Optimistic Linear Support and Successor Features as a Basis for Optimal Policy Transfer

Lucas N. Alegre [1] **Ana L. C. Bazzan** [1] **Bruno C. da Silva** [2]

Abstract

In many real-world applications, reinforcement learning (RL) agents might have to solve multiple tasks, each one typically modeled via a reward function. If reward functions are expressed linearly, and the agent has previously learned a set of policies for different tasks, *successor features* (SFs) can be exploited to combine such policies and identify reasonable solutions for new problems. However, the identified solutions are not guaranteed to be optimal. We introduce a novel algorithm that addresses this limitation. It allows RL agents to combine existing policies and *directly* identify optimal policies for arbitrary new problems, without requiring any further interactions with the environment. We first show (under mild assumptions) that the transfer learning problem tackled by SFs is equivalent to the problem of learning to optimize multiple objectives in RL. We then introduce an SF-based extension of the *Optimistic Linear Support* algorithm to learn a set of policies whose SFs form a convex coverage set. We prove that policies in this set can be combined via generalized policy improvement to construct optimal behaviors for any new linearly-expressible tasks, without requiring any additional training samples. We empirically show that our method outperforms state-of-the-art competing algorithms both in discrete and continuous domains under value function approximation.

1. Introduction

Reinforcement learning (RL) has been successfully applied to solve many complex problems in a wide range of domains (Silver et al., 2017; Vinyals et al., 2019; Bellemare et al., 2020). However, real-world problems often require optimizing multiple tasks or identifying solutions that optimize possibly conflicting objectives, such as efficiency, energy, and safety. Typically, these tasks or objectives are encoded via separate reward functions. A simple solution to solve problems in this setting is to combine the different objectives of the agent according to some weighting scheme, thereby producing a single scalar reward. This allows for standard RL algorithms to be used, but introduces the question of how to balance the relative importance of each objective (Vamplew et al., 2021). As an alternative, one can identify *multiple* decision-making policies—each one specialized in solving a particular objective or a particular weighted combination of objectives. This is useful because it allows the user of the algorithm to select a policy based on the particular task they are tackling, or based on their current relative preferences over objectives.

Consider an agent that needs to solve multiple tasks whose reward functions can be linearly-expressed (i.e., represented as the weighted sum of reward features and reward weights). In this setting, *successor features* (SFs) (Barreto et al., 2017) have been shown to be a promising approach capable of composing previously-learned policies to solve novel tasks (Barreto et al., 2018; Borsa et al., 2019; Gimelfarb et al., 2021; Nemecek & Parr, 2021). SFs, in particular, allow for the policy evaluation and policy improvement steps (which underlie most RL algorithms) to be extended to the case where the agent needs *(i)* to evaluate a policy on multiple tasks; or *(ii)* to construct a policy (appropriate for solving a novel task) by improving upon an existing set of policies. These processes are known, respectively, as *generalized policy evaluation* (GPE) and *generalized policy improvement* (GPI) (Barreto et al., 2020). This is relevant because, given the weights describing a new reward function (or task), GPI can be used to construct a policy that performs better than the existing, previously-learned ones known to the agent. However, the solutions identified by GPI are *not* guaranteed to be optimal policies. **This leads to an important open problem: how to construct a set of policies, such that combining them directly leads to the optimal policy for *any* novel linearly-expressible tasks?**

[1]Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil [2]University of Massachusetts Amherst, MA. Correspondence to: Lucas N. Alegre <lnalegre@inf.ufrgs.br>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

A closely-related problem to the one of solving multiple tasks is that of simultaneously optimizing multiple objectives. This problem has been extensively studied in the multi-objective RL (MORL) literature (Van Moffaert & Nowé, 2014; Abels et al., 2019; Yang et al., 2019; Hayes et al., 2022). When an agent's relative preferences over objectives can be expressed linearly, an optimal solution corresponds to a *convex coverage set* (CCS). This means, in particular, that given *any* new linear preferences, there exists at least one corresponding optimal policy in such a set. The *Optimistic Linear Support* (OLS) algorithm (Roijers et al., 2015; Mossalam et al., 2016) is a state-of-the-art technique for constructing a CCS by iteratively learning policies specialized in optimizing different linear preferences.

