Reward-Aware Proto-Representations in Reinforcement Learning

Hon Tik Tse **Siddarth Chandrasekar** **Marlos C. Machado***
University of Alberta, Alberta Machine Intelligence Institute (Amii)
*Canada CIFAR AI Chair
{hontik, siddart2, machado}@ualberta.ca

Abstract

In recent years, the successor representation (SR) has attracted increasing attention in reinforcement learning (RL), and it has been used to address some of its key challenges, such as exploration, credit assignment, and generalization. The SR can be seen as representing the underlying credit assignment structure of the environment by implicitly encoding its induced transition dynamics. However, the SR is reward-agnostic. In this paper, we discuss a similar representation that also takes into account the reward dynamics of the problem. We study the default representation (DR), a recently proposed representation with limited theoretical (and empirical) analysis. Here, we lay some of the theoretical foundation underlying the DR in the tabular case by (1) deriving dynamic programming and (2) temporal-difference methods to learn the DR, (3) characterizing the basis for the vector space of the DR, and (4) formally extending the DR to the function approximation case through default features. Empirically, we analyze the benefits of the DR in many of the settings in which the SR has been applied, including (1) reward shaping, (2) option discovery, (3) exploration, and (4) transfer learning. Our results show that, compared to the SR, the DR gives rise to qualitatively different, reward-aware behaviour and quantitatively better performance in several settings.

1 Introduction

Learning appropriate representations is a key challenge in reinforcement learning (RL). The successor representation (SR) [10], which represents states as the expected discounted number of visits to successor states, is a particularly promising idea. It has been shown to be an effective distance metric for reward shaping [54, 49], a promising inductive bias for temporally-extended exploration [21, 25], and an effective representation for credit assignment [27, 23], generalization [20], and zero-shot RL [47, 48, 6, 44, 1]. Furthermore, recent evidence suggests that the SR is a good computational model for explaining decision-making [28] and neural activity in the brain [41, 40].

The underlying idea behind the SR is to incorporate the temporal aspect of the problem into the representation so that two states are considered similar if they lead to similar future outcomes. Similar ideas permeate other concepts such as proto-value functions [27, 26] and slow feature analysis [39, 53]. In this work, we use the term *proto-representations* to refer to such temporal representations that implicitly capture the environment dynamics, with the SR the most prominent of them. However, the SR (and others [29, 30, 5, 12, 52]) only captures the transition dynamics of the environment, and does not take rewards into account. In this paper, we demonstrate the benefits of learning a proto-representation that also takes rewards into account.

One candidate for reward-aware proto-representations is the *default representation* (DR) [34]. Piray and Daw, in a neuroscience study, applied the DR mainly to replanning tasks and explaining

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

different cognitive phenomena, such as habits and cognitive control. Nevertheless, we claim that the DR is an interesting concept beyond neuroscience, due to its relationship with the SR and its reward-awareness. In this context, many aspects of the DR have yet to be explored. There are no efficient, general algorithms for learning the DR incrementally and online with a linear cost, akin to temporal-difference (TD) learning [42]. Additionally, given the successful applications of the eigenvectors of the SR for reward shaping [54, 49], and temporally-extended exploration [25], the eigenvectors of the DR have seen limited investigation. Finally, the behavior of the DR for a reward function not constant over non-terminal states is underexplored.

In this paper, we (1) derive methods for learning the DR using dynamic programming (DP) and temporal-difference (TD) learning [42], and prove the convergence for the DP method, (2) characterize the basis for the vector space of the DR, also describing the settings in which the DR and the SR have the same eigenvectors, and (3) define default features, taking a first step towards using the DR with function approximation.

Furthermore, we empirically investigate whether the DR provides benefits over the SR. While prior work [34, 3] has mainly focused on applying the DR to explain phenomena in neuroscience, here we consider applications more common in computational RL. We demonstrate that: (1) in environments with low-reward regions to be avoided, using the DR for reward shaping achieves superior performance over the SR, (2) using the DR for online eigenoption discovery [25], the DR exhibits reward-aware exploratory behavior and obtains higher rewards than the SR over the course of exploration, (3) the DR, similar to the SR, can be used for count-based exploration, and (4) default features enable efficient transfer learning when the rewards at terminal states change.

2 Background

In this paper, we use lowercase symbols (e.g., r, p) to denote functions, calligraphic font (e.g, $\mathcal{S}, \mathcal{A}$) to denote sets, bold lowercase symbols (e.g., $\mathbf{r}, \mathbf{e}$) to denote vectors, and bold uppercase symbols (e.g., $\mathbf{\Psi}, \mathbf{Z}$) to denote matrices. We index the (i, j)-th entry of a matrix $\mathbf{A}$ by $\mathbf{A}(i, j)$.

2.1 Reinforcement Learning and the Successor Representation

In the standard RL framework, the environment is formulated as an MDP $\langle \mathcal{S}, \mathcal{A}, r, p, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \to \mathbb{R}$ (or $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$) is the reward function, $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, and $\gamma \in [0, 1)$ is the discount factor. At every time step t, the agent observes a state, S_t, and selects an action, A_t. The environment then transitions to a next state, $S_{t+1} \sim p(\cdot|S_t, A_t)$, and the agent receives a reward, $R_t = r(S_t)$ (or $R_t = r(S_t, A_t)$). The agent's goal is to learn a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected discounted return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_t]$.

In this framework, the successor representation [SR; 10] represents states as the expected discounted number of visits to their successor states. Given a policy π, the SR, $\mathbf{\Psi}^\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$, is defined as

$$\mathbf{\Psi}^\pi(s, s') = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{\{S_t = s'\}} | S_0 = s \right], \tag{1}$$

where $\mathbb{1}$ is the indicator function. Furthermore, denoting the transition probability matrix induced by p and π by $\mathbf{P}^\pi$, the SR can be computed in closed form by $\sum_{t=0}^{\infty} \gamma^t (\mathbf{P}^\pi)^t = (\mathbf{I} - \gamma \mathbf{P}^\pi)^{-1}$. Note that in this work, we set the rows corresponding to terminal states to be all zeros for $\mathbf{P}^\pi$. Denoting the set of trajectories from s to s' by $\mathcal{T}_{s \to s'}$, we can express the entries of the SR [5] as

$$\mathbf{\Psi}^\pi(s, s') = \sum_{\tau \in \mathcal{T}_{s \to s'}} \mathbf{P}^\pi(\tau) \gamma^{\eta(\tau)}, \tag{2}$$

where $\mathbf{P}^\pi(\tau)$ is the probability of following τ under π, and $\eta(\tau)$ denotes the number of steps in τ.

Finally, the SR can be learned using temporal-difference learning [42, 10], for all $j \in \mathcal{S}$:

$$\mathbf{\Psi}^\pi(S_t, j) \longleftarrow \mathbf{\Psi}^\pi(S_t, j) + \alpha \big[\mathbb{1}_{\{S_t = j\}} + \gamma \mathbf{\Psi}^\pi(S_{t+1}, j) - \mathbf{\Psi}^\pi(S_t, j) \big]. \tag{3}$$

2.2 Linearly Solvable Markov Decision Processes

The *default representation* (DR) is defined in the framework of linearly solvable MDPs [45, 46], a simplification of MDPs in which the optimal value function can be expressed by a linear equation. In this work, building on the work by Piray and Daw [34], we define linearly solvable MDPs as

$(\mathcal{S}, \mathcal{A}, r, p, \pi_d)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \to \mathbb{R}$ is the reward function, $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, and π_d is the *default policy* that assigns non-zero probabilities to all state-action pairs. It is standard in linearly solvable MDPs to assume no discounting of rewards, i.e., $\gamma = 1$. We also make the assumption that $r(s) < 0$ for all non-terminal s. In order to make MDPs linearly solvable, the agent is tasked not only with maximizing the rewards, but also with not deviating too much from the default policy. Deviating from the default policy incurs a cost to the agent, causing the agent to also receive a penalty $\lambda \, \mathrm{KL}(p^{\pi}(\cdot|S_t) \| p^{\pi_d}(\cdot|S_t))$ at every time step t, where $\mathrm{KL}(u\|v)$ denotes Kullback-Leibler (KL) divergence between u and v, $p^{\pi}(s'|s)$ denotes the probability of transitioning to s' from s under policy π, and $\lambda > 0$ determines the relative importance of the deviation cost.

The DR, introduced under this formulation, is a reward-aware proto-representation that encodes the expected rewards for visiting successor states. Let $\mathbf{r} \in \mathbb{R}^{|\mathcal{S}|}$ be the vector of rewards at all states, and $\mathbf{P}^{\pi_d}$ be the transition probability matrix induced by π_d (i.e., $\mathbf{P}^{\pi_d}(s, s')$ is the probability of transition from s to s' under π_d), the DR, denoted by $\mathbf{Z}$, can be computed in closed form by

$$\mathbf{Z} = \Big[\mathrm{diag}\big(\exp(-\mathbf{r}/\lambda)\big) - \mathbf{P}^{\pi_d}\Big]^{-1}, \tag{4}$$

where $\exp$ denotes element-wise exponentiation.

Importantly, the DR can be used to retrieve the optimal value function with a set of linear equations [34]. Let N, T be the set of indices of non-terminal and terminal states, and $\mathcal{S}_N$ and $\mathcal{S}_T$ denote the set of non-terminal and terminal states. We have

$$\exp(\mathbf{v}^*_N/\lambda) = \mathbf{Z}_{NN}\mathbf{P}^{\pi_d}_{NT}\exp(\mathbf{r}_T/\lambda), \tag{5}$$

where $\mathbf{v}^*_N$ is the vector of optimal values for non-terminal states, and $\mathbf{P}^{\pi_d}_{NT} \in \mathbb{R}^{|\mathcal{S}_N| \times |\mathcal{S}_T|}$ denotes the matrix of transition probabilities. Note that after learning the DR, assuming access to the transition dynamics, it is possible to directly compute the optimal values for any new configurations of terminal state rewards, a fact that has been used in the past to perform transfer learning [34, 3].

3 Reward-Aware Proto-Representations

In the previous section, we presented the DR in a much more limited way than when discussing the SR. This is not by accident; up to this point, the DR has not been that well developed. Little is written about the impact of environment rewards on the DR, how those impact the space the DR spans, and we still do not have a state-action formulation of the DR. We address these issues in this section.

3.1 The Default Representation as a Generalization of the Successor Representation

In problems in which the reward signal is the same across the whole state space, the DR spans the same space spanned by the SR. However, in settings in which the rewards received by the agent vary throughout an episode (see Figure 1), the DR captures the expected rewards obtained when traveling between two states, instead of the number of transitions required to travel between states, as in the SR.

We first formalize the idea that the DR and SR span the same space in Theorem 3.1 in settings in which the agent does not have access to a reward signal. Prior work had been limited to comparing the analytical form of the SR and DR [3].

Theorem 3.1. *Suppose both the SR and DR are computed with respect to the same policy, i.e., $\pi = \pi_d$. When the reward function is constant and negative, i.e., $r(s) = r(s') < 0 \; \forall s, s' \in \mathcal{S}$, the SR and DR share the same set of eigenvectors. Furthermore, when the SR and DR are symmetric, the i-th eigenvectors of the SR and DR are equivalent, and the i-th eigenvalues of the SR ($\mu_{SR,i}$) and DR ($\mu_{DR,i}$) are related as follows:*

$$\mu_{SR,i} = \Big[\gamma\Big(\mu^{-1}_{DR,i} - \exp\big(-r(s)/\lambda\big) + \gamma^{-1}\Big)\Big]^{-1}, \tag{6}$$

where $\gamma \in (0, 1)$ is the discount factor of the SR, $r(s)$ is the state reward, and λ is the relative importance of the deviation cost of the DR.

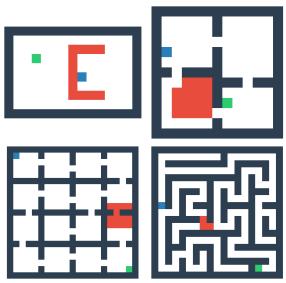

Figure 1: Episodic envs. adapted to incorporate negative rewards. Clockwise: 1) *grid task* [10], 2) *four rooms* [43], 3) *grid room* [49], and 4) *grid maze* [49]. Start state is in blue. The agent receives -1 reward at every time step unless it steps on red tiles (-20 reward) or reaches the goal in green (0 reward).

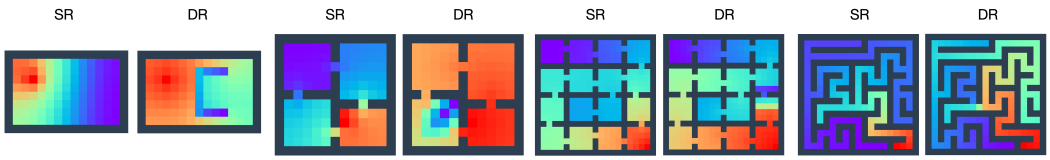

Figure 2: Top eigenvectors of the SR and the DR in the environments shown in Figure 1. We report the logarithm of the DR for better visualization due to very different magnitudes.

Proof sketch. We write the DR as $[\tilde{r}\mathbf{I} - \mathbf{P}^{\pi}]^{-1}$, where $\tilde{r} = \exp(-r(s)/\lambda)$, and perform algebraic manipulations on the expression $\mathbf{Z}\mathbf{e} = \lambda\mathbf{e}$ to obtain the eigenvectors of $[\mathbf{I} - \gamma\mathbf{P}^{\pi}]^{-1}$. Then, we show that the eigenvalues of the SR and DR are related by a function that is monotonically increasing over the range of DR eigenvalues. The complete proof is available in Appendix B.2.

