{-1}(\bar{o})}_s.$$

A.3 Reward-based MDP and SMDP abstraction

In the following we define the abstract MDP and abstract SMDPs, following the conventional notations of [Li *et al.*, 2006; Abel *et al.*, 2016; Ravindran and Barto, 2003].

Definition A.2 (Abstract MDP)**.** *Let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$ be a ground MDP. We say that $\bar{\mathcal{M}} = \langle \bar{\mathcal{S}}, \mathcal{A}, \bar{P}, \bar{r}, \gamma \rangle$ is an* abstract MDP of $\mathcal{M}$ *if there exists (1) a state abstraction mapping $f\colon \mathcal{S} \to \bar{\mathcal{S}}$, which maps each ground state to an abstract state, (2) a weight function $w\colon \mathcal{S} \to [0,1]$ for the ground states such that $\forall \bar{s} \in \bar{\mathcal{S}}$, $\sum_{s \in f^{-1}(\bar{s})} w_s = 1$, and (3) a state-dependent action mapping $g_s\colon \mathcal{A} \to \bar{\mathcal{A}}$, the abstract transition dynamics and rewards are defined as*

$$\bar{P}^{\bar{a}}_{\bar{s},\bar{s}'} = \sum_{s \in f^{-1}(\bar{s})} w_s \sum_{s' \in f^{-1}(\bar{s}')} P^{g_s^{-1}(\bar{a})}_{s,s'}, \text{ and } \quad \bar{r}^{\bar{a}}_{\bar{s}} = \sum_{s \in f^{-1}(\bar{s})} w_s r^{g_s^{-1}(\bar{a})}_s.$$

In cases in which multiple ground actions map to the same abstract action, $g_s^{-1}(\bar{a})$ picks one of the ground actions. g_s is commonly defined as the identity mapping [Li *et al.*, 2006; Abel *et al.*, 2016]. We now generalize this definition to abstract SMDPs:

Definition A.3 (Abstract SMDP)**.** *Let $\mathcal{M} = \langle \mathcal{S}, \mathcal{O}, P, r, \gamma \rangle$ be a ground SMDP. We say that $\bar{\mathcal{M}} = \langle \bar{\mathcal{S}}, \bar{\mathcal{O}}, \bar{P}, \bar{r}, \gamma \rangle$ is an* abstract SMDP of $\mathcal{M}$ *if there exists (1) a state abstraction mapping $f\colon \mathcal{S} \to \bar{\mathcal{S}}$ which maps each ground state to an abstract state, (2) a weight function $w\colon \mathcal{S} \to [0,1]$ over the ground states such that $\forall \bar{s} \in \bar{\mathcal{S}}, \sum_{s \in f^{-1}(\bar{s})} w_s = 1$, (3) a state-dependent option abstraction mapping $g_s\colon \times \mathcal{O} \to \bar{\mathcal{O}}$, and (4) the abstract transition dynamics and rewards are*

$$\bar{P}^{\bar{o}}_{\bar{s},\bar{s}'} = \sum_{s \in f^{-1}(\bar{s})} w_s \sum_{s' \in f^{-1}(\bar{s}')} P^{g_s^{-1}(\bar{o})}_{s,s'}, \text{ and } \quad \bar{r}^{\bar{o}}_{\bar{s}} = \sum_{s \in f^{-1}(\bar{s})} w_s r^{g_s^{-1}(\bar{o})}_s.$$

In cases in which multiple ground options map to the same abstract option, $g_s^{-1}(\bar{o})$ picks one of the ground options, e.g., the option of shortest duration, maximum entropy, etc.

A.4 Relation to Other MDP Abstraction Methods

The framework of MDP abstraction was first introduced by [Dean and Givan, 1997] through stochastic bisimulation, and [Ravindran and Barto, 2002] extended it to MDP homomorphisms. Later, [Li *et al.*, 2006] classified exact MDP abstraction into 5 categories and [Abel *et al.*, 2016] formulated their approximate counterparts: model-irrelevance, Q^π-irrelevance, Q^*-irrelevance, a^*-irrelevance and π^*-irrelevance abstractions. Our successor homomorphism follows the formulation of MDP homomorphism [Ravindran and Barto, 2003], which broadly fall into the category of model-irrelevance abstraction, where states are aggregated according to their one-step/multi-step transition dynamics and rewards. On the other hand, abstraction schemes which aggregate states according to their Q-values are Q^*-irrelevance abstraction and a^* abstraction, where Q^* aggregate states according to all actions, while a^* aggregate states with the same optimal action and Q-value, cf. Figure 1 for an illustration of the induced abstract MDPs. Different from prior abstraction formulations (Definition A.2) which are reward-based, our feature-based successor homomorphism produces abstract models with meaningful temporal semantics, and is robust under task changes. Furthermore, we include a generic formulation of feature-based (variable-reward) abstraction (Definition A.1), which provides a basis for potential feature-based abstractions other than successor homomorphism.

A.5 Proof for Theorem 4.1

Theorem 4.1. *Let $w_r\colon \mathbb{R}^d \to \mathbb{R}$ be a linear reward vector such that $r^a_s = w_r^T \boldsymbol{\theta}(s,a)$. Under this reward function, the value of an optimal abstract policy obtained through the $(\epsilon_P, \epsilon_\psi)$-approximate successor homomorphism is close to the optimal ground SMDP policy, where the difference is bounded by $\frac{2\kappa}{(1-\gamma)^2}$, where $\kappa = |w_r|(2\epsilon_\psi + \frac{\epsilon_P |\bar{\mathcal{S}}| \max_{s,a} |\boldsymbol{\theta}(s,a)|}{1-\gamma})$. (c.f. appendix for the proof)*

Proof. Given ϵ-Approximate Successor Homomorphism: $h = (f(s), g_s(o), w_s)$ from ground $\boldsymbol{\psi}$-SMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{O}, P, \boldsymbol{\psi}, \gamma \rangle$

to abstract $\boldsymbol{\psi}$-SMDP $\bar{\mathcal{M}} = \langle \bar{\mathcal{S}}, \bar{\mathcal{O}}, \bar{P}, \boldsymbol{\psi}, \gamma \rangle$, such that $\forall s_1, s_2 \in \mathcal{S}, o_1, o_2 \in \mathcal{O}, h(s_1, o_1) = h(s_2, o_2) \implies$

$$|\sum_{s_j \in f^{-1}(\bar{s}')} P^{o_1}_{s_1,s_j} - \sum_{s_j \in f^{-1}(\bar{s}')} P^{o_2}_{s_2,s_j}| \leq \epsilon_P \tag{12}$$

$$|\boldsymbol{\psi}^{o_1}_{s_1} - \boldsymbol{\psi}^{o_2}_{s_2}| \leq \epsilon_{\boldsymbol{\psi}} \tag{13}$$

The abstract transition dynamics and features are

$$P^{\bar{o}}_{\bar{s},\bar{s}'} = \sum_{s \in f^{-1}(\bar{s})} w_s \sum_{s' \in f^{-1}(\bar{s}')} P^{g_s^{-1}(\bar{o})}_{s,s'}, \text{ and } \boldsymbol{\psi}^{\bar{o}}_{\bar{s}} = \sum_{s \in f^{-1}(\bar{s})} w_s \boldsymbol{\psi}^{g_s^{-1}(\bar{o})}_s. \tag{14}$$

We show that given features $\boldsymbol{\theta} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ in the underlying (feature-based) MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, \boldsymbol{\theta}, \gamma \rangle$, and linear reward function on the features $w_r \in \mathbb{R}^d$, the difference in value of the optimal policy in the induced abstract SMDP and the ground SMDP is bounded by $\frac{2\kappa}{(1-\gamma)^2}$, where $\kappa = |w_r|(2\epsilon_{\boldsymbol{\psi}} + \frac{\epsilon_P |\bar{\mathcal{S}}| \max_{s,a} |\boldsymbol{\theta}(s,a)|}{1-\gamma})$.