Even though SFs and MORL address related problems (i.e., optimizing multiple tasks, or optimizing multiple objectives), they have typically been investigated independently. **In this paper we show that it is possible to combine and extend theoretical guarantees from each of these optimization frameworks *to directly identify optimal policies for any new tasks*. We refer to this as *optimal policy transfer*.** We first show (under mild assumptions) that the transfer learning problem tackled by SFs is equivalent to the problem of learning policies that solve a multi-objective Markov decision process (MOMDP) under linear preferences. We then introduce an SF-based extension of the OLS algorithm (SFOLS) capable of identifying which tasks to solve so that the SFs corresponding to their policies form a CCS. Importantly, we prove that policies in this set can be combined via GPI to directly construct optimal behaviors for *any* new linearly-expressible tasks, without requiring the agent to collect any additional training samples. Furthermore, we also prove that when an *incomplete* CCS is available, it is possible to bound the gap between an optimal solution and solutions derived via GPI. Additionally, in Appendix A.5 we show that SFOLS can be used to solve a recently-proposed, important open problem in the area of optimal skill discovery in the unsupervised setting (Eysenbach et al., 2022). Even though SFOLS was not designed to this end, it can nonetheless be used to identify optimal solutions to this open problem. Finally, we empirically show that SFOLS outperforms state-of-the-art competing algorithms in the optimal policy transfer setting, both in discrete and continuous domains under value function approximation, as well as in a zero-shot/lifelong learning setting.

2. Background

Here, we discuss important definitions (and corresponding notation) associated with RL, SFs, GPI, and MORL.

2.1. Reinforcement Learning

A RL problem (Sutton & Barto, 2018) is typically modeled as a *Markov decision process* (MDP). An MDP can be defined as a tuple $M \equiv (\mathcal{S}, \mathcal{A}, p, r, \mu, \gamma)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $p(\cdot|s,a)$ describes the distribution over next states given that the agent executed action a in state s, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ is a reward function, μ is an initial state distribution, and $\gamma \in [0,1)$ is a discounting factor. Let S_t, A_t and $R_t = r(S_t, A_t, S_{t+1})$ be random variables corresponding to the state, action, and reward, respectively, at time step t. The goal of the agent is to learn a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ that maximizes the expected discounted sum of rewards (*return*) $G_t = \sum_{i=0}^{\infty} \gamma^i R_{t+i}$. The *action-value function* of a policy π is defined as $q^\pi(s,a) \equiv \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$, where $\mathbb{E}_\pi[\cdot]$ denotes the expectation over trajectories induced by π. Given q^π, one can define a *greedy* policy $\pi'(s) \in \arg\max_a q^\pi(s,a)$. It is guaranteed that $q^{\pi'}(s,a) \geq q^\pi(s,a), \forall (s,a) \in \mathcal{S} \times \mathcal{A}$. The processes of computing q^π and π' are known, respectively, as the *policy evaluation* and *policy improvement* steps. Under certain conditions, repeatedly executing the policy evaluating and improvement steps leads to an optimal policy $\pi^*(s) \in \arg\max_a q^*(s,a)$ (Puterman, 2005).