This result is particularly interesting because the eigenvectors of the SR have been shown to be useful in settings such as reward shaping and option discovery. The top eigenvector of the SR captures temporal distance between states, and such distance has been used for reward shaping as it leads to better performance than distances that are based on the coordinates of states [49, 54]. In option discovery, it has been shown that eigenoptions, which are obtained by training a policy to maximize a reward signal induced by the eigenvectors of the SR, allow the agent to reach less explored areas of the state space, facilitating exploration [25]. We will revisit these applications in Section 6.

In problems in which the reward function is not constant, the DR is able to capture such information. While this is evident by the reward vector, $\mathbf{r}$, in Equation 4, we can re-write the DR to match the closed-form of the SR (Equation 2) to make such a relationship clearer:

$$\mathbf{Z}(s, s') = \sum_{\tau \in \mathcal{T}_{s \to s'}} \mathbf{P}^{\pi_d}(\tau) \exp(r(\tau)/\lambda), \tag{7}$$

where $r(\tau)$ is the sum of rewards obtained along trajectory τ (see derivation in Appendix B.1).

By writing Equation 7 similarly to the SR's, we see that both the SR and the DR compute a weighted average of a statistic over all trajectories from s to s', where the weights correspond to the probabilities of the trajectories under the policies w.r.t. which the SR and DR are computed. For the SR, the statistic captures the number of transitions required to travel between states, while for the DR, it captures the rewards obtained when travelling between states.

Figure 2 illustrates the eigenvectors of the DR and SR in the environments depicted in Figure 1 to help the reader develop an intuition of the similarities and differences between these two proto-representations. While the top eigenvectors of the SR reflect the proximity of states in terms of transitions, the top eigenvectors of the DR not only capture the transition dynamics of the environment, but they also accurately reflect the positions of low-reward regions in the environment.

3.2 State-Action-Dependent Rewards

The form of the DR discussed so far is defined for reward functions that depend only on the state, and cannot be applied when the reward function depends on state-action pairs, which is prevalent in RL.

To extend the DR to state-action pairs, we need to extend linearly solvable MDPs to state-action pairs, which was first presented in the work by Ringstrom et al. [37]. However, the DR was not discussed in this work, so we provide our derivation in Appendix B.3. Interestingly, the resulting equation looks similar to Equation 4. Using the overbar for vectors and matrices in the state-action setting, e.g., $\bar{\mathbf{P}}$ instead of $\mathbf{P}$, the DR for state-action-dependent rewards is:

$$\bar{\mathbf{Z}} = \left[\operatorname{diag}\left(\exp(-\bar{\mathbf{r}}/\lambda) \right) - \bar{\mathbf{P}}^{\pi_d} \right]^{-1}, \tag{8}$$

where and $\bar{\mathbf{r}} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ is the vector of state-action rewards, and $\bar{\mathbf{P}}^{\pi_d} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}||\mathcal{A}|}$ the matrix of transition probabilities between state-action pairs under π_d.

By learning the DR for the state-action space, the optimal Q-values can now be computed by

$$\exp(\bar{\mathbf{q}}_N/\lambda) = \bar{\mathbf{Z}}_{NN} \bar{\mathbf{P}}^{\pi_d}_{NT} \exp(\bar{\mathbf{r}}_T/\lambda), \tag{9}$$

where $\bar{\mathbf{q}} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ is the vector of optimal state-action values. Furthermore, the optimal policy can be computed simply as

$$\pi^*(a|s) = \frac{\pi_d(a|s)\exp(q^*(s,a)/\lambda)}{\sum_{a'}\pi_d(a'|s)\exp(q^*(s,a')/\lambda)}. \tag{10}$$

As we show in Section 5, the DR under this formulation, combined with default features, unlike prior work, allows direct computation of optimal policies when terminal rewards change, enabling efficient transfer without assuming access to transition dynamics [34, 3].

4 General Algorithms for Estimating the Default Representation

We now introduce efficient and general algorithms for learning the DR through dynamic programming and TD learning. Before the results we outline below, there were two update mechanisms for the DR: one based on the Woodbury matrix identity [34], which is limited to the tabular case; and a TD learning method recently introduced in a pre-print that is limited to settings in which the reward function is constant over non-terminal states [3]. We refer the reader back to the previous section to emphasize how different the DR is in more general settings, with rewards that vary across states.

4.1 Dynamic Programming

We first present a method to learn the DR by dynamic programming (DP), and we prove it converges. The update rule for the DP algorithm and the convergence result are presented in the theorem below.

Theorem 4.1. *Let $\mathcal{S}_N$ denote the set of non-terminal states. Assume $r(s) < 0\ \forall s \in \mathcal{S}_N$. Let $\mathbf{R} = \mathrm{diag}\big(\exp(-\mathbf{r}/\lambda)\big)$, where $\mathbf{r}$ is the vector of all state rewards. Let $\mathbf{Z}_0 = \mathbf{R}^{-1}$. The update rule*

$$\mathbf{Z}_{k+1} = \mathbf{R}^{-1} + \mathbf{R}^{-1}\mathbf{P}^{\pi_d}\mathbf{Z}_k \tag{11}$$

converges to the DR, that is, $\lim_{k\to\infty}\mathbf{Z}_k = \mathbf{Z}$.

Proof sketch. We recursively expand the RHS of Equation 11 and look at its closed-form in the limit, which we show is a Neumann series. We conclude the proof by leveraging the convergence of the Neumann series and plugging it back into the derived limit. The complete proof is in Appendix B.4.

4.2 Temporal Difference Learning

To learn the DR by TD learning, we build on the DP method and start with Equation 11. For any state s, for all j, $\mathbf{Z}(s,j)$ can be learned using DP as

$$\mathbf{Z}(s,j) = \exp\big(r(s)/\lambda\big)\Big(\mathbb{1}_{\{s=j\}} + \mathbb{E}_{s'\sim p^{\pi_d}(s'|s)}\big[\mathbf{Z}(s',j)\big]\Big), \tag{12}$$

which requires access to the transition probabilities under the default policy.

In TD learning, we do not assume access to these probabilities, but we can approximate the above equation using samples, estimating the expectation over next states s' with a sampled next state. Then, given a transition (s,a,r,s') sampled under the default policy, we can update $\mathbf{Z}(s,j)$ as

$$\mathbf{Z}(s,j) \longleftarrow \mathbf{Z}(s,j) + \alpha\Big[Y - \mathbf{Z}(s,j)\Big], \tag{13}$$

where α is the step size and Y the bootstrapping target. For non-terminal states, $Y = \exp(r/\lambda)\big(\mathbb{1}_{\{s=j\}} + \mathbf{Z}(s',j)\big)$, while for terminal states, Y = $\exp(r/\lambda)\mathbb{1}_{\{s=j\}}$. Naturally, we can leverage importance sampling [36] if the default policy is different from the behaviour policy.

5 Extending The DR Beyond Classical Tabular RL

We have been using the tabular case to formalize the ideas we are introducing; now we extend them beyond this setting. Given Equation 13, it is trivial to formalize the parameterization of the DR such that $\mathbf{Z}(s,s';\boldsymbol{\theta}) \approx \mathbf{Z}(s,s')$, for some parameters $\boldsymbol{\theta}$. In this section, we go beyond that, defining the concept of *default features*, which is more semantically meaningful when talking about the DR in terms of features. We also define a proto-representation for the maximum entropy RL framework. However, since this paper focuses on the DR, we defer the discussion of this proto-representation to Appendix B.5.

Similar to successor features [SFs; 2], we derive here a decomposition of the value function into features and weights such that these features capture the dynamics of the environment. Different from SFs, default features also consider non-terminal rewards as part of the environment dynamics.

Recall we can use the DR to obtain the optimal value function with a set of linear equations (Eq. 5):

$$\exp(\mathbf{v}_N^*/\lambda) = \mathbf{Z}_{NN}\mathbf{P}_{NT}^{\pi_d}\exp(\mathbf{r}_T/\lambda). \tag{14}$$

Inspired by the decomposition performed with SFs, we define default features (DFs) by decomposing the terminal rewards in this equation. We assume that the exponential of the terminal state rewards can be computed as $\exp\big(r(s)/\lambda\big) = \boldsymbol{\phi}(s)^\top \mathbf{w}$, where $\boldsymbol{\phi}(s), \mathbf{w} \in \mathbb{R}^d$ are the features of a terminal state, s, and the weights of the reward function, respectively. Note that this decomposition is not restrictive, since we can fully recover any terminal state rewards if $\boldsymbol{\phi}(s) = \exp\big(r(s)/\lambda\big)$.

Under this decomposition, the above equation can be written as

$$\exp(\mathbf{v}_N^*/\lambda) = \mathbf{Z}_{NN}\mathbf{P}_{NT}^{\pi_d}\exp(\mathbf{r}_T/\lambda) = \mathbf{Z}_{NN}\mathbf{P}_{NT}^{\pi_d}\boldsymbol{\Phi}\mathbf{w}, \tag{15}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{|S_T|\times d}$ is the matrix of terminal state features. We define the product $\mathbf{Z}_{NN}\mathbf{P}_{NT}^{\pi_d}\boldsymbol{\Phi}$ as the DFs matrix, and each row $\boldsymbol{\zeta}(s) = (\mathbf{Z}_{NN}\mathbf{P}_{NT}^{\pi_d}\boldsymbol{\Phi})_{s,:}$ to be the DFs of a non-terminal state. Then, the optimal value for each non-terminal state can be computed as $\boldsymbol{\zeta}(s)^\top\mathbf{w}$. Note that such a form of optimal value computation is only possible when only the rewards at terminal states change. The reason is that, when only the terminal state rewards change, $\zeta(s)$ stays the same for all non-terminal s. However, when the rewards for non-terminal states change, $\zeta(s)$ changes since $\zeta(s) = \mathbf{Z}_{NN}\mathbf{P}_{NT}^{\pi_d}\boldsymbol{\Phi}$ and $\mathbf{Z}_{NN}$ depends on non-terminal state rewards.

Building on Theorem 4.1, we can learn the DFs using TD learning from a transition (s, a, r, s') sampled under the default policy:

$$\boldsymbol{\zeta}(s) \longleftarrow \boldsymbol{\zeta}(s) + \alpha\big(\exp(r/\lambda)\boldsymbol{\zeta}(s') - \boldsymbol{\zeta}(s)\big), \tag{16}$$

where α is the step size, and we define $\boldsymbol{\zeta}(s) \coloneqq \boldsymbol{\phi}(s)$ for terminal states, s. DFs can be thought of as features propagating from the terminal states to all non-terminal states in a manner that respects the reward and transition dynamics. The definition of DFs can be easily extended to the case of state-action-dependent rewards, following the formulation in Section 3.2.

Apart from enabling function approximation, another benefit of learning the DFs over the DR is that it does not require access to the transition dynamics, $\mathbf{P}_{NT}^{\pi_d}$, to retrieve optimal values. This is especially useful when learning the DFs for state-action-dependent rewards, since we can directly retrieve the optimal policy using the corresponding state-action values (see Equation 10). After learning the DFs in this setting, even without access to environment dynamics, we can efficiently compute the optimal policy under any terminal state reward configuration, enabling transfer learning, as shown in Section 6. Note that while DFs enable directly computing the optimal policy, they are limited to the scenario where only the terminal state rewards change. On the other hand, while SFs can only compute a policy as good as the ones used to compute the SFs, they can be applied when any state rewards change, allowing more flexibility than DFs.

6 Experiments

Proto-representations, like the SR, have been used in RL for reward shaping [54, 49], option discovery [22, 25], count-based exploration [24], and transfer [2], among others. We now revisit these settings to assess the impact of using *reward-aware* representations.

6.1 Reward Shaping

Drawing inspiration from results with the SR [54, 49], we first evaluate the effectiveness of the DR as a distance metric. In reward shaping experiments, our goal is to assess whether the DR can serve as a useful shaping signal, capturing the geometry of the state space, and whether it not only guides the agent toward the goal state but also helps it avoid negative rewards.

Assuming $\mathbf{e}$ is the top eigenvector of the SR, prior work [54, 49] has used the shaping reward $\hat{r}_t = -\big(\mathbf{e}(s_{\text{goal}}) - \mathbf{e}(s_{t+1})\big)^2$, where $\mathbf{e}(s)$ denotes the entry for state s. Here, we instead use the top eigenvector $\mathbf{e}$ of the DR for potential-based reward shaping [31], defining the shaping reward as $\hat{r}_t = \gamma\mathbf{e}(s_{t+1}) - \mathbf{e}(s_t)$. We compare our approach (`DR-pot`) with three baselines: 1) potential-based

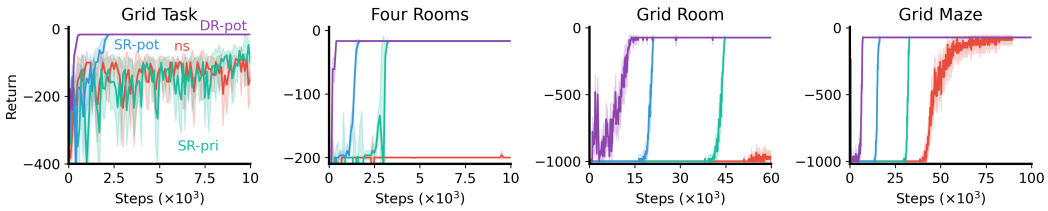

Figure 3: The avg. undiscounted return over 50 runs for potential-based reward shaping using the DR (`DR-pot`), the SR (`SR-pot`), the prior approach using the SR (`SR-prior`) [49], and no shaping (`ns`) in the environments shown in Figure 1. The shaded area indicates 95% confidence interval.

shaping with the SR's top eigenvector (`SR-pot`), 2) the distance-based SR method from [49] (`SR-pri`), and 3) no shaping (`ns`). Both the SR and DR are computed with respect to the uniform random policy.