To show the above error bound, we extend the proof of [Abel *et al.*, 2016] for the error bound induced by approximate MDP abstraction, which follows the following three steps:

Step 1: Show that $\forall s_1 \in \mathcal{S}, o_1 \in \mathcal{O}, \bar{s} = f(s_1), \bar{o} = g(o_1) \implies |Q(\bar{s}, \bar{o}) - Q(s_1, o_1)| \leq \frac{\epsilon}{1-\gamma}$.

$$r^{\bar{o}}_{\bar{s}} = w_r^T \boldsymbol{\psi}^{\bar{o}}_{\bar{s}} = \sum_{s_i \in f^{-1}(\bar{s})} w_{s_i} w_r^T \boldsymbol{\psi}^{g_{s_i}^{-1}(\bar{o})}_{s_i} = \sum_{s_i \in f^{-1}(\bar{s})} w_{s_i} r^{g_{s_i}^{-1}(\bar{o})}_{s_i} \text{ denote } r^{g_{s_i}^{-1}(\bar{o})}_{s_i} \text{ as } w_{s_i} r^{o_i}_{s_i} \tag{15}$$

$$Q(\bar{s}, \bar{o}) = \mathbb{E}[r^{\bar{o}}_{\bar{s}} + \gamma^k \max_{\bar{o}'} Q(\bar{s}', \bar{o}')] \overset{\text{Eq.(14),(15)}}{=} \sum_{s_i \in f^{-1}(\bar{s})} w_{s_i} r^{o_i}_{s_i} + \sum_{s_i \in f^{-1}(\bar{s})} w_{s_i} \sum_{\bar{s}' \in \bar{\mathcal{S}}'} \sum_{s_j \in f^{-1}(\bar{s}')} P^{g^{-1}(\bar{o})}_{s_i,s_j} \max_{\bar{o}'} Q(\bar{s}', \bar{o}') \tag{16}$$

$$Q(s_1, o_1) = \mathbb{E}[r^{o_1}_{s_1} + \gamma^k \max_{o'_j} Q(s'_j, o'_j)] = r^{o_1}_{s_1} + \sum_{\bar{s}' \in \bar{\mathcal{S}}'} \sum_{s_j \in f^{-1}(\bar{s}')} P^{o_1}_{s_1,s_j} \max_{\bar{o}'} Q(s_j, o'_j) \tag{17}$$

$$|Q(\bar{s}, \bar{o}) - Q(s_1, o_1)| \tag{18}$$

$$= |\sum_{s_i \in f^{-1}(\bar{s})} w_{s_i} (r^{o_i}_{s_i} - r^{o_1}_{s_1}) + \sum_{s_i \in f^{-1}(\bar{s})} w_{s_i} \sum_{\bar{s}' \in \bar{\mathcal{S}}'} \sum_{s_j \in f^{-1}(\bar{s}')} \left(P^{g_{s_i}^{-1}(\bar{o})}_{s_i,s_j} \max_{\bar{o}'} Q(\bar{s}', \bar{o}') - P^{o_1}_{s_1,s_j} \max_{o'_j} Q(s'_j, o'_j) \right) | \tag{19}$$

$$\leq \underbrace{|\sum_{s_i \in f^{-1}(\bar{s})} w_{s_i} w_r^T (\boldsymbol{\psi}^{o_i}_{s_i} - \boldsymbol{\psi}^{o_1}_{s_1})| + |\sum_{s_i \in f^{-1}(\bar{s})} w_{s_i} \sum_{\bar{s}' \in \bar{\mathcal{S}}'} \sum_{s_j \in f^{-1}(\bar{s}')} \left(P^{g_{s_i}^{-1}(\bar{o})}_{s_i,s_j} \max_{\bar{o}'} Q(\bar{s}', \bar{o}') - P^{o_1}_{s_1,s_j} \max_{\bar{o}'} Q(\bar{s}', \bar{o}') \right) |}_{(1)} \tag{20}$$

$$+ \underbrace{|\sum_{s_i \in f^{-1}(\bar{s})} w_{s_i} \sum_{\bar{s}' \in \bar{\mathcal{S}}'} \sum_{s_j \in f^{-1}(\bar{s}')} P^{o_1}_{s_1,s_j} \left(\max_{\bar{o}'} Q(\bar{s}', \bar{o}') - \max_{o'_j} Q(s'_j, o'_j) \right) |}_{(2)} \tag{21}$$

$$(1) \leq 2|w_r|\epsilon_{\boldsymbol{\psi}} + |\sum_{s_i \in f^{-1}(\bar{s})} w_{s_i} \sum_{\bar{s}' \in \bar{\mathcal{S}}'} \sum_{s_j \in f^{-1}(\bar{s}')} \left(P^{g_{s_i}^{-1}(\bar{o})}_{s_i,s_j} - P^{o_1}_{s_1,s_j} \right) \max_{\bar{o}'} Q(\bar{s}', \bar{o}')| \tag{22}$$

$$\leq 2|w_r|\epsilon_{\boldsymbol{\psi}} + \sum_{\bar{s}' \in \bar{\mathcal{S}}'} \epsilon_P \max_{\bar{o}'} Q(\bar{s}', \bar{o}') \leq 2|w_r|\epsilon_{\boldsymbol{\psi}} + |\bar{\mathcal{S}}|\epsilon_P \frac{|w_r| \max_{s,a} |\boldsymbol{\theta}(s,a)|}{1-\gamma} \tag{23}$$

$$\text{Denote } \kappa = |w_r| \left(2\epsilon_{\boldsymbol{\psi}} + \frac{|\bar{\mathcal{S}}|\epsilon_P \max_{s,a} |\boldsymbol{\theta}(s,a)|}{1-\gamma} \right) \tag{24}$$

$$(2) = |\underbrace{\sum_{s_i \in f^{-1}(\bar{s})} w_{s_i}}_{=1} \underbrace{\sum_{\bar{s}' \in \bar{\mathcal{S}}'} \sum_{s_j \in f^{-1}(\bar{s}')} P^{o_1}_{s_1,s_j}}_{(3) \leq \gamma} \left(\max_{\bar{o}'} Q(\bar{s}', \bar{o}') - \max_{o'_j} Q(s'_j, o'_j) \right) | \tag{25}$$

$$\leq \gamma \max_{s \in \mathcal{S}} |\max_{\bar{o}} Q(f(s), \bar{o}) - \max_{o} Q(s, o)| \quad \text{Let } o^* = \arg\max_{o} Q(s, o) \tag{26}$$

$$\leq \gamma \max_{s \in \mathcal{S}} |Q(f(s), g_s(o^*)) - Q(s, o^*)| \quad \text{which goes back to Equation (18).} \tag{27}$$