2.2. Successor Features and GPI

Let $r_{\mathbf{w}}$ be a reward function that can be linearly-expressed as $r_{\mathbf{w}}(s,a,s') = \boldsymbol{\phi}(s,a,s') \cdot \mathbf{w}$, where $\boldsymbol{\phi}(s,a,s') \in \mathbb{R}^d$ are reward features and $\mathbf{w} \in \mathbb{R}^d$ are weights. The features $\boldsymbol{\phi}(s,a,s')$ often represent aspects of the environment that the agent cares about and are related to its objective. Given a policy π, we can define the *successor features* (SFs) $\boldsymbol{\psi}(s,a) \in \mathbb{R}^d$ for a given state-action pair (s,a) as:

$$\boldsymbol{\psi}^\pi(s,a) \equiv \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i \boldsymbol{\phi}_{t+i} | S_t = s, A_t = a \right], \quad (1)$$

where $\boldsymbol{\phi}_t = \boldsymbol{\phi}(S_t, A_t, S_{t+1})$. Given the SFs $\boldsymbol{\psi}^\pi(s,a)$ associated with a particular policy π, it is possible to perform policy evaluation and directly compute the action-value function $q^\pi_{\mathbf{w}}(s,a)$ of π under *any* reward function $r_{\mathbf{w}}$:

$$q^\pi_{\mathbf{w}}(s,a) = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i \boldsymbol{\phi}_{t+i} \cdot \mathbf{w} | S_t = s, A_t = a \right] \quad (2)$$

$$= \boldsymbol{\psi}^\pi(s,a) \cdot \mathbf{w}. \quad (3)$$

Let $\boldsymbol{\psi}^\pi$ be the expected SF vector associated with π, where the expectation is with respect to the initial state distribution: $\boldsymbol{\psi}^\pi = \mathbb{E}_{S_0 \sim \mu}[\boldsymbol{\psi}^\pi(S_0, \pi(S_0))]$. Then, the value of a policy π under any given reward function $\mathbf{w}$ can be expressed as $v^\pi_{\mathbf{w}} = \boldsymbol{\psi}^\pi \cdot \mathbf{w}$. A key insight is that the definition of SFs corresponds to a form of the Bellman equation, where the features $\boldsymbol{\phi}_t$ play the role of rewards. Thus, SFs can be learned through any temporal-difference learning algorithm.

Generalized Policy Evaluation and Improvement. GPI generalizes the policy improvement step (introduced in Section 2.1) by improving a given policy, tasked with solving a particular task, based on a *set* of action-value functions, instead of a single one. Assume the agent has access to a set of previously-learned policies $\Pi = \{\pi_i\}_{i=1}^n$ and their corresponding SFs, $\Psi = \{\boldsymbol{\psi}^{\pi_i}\}_{i=1}^n$. It is possible to evaluate all policies $\pi_i \in \Pi$ under arbitrary reward functions $r_{\mathbf{w}}$, via Eq. (3): $q_{\mathbf{w}}^{\pi_i}(s,a) = \boldsymbol{\psi}^{\pi_i}(s,a) \cdot \mathbf{w}$. This is known as *generalized policy evaluation* (GPE). We define a *GPI policy* as an extension of the standard definition of policy. In particular, it is a *generalized policy* $\pi : \mathcal{S} \times \mathcal{W} \mapsto \mathcal{A}$, defined based on a policy set Π and a weight vector $\mathbf{w}$:

$$\pi^{\text{GPI}}(s;\mathbf{w}) \in \arg\max_{a \in \mathcal{A}} \max_{\pi \in \Pi} q_{\mathbf{w}}^{\pi}(s,a). \tag{4}$$

Let $q_{\mathbf{w}}^{\text{GPI}}(s,a)$ be the action-value function of policy $\pi^{\text{GPI}}(\cdot;\mathbf{w})$. The GPI theorem (Barreto et al., 2017) ensures that $q_{\mathbf{w}}^{\text{GPI}}(s,a) \geq \max_{\pi \in \Pi} q_{\mathbf{w}}^{\pi}(s,a)$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. In other words, Eq. (4) can be used to define a policy that is guaranteed to perform at least as well as any other policies $\pi_i \in \Pi$ in a new task, $\mathbf{w}$. Hence, GPI can be seen as a form of *transfer learning* (Taylor & Stone, 2009). The GPI theorem can be extended to the case where we replace q^{π_i} with an approximation, $\tilde{q}^{\pi_i}$ (Barreto et al., 2018).

