Option Transfer and SMDP Abstraction with Successor Features

Dongge Han[1] , **Sebastian Tschiatschek**[2]
[1]University of Oxford, Department of Computer Science, Oxford, United Kingdom
[2]University of Vienna, Faculty of Computer Science, Vienna, Austria
dongge.han.oxford@gmail.com, sebastian.tschiatschek@univie.ac.at

Abstract

Abstraction plays an important role in the generalisation of knowledge and skills and is key to sample efficient learning. In this work, we study joint temporal and state abstraction in reinforcement learning, where temporally-extended actions in the form of options induce temporal abstractions, while aggregation of similar states with respect to abstract options induces state abstractions. Many existing abstraction schemes ignore the interplay of state and temporal abstraction. Consequently, the considered option policies often cannot be directly transferred to new environments due to changes in the state space and transition dynamics. To address this issue, we propose a novel abstraction scheme building on successor features. This includes an algorithm for transferring abstract options across different environments and a state abstraction mechanism that allows us to perform efficient planning with the transferred options.

1 Introduction

Reinforcement learning (RL) has recently shown many remarkable successes [Mnih *et al.*, 2015; Silver *et al.*, 2017]. For efficient planning and learning in complex, long-horizon RL problems, it is often useful to allow RL agents to form different types and levels of abstraction [Sutton *et al.*, 1999; Li *et al.*, 2006]. On the one hand, temporal abstraction allows the efficient decomposition of complex problems into sub-problems. For example, the options framework [Sutton *et al.*, 1999] enables agents to execute options, i.e., temporally-extended actions (e.g., *pick up the key*), representing a sequence of primitive actions (e.g., *move forward*). On the other hand, state abstraction [Li *et al.*, 2006; Abel *et al.*, 2016] is a common approach for forming abstract environment representations through aggregating similar states into abstract states allowing for efficient planning.

In this paper we aim to combine the benefits of temporal and state abstractions for enabling transfer and reuse of options across environments. We build on the insight that environments with different dynamics and state representations may share a similar abstract representation and address the following question: How can we transfer learned options to new environments and reuse them for efficient planning and exploration? This is a challenging question because options are typically described by policies that are not transferable across different environments due to different state representations and transition dynamics. This issue is underlined by the fact that abstract options (e.g., *open a door*) can often correspond to different policies in different environments. To enable option transfer and reuse, we propose an abstract option representation that can be shared across environments. We then present algorithms that ground abstract options in new environments. Finally, we define a state abstraction mechanism that reuses the transferred options for efficient planning.

Concretely, to find a transferable abstract option representation, we propose abstract successor options, which represent options through their successor features (SF) [Dayan, 1993; Barreto *et al.*, 2017]. An SF of an option is the vector of cumulative feature expectations of executing the option and thus can act as an abstract sketch of the goals the option should achieve. By defining shared features among the environments, the abstract successor options can be transferred across environments. Abstract options then need to be grounded in the unseen environments. One way, therefore, is to find a ground option policy that maximises a reward defined as a weighted sum over the SF [Barreto *et al.*, 2019]. However, careful hand-crafting on the reward weights is needed to produce an intended option's behaviour due to interference between the different feature components. Alternatively, we formulate option grounding as a feature-matching problem where the feature expectations of the learned option should match the abstract option. Under this formulation, we present two algorithms (IRL-NAIVE and IRL-BATCH) which perform option grounding through inverse reinforcement learning (IRL). Our algorithms allow option transfer and can be used for exploration in environments with unknown dynamics. Finally, to enable efficient planning and reuse of the abstract options, we propose successor homomorphism, a state abstraction mechanism that produces abstract environment models by incorporating temporal and state abstraction via abstract successor options. We demonstrate that the abstract models enable efficient planning and yield near-optimal performance in unseen environments.

The appendix to this paper is available at https://hdg94.github.io/assets/img/abstractions_appendix.pdf.

2 Background

Markov Decision Processes (MDP). We model an agent's decision making process as an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{S}$ is a set of states the environment the agent is interacting with can be in, $\mathcal{A}$ is a set of actions the agent can take, $P\colon \mathcal{S} \times \mathcal{A} \to [0,1]^{|\mathcal{S}|}$ is a transition kernel describing transitions between states of the MDP upon taking actions, $r\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, and γ is a discount factor. An agent's behavior can be characterized by a policy $\pi\colon \mathcal{S} \to [0,1]^{|\mathcal{A}|}$, i.e., $\pi(a|s)$ is the probability of taking action a in state s.[1]

Successor Features (SF). [Barreto *et al.*, 2017] Given features $\boldsymbol{\theta}\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ associated with each state-action pair, the SF $\boldsymbol{\psi}^{\pi}_{s_0}$ is the expected discounted sum of features of state-action pairs encountered when following policy π starting from s_0, i.e.,

$$\boldsymbol{\psi}^{\pi}_{s_0} = \mathbb{E}_{\mathcal{M},\pi}[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\theta}(S_t, A_t) \mid S_0 = s_0]. \tag{1}$$