(3) is since the option lasts at least one step and terminates with probability 1:
$\sum_{\bar{s}' \in \bar{\mathcal{S}}'} \sum_{s_j \in f^{-1}(\bar{s}')} \sum_{k=1}^{\infty} P^{o_1}_{s_1,s_j,k} = 1$
Hence (3) $= \sum_{\bar{s}' \in \bar{\mathcal{S}}'} \sum_{s_j \in f^{-1}(\bar{s}')} \sum_{k=1}^{\infty} \gamma^k P^{o_1}_{s_1,s_j,k} \leq \sum_{\bar{s}' \in \bar{\mathcal{S}}'} \sum_{s_j \in f^{-1}(\bar{s}')} \gamma^{k=1} P^{o_1}_{s_1,s_j,k} = \gamma$.

$$|Q(\bar{s},\bar{o}) - Q(s_1,o_1)| \leq (1) + (2) \tag{28}$$

$$\leq \kappa + \gamma \underbrace{\max_{s \in \mathcal{S}} [|Q(f(s), g_s(o^*)) - Q(s,o^*)|]}_{(4)} \quad \text{, expand (4) again using Equation (18)} \tag{29}$$

$$\leq \kappa + \gamma \left(\kappa + \gamma \max_{s \in \mathcal{S}} [|Q(f(s), g_s(o^*)) - Q(s,o^*)|] \right) \tag{30}$$

$$\leq \kappa + \gamma\kappa + \gamma^2\kappa + \ldots \leq \frac{\kappa}{1-\gamma} \tag{31}$$

Step 2: The optimal option in the abstract MDP has a Q-value in the ground MDP that is nearly optimal, i.e.:

$$\forall s_1 \in \mathcal{S}, Q(s_1, o_1^*) - Q(s_1, g_{s_1}^{-1}(\bar{o}^*)) \leq \frac{2\kappa}{1-\gamma} \tag{32}$$

where $o_1^* = \arg\max_o Q(s_1, o)$ and $\bar{o}^* = \arg\max_{\bar{o}} Q(f(s_1), \bar{o})$.

From step 1, we have $|Q(\bar{s},\bar{o}) - Q(s_1,o_1)| \leq \frac{\kappa}{1-\gamma}$. Then, by definition of optimality,

$$Q(\bar{s}, g_{s_1}(o_1^*)) \leq Q(\bar{s}, \bar{o}^*) \implies Q(\bar{s}, g_{s_1}(o_1^*)) + \frac{\kappa}{1-\gamma} \leq Q(\bar{s}, \bar{o}^*) + \frac{\kappa}{1-\gamma} \tag{33}$$

$$\text{Therefore, by step 1 then by optimality: } Q(s_1, o_1^*) \leq Q(\bar{s}, g_{s_1}(o_1^*)) + \frac{\kappa}{1-\gamma} \leq Q(\bar{s}, \bar{o}^*) + \frac{\kappa}{1-\gamma} \tag{34}$$

$$\text{Again by step 1: } Q(\bar{s}, \bar{o}^*) \leq Q(s_1, g_{s_1}^{-1}(\bar{o}^*)) + \frac{\kappa}{1-\gamma} \tag{35}$$

$$\text{Therefore, } Q(s_1, o_1^*) \overset{Eq.(34)}{\leq} Q(\bar{s}, \bar{o}^*) + \frac{\kappa}{1-\gamma} \overset{Eq.(35)}{\leq} Q(s_1, g^{-1}(\bar{o}^*)) + \frac{2\kappa}{1-\gamma} \tag{36}$$

Step 3: The optimal abstract SMDP policy yields near optimal performance in the ground SMDP:

Denote $g^{-1}(\bar{\pi})$ as the ground SMDP policy implementing the abstract SMDP policy $\bar{\pi}$, i.e., at state s, the ground option corresponding to the abstract option $\bar{o}$ chosen by the abstract policy is $g_s^{-1}(\bar{o})$.

$$Q^{\pi^*}(s, o^*) - Q^{g^{-1}(\bar{\pi}^*)}(s, g_s^{-1}(\bar{o}^*)) \tag{37}$$

$$\leq \frac{2\kappa}{1-\gamma} + Q^{\pi^*}(s, g_s^{-1}(\bar{o}^*)) - Q^{g_s^{-1}(\bar{\pi}^*)}(s, g^{-1}(\bar{o}^*)) \text{ , by step 2} \tag{38}$$

$$= \frac{2\kappa}{1-\gamma} + [r_s^{g_s^{-1}(\bar{o}^*)} + \sum_{s' \in S} P_{s,s'}^{g_s^{-1}(\bar{o}^*)} Q^{\pi^*}(s', o^*_{s'})] - [r_s^{g_s^{-1}(\bar{o}^*)} + \sum_{s' \in S} P_{s,s'}^{g_s^{-1}(\bar{o}^*)} Q^{g^{-1}(\bar{\pi}^*)}(s', g^{-1}(o^*_{\bar{s}'}))] \tag{39}$$

$$= \frac{2\kappa}{1-\gamma} + \underbrace{\sum_{s' \in S} P_{s,s'}^{g_s^{-1}(\bar{o}^*)}}_{(1) \leq \gamma} [Q^{\pi^*}(s', o^*_{s'}) - Q^{g^{-1}(\bar{\pi}^*)}(s', g_{s'}^{-1}(o^*_{\bar{s}'}))] \tag{40}$$

$$\leq \frac{2\kappa}{1-\gamma} + \gamma \max_{s' \in S} [|Q^{\pi^*}(s', o^*_{s'}) - Q^{g^{-1}(\bar{\pi}^*)}(s', g_{s'}^{-1}(o^*_{\bar{s}'}))|] \tag{41}$$

$$\overset{\text{Eq.(37)}}{\leq} \frac{2\kappa}{1-\gamma} + \gamma \left(\frac{2\kappa}{1-\gamma} + \gamma \max_{s' \in S} [Q^{\pi^*}(s', g_{s'}^{-1}(o^*_{\bar{s}'})) - Q^{g_{s'}^{-1}(\bar{\pi}^*)}(s', g^{-1}(o^*_{\bar{s}'}))] \right) \tag{42}$$

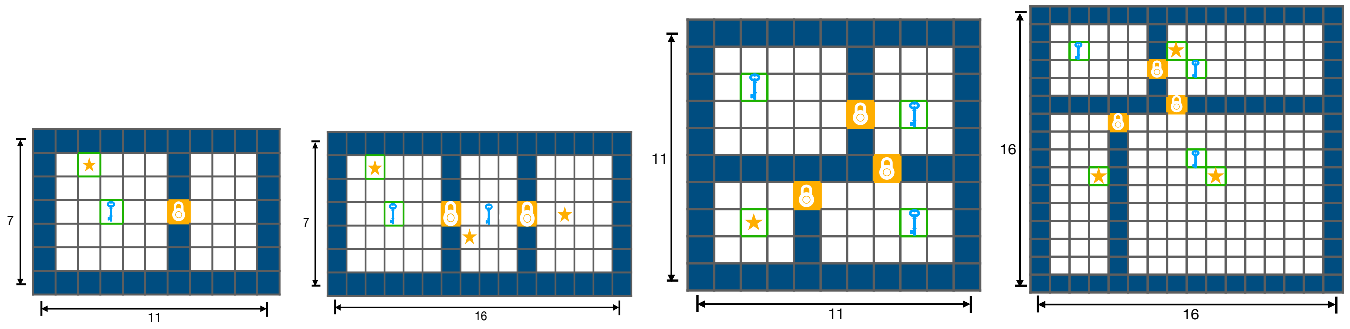
$$\leq \frac{2\kappa}{1-\gamma} + \gamma \frac{2\kappa}{1-\gamma} + \gamma^2 \frac{2\kappa}{1-\gamma} \ldots \tag{43}$$

$$\leq \frac{2\kappa}{(1-\gamma)^2} \tag{44}$$

where (1) is because the option terminates with probability 1 and takes at least 1 step. □

A.6 Additional Experiment Details

Additional details on the experimental settings



(a) 2 Rooms, 157 states (b) 3 Rooms, 487 states (c) 4 Rooms (small), 348 states (d) 4 Rooms (large), 1463 states

Figure 4: Object-room layouts used in our experiments. In each object room instance, blue grids are the walls, orange grids are doors, and there are two types of objects: stars and keys, which can be picked-up by the agent. The agent can pick up a key and use it to open a door.