We carry out the experiments in the environments shown in Figure 1. Importantly, the shortest path from the start state to the goal state passes through low-reward regions and is sub-optimal, so the agent needs to learn to avoid low-reward regions in the environment to achieve the optimal return. For the reward shaping approaches, we train a Q-learning [51] agent using a convex combination of the original environment reward, r_t, and the shaping reward, $\hat{r}_t$, resulting in the expression $(1-\beta)r_t + \beta\hat{r}_t$, where $\beta \in [0, 1]$ is a hyperparameter. Note that we assume access to the eigenvectors of the SR and DR prior to training the agent with the potential-based reward. Future work can explore learning the eigenvectors and the policy simultaneously. For the no shaping baseline, we simply train the Q-learning agent using the original environment reward. We use $\gamma = 0.99$, $\epsilon = 0.05$ for ϵ-greedy exploration, $\lambda = 1.3$ for the DR, and perform a grid search over the Q-learning's step size ($[0.1, 0.3, 1.0]$) and β ($[0.25, 0.5, 0.75, 1.0]$). We run 20 seeds for each hyperparameter setting, and after identifying the best hyperparameters, re-run 50 seeds to avoid maximization bias.

Figure 3 shows the undiscounted returns of our approach and the baselines in the environments shown in Figure 1. First, we observe that compared to prior methods, potential-based reward shaping is a better way of utilizing the eigenvectors of the proto-representations. Second, we observe that `DR-pot` performs significantly better than any of the baselines, including the SR-based approaches. This is because, as shown in Figure 2, the top eigenvector of the DR captures the location of low-reward regions, and is capable of guiding the agent along the optimal path that does not cross them, whereas the top eigenvector of the SR fails to do so, and simply leads the agent along the shortest sub-optimal path to the goal. As per our result in Section 3.1, the performance using the SR and the DR is very similar in the absence of low-reward regions (see Appendix C.2).

6.2 Option Discovery

The options framework [35, 43] allows the agent to interact with the environment in a temporally-extended manner. Prior work [25] demonstrated that the SR can be used for discovering options. In particular, their iterative online eigenoption discovery method using the SR, called *covering eigenoptions (CEO)*, greatly reduces the average number of steps required to visit every state, and enables exploring the state space more uniformly. Furthermore, CEO combined with deep learning has shown promising results and outperformed state-of-the-art baselines in a wide variety of settings [18]. Here, we consider using the DR for iterative online eigenoption discovery.

Our approach, *reward-aware covering eigenoptions (RACE)*, is similar to CEO but learns the DR instead of the SR, using its top eigenvector to define the intrinsic reward. In RACE, at every iteration, the agent collects samples to learn the DR, the eigenvector of which is then used to compute an eigenoption. The eigenoption will be used in the next RACE iteration to collect samples. This forms a cycle of using eigenoptions to improve the learned representation, and using the learned representation to then refine eigenoptions. The full algorithm and hyperparameters are in Appendix C.3. We compare RACE with CEO and a uniform random walk in the environments from Figure 1, using 100-step episodes with no terminal states. To evaluate exploration, we measure the percentage of states visited; to assess risk awareness, we track the average reward per time step. Our goal is to test whether RACE enables reward-aware exploration. The results are depicted in Figure 4, where each point represents the average performance of one hyperparameter setting over 10 seeds.

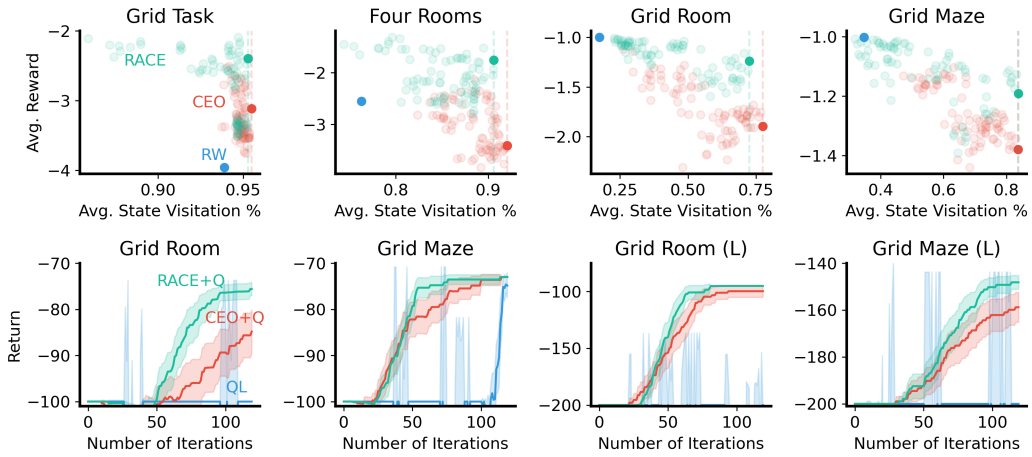

Figure 4: *Top row*: Average reward vs. state visitation percentage for various hyperparameter settings of iterative online eigenoption discovery via CEO, RACE, and random walk (RW) in the environments from Figure 1. For reference, solid dots mark settings with the highest visitation. *Bottom row*: Undiscounted return of RACE+Q, CEO+Q, and QL (baseline), averaged over 50 seeds. Rightmost environments are shown in Figure 10. Shaded areas indicate 95% confidence intervals.

RACE is better than CEO at avoiding low-reward regions. Interestingly, because of its reward-awareness, RACE has a slightly lower average state visitation percentage than CEO. The agent sometimes needs to take detours to visit states that are obstructed by low-reward regions, thus requiring a larger number of steps to visit other states (see Figure 9, in Appendix C.3). Again, a consequence of our Theorem 3.1 is that CEO and RACE behave very similarly in environments with constant rewards across the whole state space (see Figure 8, in Appendix C.3). Finally, RACE's exploratory behaviour also leads to faster learning when accumulating rewards. Results combining iterative online eigenoption discovery with Q-learning [51] are shown in Figure 4. Appendix C.3 provides details about these experiments. Interestingly, RACE+Q performs much better than CEO+Q because CEO does not distinguish between high-reward and low-reward paths, and can reach the goal state following a sub-optimal one, causing offline Q-learning to first learn the sub-optimal path.

6.3 Count-Based Exploration

Machado et al. [24] have shown that the norm of the SR, while it is being learned, encodes state-visitation counts. While reward-aware representations are somewhat at odds with pure exploration, given their risk-averse nature, we can still ask whether the norm of the DR can be used as a density model for pseudo-counts [4]. Here, we evaluate the use of the ℓ_2 norm of the DR as an exploration bonus. Specifically, the intrinsic reward is defined as $r_{\text{intr}}(s,a) = \beta \cdot \log(\|\bar{\mathbf{Z}}_{sa,:}\|_2)$, where β is a scaling factor.

Table 1: Model-free count-based exploration. Values in thousands ($\times 10^3$). 95% conf. intervals shown in parentheses.

Environment	Sarsa	+SR	+DR
RiverSwim	25 (0.8)	1,206 (566)	2,964 (252)
SixArms	265 (157)	1,066 (2,708)	3,518 (4,571)

We follow Machado et al.'s [24] experimental setup, and we consider two traditional exploration problems: Riverswim and SixArms. In Table 1, we report the performance (total undiscounted return) of Sarsa [38] with ϵ-greedy exploration, Sarsa + SR [24], and Sarsa + DR. Details about hyperameters are available in Appendix C.4. All results are averaged over 100 independent runs.

Our results suggest that the norm of the DR can indeed work as a density function since its use has led to performance orders of magnitude better than random exploration. Maybe surprisingly, Sarsa+DR even outperforms, with statistical significance, Sarsa+SR in RiverSwim. This is an interesting result, as, to learn the optimal policy, the agent needs to overcome its aversion to negative rewards, which it is encoding in its representation; but maybe this is precisely why it is somewhat more effective, as such a representation allows it to learn faster. This interplay is an interesting topic of future work.

6.4 Transfer Learning

The DR has been applied in the transfer learning setting to compute the optimal policy when the rewards at terminal states change [3, 34]. However, such computations require access to the transition dynamics of the environment. With our extension of the DR to state-action-dependent rewards and default features, it is now possible to compute the optimal policy without requiring access to the transition dynamics. We compare our approach with successor features [SFs; 2], which allow efficient transfer when the rewards of any states change. Note, however, that in this work we only consider the setting when only the rewards at terminal states change, since this is the setting that DFs can be applied to. The comparison of DFs and SFs under this setting does not serve as evidence that SFs is not effective for the setting it is designed for.

We perform experiments in the environment shown in Figure 5, consisting of four goal states. We describe our procedure for learning and using the SFs and DFs for transfer in Appendix C.5, as well as the features ϕ used. Figure 5 shows the performance of the DFs, the SFs computed with respect to different numbers of policies, and the optimal performance obtained by training an optimal policy under each new terminal state reward configuration using Q-learning [51]. As previously mentioned in Section 5, in this setting, DFs is extremely effective because it directly computes the optimal policy for new terminal state reward configurations, while SFs only computes a policy that is at least as good as the policies used to compute the SFs. Note, however, that DFs computes an optimal policy in the linearly solvable MDP setting that balances the reward function and the cost of deviating from the default policy.

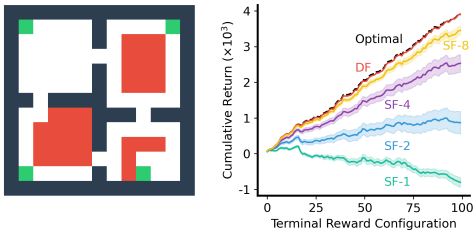

Figure 5: Left: Four rooms with multiple goals. Right: Cumulative return across new terminal state reward configurations. Curves are averaged over 50 runs. The shaded area shows 95% conf. interval.

7 Related Work

Throughout this paper, we have discussed the relevant related work for both the theoretical development and the empirical evaluation of our work. A line of work we did not discuss that might appear relevant for learning representations that capture both reward and transition dynamics is based on bisimulation metrics [13, 7], which quantify behavioural similarity between pairs of states by comparing their immediate rewards and distributions over the following states. These works enforce the bisimulation metric structure on the representation space by encouraging the difference between state representations to approximate the bisimulation metric [56, 8, 55, 17]. However, such approaches learn a distance-preserving mapping, where behaviorally similar states are mapped to nearby representations. This is in contrast to the DR or the SR, which encodes the environment dynamics directly in a semantically meaningful manner (see Eq. 7 and 2). As an immediate result, the DR supports planning (see Eq. 5) as well as other use cases discussed here, such as exploration and option discovery, in a manner not possible with bisimulation-metric-based representations.

8 Conclusion

In this work, we advance the theoretical foundation of the default representation (DR), a reward-aware proto-representation. Our contributions include: (1) deriving dynamic programming and TD methods to learn the DR, (2) analyzing its eigenspectrum, and (3) extending it to function approximation via default features. Empirically, we show that the DR, by incorporating reward structure, yields qualitatively different behavior than the SR across tasks such as reward shaping, option discovery, exploration, and transfer.

While the DR provides benefits in environments where being reward-aware is important, there are some limitations. First, naively computing the DR and its eigendecomposition can lead to numerical instabilities due to taking the exponential of negative rewards, as discussed in Appendix C.1. This could be problematic, especially in more complicated environments with long horizons. Second, as we show in Figures 6 and 8 in Appendix C, in environments in which all non-terminal state rewards

are the same, the DR will perform as well as the SR. Given the numerical issues associated with the DR, the SR may be a better choice in these environments. Third, while the DR captures the environment dynamics more fully, this comes at the expense of flexibility. While the SR, which is reward-agnostic, can allow transfer learning when any state rewards change, the DR, which is reward-aware, can only perform transfer in settings where only the terminal state rewards change.

Finally, this work lays a foundation for the DR as a potential middle ground between reward-agnostic and risk-sensitive approaches. While we focused on the tabular case for clarity and to be able to evaluate the DR across a range of use cases where reward-agnostic proto-representations have been applied, restricting the scope to this setting remains a limitation of this work. Future work should explore the use of the DR in more complex settings, starting with adapting neural network methods developed for approximating the SR [e.g., 19, 14, 9].

Acknowledgments and Disclosure of Funding

We thank Brett Daley for useful discussions and feedback. This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada CIFAR AI Chair Program, and Alberta Innovates. It was also enabled in part by computational resources provided by the Digital Research Alliance of Canada.

References

[1] S. Agarwal, H. Sikchi, P. Stone, and A. Zhang. Proto successor measure: Representing the behavior space of an RL agent. In *International Conference on Machine Learning (ICML)*, 2025.

[2] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, D. Silver, and H. van Hasselt. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[3] A. Bazarjani and P. Piray. Efficient learning of predictive maps for flexible planning. *PsyArXiv*, 2025. doi: 10.31234/osf.io/ak57f. URL `osf.io/preprints/psyarxiv/ak57f_v1`.

[4] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[5] L. Blier, C. Tallec, and Y. Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *CoRR*, abs/2101.07123, 2021.

[6] D. Borsa, A. Barreto, J. Quan, D. J. Mankowitz, H. van Hasselt, R. Munos, D. Silver, and T. Schaul. Universal successor features approximators. In *International Conference on Learning Representations (ICLR)*, 2019.