Options and Semi-Markov Decision Processes (SMDPs). Temporally extended actions, i.e., sequences of actions over multiple time steps, are often represented as options $o = \langle I^o, \pi^o, \beta^o \rangle$, where $I^o \subseteq \mathcal{S}$ is a set of states that an option can be initiated at (initiation set), $\pi^o\colon \mathcal{S} \to [0,1]^{|\mathcal{A}|}$ is the option's policy, and $\beta^o\colon \mathcal{S} \to [0,1]$ is the termination condition, i.e., the probability that o terminates in state $s \in \mathcal{S}$. The transition dynamics and rewards induced by an option o are

$$P^o_{s,s'} = \sum_{k=1}^{\infty} P(s, o, s', k)\gamma^k,$$
$$r^o_s = \mathbb{E}[r_t + \ldots + \gamma^k r_{t+k} \mid \mathcal{E}(o, s, t)],$$

where $P(s, o, s', k)$ is the probability of transiting from s to s' in k steps when following option policy π^o and terminating in s', $\mathcal{E}(o, s, t)$ is the event that option o is initiated at time t in state s, and $t + k$ is the random time at which o terminates.

An SMDP [Sutton *et al.*, 1999] is an MDP with a set of options, i.e., $\mathcal{M} = \langle \mathcal{S}, \mathcal{O}, P, r, \gamma \rangle$, where $P\colon \mathcal{S} \times \mathcal{O} \to [0,1]^{|S|}$ are the options' transition dynamics, and $r\colon \mathcal{S} \times \mathcal{O} \to \mathbb{R}$ are the options' rewards. A family of variable-reward SMDPs [Mehta *et al.*, 2008] (denoted as $\boldsymbol{\psi}$-SMDP) is defined as $\mathcal{M} = \langle \mathcal{S}, \mathcal{O}, P, \boldsymbol{\psi}, \gamma \rangle$, where $\boldsymbol{\psi}^o_s$ is the SF of option o starting at state s. A $\boldsymbol{\psi}$-SMDP induces an SMDP if the reward is linear in the features, i.e., $r^o_s = w_r^T \boldsymbol{\psi}^o_s$.

Inverse Reinforcement Learning (IRL). IRL is an approach to learning from demonstrations [Abbeel and Ng, 2004] with unknown rewards. A common assumption is that the rewards are linear in some features, i.e., $r^a_s = w_r^T \boldsymbol{\theta}(s, a)$, where $w_r \in \mathbb{R}^d$ is a real-valued weight vector specifying the reward of observing the different features. Based on this assumption, [Abbeel and Ng, 2004] observed that for a learner's policy to perform as well as the expert's, it suffices that their feature expectations match. Therefore, IRL has been widely posed as a feature-matching problem [Abbeel and Ng, 2004; Syed *et al.*, 2008; Ho and Ermon, 2016], where the learner tries to match the expert's feature expectation.

[1] We only consider stationary policies in this paper.

3 Options as Successor Features

In this section, we first present abstract successor options, an abstract representation of options through successor features that can be shared across multiple MDPs $\{\mathcal{M}_i\}_{i=1}^k$. Therefore, we assume a *feature function* which maps state-action pairs for each MDP to a shared feature space, i.e., $\boldsymbol{\theta}_{\mathcal{M}_i}\colon \mathcal{S}_i \times \mathcal{A}_i \to \mathbb{R}^d$. Such features are commonly adopted by prior works [Barreto *et al.*, 2017; Syed *et al.*, 2008; Abbeel and Ng, 2004], and can be often obtained through feature extractors such as an object detector [Girshick, 2015]. We then present two algorithms that enable us to transfer abstract successor options to new environments with and without known dynamics by grounding the abstract options.

3.1 Abstract Successor Options

Abstract successor options are vectors in $\mathbb{R}^d$, representing cumulative feature expectations that the corresponding ground options should realise.

Definition 3.1 (Abstract Successor Options, Ground Options)**.** *Let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$ be an MDP and $\boldsymbol{\theta}_{\mathcal{M}}\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ be the feature function. An* abstract successor option *is a vector $\boldsymbol{\psi}^{\bar{o}} \in \mathbb{R}^d$. For brevity, we will often denote $\boldsymbol{\psi}^{\bar{o}} \in \mathbb{R}^d$ as $\bar{o}$. Let $g_s\colon \mathcal{O} \to \mathbb{R}^d$ denote a state-dependent mapping such that $o \mapsto \boldsymbol{\psi}^o_s = \mathbb{E}[\sum_{\kappa=0}^{k} \gamma^\kappa \boldsymbol{\theta}_{\mathcal{M}}(S_{t+\kappa}, A_{t+\kappa}) | \mathcal{E}(o, s, t)]$. The options $o \in g_s^{-1}(\bar{o})$ are referred to as* ground options *of $\bar{o}$ in state s.*

The initiation set of an abstract successor option $\bar{o}$ can now be naturally defined as $I^{\bar{o}} := \{s \in \mathcal{S} \mid \exists o\colon g_s(o) = \boldsymbol{\psi}^{\bar{o}}\}$. In the following section, we present an algorithm for grounding abstract successor options in an MDP $\mathcal{M}$ using IRL.

3.2 Grounding Abstract Successor Options

The challenge of grounding abstract successor options, i.e., finding ground policies which realize an abstract successor option's feature vector, corresponds to a feature-matching problem. This problem, although with a different aim, has been extensively studied in IRL, where a learner aims to match the expected discounted cumulative feature expectations of an expert [Syed *et al.*, 2008; Abbeel and Ng, 2004]. Inspired by IRL, we present two algorithms for abstract option grounding: IRL-NAIVE and IRL-BATCH, where the latter algorithm enjoys better runtime performance while achieving similar grounding performance (cf. experiments).