Object-Rooms: N rooms are connected by doors with keys and stars inside. There are 6 actions, i.e., $\mathcal{A} = \{$Up, Down, Left, Right, Pick up, Open$\}$. The agent can pick up the keys and stars, and use a key to open a door next to it. The agent starts from the (upper) left room.

Settings for Table 1: The source room where the agent demonstrates and encodes the abstract options is the 2 Room variant. The 2-4 target rooms setting used for grounding the options are the 2 Rooms in Figure 4 (a), 3 Rooms in Figure 4 (b), and 4 Rooms (large) in Figure 4 (d) variants.

Settings for Figure 1 and Figure 11: The abstract ψ-SMDP is generated using the setting 4 Room (small) in Figure 4 (c), where the first 3 rooms each contain a key and a star is in the final room. For Figure 11, the task specifications are as follows: We refer to transfer as the task transfer, i.e., change of reward function. The total reward is discounted and normalized by the maximum reward achieved.

- dense reward (no transfer): the agent receives a reward for each key picked up, door opened, and star picked up, i.e., the reward vector $w_r = [1, 1, 1]$ over the features (key, open door, star).
- sparse reward (no transfer): the agent receives a reward for each door opened, i.e., the reward vector $w_r = [0, 1, 0]$.
- transfer (w. overlap): overlap refers to the overlap between reward function in the source and target task. In the source task, the agent receives a reward for each key picked up, and each door opened, i.e., the reward vector $w_r = [1, 1, 0]$. In the target task, the agent receives a reward for each door opened and star picked up, i.e., the reward vector $w_r = [0, 1, 1]$.
- transfer (w.o. overlap): In the source task, the agent receives a reward for each star picked up, i.e., the reward vector $w_r = [0, 0, 1]$. In the target task, the agent receives a reward for each key picked up, i.e., the reward vector $w_r = [1, 0, 0]$.

Settings for Figure 3 and Figure 10: The abstract ψ-SMDP is generated using the 3 Room setting in Figure 4 (b), where the first 2 rooms each contains a key and a star, and the final room contains a star. For figure 3, the task specifications is the same as the above description for Figure 11.

Additional Experiment Results:

In this section, we present additional experimental results.

Classic 4-Rooms. Figure 5 shows the ground option policy learned by *IRL-batch* on the 4-Room domain. The option is learned by solving 1 linear program by *IRL-batch*, i.e., the state-action visitation frequencies returned by the IRL program corresponds to an optimal policy for all starting states. Please refer to the figure for details of the option.

Minecraft Door-Rooms Experiments: As introduced in Section 5, to test our batched option grounding algorithm (Algorithm 3) on new environments with unknown transition dynamics, we built two settings in the Malmo Minecraft environment: *Bake-Rooms* and *Door-Rooms*. The results on Bake-Rooms can be found in Figure 2 in the main text. Here, we present the results on the Door-Rooms setting shown in Figure 6.

Training and Results: We compare our algorithm with the following two baselines: eigenoptions [Machado *et al.*, 2018] and random walk. For this experiment, each agent runs for 20 iterations, with 200 steps per iteration as follows: In the first iteration, all agents execute randomly chosen actions. After each iteration, the agents construct an MDP graph based on collected transitions from all prior iterations. The eigenoption agent computes $k = 1$ eigenoption of the second smallest eigenvalue (Fiedler vector) using the normalized graph Laplacian, while our algorithm grounds the $k = 1$ abstract option: *open*

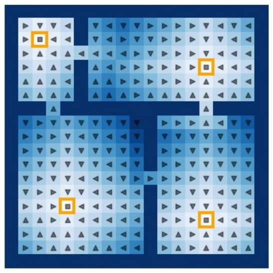
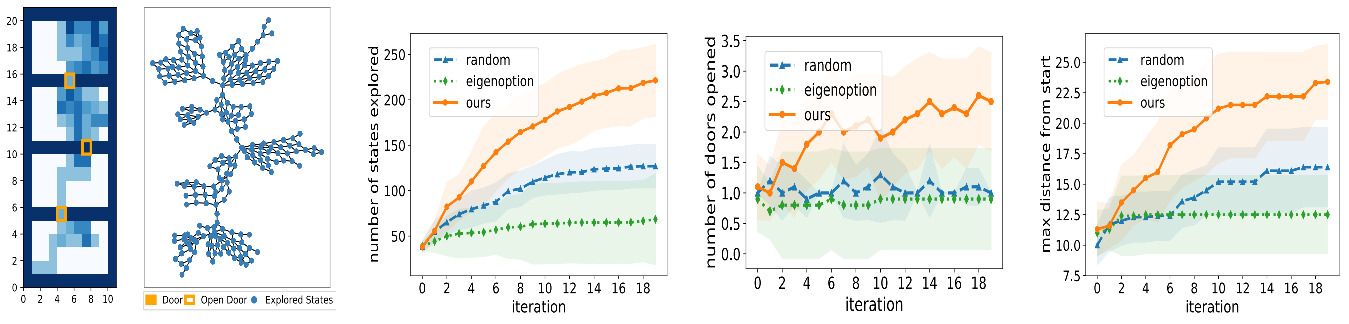


Figure 5: Ground options learned in the 4-Room domain via Algorithm 3 (presented in the Appendix). The features correspond to indicators of the room centers (marked in the orange square). Hence, the ground option should move to any one of the four room centres. The option policy accurately takes the agent from each starting state to a nearby room centre and then terminates.



(a) State visitation (b) number of states explored (c) number of doors opened (d) distance from start

Figure 6: (Minecraft Door-Rooms) Exploring and grounding options in unknown environments. 4 rooms are connected by doors, the agent starts from the bottom, opens doors and enters the other rooms. The random agent takes random actions. Our agent starts with an iteration of random walk, then after each iteration, it computes the constructed MDP and fits the "go to and open door" option, (while the eigenoptions agent finds an eigenoption), and uses the options for exploration in the next iteration. (a) shows the state visitation frequency in iteration 20 of our agent (Algorithm 3), and the MDP constructed throughout the 20 iterations. (b)-(d) compares our agent with the baselines on the average number of the explored states, doors opened and max distance traversed. The results are averaged over 10 seeds and shaded regions show the standard deviation across seeds. Our agent using abstract successor options explores on average twice as many states as the baselines, as well as quickly learns to open doors and navigating to new rooms.