[7] P. S. Castro. Scalable methods for computing state similarity in deterministic Markov decision processes. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[8] P. S. Castro, T. Kastner, P. Panangaden, and M. Rowland. MICo: Improved representations via sampling-based state similarity for Markov decision processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[9] R. Chua, A. Ghosh, C. Kaplanis, B. A. Richards, and D. Precup. Learning successor features the simple way. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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

[11] K. Dvijotham and E. Todorov. Inverse optimal control with linearly-solvable MDPs. In *International Conference on Machine Learning (ICML)*, 2010.

[12] J. Farebrother, J. Greaves, R. Agarwal, C. L. Lan, R. Goroshin, P. S. Castro, and M. G. Bellemare. Proto-value networks: Scaling representation learning with auxiliary tasks. In *International Conference on Learning Representations (ICLR)*, 2023.

[13] N. Ferns, P. Panangaden, and D. Precup. Metrics for finite Markov decision processes. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2004.

[14] D. Gomez, M. Bowling, and M. C. Machado. Proper Laplacian representation learning. In *International Conference on Learning Representations (ICLR)*, 2024.

[15] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning (ICML)*, 2017.

[16] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

[17] M. Kemertas and T. Aumentado-Armstrong. Towards robust bisimulation metric learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[18] M. Klissarov and M. C. Machado. Deep Laplacian-based options for temporally-extended exploration. In *International Conference on Machine Learning (ICML)*, 2023.

[19] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman. Deep successor reinforcement learning. *CoRR*, abs/1606.02396, 2016.

[20] C. Le Lan, S. Tu, A. Oberman, R. Agarwal, and M. G. Bellemare. On the generalization of representations in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

[21] M. C. Machado. *Efficient exploration in reinforcement learning through time-based representations*. PhD thesis, University of Alberta, Canada, 2019.

[22] M. C. Machado, M. G. Bellemare, and M. Bowling. A Laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

[23] M. C. Machado, C. Rosenbaum, X. Guo, M. Liu, G. Tesauro, and M. Campbell. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations (ICLR)*, 2018.

[24] M. C. Machado, M. G. Bellemare, and M. Bowling. Count-based exploration with the successor representation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[25] M. C. Machado, A. Barreto, D. Precup, and M. Bowling. Temporal abstraction in reinforcement learning with the successor representation. *Journal of Machine Learning Research*, 24(80): 1–69, 2023.

[26] S. Mahadevan. Proto-value functions: Developmental reinforcement learning. In *International Conference on Machine learning (ICML)*, 2005.

[27] S. Mahadevan and M. Maggioni. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, 8(10):2169–2231, 2007.

[28] I. Momennejad, E. M. Russek, J. H. Cheong, M. M. Botvinick, N. D. Daw, and S. J. Gershman. The successor representation in human reinforcement learning. *Nature Human Behaviour*, 1(9): 680–692, 2017.

[29] T. Moskovitz, S. R. Wilson, and M. Sahani. A first-occupancy representation for reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022.

[30] T. Moskovitz, S. Hromadka, A. Touati, D. Borsa, and M. Sahani. A state representation for diminishing rewards. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[31] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning (ICML)*, 1999.

[32] J. Peters, K. Mulling, and Y. Altun. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

[33] S. U. Pillai, T. Suel, and S. Cha. The Perron-Frobenius theorem: some of its applications. *IEEE Signal Processing Magazine*, 22(2):62–75, 2005.

[34] P. Piray and N. D. Daw. Linear reinforcement learning in planning, grid fields, and cognitive control. *Nature Communications*, 12(1):4942, 2021.

[35] D. Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 2000.

[36] D. Precup, R. S. Sutton, and S. Singh. Eligibility traces for off-policy policy evaluation. In *International Conference on Machine Learning (ICML)*, 2000.

[37] T. J. Ringstrom, M. Hasanbeig, and A. Abate. Goal kernel planning: Linearly-solvable non-Markovian policies for logical tasks with goal-conditioned options. *CoRR*, abs/2007.02527, 2025.

[38] G. A. Rummery and M. Niranjan. On-line Q-Learning using connectionist systems. CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994.

[39] H. Sprekeler. On the relation of slow feature analysis and Laplacian eigenmaps. *Neural Computation*, 23(12):3287–3302, 2011.

[40] K. L. Stachenfeld, M. M. Botvinick, and S. J. Gershman. Design principles of the hippocampal cognitive map. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[41] K. L. Stachenfeld, M. M. Botvinick, and S. J. Gershman. The hippocampus as a predictive map. *Nature Neuroscience*, 20(11):1643–1653, 2017.

[42] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3: 9–44, 1988.

[43] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[44] A. Tirinzoni, A. Touati, J. Farebrother, M. Guzek, A. Kanervisto, Y. Xu, A. Lazaric, and M. Pirotta. Zero-shot whole-body humanoid control via behavioral foundation models. In *International Conference on Learning Representations (ICLR)*, 2025.

[45] E. Todorov. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2006.

[46] E. Todorov. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478–11483, 2009.

[47] A. Touati and Y. Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[48] A. Touati, J. Rapin, and Y. Ollivier. Does zero-shot reinforcement learning exist? In *International Conference on Learning Representations (ICLR)*, 2023.

[49] K. Wang, K. Zhou, Q. Zhang, J. Shao, B. Hooi, and J. Feng. Towards better Laplacian representation in reinforcement learning with generalized graph drawing. In *International Conference on Machine Learning (ICML)*, 2021.

[50] K. Wang, K. Zhou, J. Feng, B. Hooi, and X. Wang. Reachability-aware Laplacian representation in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.

[51] C. Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge, 1989.

[52] H. Wiltzer, J. Farebrother, A. Gretton, Y. Tang, A. Barreto, W. Dabney, M. G. Bellemare, and M. Rowland. A distributional analogue to the successor representation. In *International Conference on Machine Learning (ICML)*, 2024.

[53] L. Wiskott and T. J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

[54] Y. Wu, G. Tucker, and O. Nachum. The Laplacian in RL: Learning representations with efficient approximations. In *International Conference on Learning Representations (ICLR)*, 2019.

[55] H. Zang, X. Li, and M. Wang. Simsr: Simple distance-based state representations for deep reinforcement learning. In *AAAI conference on Artificial Intelligence (AAAI)*, 2022.

[56] A. Zhang, R. T. McAllister, R. Calandra, Y. Gal, and S. Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations (ICLR)*, 2021.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction are supported by theoretical or empirical evidence.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in the Conclusion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide complete proofs or derivation for our theoretical results in Appendix B.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide sufficient details of our algorithms and baselines for reproduction in Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide source code in the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide description of experimental settings sufficient for understanding in the main text, and provide additional details, e.g., hyperparameters and hyperparameter search procedures, in Appendix C.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report 95% confidence intervals or visualize all independent runs wherever possible, and discuss statistical significance in Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We discuss compute resources in Appendix E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: As explained in Appendix A, this paper does not have direct societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the creators of the code used in the paper are properly credited, and license information is mentioned.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We follow the NeurIPS code submission guidelines for our submitted code.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs have only been used for editing and formatting purposes.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Broader Impact

This work investigates a novel form of representation in reinforcement learning—a foundational area within the field of artificial intelligence. Given the fundamental and theoretical nature of this research, it does not pose any foreseeable negative societal impacts. As with most projects focused on core scientific inquiry in AI, the associated risks are minimal, and no significant mitigation strategies are required.

B Theory

In this section, we present the detailed derivation of the theoretical results in the main text.

B.1 Derivation of Equations 2 and 7

We present the full derivation of Equations 2 and 7.

B.1.1 Derivation of Equation 7

Equation 7 states that the (s_i, s_j)-th entry of the DR can be written as:

$$\mathbf{Z}(s_i, s_j) = \sum_{\tau \in \mathcal{T}_{s_i \to s_j}} \mathbf{P}^{\pi_d}(\tau) \exp\big(r(\tau)/\lambda\big). \tag{17}$$

Proof. Recall that the analytical form of the DR [34] is $\mathbf{Z} = \Big[\operatorname{diag}\big(\exp(-\mathbf{r}/\lambda)\big) - \mathbf{P}^{\pi_d}\Big]^{-1}$. To simplify notation, let $\mathbf{R} = \operatorname{diag}\big(\exp(-\mathbf{r}/\lambda)\big)$. We then have

$$\begin{aligned}
\mathbf{Z} &= (\mathbf{R} - \mathbf{P}^{\pi_d})^{-1} && (18)\\
(\mathbf{R} - \mathbf{P}^{\pi_d})\mathbf{Z} &= \mathbf{I} && (19)\\
\mathbf{R}\mathbf{Z} - \mathbf{P}^{\pi_d}\mathbf{Z} &= \mathbf{I} && (20)\\
\mathbf{R}\mathbf{Z} &= \mathbf{I} + \mathbf{P}^{\pi_d}\mathbf{Z} && (21)\\
\mathbf{Z} &= \mathbf{R}^{-1} + \mathbf{R}^{-1}\mathbf{P}^{\pi_d}\mathbf{Z}. && (22)
\end{aligned}$$

From the above equation, we know that

$$\mathbf{Z}(s_i, s_j) = \exp\big(r(s_i)/\lambda\big)\mathbb{1}_{\{s_i = s_j\}} + \sum_{s'} \mathbf{P}^{\pi_d}(s_i, s') \exp\big(r(s_i)/\lambda\big)\mathbf{Z}(s', s_j), \tag{23}$$

the right-hand side of which can be repeatedly expanded to give

$$\begin{aligned}
\mathbf{Z}(s_i, s_j) = &\exp\big(r(s_i)/\lambda\big)\mathbb{1}_{\{s_i = s_j\}}\\
&+ \sum_{s'} \mathbf{P}^{\pi_d}(s_i, s') \exp\Big(\big(r(s_i) + r(s')\big)/\lambda\Big)\mathbb{1}_{\{s' = s_j\}}\\
&+ \sum_{s'} \mathbf{P}^{\pi_d}(s_i, s') \sum_{s''} \mathbf{P}^{\pi_d}(s', s'') \exp\Big(\big(r(s_i) + r(s') + r(s'')\big)/\lambda\Big)\mathbb{1}_{\{s'' = s_j\}}\\
&+ \dots.
\end{aligned} \tag{24}$$

In the above equation, it can be seen that the expression enumerates over trajectories from state s_i to state s_j with different lengths. The first row corresponds to a single trajectory with length 1 (containing one state), which is the simple case when the agent starts and ends in s_j without taking an action. The second row enumerates all trajectories from s_i to s_j with length 2, and so on.

The above equation can be written in the following more compact form:

$$\mathbf{Z}(s_i, s_j) = \sum_{\tau \in \mathcal{T}_{s_i \to s_j}} \mathbf{P}^{\pi_d}(\tau) \exp\big(r(\tau)/\lambda\big), \tag{25}$$

where $\mathcal{T}_{s_i \to s_j}$ is the set of trajectories from state s_i to state s_j, $\mathbf{P}^{\pi_d}(\tau)$ is the probability of following trajectory τ under the policy π_d, and $r(\tau)$ is the sum of rewards obtained in trajectory τ. □

B.1.2 Derivation of Equation 2

Equation 2 appeared in the work by Blier et al. [5], but the derivation was not shown. For completeness, we provide the derivation here, which is similar to that for Equation 7. Equation 2 states that the (s_i, s_j)-th entry of the SR can be written as:

$$\mathbf{\Psi}^{\pi}(s_i, s_j) = \sum_{\tau \in \mathcal{T}_{s_i \to s_j}} \mathbf{P}^{\pi}(\tau)\gamma^{\eta(\tau)}. \tag{26}$$

Proof. Recall that the analytical form of the SR [10] is $\mathbf{\Psi}^{\pi} = (\mathbf{I} - \gamma\mathbf{P}^{\pi})^{-1}$. We know that

$$\mathbf{\Psi}^{\pi} = (\mathbf{I} - \gamma\mathbf{P}^{\pi})^{-1} \tag{27}$$

$$(\mathbf{I} - \gamma\mathbf{P}^{\pi})\mathbf{\Psi}^{\pi} = \mathbf{I} \tag{28}$$

$$\mathbf{\Psi}^{\pi} = \mathbf{I} + \gamma\mathbf{P}^{\pi}\mathbf{\Psi}^{\pi}, \tag{29}$$

from which we know that

$$\mathbf{\Psi}^{\pi}(s_i, s_j) = \mathbb{1}_{\{s_i = s_j\}} + \sum_{s'} \mathbf{P}^{\pi}(s_i, s')\gamma\mathbf{\Psi}^{\pi}(s', s_j). \tag{30}$$

Expanding the right-hand side, we have

$$\mathbf{\Psi}^{\pi}(s_i, s_j) = \mathbb{1}_{\{s_i = s_j\}} \tag{31}$$

$$+ \sum_{s'} \mathbf{P}^{\pi}(s_i, s')\gamma\mathbb{1}_{\{s' = s_j\}} \tag{32}$$

$$+ \sum_{s'} \mathbf{P}^{\pi}(s_i, s') \sum_{s''} \mathbf{P}^{\pi}(s', s'')\gamma^2\mathbb{1}_{\{s'' = s_j\}} \tag{33}$$

$$+ \dots, \tag{34}$$

which can be written as

$$\mathbf{\Psi}^{\pi}(s_i, s_j) = \sum_{\tau \in \mathcal{T}_{s_i \to s_j}} \mathbf{P}^{\pi}(\tau)\gamma^{\eta(\tau)}, \tag{35}$$

where $\mathcal{T}_{s_i \to s_j}$ is the set of trajectories from state s_i to state s_j, $\mathbf{P}^{\pi}(\tau)$ is the probability of following trajectory τ under the policy π, and $\eta(\tau)$ is the number of steps it takes to visit s_j from s_i in trajectory τ. Note that when $\tau = (s_i) = (s_j)$, $\eta(\tau) = 0$. □

B.2 Theorem 3.1

Theorem 3.1. *Suppose both the SR and DR are computed with respect to the same policy, i.e., $\pi = \pi_d$. When the reward function is constant and negative, i.e., $r(s) = r(s') < 0\ \forall s, s' \in \mathcal{S}$, the SR and DR share the same set of eigenvectors. Furthermore, when the SR and DR are symmetric, the i-th eigenvectors of the SR and DR are equivalent, and the i-th eigenvalues of the SR ($\mu_{SR,i}$) and DR ($\mu_{DR,i}$) are related as follows:*

$$\mu_{SR,i} = \left[\gamma\Big(\mu_{DR,i}^{-1} - \exp\big(-r(s)/\lambda\big) + \gamma^{-1}\Big)\right]^{-1}, \tag{36}$$

where $\gamma \in (0, 1)$ is the discount factor of the SR, $r(s)$ is the state reward, and λ is the relative importance of the deviation cost of the DR.