IRL-NAIVE (Algorithm 1). is a naive algorithm for grounding an abstract option $\bar{o}$. The algorithm uses feature matching based IRL to find for each possible starting state s_{start} an option realizing the feature vector $\boldsymbol{\psi}^{\bar{o}}$. First, we create an augmented MDP which enables termination of options. To do this, we augment the action space with a terminate action a_{T} and append a null state s_{null} to the state space. Specifically, a_{T} will take the agent from any "regular" state to the null state. Taking any action in the null state will lead to itself, and yield a zero feature, i.e., $\boldsymbol{\theta}(s_{\text{null}}, \cdot) = \mathbf{0}$. With the augmented MDP, we can compute an option policy via IRL. Several IRL formulations exist, and we adopt the linear programming (LP) approach by [Syed *et al.*, 2008] (Algorithm 2

Algorithm 1 Option Grounding (IRL-NAIVE)

Input: $\mathcal{M}{=}\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, abstract option $\bar{o}$, $\epsilon_{\text{thresh.}}$
Output: initiation set $I^{\bar{o}}$, dict. of ground option policies $\Pi^{\bar{o}}$, dict. of termination probabilities $\Xi^{\bar{o}}$

// Construct augmented MDP
$\mathcal{S}' \leftarrow \mathcal{S} \cup \{s_{\text{null}}\}, \mathcal{A}' \leftarrow \mathcal{A} \cup \{a_{\text{T}}\}, P' \leftarrow P$
$\forall s \in \mathcal{S}' \colon P'(s_{\text{null}}|s, a_{\text{T}}) = 1$
$\forall a \in \mathcal{A}' \colon P'(s_{\text{null}} \mid s_{\text{null}}, a) = 1$
// Find ground options
for all $s_{\text{start}} \in \mathcal{S}$, **do**
  $\epsilon, \pi^o_{s_{\text{start}}} \leftarrow \text{IRL}(\mathcal{S}', \mathcal{A}', P', \gamma, s_{\text{start}}, \boldsymbol{\psi}^{\bar{o}})$
  if $\epsilon \leq \epsilon_{\text{thresh.}}$ **then**
    $I^{\bar{o}} \leftarrow I^{\bar{o}} \cup \{s_{\text{start}}\}$
    $\Pi^{\bar{o}}(s_{\text{start}}) = \pi^o_{s_{\text{start}}}$
    $\Xi^{\bar{o}}(s_{\text{start}}) = \beta^o$, where $\beta^o(s) = \pi^o_{s_{\text{start}}}(a_{\text{T}}|s)$
  end if
end for
return $I^{\bar{o}}, \Pi^{\bar{o}}, \Xi^{\bar{o}}$

in Appendix A.1). Specifically, the LP approach finds discounted visitation frequencies for all state-action pairs such that the induced feature expectations match the abstract successor option. Then the option policy π^o (including option termination) can be deduced from the state-action visitation frequencies. Finally, if the discrepancy ϵ of the successor feature of the learned option and the abstract option is below a specified threshold $\epsilon_{\text{thresh.}}$, the start state s_{start} will be added to the initiation set.

IRL-BATCH (Algorithm 3, Appendix A.1). IRL-NAIVE needs to solve an IRL problem for each state in $\mathcal{S}$ which is computationally demanding. To improve the efficiency of abstract option grounding, we propose IRL-BATCH, a learning algorithm that performs IRL for starting states in batches. The main challenge is that the state-action visitation frequencies found by the LP for matching the abstract option's feature vector may be a mixture of different option policies from different starting states. To approach this issue, IRL-BATCH uses a recursive procedure with a batched IRL component (cf. Algorithm 4 in Appendix A.1) which effectively regularises the learned option policy. This algorithm can significantly reduce the number of IRL problems that need to be solved while preserving near-optimal performance, cf. Table 1.

3.3 Transfer Via Abstract Successor Options

Since features are assumed to be shared across MDPs, an existing option from a source environment can be transferred to a target environment as follows: first find the abstract successor option $\bar{o}$ by computing the ground option's successor feature in the source environment, and then ground $\bar{o}$ in the target environment using the IRL-NAIVE or IRL-BATCH algorithm, i.e., obtain $o' \in g_s^{-1}(\bar{o})$. When the transition dynamics of the target environment are unknown, exploration is needed before option grounding to construct the (approximate) MDP transition graph, e.g., through a random walk. However, random walks can often get trapped in local communities [Pons and Latapy, 2005] and thus can be inefficient for exploration. On the other hand, unlike solving a long-horizon task with sparse rewards, options are often relatively localised, and hence can be readily found with partially constructed transition graphs. This enables simultaneous exploration and option grounding: given a set of abstract options $\bar{\mathcal{O}}$, we start with an iteration of a random walk which constructs an approximate MDP $\tilde{\mathcal{M}}_0$. In each subsequent iteration k, we ground the abstract options using $\tilde{\mathcal{M}}_{k-1}$, and update the MDP $\tilde{\mathcal{M}}_k$ by exploring with a random walk using both primitive actions and the computed ground options. We show empirically that, using our approach, ground options can be learned quickly and significantly improve the exploration efficiency, cf. Section 5.1.