2.3. Multi-Objective Reinforcement Learning

The multi-objective RL setting (MORL) is used to model problems where an agent is tasked with optimizing possibly conflicting objectives, each one modeled via a separate reward function. Formally, MORL problem are modeled as multi-objective MDPs (MOMDPs), which differ from regular MDPs in that the reward function is a vector-valued function $\mathbf{r} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}^m$, where m is the number of objectives. Then, the multi-objective action-value function of a given policy π is defined as:

$$\mathbf{q}^{\pi}(s,a) \equiv \mathbb{E}_{\pi}\left[\sum_{i=0}^{\infty} \gamma^i \mathbf{R}_{t+i} | S_t = s, A_t = a\right], \tag{5}$$

where $\mathbf{q}^{\pi}(s,a)$ is a m-dimensional vector with the i-th entry corresponding to the expect return of policy π (given the state-action pair (s,a)) under the i-th objective. Let $\mathbf{v}^{\pi} \in \mathbb{R}^m$ be the multi-objective value of the policy π under the initial state distribution μ: $\mathbf{v}^{\pi} = \mathbb{E}_{S_0 \sim \mu}\left[\mathbf{q}^{\pi}(S_0, \pi(S_0))\right]$, where v_i^{π} is the value of policy π under the i-th objective. Let a *user utility function* (or scalarization function) $u : \mathbb{R}^m \mapsto \mathbb{R}$ be a mapping from the multi-objective value of policy π, $\mathbf{v}^{\pi}$, to a scalar. Utility functions often linearly combine the value of a policy under each of the m objectives using a set of weights $\mathbf{w}$: $u(\mathbf{v}^{\pi}, \mathbf{w}) = \mathbf{v}^{\pi} \cdot \mathbf{w}$, where each element of $\mathbf{w} \in \mathbb{R}^m$ specifies the relative importance of each objective. For any given constant $\mathbf{w}$ (i.e., a particular way of weighting objectives), the original MOMDP collapses into an MDP with reward function $r_{\mathbf{w}}(s,a,s') = \mathbf{r}(s,a,s') \cdot \mathbf{w}$. Let a *Pareto frontier* be a set of nondominated multi-objective value functions $\mathbf{v}^{\pi}$: $\mathcal{F} \equiv \{\mathbf{v}^{\pi} | \nexists \pi' \text{s.t.} \mathbf{v}^{\pi'} \succ_p \mathbf{v}^{\pi}\}$, where $\succ_p$ is the *Pareto dominance relation* $\mathbf{v}^{\pi} \succ_p \mathbf{v}^{\pi'} \iff (\forall i : v_i^{\pi} \geq v_i^{\pi'}) \wedge (\exists i : v_i^{\pi} > v_i^{\pi'})$. We define the optimal solution to a MOMDP as the set of all policies π such that $\mathbf{v}^{\pi}$ is in the Pareto frontier. Given a linear utility function u, we can define a *convex coverage set* (CCS) (Roijers et al., 2013) as a finite convex subset of $\mathcal{F}$, such that there exists a policy in the set that is optimal with respect to any linear preference $\mathbf{w}$. In other words, the CCS is the set of nondominated multi-objective value functions $\mathbf{v}^{\pi}$ where the dominance relation is now defined over scalarized values:

$$\text{CCS} \equiv \{\mathbf{v}^{\pi} \in \mathcal{F} \mid \exists \mathbf{w} \text{ s.t. } \forall \mathbf{v}^{\pi'} \in \mathcal{F}, \mathbf{v}^{\pi} \cdot \mathbf{w} \geq \mathbf{v}^{\pi'} \cdot \mathbf{w}\}, \tag{6}$$

where $\mathbf{w}$ is a vector of weights used by the scalarization function u. The above definition implies that the optimal solution to a MOMDP, under linear preferences, is a finite convex subset of the Pareto frontier.