Proof. Let $\mu_i, \mathbf{e}_i$ be the i-th eigenvalue and eigenvector of the DR. Assuming that the reward function is constant, and letting $\tilde{r} = \exp\big(-r(s)/\lambda\big)$, the DR can be written as $[\tilde{r}\mathbf{I} - \mathbf{P}^{\pi}]^{-1}$. Then, we have:

$$[\tilde{r}\mathbf{I} - \mathbf{P}^{\pi}]^{-1}\mathbf{e}_i = \mu_i\mathbf{e}_i \tag{37}$$

$$[\tilde{r}\mathbf{I} - \mathbf{P}^{\pi}]\mathbf{e}_i = \mu_i^{-1}\mathbf{e}_i \tag{38}$$

$$-\gamma\mathbf{P}^{\pi}\mathbf{e}_i = \gamma(\mu_i^{-1} - \tilde{r})\mathbf{e}_i \tag{39}$$

$$(\mathbf{I} - \gamma\mathbf{P}^{\pi})\mathbf{e}_i = \gamma(\mu_i^{-1} - \tilde{r} + \gamma^{-1})\mathbf{e}_i \tag{40}$$

$$(\mathbf{I} - \gamma\mathbf{P}^{\pi})^{-1}\mathbf{e}_i = \big[\gamma(\mu_i^{-1} - \tilde{r} + \gamma^{-1})\big]^{-1}\mathbf{e}_i \tag{41}$$

$$\mathbf{\Psi}^{\pi}\mathbf{e}_i = \left[\gamma\Big(\mu_i^{-1} - \exp\big(-r(s)/\lambda\big) + \gamma^{-1}\Big)\right]^{-1}\mathbf{e}_i \tag{42}$$

We have thus shown that the SR and DR share the same set of eigenvectors. When the SR and DR are symmetric, their eigenvalues are real, and we can further show that the orders of the eigenvectors of the SR and DR by their corresponding eigenvalues are identical.

From Eq. 42, we know that the eigenvalues of the SR and DR are related by the following function:

$$\mu_{SR} = f(\mu_{DR}) = [\gamma(\mu_{DR}^{-1} - \exp(-r(s)/\lambda) + \gamma^{-1})]^{-1} \tag{43}$$

Taking the derivative, we have

$$f'(\mu_{DR}) = \gamma^{-1}[1 + \mu_{DR}(\gamma^{-1} - \exp(-r(s)/\lambda))]^{-2}. \tag{44}$$

When $\gamma = \exp(r(s)/\lambda)$, $f'(\mu_{DR})$ is always positive, so the orders of the eigenvectors of the SR and DR are identical. When $\gamma \neq \exp(r(s)/\lambda)$, $f'(\mu_{DR}) > 0$ except at the vertical asymptote at $\mu_{DR} = \frac{-1}{\gamma^{-1} - \exp(-r(s)/\lambda)}$. We now show that all eigenvalues of the DR lie on the same side of the vertical asymptote, and so the orders of eigenvectors are still identical. We do so by first deriving lower and upper bounds on the eigenvalues of the DR.

First, we show that all eigenvalues of the DR are positive. We start with the inverse of the DR, $\mathbf{Z}^{-1} = \text{diag}(\exp(-\mathbf{r}/\lambda)) - \mathbf{P}^{\pi_d}$. Consider a row i of $\mathbf{Z}^{-1}$ corresponding to a non-terminal state. We know that $\mathbf{Z}^{-1}(i,i) = \exp(-r(s)/\lambda) - \mathbf{P}^{\pi_d}(i,i) > 1 - \mathbf{P}^{\pi_d}(i,i) = \sum_{j \neq i} |\mathbf{Z}^{-1}(i,j)|$. For a row i corresponding to a terminal state, $\mathbf{Z}^{-1}(i,i) > \sum_{j \neq i} |\mathbf{Z}^{-1}(i,j)|$ easily holds. Then, by the Gershgorin circle theorem, we know that this matrix has all positive eigenvalues, and thus, all eigenvalues of the DR are positive.

Next, we derive an upper bound on the eigenvalues of the DR. Also applying the Gershgorin circle theorem on the inverse of the DR, we know that the set of eigenvalues of the DR lies within the union of the Gershgorin discs. For the i-th row of the inverse of the DR, if it corresponds to a non-terminal state, we have a disc centered at $\exp(-r(s)/\lambda) - \mathbf{P}^{\pi_d}(i,i)$ with radius $1 - \mathbf{P}^{\pi_d}(i,i)$. For a terminal state, the disc is centered at $\exp(-r(s)/\lambda)$ with radius 0. It is then not hard to see that the minimum of the union of the discs is simply $\exp(-r(s)/\lambda) - 1$. Since the lower bound on the eigenvalues of the inverse of the DR is $\exp(-r(s)/\lambda) - 1$, the upper bound on the eigenvalues of the DR is $\frac{1}{\exp(-r(s)/\lambda)-1}$.

We now know that all eigenvalues of the DR lie in the range $\left(0, \frac{1}{\exp(-r(s)/\lambda)-1}\right]$. We now show that the vertical asymptote of f, if it exists, always lies outside of this range. We focus on the following two cases: 1) $\gamma < \exp(r(s)/\lambda)$, and 2) $\gamma > \exp(r(s)/\lambda)$.

First, when $\gamma < \exp(r(s)/\lambda)$, it can be easily shown that the vertical asymptote $\frac{-1}{\gamma^{-1} - \exp(-r(s)/\lambda)} < 0$. Since all eigenvalues are positive, they all lie on the right hand side of the vertical asymptote.

For the remaining case, we first have that

$$\gamma < 1 \tag{45}$$

$$\gamma^{-1} > 1 \tag{46}$$

$$\gamma^{-1} - \exp(-r(s)/\lambda) > 1 - \exp(-r(s)/\lambda). \tag{47}$$

Since $\gamma > \exp(r(s)/\lambda)$ and we assume $r(s) < 0$, both sides of the inequality are negative. Then,

$$\frac{1}{\gamma^{-1} - \exp(-r(s)/\lambda)} < \frac{1}{1 - \exp(-r(s)/\lambda)} \tag{48}$$

$$\frac{-1}{\gamma^{-1} - \exp(-r(s)/\lambda)} > \frac{1}{\exp(-r(s)/\lambda) - 1}. \tag{49}$$

Since all eigenvalues are less than or equal to $\frac{1}{\exp(-r(s)/\lambda)-1}$, they all lie on the left hand side of the vertical asymptote. We have thus shown that when a vertical asymptote exists, it lies outside of the range of eigenvalues, and so the orders of the eigenvectors of the SR and DR are still identical. □

B.3 Extension of the DR to State-Action-Dependent Reward

In this subsection, we derive the DR for the case when the reward function depends on both the state and action. The extension of linearly solvable MDPs to the state-action case was first presented by

Ringstrom et al. [37]. However, since (1) the DR was not defined in this work, and (2) there are minor differences in the formulation, derivation, and notation, we present the full derivation below nonetheless. Note, also, that the derivation bears similarity to relative entropy policy search [32], which constrains the KL divergence between an observed data distribution and that generated by the policy.

For this setting, instead of formulating the deviation cost as $\mathrm{KL}\left(p^{\pi}(\cdot|S_t)\|p^{\pi_d}(\cdot|S_t)\right)$, we formulate the deviation cost from the default policy as $\mathrm{KL}\left(p^{\pi}(\cdot,\cdot|S_t,A_t)\|p^{\pi_d}(\cdot,\cdot|S_t,A_t)\right)$, where $p^{\pi}(\cdot,\cdot|S_t,A_t)$ denotes the distribution over the next state-action pairs given the current state-action pair (S_t,A_t) and the policy π. Then, at every time step t, the agent receives a reward of $\tilde{r}(S_t,A_t) = r(S_t,A_t) - \lambda\,\mathrm{KL}\left(p^{\pi}(\cdot,\cdot|S_t,A_t)\|p^{\pi_d}(\cdot,\cdot|S_t,A_t)\right)$, where $\lambda > 0$ determines the relative importance of the deviation cost.

Let T be the random variable denoting the final time step of an episode. The return starting from time t is then

$$G_t = \tilde{r}(S_t,A_t) + \cdots + \tilde{r}(S_{T-1},A_{T-1}) + r(S_T,A_T), \tag{50}$$

where we allow the agent to select an action A_t at the terminal state S_t to obtain a final reward $r(S_T,A_T)$.

The state-action value function is defined as $q^{\pi}(s,a) = \mathbb{E}_{\pi}\big[G_t|S_t=s,A_t=a\big]$. For terminal states, $q^{\pi}(s,a)$ is simply $r(s,a)$. For non-terminal states, we have

$$\begin{aligned}
q^{\pi}(s,a) &= \mathbb{E}_{\pi}\big[G_t|S_t=s,A_t=a\big] && (51)\\
&= \mathbb{E}_{\pi}\big[\tilde{r}(S_t,A_t) + \cdots + \tilde{r}(S_{T-1},A_{T-1}) + r(S_T,A_T)|S_t=s,A_t=a\big] && (52)\\
&= \tilde{r}(s,a) + \mathbb{E}_{\pi}\big[\tilde{r}(S_{t+1},A_{t+1}) + \cdots + \tilde{r}(S_{T-1},A_{T-1}) + r(S_T,A_T)|S_t=s,A_t=a\big] && (53)\\
&= \tilde{r}(s,a) + \sum_{s',a'} p^{\pi}(s',a'|s,a)\mathbb{E}_{\pi}\big[\tilde{r}(S_{t+1},A_{t+1}) + \cdots + && (54)\\
&\qquad \tilde{r}(S_{T-1},A_{T-1}) + r(S_T,A_T)|S_t=s,A_t=a,S_{t+1}=s',A_{t+1}=a'\big] && (55)\\
&= \tilde{r}(s,a) + \sum_{s',a'} p^{\pi}(s',a'|s,a)\mathbb{E}_{\pi}\big[\tilde{r}(S_{t+1},A_{t+1}) + \cdots + && (56)\\
&\qquad \tilde{r}(S_{T-1},A_{T-1}) + r(S_T,A_T)|S_{t+1}=s',A_{t+1}=a'\big] && (57)\\
&= \tilde{r}(s,a) + \sum_{s',a'} p^{\pi}(s',a'|s,a)q^{\pi}(s',a') && (58)\\
&= \tilde{r}(s,a) + \mathbb{E}_{s',a'\sim p^{\pi}(\cdot,\cdot|s,a)}\big[q^{\pi}(s',a')\big]. && (59)
\end{aligned}$$

Then, the optimal state-action value function for non-terminal states satisfies

$$\begin{aligned}
q^*(s,a) &= \max_{\pi}\left\{\tilde{r}(s,a) + \mathbb{E}_{s',a'\sim p^{\pi}(\cdot,\cdot|s,a)}\big[q^*(s',a')\big]\right\} && (60)\\
&= \max_{\pi}\left\{r(s,a) - \lambda\,\mathrm{KL}\left(p^{\pi}(\cdot,\cdot|s,a)\|p^{\pi_d}(\cdot,\cdot|s,a)\right) + \mathbb{E}_{s',a'\sim p^{\pi}(\cdot,\cdot|s,a)}\big[q^*(s',a')\big]\right\} && (61)\\
&= r(s,a) + \max_{\pi}\left\{-\lambda\mathbb{E}_{s',a'\sim p^{\pi}(\cdot,\cdot|s,a)}\left[\ln\frac{p^{\pi}(s',a'|s,a)}{p^{\pi_d}(s',a'|s,a)} - \frac{q^*(s',a')}{\lambda}\right]\right\} && (62)\\
&= r(s,a) + \max_{\pi}\left\{-\lambda\mathbb{E}_{s',a'\sim p^{\pi}(\cdot,\cdot|s,a)}\left[\ln\frac{p^{\pi}(s',a'|s,a)}{p^{\pi_d}(s',a'|s,a)\exp\left(q^*(s',a')/\lambda\right)}\right]\right\}. && (63)
\end{aligned}$$

The expectation term above resembles a KL divergence, but the denominator of the fraction is not properly normalized. Define the normalization factor $C = \sum_{s',a'} p^{\pi_d}(s',a'|s,a)\exp\left(q^*(s',a')/\lambda\right)$.