4 Abstraction with Successor Homomorphism

Parallel to temporal abstraction, state abstraction [Dean and Givan, 1997; Li *et al.*, 2006; Abel *et al.*, 2016] aims to form abstract MDPs in order to reduce the complexity of planning while ensuring close-to-optimal performance compared with using the original MDP. Typically, an abstract MDP is formed by aggregating similar states into an abstract state. However, most prior methods do not take into account temporal abstraction thus often requiring strict symmetry relations among the aggregated states. To overcome this limitation, we propose a state abstraction mechanism based on our abstract successor options and ψ-SMDPs (cf. Sec. 2), which naturally models the decision process with the abstract options. In particular, we propose *successor homomorphisms*, which define criteria for state aggregation and induced *abstract ψ-SMDPs*. We show that planning with these abstract ψ-SMDPs yields near-optimal performance. Moreover, the abstract ψ-SMDPs can inherit meaningful semantics from the options, cf. Fig. 1.

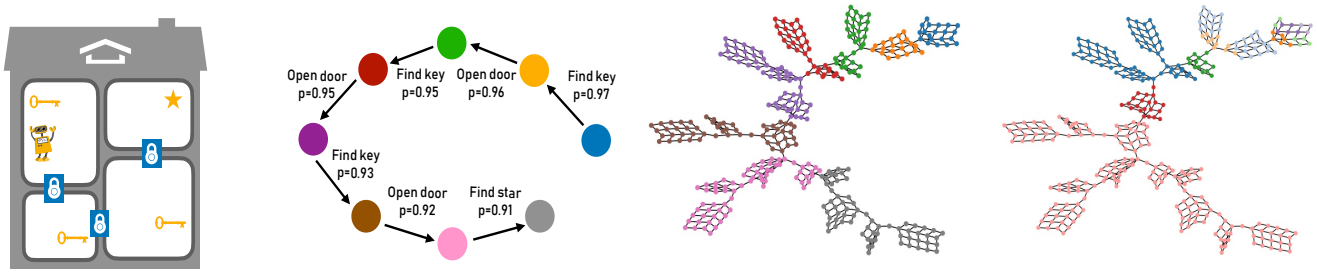
Specifically, an abstract ψ-SMDP is a tuple $\bar{\mathcal{M}} = \langle \bar{\mathcal{S}}, \bar{\mathcal{O}}, \bar{P}, \bar{\boldsymbol{\psi}}, \gamma \rangle$, where $\bar{\mathcal{S}}$ is a set of abstract states, $\bar{\mathcal{O}}$ are the abstract options, whose transition dynamics between the abstract states are described by $\bar{P}^{\bar{o}}_{\bar{s},\bar{s}'}$, and $\bar{\boldsymbol{\psi}}^{\bar{o}}_{\bar{s}}$ is the SF of executing $\bar{o}$ from $\bar{s}$. A successor homomorphism $h = (f(s), g_s(o), w_s)$ maps a (ground) ψ-SMDP to an abstract ψ-SMDP, where $f(s)$ aggregates ground states with approximately equal option transition dynamics into abstract states, $g_s(o)$ is our state-dependent mapping between ground and abstract options, and w_s is a weight function which weighs the aggregated ground states towards each abstract state.

Definition 4.1 (ϵ-Approximate Successor Homomorphism)**.** *Let $h = (f(s), g_s(o), w_s)$ be a mapping from a ground $\boldsymbol{\psi}$-SMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{O}, P, \boldsymbol{\psi}, \gamma \rangle$ to an abstract $\boldsymbol{\psi}$-SMDP $\bar{\mathcal{M}} = \langle \bar{\mathcal{S}}, \bar{\mathcal{O}}, \bar{P}, \bar{\boldsymbol{\psi}}, \gamma \rangle$, where $f \colon \mathcal{S} \to \bar{\mathcal{S}}$, $g_s \colon \mathcal{O} \to \bar{\mathcal{O}}$ and $w \colon \mathcal{S} \to [0, 1]$ s.t. $\forall \bar{s} \in \bar{\mathcal{S}}, \sum_{s \in f^{-1}(\bar{s})} w_s = 1$. h is an $(\epsilon_P, \epsilon_\psi)$*-approximate successor homomorphism *if for $\epsilon_P, \epsilon_\psi > 0$, $\forall s_1, s_2 \in \mathcal{S}, o_1, o_2 \in \mathcal{O}$,*

$$h(s_1, o_1) = h(s_2, o_2)^2 \implies$$
$$\forall s' \in \mathcal{S} \mid \sum_{s'' \in f^{-1}(f(s'))} P^{o_1}_{s_1,s''} - P^{o_2}_{s_2,s''} | \leq \epsilon_P,$$
$$\text{and } \|\boldsymbol{\psi}^{o_1}_{s_1} - \boldsymbol{\psi}^{o_2}_{s_2}\|_1 \leq \epsilon_\psi.$$

Having defined the abstract states, the transition dynamics and features of the abstract ψ-SMDP $\bar{\mathcal{M}}$ can be computed by:

[2] $h(s_1, o_1) = h(s_2, o_2) \Leftrightarrow f(s_1) = f(s_2)$ and $g_{s_1}(o_1) = g_{s_2}(o_2)$



(b) Abstract ψ-SMDP (c) Successor Homomorphism (d) Q (all): Q^*-irrelevance Abstraction

Figure 1: MDP abstraction in the Object-Rooms domain. (b) Abstract ψ-SMDP induced by our successor homomorphism from the ground MDP as shown in (c), the abstract states in (b) correspond to aggregated ground states of the same colour in (c). (d) Abstraction induced by approximate Q^*-irrelevance abstraction (cf. Appendix A.4) for *find key*; the abstraction does not carry temporal semantics, and is not reusable for other tasks e.g., *find star*.. Another example can be found in Figure 10 in Appendix A.6 and more details are in the experiments section.