3. Optimal Policy Transfer

As mentioned in Section 1, a key contribution of this paper is to combine and extend theoretical guarantees from the SFs and MORL literature to construct new methods capable of directly identifying optimal policies for any new tasks. We refer to this as the optimal policy transfer problem. We first formally define such a problem. Then, we demonstrate how to map any transfer learning problem defined within the SFs framework to an equivalent multi-objective problem modeled as a MOMDP under linear utility functions. In Section 4 we derive a principled method with theoretical guarantees for constructing a CCS. These contributions are relevant because, as will be shown, performing GPI over the policies in the CCS is a sufficient condition to ensure that—given any novel linearly-expressible tasks—the resulting policy will be optimal. **We will show that by mapping SF transfer learning problems to equivalent multi-objective problems, and by exploiting properties of GPI and CCS, we can construct a principle algorithm that achieves our goal of performing optimal policy transfer.**

3.1. Problem Formulation

In the SFs literature, transfer learning is defined as the problem of combining existing policies to identify a (typically sub-optimal, but reasonable) policy for a novel task. Let $\mathcal{M}^{\phi}$ be the—possibly infinite—set of MDPs associated with all linearly-expressible reward functions:

$$\mathcal{M}^{\phi} \equiv \{(\mathcal{S}, \mathcal{A}, p, r_{\mathbf{w}}, \mu, \gamma) | r_{\mathbf{w}}(s,a,s') = \boldsymbol{\phi}(s,a,s') \cdot \mathbf{w}\}. \tag{7}$$

Assume an agent has learned a set of policies, Π, for solving some set of tasks, $\mathcal{M} \subset \mathcal{M}^{\phi}$. By using SF and GPI, it is

possible to perform transfer knowledge by composing such policies to construct a new specialized policy for solving a novel new task $M \notin \mathcal{M}$. The question of which set of policies, Π, should be learned by the agent to facilitate transfer learning is an open problem. Our goal is to construct a policy set Π such that the value of the GPI policy π^{GPI}, derived from Π, is as close as possible to the value of the optimal policy for *any* tasks $\mathbf{w} \in \mathcal{W}$. That is,

$$\arg\min_{\Pi} \mathbb{E}_{\mathbf{w}\sim\mathcal{W}} \left[\mathcal{L}(\pi^{\text{GPI}}, \mathbf{w})\right], \tag{8}$$

where the expectation is over tasks $\mathbf{w}$ drawn uniformly at random from the set $\mathcal{W}$ and $\mathcal{L}(\pi, \mathbf{w}) = v^*_{\mathbf{w}} - v^{\pi}_{\mathbf{w}}$, where $v^*_{\mathbf{w}}$ is the value of the optimal policy under a given reward function $r_{\mathbf{w}}$: $v^*_{\mathbf{w}} = \max_\pi \boldsymbol{\psi}^\pi \cdot \mathbf{w}$.

Without loss of generality, we consider weight vectors $\mathbf{w} \in \mathcal{W}$ that induce convex combinations of the features; that is, $\sum_i w_i = 1$ and $w_i \geq 0, \forall i$. This is common practice in the MORL literature (Yang et al., 2019), as it does not alter the optimal policies of the MDPs since optimal policies are invariant with respect to the scale of the rewards.

3.2. Bridging Successor Features and MORL

In this section, we show that any transfer learning problem within the SF framework can be mapped into an equivalent problem of learning multiple policies in MORL. We do so by transforming a set of MDPs, as defined in Eq. (7), into a MOMDP, such that the set of optimal policies for solving all tasks in Eq. (7) is equal to the set of policies that solve the corresponding MOMDP; that is, the policies in the CCS.