The above equation then becomes

$$q^*(s,a) = r(s,a) + \max_{\pi} \left\{ -\lambda \mathbb{E}_{s',a' \sim p^{\pi}(\cdot,\cdot|s,a)} \left[\ln \frac{p^{\pi}(s',a'|s,a)/C}{p^{\pi_d}(s',a'|s,a) \exp\left(q^*(s',a')/\lambda\right)/C} \right] \right\} \tag{64}$$

$$= r(s,a) + \lambda \ln C + \max_{\pi} \left\{ -\lambda \, \mathrm{KL}\left(p^{\pi}(\cdot,\cdot|s,a) \| p^{\pi_d}(\cdot,\cdot|s,a) \exp\left(q^*(\cdot,\cdot)/\lambda\right)/C \right) \right\} \tag{65}$$

$$= r(s,a) + \lambda \ln C, \tag{66}$$

where the last equality follows from the fact that the minimum of the KL divergence is 0. Note that in order to make the KL divergence 0, the optimal policy needs to satisfy

$$p^{\pi^*}(s',a'|s,a) \propto p^{\pi_d}(s',a'|s,a) \exp\left(q^*(s',a')/\lambda\right) \tag{67}$$

$$\pi^*(a'|s') p(s'|s,a) \propto \pi_d(a'|s') p(s'|s,a) \exp\left(q^*(s',a')/\lambda\right) \tag{68}$$

$$\pi^*(a'|s') \propto \pi_d(a'|s') \exp\left(q^*(s',a')/\lambda\right) \tag{69}$$

Then, given the optimal state-action value function and the default policy, it is straightforward to retrieve the optimal policy, π^*:

$$\pi^*(a|s) = \frac{\pi_d(a|s) \exp\left(q^*(s,a)/\lambda\right)}{\sum_{a'} \pi_d(a'|s) \exp\left(q^*(s,a')/\lambda\right)} \tag{70}$$

Substituting C back to Equation 66, we have

$$q^*(s,a) = r(s,a) + \lambda \ln \sum_{s',a'} p^{\pi_d}(s',a'|s,a) \exp\left(q^*(s',a')/\lambda\right) \tag{71}$$

$$q^*(s,a)/\lambda = r(s,a)/\lambda + \ln \sum_{s',a'} p^{\pi_d}(s',a'|s,a) \exp\left(q^*(s',a')/\lambda\right) \tag{72}$$

$$\exp\left(q^*(s,a)/\lambda\right) = \exp\left(r(s,a)/\lambda\right) \left(\sum_{s',a'} p^{\pi_d}(s',a'|s,a) \exp\left(q^*(s',a')/\lambda\right) \right) \tag{73}$$

We now represent the above equation in matrix form. Let $\bar{\mathbf{q}} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ be the vector of optimal state-action values, $\bar{\mathbf{r}} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ be the vector of state-action rewards, and $\bar{\mathbf{P}}^{\pi_d} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}||\mathcal{A}|}$ be the matrix of transition probabilities between state-action pairs under the default policy, i.e., $\bar{\mathbf{P}}^{\pi_d}(sa, s'a') = p^{\pi_d}(s',a'|s,a)$. Furthermore, let N, T be the set of indices of non-terminal and terminal state-action pairs respectively. The above equation, defined for non-terminal states, can then be written as

$$\exp(\bar{\mathbf{q}}_N/\lambda) = \mathrm{diag}(\exp(\bar{\mathbf{r}}_N/\lambda))(\bar{\mathbf{P}}^{\pi_d}_{NN} \exp(\bar{\mathbf{q}}_N/\lambda) + \bar{\mathbf{P}}^{\pi_d}_{NT} \exp(\bar{\mathbf{q}}_T/\lambda)) \tag{74}$$

$$\mathrm{diag}(\exp(-\bar{\mathbf{r}}_N/\lambda)) \exp(\bar{\mathbf{q}}_N/\lambda) = \bar{\mathbf{P}}^{\pi_d}_{NN} \exp(\bar{\mathbf{q}}_N/\lambda) + \bar{\mathbf{P}}^{\pi_d}_{NT} \exp(\bar{\mathbf{r}}_T/\lambda) \tag{75}$$

$$\left[\mathrm{diag}(\exp(-\bar{\mathbf{r}}_N/\lambda)) - \bar{\mathbf{P}}^{\pi_d}_{NN} \right] \exp(\bar{\mathbf{q}}_N/\lambda) = \bar{\mathbf{P}}^{\pi_d}_{NT} \exp(\bar{\mathbf{r}}_T/\lambda) \tag{76}$$

$$\exp(\bar{\mathbf{q}}_N/\lambda) = \left[\mathrm{diag}(\exp(-\bar{\mathbf{r}}_N/\lambda)) - \bar{\mathbf{P}}^{\pi_d}_{NN} \right]^{-1} \bar{\mathbf{P}}^{\pi_d}_{NT} \exp(\bar{\mathbf{r}}_T/\lambda) \tag{77}$$

The DR for state-action-dependent rewards is then $\left[\mathrm{diag}(\exp(-\bar{\mathbf{r}}_N/\lambda)) - \bar{\mathbf{P}}^{\pi_d}_{NN} \right]^{-1}$ for non-terminal states, and $\left[\mathrm{diag}(\exp(-\bar{\mathbf{r}}/\lambda)) - \bar{\mathbf{P}}^{\pi_d} \right]^{-1}$ for all states.

B.4 Convergence Proof for Dynamic Programming Algorithm

Theorem 4.1. *Let $\mathcal{S}_N$ denote the set of non-terminal states. Assume $r(s) < 0 \; \forall s \in \mathcal{S}_N$. Let $\mathbf{R} = \mathrm{diag}(\exp(-\mathbf{r}/\lambda))$, where $\mathbf{r}$ is the vector of all state rewards. Let $\mathbf{Z}_0 = \mathbf{R}^{-1}$. The update rule*

$$\mathbf{Z}_{k+1} = \mathbf{R}^{-1} + \mathbf{R}^{-1} \mathbf{P}^{\pi_d} \mathbf{Z}_k \tag{78}$$

converges to the DR, that is, $\lim_{k \to \infty} \mathbf{Z}_k = \mathbf{Z}$.

Proof. Recursively expanding the right hand side of Equation 78, we have

$$\mathbf{Z}_{k+1} = \mathbf{R}^{-1} + (\mathbf{R}^{-1}\mathbf{P}^{\pi_d})\mathbf{R}^{-1} + \cdots + (\mathbf{R}^{-1}\mathbf{P}^{\pi_d})^{k+1}\mathbf{R}^{-1} \tag{79}$$

$$= \left[\sum_{t=0}^{k+1}(\mathbf{R}^{-1}\mathbf{P}^{\pi_d})^t\right]\mathbf{R}^{-1}. \tag{80}$$

Taking the limit, we have

$$\lim_{k\to\infty}\mathbf{Z}_k = \left[\sum_{t=0}^{\infty}(\mathbf{R}^{-1}\mathbf{P}^{\pi_d})^t\right]\mathbf{R}^{-1}. \tag{81}$$

Note that $\left[\sum_{t=0}^{\infty}(\mathbf{R}^{-1}\mathbf{P}^{\pi_d})^t\right]$ is a Neumann series. We now establish the convergence of this Neumann series. Since $\sum_{s'}\mathbf{P}^{\pi_d}(s,s')$ is equal to 1 for every non-terminal s, and 0 for all terminal s, we have

$$\|\mathbf{R}^{-1}\mathbf{P}^{\pi_d}\|_\infty = \max_{s\in\mathcal{S}}\exp\big(r(s)/\lambda\big)\sum_{s'}\mathbf{P}^{\pi_d}(s,s') = \max_{s\in\mathcal{S}_N}\exp\big(r(s)/\lambda\big) < 1, \tag{82}$$

where the final inequality follows from our assumption that $r(s) < 0$. Equation 82 is a sufficient condition for the convergence of the Neumann series, so we know that this series converges to $(\mathbf{I} - \mathbf{R}^{-1}\mathbf{P}^{\pi_d})^{-1}$. Plugging this back, we have

$$\lim_{k\to\infty}\mathbf{Z}_k = (\mathbf{I} - \mathbf{R}^{-1}\mathbf{P}^{\pi_d})^{-1}\mathbf{R}^{-1} \tag{83}$$

$$= \big[\mathbf{R}^{-1}(\mathbf{R} - \mathbf{P}^{\pi_d})\big]^{-1}\mathbf{R}^{-1} \tag{84}$$

$$= \big[\mathbf{R}\mathbf{R}^{-1}(\mathbf{R} - \mathbf{P}^{\pi_d})\big]^{-1} \qquad \text{because } \mathbf{B}^{-1}\mathbf{A}^{-1} = (\mathbf{A}\mathbf{B})^{-1} \tag{85}$$

$$= \Big[\operatorname{diag}\big(\exp(-\mathbf{r}/\lambda)\big) - \mathbf{P}^{\pi_d}\Big]^{-1} = \mathbf{Z}. \tag{86}$$

□

B.5 Proto-Representation in Maximum Entropy RL

Similar to how the SR is a proto-representation derived in the standard RL formulation, and the DR in the linearly solvable MDPs, we can derive proto-representations for other formulations. Specifically, due to its popularity, and for completeness, we introduce a proto-representation in the maximum entropy (MaxEnt) RL framework [16, 15]. We call it *maximum entropy representation (MER)*.

In our MaxEnt RL formulation, at each time step t, the agent receives both the reward $r(S_t)$ and an entropy bonus $\lambda\mathcal{H}\big(p^\pi(\cdot|S_t)\big)$, where $p^\pi(\cdot|S_t)$ denotes transition probabilities over successor states given policy π, and $\lambda > 0$ controls the weight of the entropy term. Assuming $\gamma = 1$, we define the MER as:

Definition 4.1. Let $\mathbf{r}$ be the vector of state rewards, and $\mathbf{A}$ be the adjacency matrix, i.e., $A(s,s')$ is equal to 1 if s' can be reached in one step from s, and 0 otherwise. The MER is defined as

$$\mathbf{M} = \Big[\operatorname{diag}\big(\exp(-\mathbf{r})/\lambda\big) - \mathbf{A}\Big]^{-1}. \tag{87}$$

The MER differs from the DR (Equation 4) in that it uses the adjacency matrix rather than the default policy's transition probabilities. This connection arises because maximum entropy RL and linearly solvable MDPs are closely related as one can be transformed into the other [11]. As a result, most of our contributions readily extend to the MER. Preliminary analyses showed that the MER also behaves similarly to the DR, so we focus our experiments on the DR. Still, the MER and DR are distinct proto-representations with different formulations, assumptions, and value functions, which can lead to different optimal policies. While we did not observe substantial differences in the settings we considered, exploring them in other regimes remains an open direction.

We now provide the derivation of the MER. To derive the MER for deterministic transitions, we can simply formulate the entropy as $\mathcal{H}\big(\pi(\cdot|S_t)\big)$. However, to derive the MER for stochastic transitions, we need to formulate the entropy slightly differently. We assume that at every time step t, the

agent receives a reward of $\tilde{r}(S_t) = r(S_t) + \lambda \mathcal{H}\big(p^\pi(\cdot|S_t)\big)$, where $p^\pi(\cdot|S_t)$ denotes the transition probabilities over the successor states of S_t given policy π, and $\lambda > 0$ determines the relative importance of the entropy term.

Let T be a random variable denoting the final time step of an episode. The return starting from time t is then

$$G_t = \tilde{r}(S_t) + \cdots + \tilde{r}(S_{T-1}) + r(S_T). \tag{88}$$

Note that similar to prior work [46, 34], we allow the agent to obtain a final reward of $r(S_T)$ at the terminal state, S_T.

The value function is defined as $v^\pi(s) = \mathbb{E}_\pi\big[G_t|S_t = s\big]$. For terminal states, $v^\pi(s)$ is simply $r(s)$. For non-terminal states, we have

$$\begin{aligned}
v^\pi(s) &= \mathbb{E}_\pi\big[G_t|S_t = s\big] && (89)\\
&= \mathbb{E}_\pi\big[\tilde{r}(S_t) + \cdots + \tilde{r}(S_{T-1}) + r(S_T)|S_t = s\big] && (90)\\
&= \tilde{r}(s) + \sum_{s'} p^\pi(s'|s)\mathbb{E}_\pi\big[\tilde{r}(S_{t+1}) + \cdots + \tilde{r}(S_{T-1}) + r(S_T)|S_t = s, S_{t+1} = s'\big] && (91)\\
&= \tilde{r}(s) + \sum_{s'} p^\pi(s'|s)\mathbb{E}_\pi\big[\tilde{r}(S_{t+1}) + \cdots + \tilde{r}(S_{T-1}) + r(S_T)|S_{t+1} = s'\big] && (92)\\
&= \tilde{r}(s) + \sum_{s'} p^\pi(s'|s)v^\pi(s') && (93)\\
&= \tilde{r}(s) + \mathbb{E}_{s'\sim p^\pi(\cdot|s)}\big[v^\pi(s')\big]. && (94)
\end{aligned}$$

The optimal value function for non-terminal states satisfies

$$\begin{aligned}
v^*(s) &= \max_\pi \Big\{\tilde{r}(s) + \mathbb{E}_{s'\sim p^\pi(\cdot|s)}\big[v^*(s')\big]\Big\} && (95)\\
&= \max_\pi \Big\{r(s) + \lambda\mathcal{H}\big(p^\pi(\cdot|s)\big) + \mathbb{E}_{s'\sim p^\pi(\cdot|s)}\big[v^*(s')\big]\Big\} && (96)\\
&= r(s) + \max_\pi \Big\{ - \lambda\mathbb{E}_{s'\sim p^\pi(\cdot|s)}\big[\ln p^\pi(s'|s)\big] + \mathbb{E}_{s'\sim p^\pi(\cdot|s)}\big[v^*(s')\big]\Big\} && (97)\\
&= r(s) + \max_\pi \left\{-\lambda\mathbb{E}_{s'\sim p^\pi(\cdot|s)}\left[\ln\frac{p^\pi(s'|s)}{\exp(v^*(s')/\lambda)}\right]\right\}. && (98)
\end{aligned}$$

The expectation term above resembles a KL divergence, but $\exp\big(v^*(s')/\lambda\big)$ is not a normalized distribution. To normalize it, we define $C(s) = \sum_{s'\in\mathcal{S}_s}\exp\big(v^*(s')/\lambda\big)$, where $\mathcal{S}_s = \big\{s'|\exists\pi : p^\pi(s'|s) > 0\big\}$ denotes the set of successor states of state s. We then have

$$\begin{aligned}
v^*(s) &= r(s) + \max_\pi \left\{-\lambda\mathbb{E}_{s'\sim p^\pi(\cdot|s)}\left[\ln\frac{p^\pi(s'|s)/C(s)}{\exp\big(v^*(s')/\lambda\big)/C(s)}\right]\right\} && (99)\\
&= r(s) + \max_\pi \left\{-\lambda\mathbb{E}_{s'\sim p^\pi(\cdot|s)}\left[\ln\frac{p^\pi(s'|s)}{\exp\big(v^*(s')/\lambda\big)/C(s)}\right] + \lambda\ln C(s)\right\} && (100)\\
&= r(s) + \lambda\ln C(s) + \max_\pi \Big\{ - \lambda\,\mathrm{KL}\Big(p^\pi(\cdot|s)\|\exp\big(v^*(\cdot)/\lambda\big)/C(s)\Big)\Big\} && (101)\\
&= r(s) + \lambda\ln C(s), && (102)
\end{aligned}$$

where the final equality follows from the fact that the minimum value of KL divergence is 0.