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

where $\sum_{s''\in f^{-1}(f(s'))} P^{o}_{s,s''}$ refers to the transition probability from ground state s to an abstract state $\bar{s}' = f(s')$ following option o. Intuitively, two states mapping to the same abstract state have approximately equal option transition dynamics towards all abstract states, and the corresponding ground options induce similar successor features.

For efficient computation of the abstract model, the transition dynamics condition can be alternatively defined on the ground states s.t. $h(s_1, o_1) = h(s_2, o_2) \implies \forall s' \in S$,

$$|P^{o_1}_{s_1,s'} - P^{o_2}_{s_2,s'}| \leq \epsilon_P, \text{ and } \quad \|\boldsymbol{\psi}^{o_1}_{s_1} - \boldsymbol{\psi}^{o_2}_{s_2}\|_1 \leq \epsilon_\psi, \quad (2)$$

which states that two states mapping to the same abstract state have approximately equal option transition dynamics towards all *ground* states, and the corresponding ground options induce similar SF. In general, Def 4.1 and Eq. (2) can result in different abstractions, but similar performance bounds could be derived. In cases in which multiple ground options map to the same abstract option, $g_s^{-1}(\bar{o})$ picks one of the ground options. An example abstract $\boldsymbol{\psi}$-SMDP induced by our approximate successor homomorphism is shown in Fig. 1.

Our successor homomorphism combines state and temporal abstraction, by aggregating states with similar multi-step transition dynamics and feature expectations, in contrast to one-step transition dynamics and rewards. This formulation not only relaxes the strong symmetry requirements among the aggregated states but also provides the induced abstract ψ-SMDP with semantic meanings obtained from the options. Furthermore, each abstract $\boldsymbol{\psi}$-SMDP instantiates a family of abstract SMDPs (cf. Def. A.3, Appendix A.3) by specifying a task, i.e., reward weight w_r on the features. Extending results from [Abel *et al.*, 2016; Abel *et al.*, 2020] to our setting, we can guarantee performance of planning with the abstract ψ-SMDP across different tasks.

Theorem 4.1. *Let $w_r \in \mathbb{R}^d$ be a 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 $\kappa = |w_r|(2\epsilon_\psi + \frac{\epsilon_P |\bar{\mathcal{S}}| \max_{s,a} |\boldsymbol{\theta}(s,a)|}{1-\gamma})$. (cf. appendix for the proof)*

Goal	Algorithm	2 Rooms success	2 Rooms *LP*	3 Rooms success	3 Rooms *LP*	4 Rooms success	4 Rooms *LP*
Find Key	IRL-NAIVE	1.0	157	1.0	487	1.0	1463
	IRL-BATCH	1.0	**2**	1.0	**2**	1.0	**2**
	OK	1.0	-	1.0	-	1.0	-
Find Star	IRL-NAIVE	1.0	157	1.0	487	1.0	1463
	IRL-BATCH	1.0	**2**	1.0	**2**	1.0	**2**
	OK	1.0	-	1.0	-	1.0	-
Find Key, Open Door	IRL-NAIVE	1.0	157	1.0	487	**1.0**	1463
	IRL-BATCH	**1.0**	**3**	**1.0**	**15**	0.95	**29**
	OK	0.08	-	0.56	-	0.71	-
Find Key, Open Door, Find Star	IRL-NAIVE	1.0	157	1.0	487	1.0	1463
	IRL-BATCH	**1.0**	**4**	**1.0**	**16**	**1.0**	**7**
	OK	0.0	-	0.51	-	0.66	-

Table 1: Performance and efficiency of option grounding in Object-Rooms. We show the success rate (success) of the learned options for achieving all specified goals across all starting states in the initiation set, and the number of LPs used to find the option policy.

5 Experiments

In this section, we empirically evaluate our proposed abstraction scheme for option transfer to new environments with known and unknown transition dynamics, and planning with abstract SMDPs. Additional details for all experiments are available in Appendix A.6.