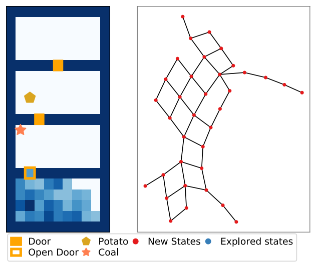
door and go to door. In the next iteration, the agents perform random walks with both the primitive actions and the acquired options, update the MDP graphs, compute new options, . . .

Figure 6 shows our obtained results. Figure 6(a) shows the state visitation frequencies of our algorithm in the 20th iteration and the constructed MDP graph. The agent starts from the bottom room (R1), and learns to navigate towards the door, open the door and enter the next rooms. Figures 6(b)-(d) compare the agents in terms of the total number of states explored, number of doors opened and the maximum distance from the starting location. Note that the door layouts are different from the Bake-Rooms environment. Our agent explores on average more than twice as many states as the two baselines, quickly learns to open the doors and navigate to new rooms, while the baselines on average only learn to open the first door and mostly stay in the starting room.

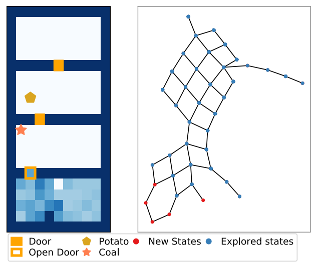
More details on Minecraft Bake-Rooms: Besides Figure 2, we now present more details of our option grounding algorithm in environments with unknown transition dynamics in the Minecraft Bake-Rooms experiment. Figures 7, 8 and 9 show the state visitation frequencies and MDP graph constructed over 20 iterations by our algorithm. The agent starts from the bottom room R1, a coal dropper is in R2, a potato dropper is in R3. The rooms are connected by doors which can be opened by the agent. For clarity of presentation, we show the undirected graph constructed. Blue nodes denote explored states and red nodes denote new states explored in the respective iteration.

The shown figures demonstrate that our agent learns to open the door, and open the door and enter R2 to collect coal in iteration 2, while the eigenoptions agent learns to collect coal in iteration 18, and the random agent collects a coal block in iteration 14. Our agent learns to collect potato in R3 in iteration 3, while the eigenoptions agent learns this in iteration 19, and the random agent has not reached R3 within 20 iterations.

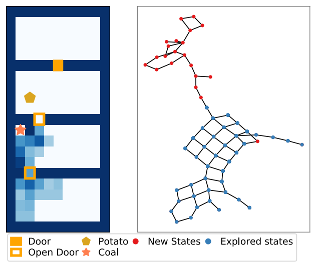
Additional results on abstraction: Figure 10 shows the abstract MDPs in the Object-Rooms with $N = 3$ rooms. Figure 10(a) is the abstract ψ-SMDP model induced by our approximate successor homomorphism, the colors of the nodes match their corresponding ground states in the ground MDP shown in Figure 10(b). The edges with temporal semantics correspond to abstract successor options and the option transition dynamics. To avoid disconnect graphs, we can augment the abstract successor options with shortest path options, which connect ground states of disconnected abstract states to their nearest abstract