Then, we have that

$$\begin{aligned}
v^*(s) &= r(s) + \lambda\ln\sum_{s'\in\mathcal{S}_s}\exp\big(v^*(s')/\lambda\big) && (103)\\
v^*(s)/\lambda &= r(s)/\lambda + \ln\sum_{s'\in\mathcal{S}_s}\exp\big(v^*(s')/\lambda\big) && (104)\\
\exp\big(v^*(s)/\lambda\big) &= \exp\big(r(s)/\lambda\big)\left(\sum_{s'\in\mathcal{S}_s}\exp\big(v^*(s')/\lambda\big)\right) && (105)
\end{aligned}$$

Now, we express the above equation in a matrix form. Let $\mathbf{v}$ be the vector of optimal state values, $\mathbf{r}$ be the vector of state rewards, and $\mathbf{A}$ be adjacency matrix, i.e. $A(s, s') = \mathbb{1}_{\{s' \in \mathcal{S}_s\}}$. Furthermore, let N, T be the set of indices of non-terminal and terminal states respectively. The above equation, defined for non-terminal states, can be written as

$$\exp(\mathbf{v}_N/\lambda) = \mathrm{diag}(\exp(\mathbf{r}_N)/\lambda)\big(\mathbf{A}_{NN}\exp(\mathbf{v}_N/\lambda) + \mathbf{A}_{NT}\exp(\mathbf{v}_T/\lambda)\big) \tag{106}$$

$$\mathrm{diag}\left(\exp(-\mathbf{r}_N)/\lambda\right)\exp(\mathbf{v}_N/\lambda) = \mathbf{A}_{NN}\exp(\mathbf{v}_N/\lambda) + \mathbf{A}_{NT}\exp(\mathbf{r}_T/\lambda) \tag{107}$$

$$\Big[\mathrm{diag}\left(\exp(-\mathbf{r}_N)/\lambda\right) - \mathbf{A}_{NN}\Big]\exp(\mathbf{v}_N/\lambda) = \mathbf{A}_{NT}\exp(\mathbf{r}_T/\lambda) \tag{108}$$

$$\exp(\mathbf{v}_N/\lambda) = \Big[\mathrm{diag}\left(\exp(-\mathbf{r}_N)/\lambda\right) - \mathbf{A}_{NN}\Big]^{-1}\mathbf{A}_{NT}\exp(\mathbf{r}_T/\lambda) \tag{109}$$

We define the MER for non-terminal states as $\Big[\mathrm{diag}\left(\exp(-\mathbf{r}_N)/\lambda\right) - \mathbf{A}_{NN}\Big]^{-1}$. A more general definition of the MER for all states is $\Big[\mathrm{diag}\left(\exp(-\mathbf{r})/\lambda\right) - \mathbf{A}\Big]^{-1}$.

C More Experiment Details

We provide additional experiment details omitted from the main text due to space constraints.

C.1 Numerical Considerations

Reward shaping and option discovery experiments involve the top eigenvector of the DR. We describe numerical considerations when performing eigendecomposition of the DR. First, in practice, we perform eigendecomposition of the symmetrized DR, $\mathrm{Sym}(\mathbf{Z}) = (\mathbf{Z} + \mathbf{Z}^\top)/2$, to ensure real eigenvalues and eigenvectors. Second, as the DR involves exponentiating negative rewards (see Eq. 7), the magnitude of the DR entries, especially the off-diagonal entries, can become very small. To mitigate the resulting numerical issues, we use the library `python-flint`.[1] Note, however, that the improved precision comes at the cost of increased runtime. As a larger λ has the effect of reducing the magnitude of negative trajectory returns in the DR (see Eq. 7), it can alleviate numerical instabilities. In our experiments, we initially started with $\lambda = 1$, and slowly increased λ by 0.1 until we settled on $\lambda = 1.3$, which, paired with the use of `python-flint`, resulted in no numerical issues.

Finally, the magnitude of the top eigenvector entries can be very small. Therefore, in practice, we use the logarithm of the top eigenvector in place of the top eigenvector. We now show that under mild assumptions, the top eigenvector of the DR is positive, allowing us to take the logarithm. We first present a mild assumption:

Assumption C.1. There is only one start state, s_0, and it is possible to reach any state from s_0 under the default policy, i.e., $\exists \tau_{s_0 \to s} : \mathbf{P}^{\pi_d}(\tau_{s_0 \to s}) > 0$ for all $s \in \mathcal{S}$.

To avoid introducing unnecessary bias, we use the uniform random policy as the default policy in all of our experiments, which is the standard practice [34, 3]. We also perform experiments in grid world environments, where it is possible to reach any state from the start state. Under these conditions, Assumption C.1 easily holds. We now present the following proposition:

Proposition C.2. *Under Assumption C.1, the top eigenvector of* $\mathrm{Sym}(\mathbf{Z})$ *is positive.*

Proof. We can rearrange the rows of $\mathbf{Z}$ so that the first row corresponds to the start state, s_0. By Assumption C.1 and Eq. 7, the first row of $\mathbf{Z}$ has all positive entries, i.e., $\mathbf{Z}(s_0, s) > 0 \; \forall s \in \mathcal{S}$, while all other entries are non-negative. After symmetrization, $\mathrm{Sym}(\mathbf{Z})$ has positive first row and first column. It is not hard to see, then, that $\mathrm{Sym}(\mathbf{Z})^2$ has all positive entries, and $\mathrm{Sym}(\mathbf{Z})$ is a primitive matrix [33]. Finally, by the Perron's theorem [33], the largest eigenvalue of $\mathrm{Sym}(\mathbf{Z})$ is positive, and the corresponding eigenvector is positive. □

Proposition C.2 applies to the case when the DR is computed in closed form. We now extend the proposition to the case when the DR is learned by TD learning. When learning the DR using TD

[1] https://github.com/flintlib/python-flint, MIT license

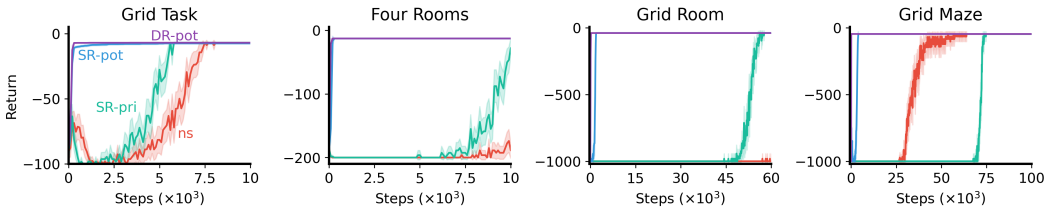

Figure 6: The average undiscounted return of potential-based reward shaping using the DR (`DR-pot`) and the SR (`SR-pot`), the prior distance-based reward shaping using the SR (`SR-prior`) [49], and no shaping (`ns`) over 50 independent runs in the variations of the environments shown in Figure 1 without low-reward regions. The shaded area indicates 95% confidence interval.

learning (Eq. 13), the agent interacts with the environment and learn the DR in an incremental, online manner. Let $\mathcal{S}_V$ be the set of states visited by the agent. In this setting, the top eigenvector refers to the top eigenvector of the sub-matrix of the symmetrized DR corresponding to the states in $\mathcal{S}_V$.

Proposition C.3. *Assume that there is only one start state s_0. Let $\mathcal{D}$ be a dataset containing the transitions collected by an agent starting from s_0 and following the default policy. Importantly, the transitions are stored in the order in which they were collected. Initialize $\mathbf{Z}$ as the identity matrix. When learning the DR by TD learning (Eq. 13) with a step size $\alpha \in (0, 1)$, a backward sweep through $\mathcal{D}$ guarantees that the top eigenvector of $\mathrm{Sym}(\mathbf{Z}_{VV})$ is positive, where $\mathbf{Z}_{VV}$ is the sub-matrix of $\mathbf{Z}$ that corresponds to the states in $\mathcal{S}_V$.*

Proof. We can rearrange the rows of $\mathbf{Z}_{VV}$ so that the first row corresponds to the start state, s_0. The dataset, $\mathcal{D}$, consists of one or more trajectories starting from s_0. When iterating through a trajectory in a backward manner, for the row in $\mathbf{Z}_{VV}$ corresponding to a state, s, all the entries corresponding to states visited after s in the trajectory will be updated to have positive values. It is then not hard to see that after iterating through all transitions, the first row of $\mathbf{Z}_{VV}$ has all positive entries, while the remaining rows are non-negative. After symmetrization, $\mathrm{Sym}(\mathbf{Z}_{VV})$ has positive first row and first column. It is then easy to see that $\mathrm{Sym}(\mathbf{Z}_{VV})$ is a primitive matrix and, by Perron's theorem [33], its top eigenvector is positive. □

For MDPs with an initial distribution, p_0, over multiple start states, we can define a new start state that transitions to these start states under p_0, no matter what action is taken. The reward for this new start state also does not affect the optimal policy, and only needs to be negative (see Theorem 4.1). Proposition C.2 and C.3 can then be applied.

In practice, when learning the DR using TD learning, for example, in RACE (Algorithm 1), we initialize the DR as the identity matrix, and perform at least one backward sweep through the dataset of collected transitions. We then compute the eigendecomposition for $\mathrm{Sym}(\mathbf{Z}_{\mathrm{VV}})$, and take the logarithm of the resulting top eigenvector. To project this transformed eigenvector in $\mathbb{R}^{|\mathcal{S}_V|}$ back to $\mathbb{R}^{|\mathcal{S}|}$, we simply augment this vector with zeros for states in $\mathcal{S} \setminus \mathcal{S}_V$.

C.2 Reward Shaping

Figure 6 shows reward-shaping results in the variations of the environments shown in Figure 1 without low-reward regions. In these environments, we do not observe a significant performance difference between potential-based reward shaping using the DR and the SR. This is because in these environments, all states apart from the terminal state have the same reward. Although Theorem 3.1 cannot be directly applied since the terminal state has a different reward (0) than the rest of the states (-1), it is reasonable to expect that the DR and the SR have similar eigenvectors in this setting, leading to similar results.

C.3 Option Discovery

In Algorithm 1, we present the full algorithm of RACE, an instance of the framework of representation-driven option discovery [ROD; 25] that relies on the DR instead of the SR for option

Algorithm 1 Reward-Aware Covering Eigenoptions (RACE)

Input: α ; ▷ Step size for learning the DR
λ ; ▷ Relative importance of the deviation cost for the DR
N_{learn} ; ▷ Number of iterations through collected dataset to learn the DR
N_{option} ; ▷ Number of latest eigenoptions to keep
p_{option} ; ▷ Probability of sampling an option instead of a primitive action
α_0, γ_0 ; ▷ Step size and discount factor for learning the options' policies
N_{steps} ; ▷ Number of interactions with the environment in each ROD iteration
N_{iter} ; ▷ Number of iterations of the ROD cycle
$\mathcal{D} \longleftarrow \emptyset$
$\Omega \longleftarrow \emptyset$
$\mathbf{Z} \longleftarrow \mathbf{I}$
for $i \longleftarrow 0$ to N_{iter} **do**
▷ Collect samples
$\mathcal{D}_{\text{curr}} \longleftarrow \emptyset$
for $j \longleftarrow 0$ **to** N_{steps} **do**
With prob. $1 - p_{\text{option}}$ randomly sample, uniformly, a primitive action a; otherwise uniformly sample an option ω from Ω
if primitive action was sampled **then**
In state s, take action a and observe state s' and reward r
$\mathcal{D}_{\text{curr}} \longleftarrow \mathcal{D}_{\text{curr}} \| (s, a, r, s')$ ▷ Append transition to current dataset
else
while option ω not terminated **do**
In state s, take action a sampled under the option policy, and observe state s' and reward r
$\mathcal{D}_{\text{curr}} \longleftarrow \mathcal{D}_{\text{curr}} \| (s, a, r, s')$ ▷ Append transition to current dataset
end while
end if
end for
▷ Learn the default representation
Update $\mathbf{Z}$ by N_{learn} sweeps through $\mathcal{D}_{\text{curr}}$ using Equation 13 with step size α and deviation importance λ
▷ Learn eigenoption
$\mathcal{D} \longleftarrow \mathcal{D} \| \mathcal{D}_{\text{curr}}$
Compute the top eigenvector $\mathbf{e}$ of $\mathbf{Z}$
Use Q-learning to compute an eigenoption ω' using $\mathbf{e}$, $\mathcal{D}$, α_0, and γ_0
▷ Add the learned eigenoption to the set of eigenoptions
Add w' to Ω, and remove the oldest option from Ω if $|\Omega| > N_{\text{option}}$
end for

discovery. Note that by learning the SR instead of the DR in Algorithm 1, we recover CEO [25], and therefore omit the corresponding pseudocode for brevity. In this version of RACE, given the complexity of performing importance sampling with options, we treat transitions generated from options the same as those generated from the default policy, and use both for learning the DR.