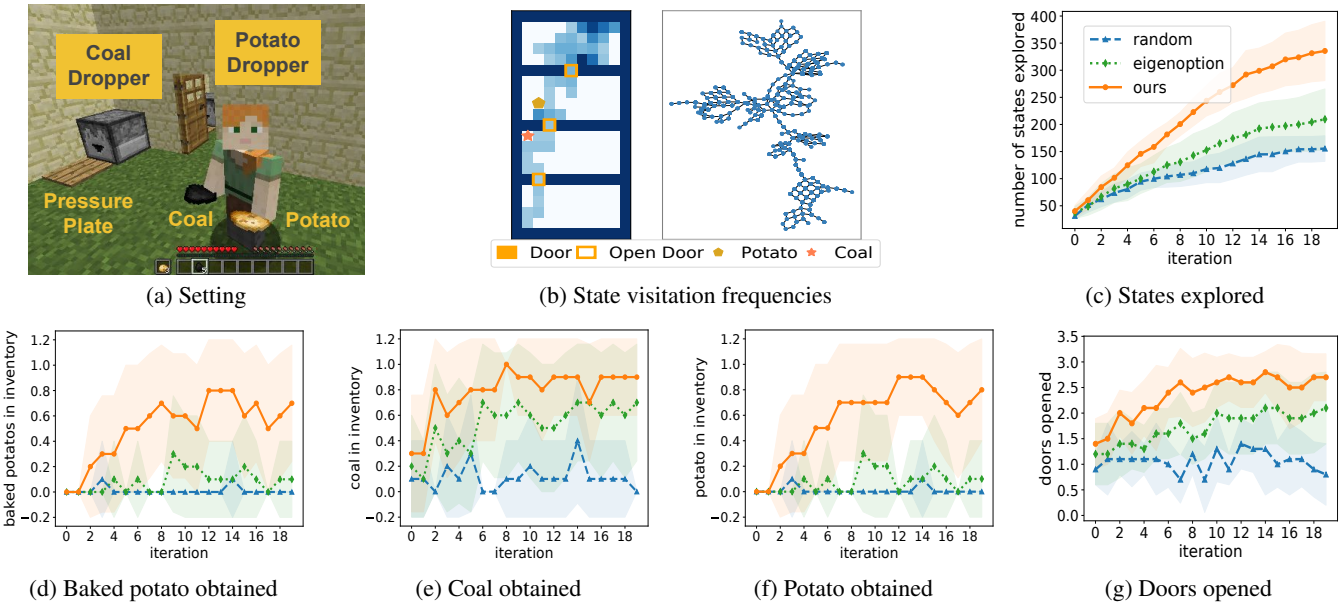
5.1 Option Transfer

We evaluate the performance and efficiency of our option grounding algorithm for transferring options given by expert demonstrations to new environments.

Transfer to Environments with Known Dynamics

We compare our option grounding algorithms with the baseline algorithm Option Keyboard (OK) [Barreto *et al.*, 2019] on the Object-Rooms Setting.

• IRL-NAIVE *(Algorithm 1) and* IRL-BATCH *(Appendix; Algorithm 3).* Our both algorithms *transfer* an expert's demonstration from a 2-Room source environment to a target environment of 2-4 Rooms, by first encoding the expert's



(a) Setting (b) State visitation frequencies (c) States explored

(d) Baked potato obtained (e) Coal obtained (f) Potato obtained (g) Doors opened

Figure 2: Exploring and grounding options in unknown environments in Minecraft. Figures (c)-(g) share legends.

demonstration as an abstract option, then grounding the abstract option in the target environment.

• *Option Keyboard (*OK*)* [Barreto *et al.*, 2019]: OK first trains primitive options (find-key, open-door, find-star) to maximise a cumulant (pseudo-reward on the history). The cumulants are hand-crafted in the target environment, where the agent is rewarded when terminating the option after achieving the goal. Then the composite options are synthesized via generalized policy improvement (GPI).

Object-Rooms Setting. N rooms with keys and stars inside are connected by doors. There are 6 actions: $\mathcal{A} = \{$Up, Down, Left, Right, Pick up, Open$\}$. The agent can pick up the keys and stars, and use the keys to open doors. See Figure 4 in Appendix A.6 for an illustration.

Results. Table 1 shows the success rate of the computed options for achieving the required goals across all starting states in the initiation sets. Both IRL-NAIVE and IRL-BATCH successfully transfer the options. Moreover, IRL-BATCH significantly reduces the number of LPs solved. The OK composed options have a low success rate for achieving the required goals, e.g., for grounding "find key, open door, find star" in the 3-Room setting, the learned options often terminate after achieving 1-2 subgoals.

Transfer to Environments with Unknown Dynamics

To perform grounding with unknown transition dynamics, the agent simultaneously explores and computes ground options.

Bake-Rooms Setting (Minecraft). The considered environment is based on Malmo, a standard AI research platform using Minecraft [Johnson *et al.*, 2016], as shown in Figure 2: 4 rooms (R1-R4 from bottom to top) are connected by doors, and the agent starts in R1. To obtain a baked potato, the agent needs to open doors, collect coal from a coal dropper in R2, collect potato from a dropper in R3, and issue a *craft* command to bake the potato. Agents observe their current location, objects in nearby 3×3 grids, and the items in inventory.