Recall that $\boldsymbol{\phi}(s, a, s')$ is a d-dimensional vector containing the d reward features used to construct SFs. We construct a MOMDP with $m{=}d$ objectives such that its m-dimensional reward function, $\mathbf{R}(s, a, s')$, is defined as $\boldsymbol{\phi}(s, a, s')$. That is, for any s, a, s', we define $R_i(s, a, s') \equiv \phi_i(s, a, s')$, where R_i is the reward function associated with the i-th objective of the MOMDP. Let $\mathbf{q}^\pi(s, a)$ be the multi-objective action-value function of the MOMDP. Then,

$$\mathbf{q}^\pi(s, a) \equiv \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i \mathbf{R}_{t+i} | S_t = s, A_t = a\right] \tag{9}$$

$$= \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i \boldsymbol{\phi}_{t+i} | S_t = s, A_t = a\right] \tag{10}$$

$$\equiv \boldsymbol{\psi}^\pi(s, a). \tag{11}$$

Therefore, any algorithms capable of learning SFs $\boldsymbol{\psi}^\pi(s, a)$ can be used to learn the multi-objective action-value $\mathbf{q}^\pi(s, a)$ of a corresponding MOMDP, and vice-versa. Recall that $\boldsymbol{\psi}^\pi$ is the expected SF vector associated with an arbitrary policy π. Under the previously-introduced definition that $\mathbf{R}(s, a, s') = \boldsymbol{\phi}(s, a, s')$, we can show that the multi-objective policy value is equal to the expected SF vector: $\mathbf{v}^\pi \equiv \mathbb{E}_{S_0\sim\mu}[\mathbf{q}^\pi(S_0, \pi(S_0))] = \mathbb{E}_{S_0\sim\mu}[\boldsymbol{\psi}^\pi(S_0, \pi(S_0))] = \boldsymbol{\psi}^\pi$, where the second equality follows from the identity in Eq. (11). Let $\boldsymbol{\psi}^\pi$ be the SF vector associated with any $\mathbf{v}^\pi \in \mathcal{F}$. Then, the original definition of CCS can be rewritten by replacing each occurrence of $\mathbf{v}^\pi \in \mathcal{F}$ with its corresponding $\boldsymbol{\psi}^\pi$:

$$\text{CCS} \equiv \{\mathbf{v}^\pi \in \mathcal{F} \mid \exists \mathbf{w} \text{ s.t. } \forall \mathbf{v}^{\pi'} \in \mathcal{F}, \mathbf{v}^\pi \cdot \mathbf{w} \geq \mathbf{v}^{\pi'} \cdot \mathbf{w}\} \tag{12}$$

$$= \{\boldsymbol{\psi}^\pi \mid \exists \mathbf{w} \text{ s.t. } \forall \boldsymbol{\psi}^{\pi'}, \boldsymbol{\psi}^\pi \cdot \mathbf{w} \geq \boldsymbol{\psi}^{\pi'} \cdot \mathbf{w}\} \tag{13}$$

$$= \{\boldsymbol{\psi}^\pi \mid \exists \mathbf{w} \text{ s.t. } \forall \pi', v^{\pi}_{\mathbf{w}} \geq v^{\pi'}_{\mathbf{w}}\}. \tag{14}$$

Let Π_{CCS} be the set of all policies whose expected SF $\boldsymbol{\psi}^\pi$ is in the CCS. All such policies have the property that their value is greater than the value of any other policies, on at least one task, $\mathbf{w}$. Let $\mathbf{w}_n$ be *any* new task of interest, and let π^*_n be an optimal policy for solving this task. Because π^*_n is optimal, it follows that its value, $v^{\pi^*_n}_{\mathbf{w}}$, is greater than the value of all other policies, π', on task $\mathbf{w}$. That is, $v^{\pi^*_n}_{\mathbf{w}} \geq v^{\pi'}_{\mathbf{w}}$. Thus, by the definition of CCS in Eq. (14), it must be the case that $\boldsymbol{\psi}^{\pi^*_n} \in$ CCS and $\pi^*_n \in \Pi_{\text{CCS}}$. This implies that for *any* novel task, whose linear reward function is represented by weights $\mathbf{w}_n$, a corresponding optimal policy is in Π_{CCS}.