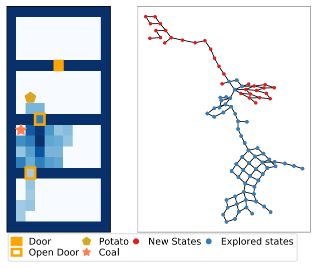
(a) iteration 0



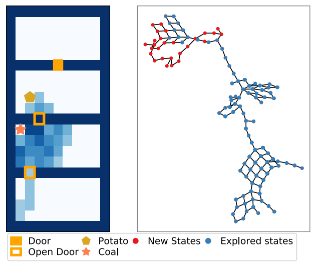
(b) iteration 1



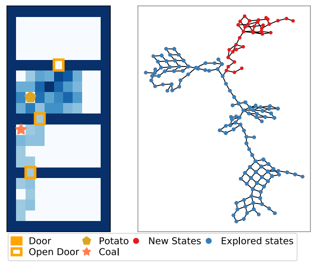
(c) iteration 2



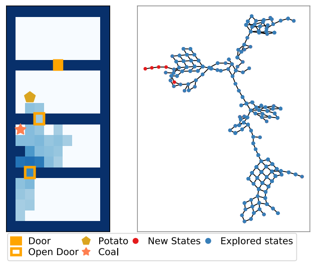
(d) iteration 3



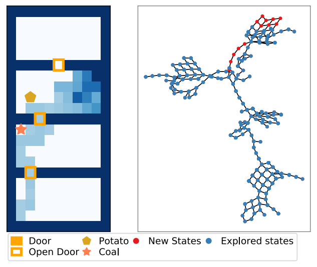
(e) iteration 4



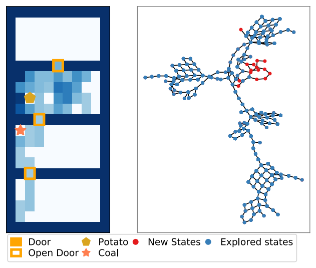
(f) iteration 5



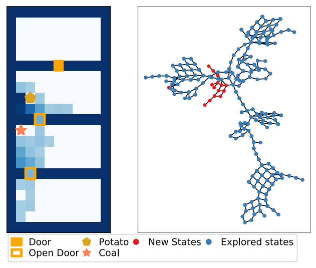
(g) iteration 6



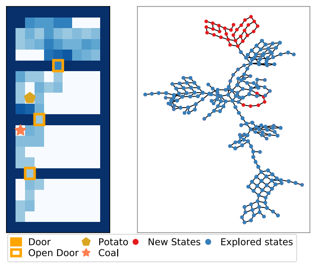
(h) iteration 7



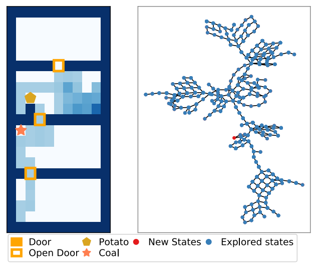
(i) iteration 8



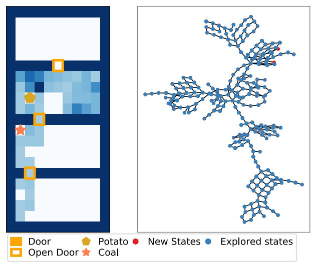
(j) iteration 9



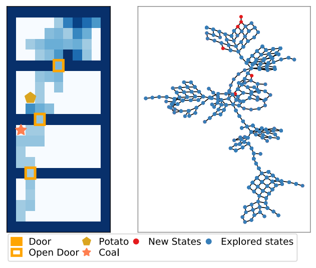
(k) iteration 10



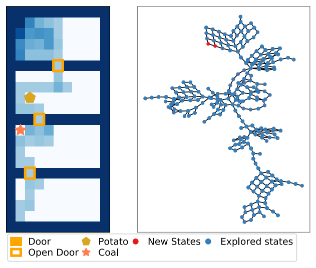
(l) iteration 11



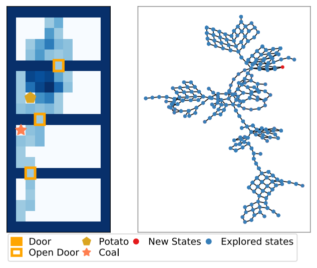
(m) iteration 12



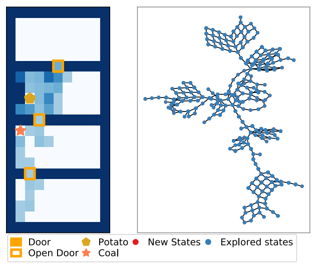
(n) iteration 13



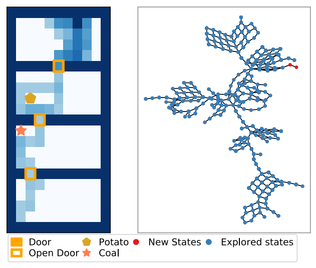
(o) iteration 14



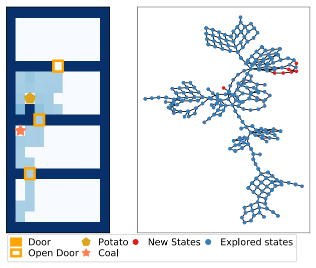
(p) iteration 15



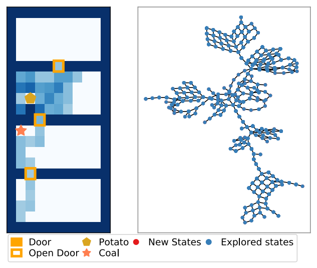
(q) iteration 16



(r) iteration 17

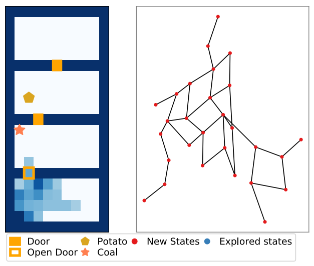


(s) iteration 18

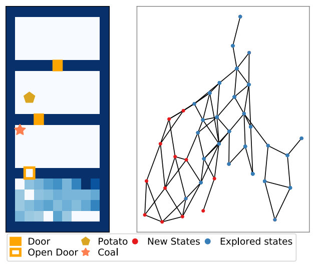


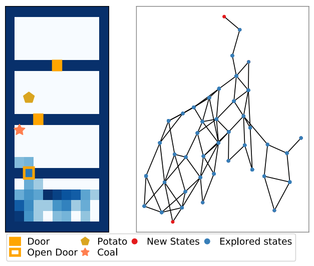
(t) iteration 19

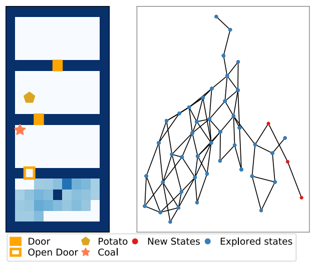
Figure 7: (Our Agent - Algorithm 3) State visitation frequencies and constructed graph per iteration in Minecraft Bake-Rooms