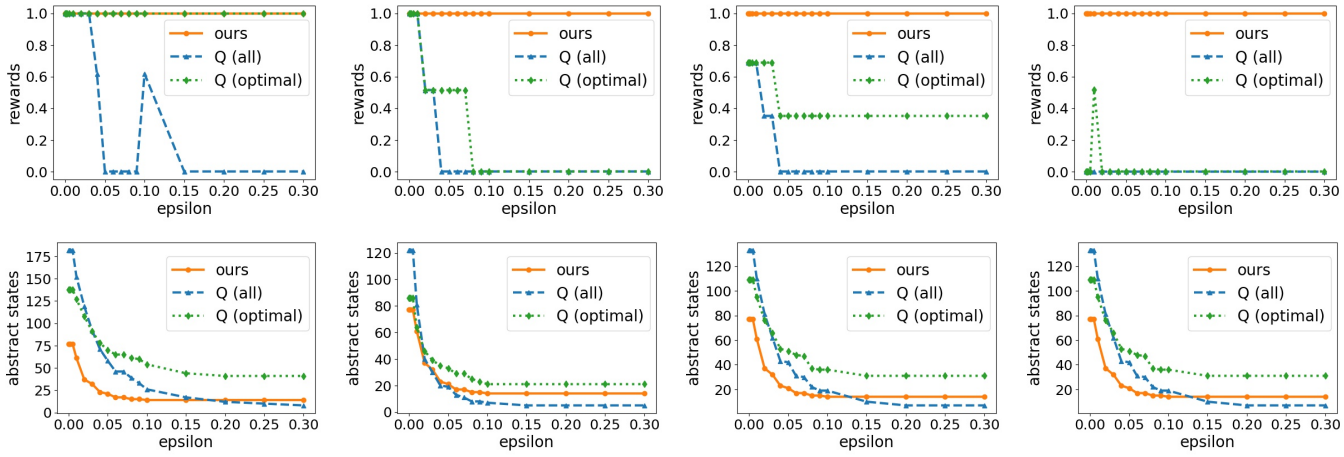
Baselines and learning. We compare our algorithm IRL-BATCH with the baselines (i) *random walk* and (ii) *eigenoptions* [Machado *et al.*, 2018]. Each agent runs for 20 iterations, each consisting of 200 steps. In the first iteration, all agents execute random actions. After each iteration, the agents construct an MDP graph based on collected transitions from all prior iterations. The eigenoption agent computes 3 eigenoptions of smallest eigenvalues using the normalized graph Laplacian, while IRL-BATCH grounds the 3 abstract options: *1. open and go to door, 2. collect coal, 3. collect potato*. In the next iteration, agents perform a random walk with both primitive actions and the acquired options, update the MDP graph, compute new options, and so on.

Results. Figure 2 (a) shows the state visitation frequency of our algorithm in the 14^{th} iteration and a constructed transition graph (more details in Figure 7, Appendix A.6). The trajectory shows that from R1 (bottom), the agent learns to open door, collect coal in R2, open door, collect potato in R3, and navigate around the rooms. (c) shows that our algorithm explores on average more than 50 percent more states than both baselines in 20 iterations. (d) - (g) show the objects collected by the agents (max count is 1) and the number of doors opened. The agent using our algorithm learns to open door and collect coal within 2 iterations, and it learns to bake a potato 50 percent of the time within 5 iterations. In all cases, our algorithm outperforms the baselines. The results are averaged over 10 seeds, shaded regions show standard deviations.

5.2 Planning with Abstract SMDP

We compare the performance of planning using our abstract ψ-SMDPs found by successor homomorphism, with two baseline abstraction methods [Li *et al.*, 2006; Abel *et al.*, 2016] on the *Object-Rooms* environment.

1. Q (all): (Q^*-irrelevance abstraction), if $f(s_1) = f(s_2)$,



(a) dense reward (b) sparse reward (c) transfer (w. overlap) (d) transfer (w.o. overlap)

Figure 3: 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 MDPs. The x-axes are the distance thresholds ϵ. *transfer* refers to task transfer (i.e., different reward function). Further details are available in Appendix A.6.

then $\forall a, |Q^*(s_1, a) - Q^*(s_2, a)| \leq \epsilon$, where $Q^*(s, a)$ is the optimal Q-function for the considered MDP.

2. Q (optimal): (a^*-irrelevance abstraction), if $f(s_1) = f(s_2)$ then s_1 and s_2 share the same optimal action a^* and $|Q^*(s_1, a^*) - Q^*(s_2, a^*)| \leq \epsilon$.

Training and results. 3 abstract options are given: 1. open door, 2. find key, and 3. find star. To find the abstract model induced by our successor homomorphism, we first ground each abstract option. Then we form the abstract state by clustering the ground states s according to the pairwise distance of their option termination state distributions $\max_{s',\bar{o}} |P^{g_{s_1}^{-1}(\bar{o})}_{s_1,s'} - P^{g_{s_2}^{-1}(\bar{o})}_{s_2,s'}|$ through agglomerative clustering. Finally we compute the abstract transition dynamics and feature function. For Q (all) and Q (optimal), we first perform Q-value iteration on a source task to obtain the optimal Q-values, then cluster the states by their pairwise differences, e.g., $|Q^*(s_1, \cdot) - Q^*(s_2, \cdot)|_\infty$ for the Q (all) approach, then compute the abstract transition dynamics.

Figure 3 shows the results of using the induced abstract models for planning. Ours perform well across all tested settings with few abstract states. Since abstract ψ-SMDPs do not depend on rewards, the abstract model is robust across tasks (with varying reward functions) and sparse rewards settings. In contrast, abstraction schemes based on the reward functions perform worse when the source task for performing abstraction is different from the target task for planning.