The above realization implies that if one can learn a CCS, it is possible to *directly identify an optimal policy for any linearly-expressible tasks*. Thus, we may reuse algorithms (from the MORL literature) tailored to construct CCS's to derive a set of policies, Π_{CCS}, such that, given any MDP with a linear reward function (that is, MDPs $M \in \mathcal{M}^\phi$), its corresponding optimal policy is in Π_{CCS}. That is, knowledge of a CCS allows the agent to directly identify optimal solutions to any MDPs with linear reward functions.

3.3. Theoretical Results

We now prove that learning a CCS in the form of Eq. 14 allows for the problem defined in Eq. (8) to be solved. In particular, we show that by learning a CCS and performing GPI on the corresponding policy set, Π_{CCS}, one can guarantee optimal performance of the GPI policy on *all* reward weight vectors $\mathbf{w} \in \mathcal{W}$. Finally, we also introduce a bound on the performance of the GPI policy when only a partial CCS is available. The proofs of all lemmas and theorems in this section can be found in Appendix A.

First, we define a weaker strategy for performing policy transfer that is commonly used in MORL settings. Let a *Set Max Policy* (SMP) (Zahavy et al., 2021a) be the best policy in some set Π for a given reward weight vector $\mathbf{w}$:

$$\pi^{\text{SMP}}(s; \mathbf{w}) = \pi'(s), \text{ where } \pi' = \arg\max_{\pi\in\Pi} v^{\pi}_{\mathbf{w}}. \tag{15}$$

Let the value of this policy be $v^{\text{SMP}}_{\mathbf{w}} = \max_{\pi\in\Pi} v^{\pi}_{\mathbf{w}}$. Zahavy et al. (2021a) showed that for any weight vector $\mathbf{w} \in \mathcal{W}$ and policy set Π, it follows that $v^{\text{GPI}}_{\mathbf{w}} \geq v^{\text{SMP}}_{\mathbf{w}}$.

Lemma 3.1. *Let Π be a set of policies and $\mathbf{w}$ an arbitrary weight vector. If an optimal policy for the reward $r_{\mathbf{w}}$ is in Π, then $v^{SMP}_{\mathbf{w}} = v^{*}_{\mathbf{w}}$.*

Theorem 3.2. *Let $\Pi \equiv \{\pi_i\}_{i=1}^{n}$ be a set of policies such that the set of their expected SFs, $\Psi = \{\boldsymbol{\psi}^{\pi_i}\}_{i=1}^{n}$, constitute a CCS (Eq. (14)). Then, given any weight vector $\mathbf{w}\in\mathcal{W}$, the GPI policy $\pi^{GPI}(s;\mathbf{w}) \in \arg\max_{a\in\mathcal{A}} \max_{\pi\in\Pi} q^{\pi}_{\mathbf{w}}(s,a)$ is optimal with respect to $\mathbf{w}$: $v^{GPI}_{\mathbf{w}} = v^{*}_{\mathbf{w}}$.*

Theorem 3.2 shows that learning a CCS in the form of Eq. (14) guarantees optimal behavior when GPI is used to identify a policy for any given task. However, performing GPI also provides improvement upon a set of policies, Π, even when an incomplete CCS is available:

Definition 3.3. A SF set $\Psi = \{\boldsymbol{\psi}^{\pi_i}\}_{i=1}^{n}$ is an ϵ-CCS if

$$\forall \mathbf{w} \in \mathcal{W},\ \max_{\boldsymbol{\psi}^{\pi}\in \text{CCS}} \boldsymbol{\psi}^{\pi}\cdot\mathbf{w} - \max_{\boldsymbol{\psi}^{\pi}\in\Psi} \boldsymbol{\psi}^{\pi}\cdot\mathbf{w} \leq \epsilon \\ \Rightarrow v^{*}_{\mathbf{w}} - v^{\text{SMP}}_{\mathbf{w}} \leq \epsilon.$$