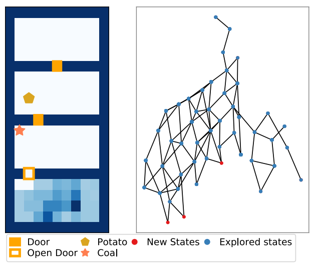


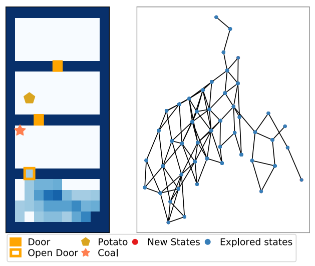
(a) iteration 0

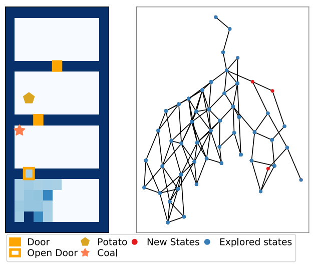

(b) iteration 1

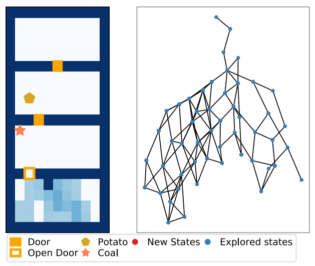

(c) iteration 2

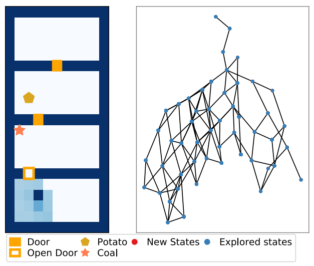

(d) iteration 3

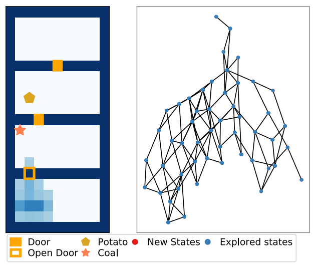

(e) iteration 4

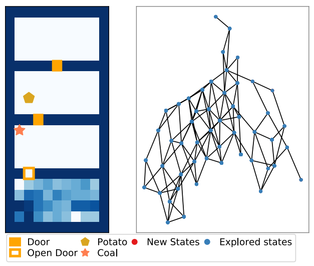

(f) iteration 5

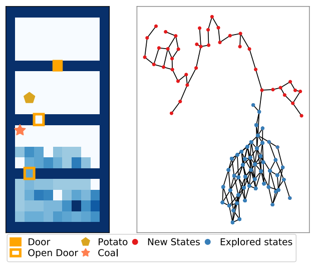

(g) iteration 6

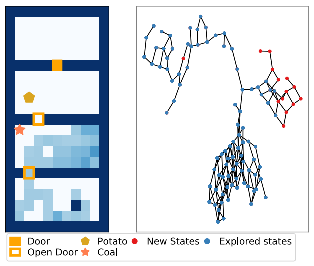

(h) iteration 7

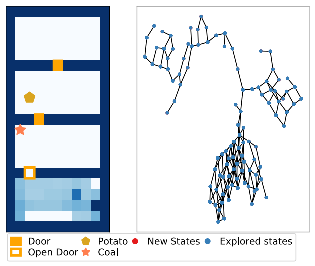

(i) iteration 8

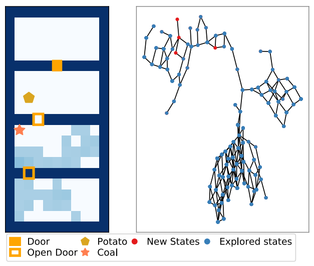

(j) iteration 9

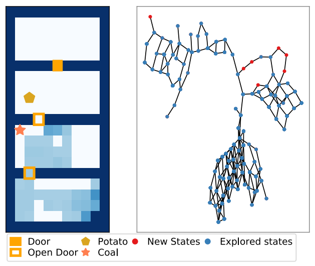

(k) iteration 10

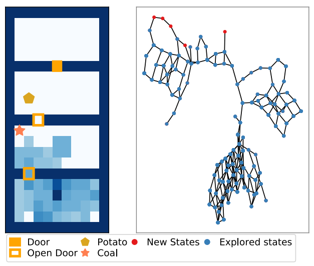

(l) iteration 11

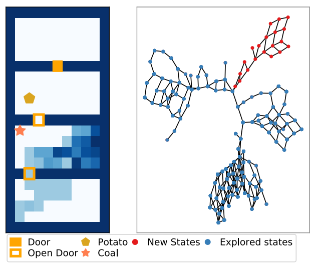

(m) iteration 12

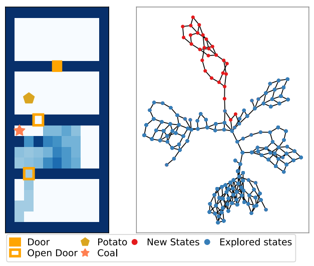

(n) iteration 13


(o) iteration 14


(p) iteration 15


(q) iteration 16


(r) iteration 17


(s) iteration 18

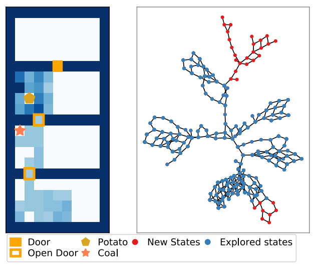
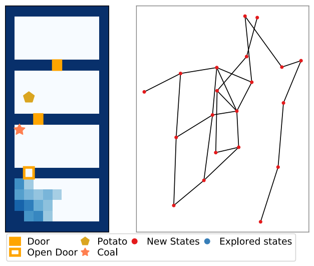
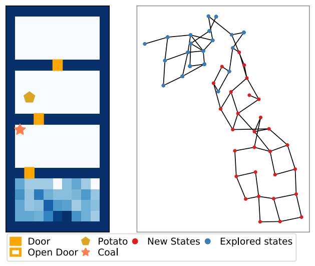

(t) iteration 19

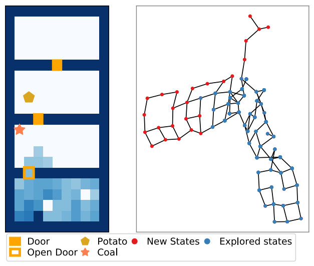
Figure 8: (Eigenoptions Agent) State visitation frequencies and constructed graph per iteration in Minecraft Bake-Rooms