We share the following hyperparameters for RACE and CEO: $\alpha_0 = 0.1$, $\gamma_0 = 0.99$, $N_{\text{step}} = 100$, $N_{\text{iter}} = 50$ for grid task and four rooms, and $N_{\text{iter}} = 120$ for grid room and grid maze. We use $\gamma = 0.99$ for learning the SR, $\lambda = 1.3$ for learning the DR, and sweep over the remaining hyperparameters, as described in Table 2. We perform 10 independent runs for each hyperparameter setting.

In Figure 4, we used solid dots to highlight the hyperparameter settings that lead to the highest state visitation percentage. We show the learning curves for these hyperparameter settings in Figure 7. We can see that RACE maintains a similar state visitation percentage as CEO, but obtains much higher rewards when exploring the environments. This is because RACE, being reward-aware, avoids low-reward regions when exploring the environments. The random walk, on the other hand, struggles to

Table 2: Hyperparameter search for eigenoption discovery.

Name	Description	Values
p_{option}	Probability of selecting an option	$[0.01, 0.05, 0.1]$
N_{learn}	Number of iterations through the collected dataset to learn the SR/DR	$[1, 10, 100]$
α	Step size for learning the SR/DR	$[0.01, 0.03, 0.1]$
N_{option}	Number of latest options to keep	$[1, 8, 1000]$

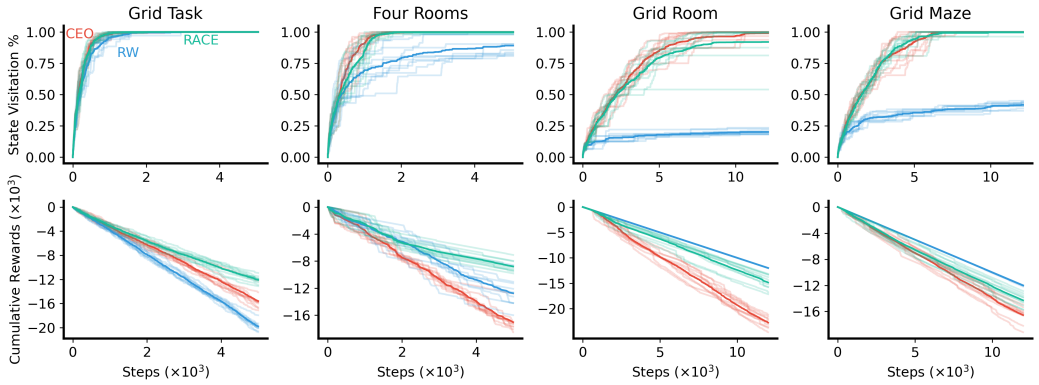

Figure 7: The state visitation percentage (top) and cumulative rewards (bottom) for the highlighted hyperparameter settings in Figure 4 of iterative online eigenoption discovery using the SR (CEO) and the DR (RACE), and the random walk (RW). The solid line shows the average over 10 seeds, while the individual seeds are shown in lighter shade.

explore the state space, especially for the larger environments like grid room and grid maze, and does not encounter low-reward regions, causing it to obtain even higher cumulative rewards than RACE.

Figure 8 shows the state visitation percentage for CEO, RACE, and the random walk in the variations of the environments in Figure 1 without any low-reward regions. As the reward function is constant, the top eigenvectors of the SR and the DR are guaranteed by Theorem 3.1 to be identical. It thus does not come as a surprise that we observe very similar state visitation percentages for CEO and RACE.

We now describe details on combining iterative online eigenoption discovery with Q-learning [51]. Specifically, we use iterative online eigenoption discovery to collect transition data, and then perform offline Q-learning using the collected data. We compare RACE with Q-learning (RACE+Q) with CEO with Q-learning (CEO+Q), and a Q-learning baseline (QL). For the Q-learning baseline, at every iteration, we use the ϵ-greedy policy induced by the current Q-values to collect transitions, and then update the Q-values using the collected transitions.

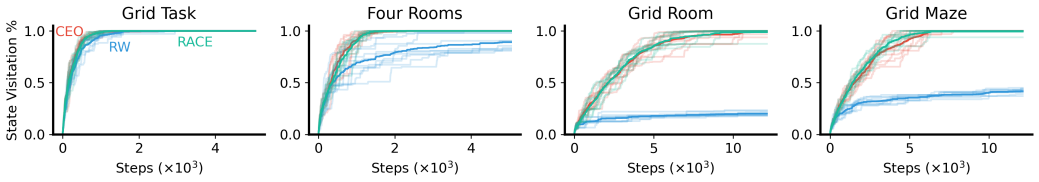

Figure 8: State visitation percentage for CEO, RACE, and the random walk (RW) in the environments shown in Figure 1 without low-reward regions. Shown in lighter shade are the 10 individual seeds.

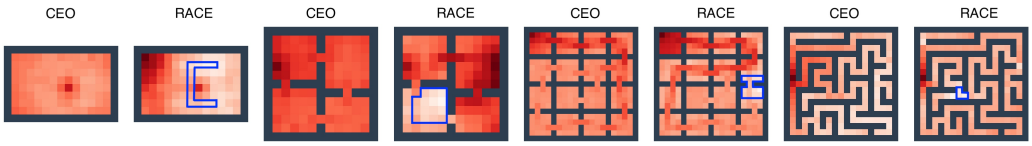

Figure 9: Cumulative state visits for CEO and RACE in the environments from Figure 1 averaged over 10 seeds, where darker red indicates more visits. The low-reward regions are enclosed in blue. Although CEO is also applied to the problem with low-reward regions, we emphasized them only for RACE because CEO does not see it. As indicated by the lighter red, RACE visits low-reward regions much less than CEO. It is especially clear in grid room (third env. from the left) that RACE takes detours to visit the bottom rooms without passing through low-reward regions.

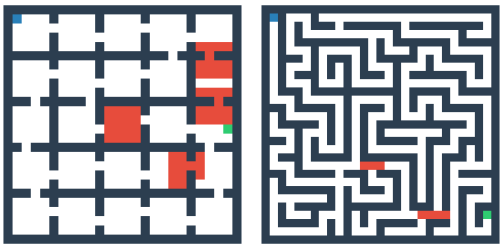

Figure 10: Left: Larger version of Grid Room (Grid Room (L)); Right: Larger version of Grid Maze (Grid Maze (L)), adapted from prior work [50].

For RACE+Q and CEO+Q, we follow the same hyperparameter search procedure as before (see Table 2). For the Q-learning baseline, we perform a grid search over the Q-value initialization ($[-1000, -100, -10, 0]$), ϵ for ϵ-greedy exploration ($[0.01, 0.05, 0.1, 0.15, 0.2]$), and step size ($[0.01, 0.03, 0.1, 0.3, 1]$). We run 10 seeds for each hyperparameter setting, and after identifying the best hyperparameters, re-run 50 seeds to avoid maximization bias. As we do not see much performance difference in simpler environments, we introduce two new environments that are larger versions of Grid Room (Grid Room (L)) and Grid Maze (Grid Maze (L)), shown in Figure 10.

Table 3: Hyperparameters for Count-Based Exploration (Sarsa + DR).

Environment	α	η	β	λ	$\text{transform}(\mathbf{x})$
RIVERSWIM	0.5	0.25	100	1	$\log(\|\mathbf{x}\|_2)$
SIXARMS	0.5	0.01	0.1	1.5	$\log(\|\mathbf{x}\|_2)$

C.4 Count-Based Exploration

To facilitate the learning and application of the DR for exploration in the Riverswim and Sixarms environments, we rescale the environment rewards to lie within the range $[-1, 0]$. Importantly, this rescaling is applied solely for learning the DR; the Sarsa agent continues to operate using the original, unscaled rewards for action-value estimation. Note that we use the DR defined for state-action-dependent rewards, $\bar{\mathbf{Z}}$.

For Sarsa and Sarsa+SR, we adopt the code (MIT license) and hyperparameters specified by the original authors [24]. It is to be noted that while they use $r_{\text{intr}}(s) = \beta \cdot \frac{1}{\|\mathbf{\Psi}^{\pi}_{s,:}\|_1}$ as the intrinsic reward [24], we use $r_{\text{intr}}(s, a) = \beta \cdot \log(\|\bar{\mathbf{Z}}_{sa,:}\|_2)$. We empirically find that the logarithmic transformation enhances performance when leveraging the DR for exploration.

For Sarsa+DR, we sweep over different values of $\eta, \alpha, \beta, \lambda$, with $\eta \in \{0.01, 0.1, 0.25, 0.5\}$, $\alpha \in \{0.01, 0.1, 0.25, 0.5\}$, $\beta \in \{0.1, 1, 10, 100\}$ and $\lambda \in \{1, 1.5, 2\}$. Here, η and α denote the step sizes for updating the Q-values and the DR, respectively. β is the scaling factor for the intrinsic reward and λ controls the relative importance of the deviation cost. We use $\epsilon = 0.01$ for ϵ-greedy exploration. Additionally, we experimented with different transformations of the DR for computing the intrinsic reward, specifically, $\text{transform}(\mathbf{x}) \in \{\|\mathbf{x}\|_1, \|\mathbf{x}\|_2, \log(\|\mathbf{x}\|_1), \log(\|\mathbf{x}\|_2)\}$, where $\mathbf{x}$ is a row of $\bar{\mathbf{Z}}$. The best hyperparameters, shown in Table 3, were evaluated on 100 independent runs.

C.5 Transfer

We describe the details of the features used for transfer learning. We assume that the agent has access to features that can perfectly represent terminal rewards for the DFs, and features that can perfectly represent the reward function for the SFs. For the DR, as there are four terminal states in the environment, and we assume the agent receives the same reward at a terminal state regardless of which action is chosen, we represent the features as a 4-dimensional one-hot vector, where each entry in the vector corresponds to one terminal state.

For the SR, we have to represent the reward function for all states. There are three types of states in the environment: empty state (indicated by white tile), low-reward state (indicated by red tile), and goal state (indicated by green tile). All empty states have the same reward of -1, and all low-reward states have the same reward of -20. Therefore, we represent state features as a 6-dimensional one-hot vector, where one entry is activated when the state is an empty state, one entry is activated when the state is a low-reward state, and the remaining four entries correspond to the four terminal states.

Given the features, we compute the DFs and the SFs under randomly sampled configurations of terminal state rewards, where the reward at each terminal state is sampled independently from a normal distribution with 0 mean and 50 standard deviation. We learn the DFs by following the default policy. While we only need to learn the DFs under the default policy, we can learn the SFs with respect to a set of policies learned under different reward functions. The better the set of reward functions covers the space of reward functions, the better the transfer policy computed using the SFs. We consider the number of reward functions to be 1, 2, 4, and 8. For each randomly sampled reward function, we compute the SFs with respect to the optimal policy learned under this sampled reward function.

In this work, we ensure that the DFs and the SFs are well learned by allowing the agent to interact with the environment for a large number of steps. For the DFs, the agent interacts with the environment while following the default policy for 100K steps. For the SFs, we first use Q-learning [51] to learn an optimal policy for 100K steps. We then learn the SFs while following the optimal policy for another 100K steps. We use $\lambda = 1.3$ and the uniform random policy as the default policy for the DFs, $\gamma = 0.99$ for the SFs, and use a step size of 0.1 for both approaches. We perform 50 independent runs.

Note that the DFs computes an optimal policy in the linearly solvable MDP setting that balances the reward function and the cost of deviating from the default policy. The optimal policies computed by the DFs are then softly biased towards the default policy. We mitigate this bias by approximating the optimal transfer policies (see Eq. 10) using deterministic policies greedy over the optimal Q-values.

D Statistical Significance

We report 95% confidence intervals or directly visualize all independent runs if possible. The 95% confidence intervals capture randomness across independent runs, where the primary source of randomness is random exploration, e.g., ϵ-greedy exploration. In transfer learning experiments, randomness also arises from sampling terminal state reward configurations. As we perform a large number of independent runs in our experiments ($N \geq 50$), we assume the sampling distribution is normal and compute the 95% confidence interval as $\pm 1.96 \frac{\hat{\sigma}}{\sqrt{N}}$, where $\hat{\sigma}$ is the sample standard deviation.

E Compute Resources

We use CPUs for all of our experiments. We describe the runtimes for experiments involving the DR. For reward-shaping experiments, each independent run takes under 10 minutes. For eigenoption discovery experiments, each independent run takes less than 2 hours in grid task and four rooms, and takes around 10 hours in grid room and grid maze. For count-based exploration experiments, each independent run takes less than one minute. For transfer experiments, each independent run takes less than 5 minutes. Due to the large number of independent runs performed for hyperparameter search and preliminary experiments, we estimate the total compute used for the project to be 10.5 CPU core years.