6 Related Work

Agent-space options [Konidaris and Barto, 2007] are one of our conceptual inspirations. In our work, we tie the agent-space to the problem-space through features.

Successor features (SF). Successor representations (SR) were first introduced by [Dayan, 1993]. [Barreto *et al.*, 2017] proposed SF which generalised SR. [Machado *et al.*, 2018] discovered eigenoptions from SF and showed their equivalence to options derived from the graph Laplacian. [Ramesh *et al.*, 2019] discovered successor options via clustering over the SR. The Option Keyboard [Barreto *et al.*, 2019] is a pioneering work for option combinations: primitive options are first trained to maximise cumulants (i.e., pseudo-rewards on histories), then by putting preference weights over the cumulants, new options are synthesized to maximise the weighted sum of cumulants. For the purpose of option transfer and grounding, however, this may yield imprecise option behaviours due to the interference between cumulants. In contrast, our successor feature-matching formulation generates precise option behaviours as demonstrated in Table 1.

MDP Abstraction. MDP abstraction through bisimulation was first considered by [Dean and Givan, 1997]. [Ravindran and Barto, 2002] introduced MDP homomorphisms, which account for action equivalence with a state-dependent action mapping. [Li *et al.*, 2006] proposed a unified theory of abstraction, and approximate abstraction mechanisms were studied by [Ferns *et al.*, 2004] and [Abel *et al.*, 2016]. Most relevant to our work are SMDP homomorphisms [Ravindran and Barto, 2003] and theoretical formulations for reward-based abstractions in SMDPs in [Abel *et al.*, 2020]. Different from prior abstraction formulations which are reward-based, our feature-based successor homomorphism produces abstract models which are reusable across tasks.

7 Conclusion

We studied temporal and state abstraction in RL using SF. Specifically, we developed an abstract option representation with SF and presented algorithms that transfer existing ground options to new environments. Based on the abstract options, we developed the successor homomorphism, which produces abstract ψ-SMDPs that can be used to perform efficient planning with near-optimal performance guarantees. We demonstrated empirically that our algorithms transfer the options effectively and efficiently, and showed that our abstract ψ-SMDP models exhibit meaningful temporal semantics, with near-optimal planning performance across tasks.

References

[Abbeel and Ng, 2004] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004.

[Abel *et al.*, 2016] David Abel, David Hershkowitz, and Michael Littman. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, pages 2915–2923. PMLR, 2016.

[Abel *et al.*, 2020] David Abel, Nate Umbanhowar, Khimya Khetarpal, Dilip Arumugam, Doina Precup, and Michael Littman. Value preserving state-action abstractions. In *International Conference on Artificial Intelligence and Statistics*, pages 1639–1650. PMLR, 2020.

[Barreto *et al.*, 2017] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4058–4068, 2017.

[Barreto *et al.*, 2019] André Barreto, Diana Borsa, Shaobo Hou, Gheorghe Comanici, Eser Aygün, Philippe Hamel, Daniel K Toyama, Jonathan J Hunt, Shibl Mourad, David Silver, et al. The option keyboard: Combining skills in reinforcement learning. *Advances in Neural Information Processing Systems*, 2019.

[Dayan, 1993] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

[Dean and Givan, 1997] Thomas Dean and Robert Givan. Model minimization in markov decision processes. In *AAAI Conference on Artificial Intelligence*, 1997.

[Ferns *et al.*, 2004] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Uncertainty in Artifical Intelligence*, 2004.

[Girshick, 2015] Ross Girshick. Fast r-cnn. In *International Conference on Computer Vision*, pages 1440–1448, 2015.

[Ho and Ermon, 2016] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems*, 2016.

[Johnson *et al.*, 2016] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *International Joint Conference on Artificial Intelligence*, 2016.

[Konidaris and Barto, 2007] George Dimitri Konidaris and Andrew G Barto. Building portable options: Skill transfer in reinforcement learning. In *International Joint Conference on Artificial Intelligence*, pages 895–900, 2007.

[Li *et al.*, 2006] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *International Symposium on Artificial Intelligence and Mathematics*, 2006.

[Machado *et al.*, 2018] Marlos C Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations*, 2018.

[Mehta *et al.*, 2008] Neville Mehta, Sriraam Natarajan, Prasad Tadepalli, and Alan Fern. Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning*, 73(3):289, 2008.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[Pons and Latapy, 2005] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *International Symposium on Computer and Information Sciences*, pages 284–293. Springer, 2005.

[Ramesh *et al.*, 2019] Rahul Ramesh, Manan Tomar, and Balaraman Ravindran. Successor options: An option discovery framework for reinforcement learning. In *International Joint Conference on Artificial Intelligence*, 2019.

[Ravindran and Barto, 2002] Balaraman Ravindran and Andrew G Barto. Model minimization in hierarchical reinforcement learning. In *International Symposium on Abstraction, Reformulation, and Approximation*, pages 196–211. Springer, 2002.

[Ravindran and Barto, 2003] Balaraman Ravindran and Andrew G Barto. Smdp homomorphisms: an algebraic approach to abstraction in semi-markov decision processes. In *International Joint Conference on Artificial Intelligence*, pages 1011–1016, 2003.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[Sutton *et al.*, 1999] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[Syed *et al.*, 2008] Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. In *International Conference on Machine Learning*, pages 1032–1039, 2008.