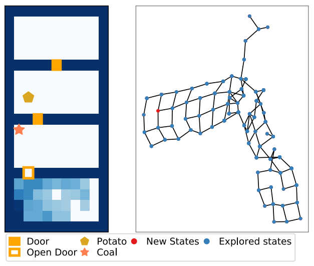
(a) iteration 0



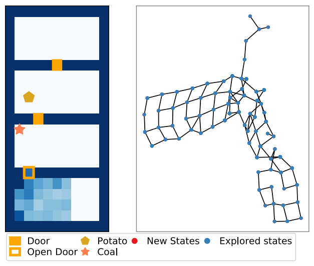
(b) iteration 1



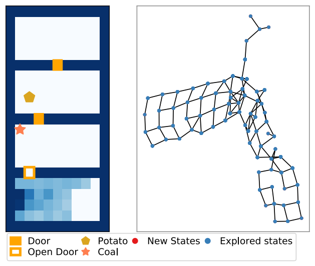
(c) iteration 2



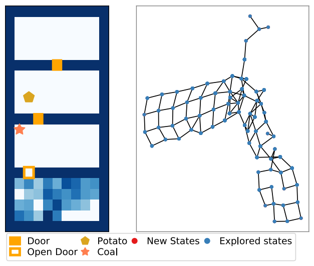
(d) iteration 3



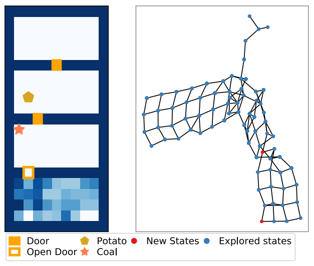
(e) iteration 4



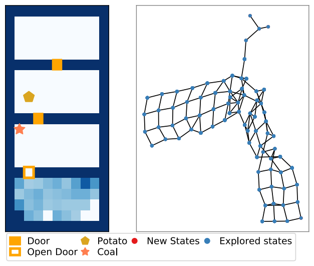
(f) iteration 5



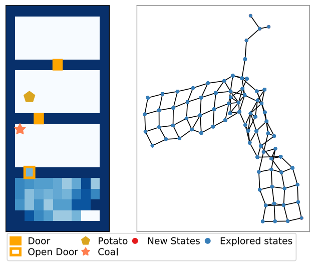
(g) iteration 6



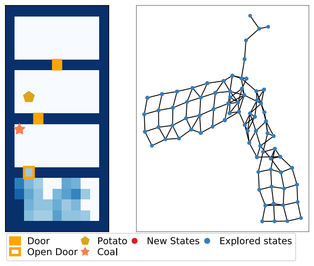
(h) iteration 7



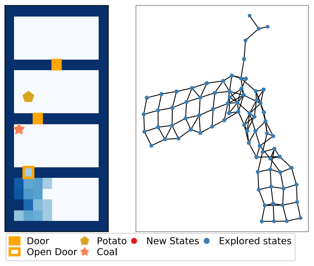
(i) iteration 8



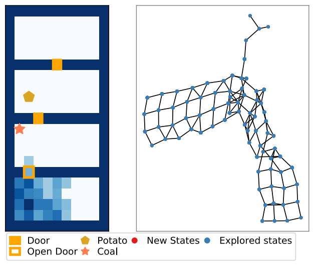
(j) iteration 9



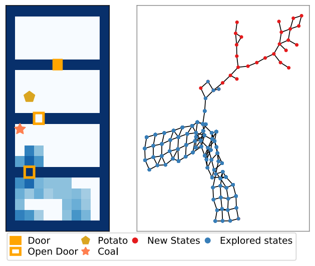
(k) iteration 10



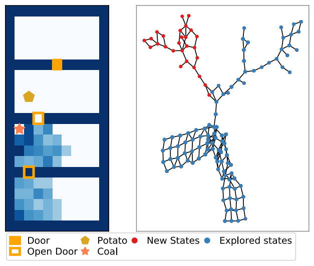
(l) iteration 11



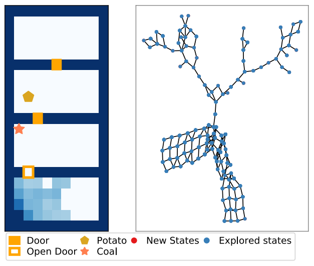
(m) iteration 12



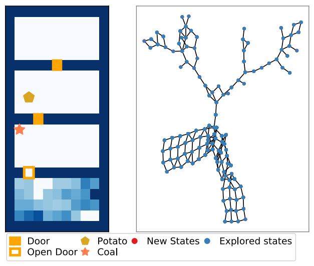
(n) iteration 13



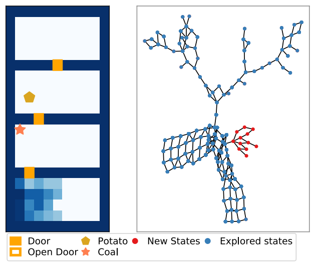
(o) iteration 14



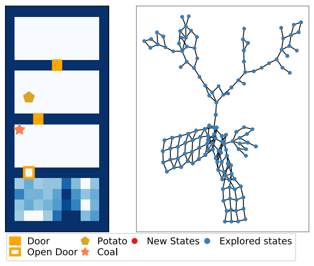
(p) iteration 15



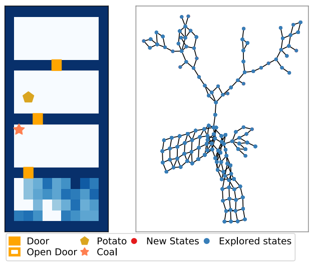
(q) iteration 16



(r) iteration 17

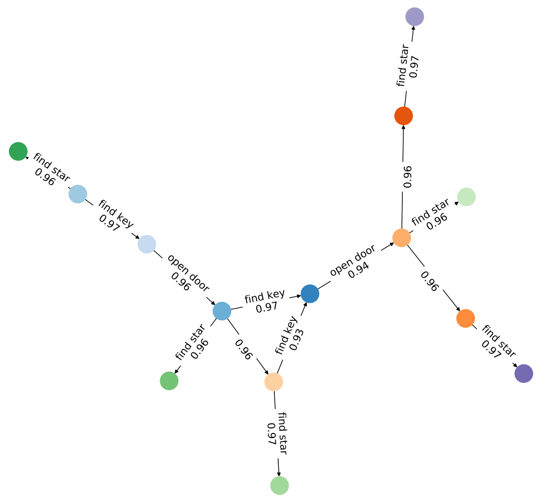


(s) iteration 18

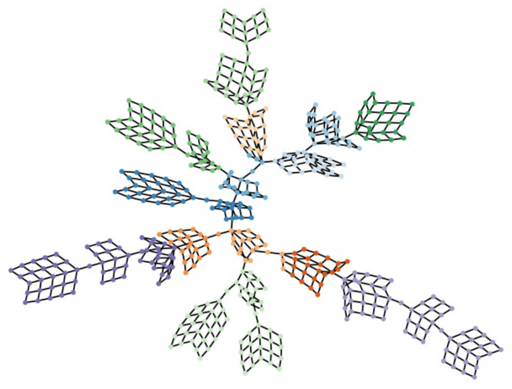


(t) iteration 19

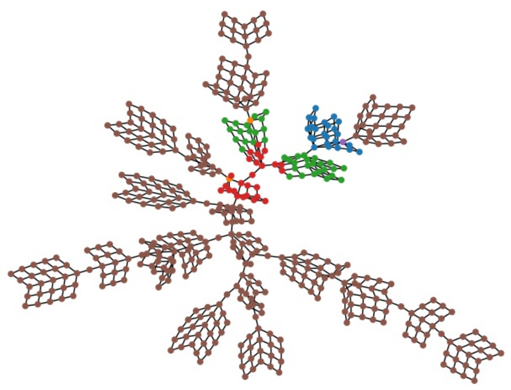
Figure 9: (Random Agent) State visitation frequencies and constructed graph per iteration in Minecraft Bake-Rooms



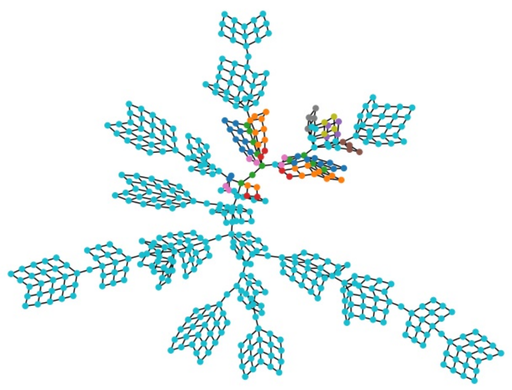
(a) Abstract ψ-SMDP



(b) Ground MDP



(c) Q-Abstraction w. All Actions

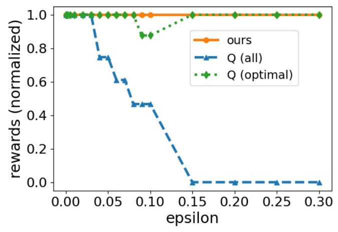


(d) Q-Abstraction w. Optimal Action

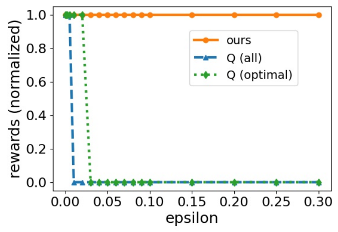
Figure 10: Abstract MDP with different abstraction schemes. (a) is the abstract MDP induced by our proposed successor homomorphism and (b) shows how the ground states are mapped to the abstract states. Node colors correspond to the abstract states in (a). (c) and (d) are abstraction induced by the Q^*-irrelevance and a^*-irrelevance abstraction schemes.

states. Figure 10(c) and (d) show the abstract MDPs induced by the Q^*-irrelevance (Q-all) and a^*-irrelevance (Q-optimal) abstraction methods, for the task *find key*. The distance threshold $\epsilon = 0.1$.

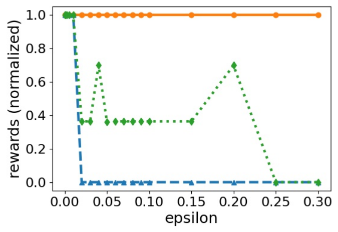
Figure 11 shows the results of using the abstract MDP for planning in the Object-Rooms with $N = 4$ rooms. Please refer to Section A.6 for a detailed description of the settings. Our successor homomorphism model performs well across all tested settings with few abstract states (number of clusters). Since successor homomorphism does not depend on rewards, the abstract model can transfer across tasks (with varying reward functions), and is robust under sparse rewards settings. Whereas abstraction schemes based on the reward function perform worse when the source task for performing abstraction is different from the target task where the abstract MDP is used for planning.



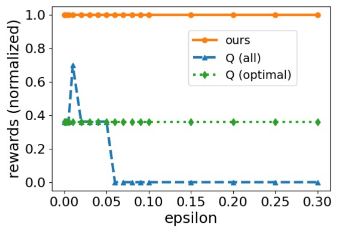
(a) no transfer (dense reward)



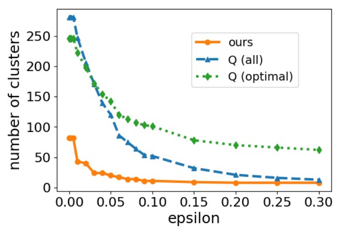
(b) no transfer (sparse reward)



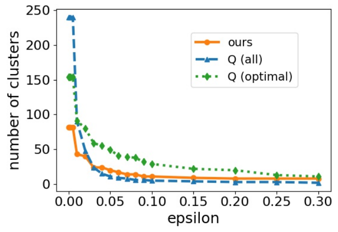
(c) transfer (w. overlap)



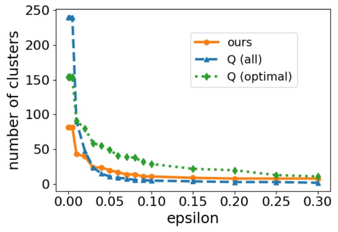
(d) transfer (w.o. overlap)



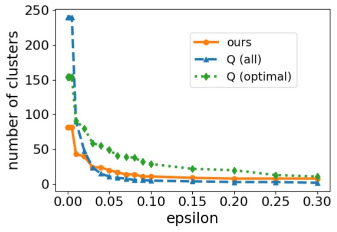
(e) no transfer (dense reward)



(f) no transfer (sparse reward)



(g) transfer (w. overlap)



(h) transfer (w.o. overlap)

Figure 11: Performance of planning with the abstract MDPs. The upper row shows the total rewards (normalized by the maximum possible total rewards) obtained, and the lower row shows the corresponding number of abstract states of the abstract MDP. The x-axes are the distance thresholds ϵ. transfer refers to task transfer (i.e., different reward function)