Definition 3.4. Given a SF set $\Psi = \{\boldsymbol{\psi}^{\pi_i}\}_{i=1}^{n}$, the GPI-expanded SF set Ψ^{GPI} is defined as the set

$$\Psi^{\text{GPI}} = \{\boldsymbol{\psi}^{\pi} |\ \pi \in \{\pi^{\text{GPI}}(\cdot;\mathbf{w}) \text{ for all } \mathbf{w}\in\mathcal{W}\}\}.$$

Theorem 3.5. *Let $\Pi = \{\pi^{*}_{i}\}_{i=1}^{n}$ be a set of optimal policies with respect to weights $\{\mathbf{w}_i\}_{i=1}^{n}$, such that their SF set $\Psi = \{\boldsymbol{\psi}^{\pi^{*}_{i}}\}_{i=1}^{n}$ is an ϵ_1-CCS according to Def. (3.3). Let $\boldsymbol{\phi}_{max} = \max_{s,a} ||\boldsymbol{\phi}(s,a)||$. Then, the GPI-expanded SF set Ψ^{GPI} is an ϵ_2-CCS where:*

$$\epsilon_2 \leq \min\{\epsilon_1, \frac{2}{1-\gamma}\boldsymbol{\phi}_{max} \max_{\mathbf{w}\in\mathcal{W}} \min_{i} ||\mathbf{w}-\mathbf{w}_i||\}.$$

Intuitively, Theorem (3.5) implies that if a CCS is incomplete, it is possible to bound the performance gap between the GPI policy (considering an adversarially-chosen task, $\mathbf{w}$) and the optimal policy for that task.

4. Constructing a Set of Policies with Optimistic Linear Support

In this section, we introduce SFOLS, an SF-based extension of the OLS algorithm (Roijers, 2016). It incrementally learns a set of policies, Π, such that their corresponding successor feature set, Ψ, converges to the CCS (Eq. (14)). The policies in Π, at any given iteration of SFOLS, can be combined via GPI to derive an expanded set of solutions Ψ^{GPI}. Because SFOLS monotonically increases the size of Π, and because it converges to the CCS (as discussed later in this section), it follows from Theorem (3.5) that the optimality gap in Eq. (3.5) decreases to zero. In other words, SFOLS is guaranteed to converge to a set of policies, Π, whose combination via GPI is capable of directly produce optimal solutions to any linearly-expressible tasks. Pseudocode for SFOLS is shown in Algorithm 1.

Algorithm 1 SFs Optimistic Linear Support (SFOLS)

1: **Initialize:** $\Pi\leftarrow\{\}$; $\Psi\leftarrow\{\}$; $\mathcal{W}_{exp}\leftarrow\{\}$; $Q\leftarrow\{\}$;
2: **for** each extremum of the weight simplex $\mathbf{w}_e \in \mathcal{W}$ **do**
3: Add $\mathbf{w}_e$ to Q with maximum priority
4: **end for**
5: **repeat**
6: $\mathbf{w} \leftarrow$ pop weight with maximum priority in Q
7: $\pi, \boldsymbol{\psi}^{\pi} \leftarrow$ solve task $(\mathcal{S}, \mathcal{A}, p, r_{\mathbf{w}}, \mu, \gamma)$
8: Add $\mathbf{w}$ to $\mathcal{W}_{exp}$
9: **if** $\boldsymbol{\psi}^{\pi} \notin \Psi$ **then**
10: Remove from Q all $\mathbf{w}'$ s.t. $\boldsymbol{\psi}^{\pi}\cdot\mathbf{w}' > v^{\text{SMP}}_{\mathbf{w}'}$
11: $\mathcal{W}_c \leftarrow \text{CornerWeights}(\boldsymbol{\psi}^{\pi}, \mathbf{w}, \Psi)$
12: Add $\boldsymbol{\psi}^{\pi}$ to Ψ and π to Π
13: **for** $\mathbf{w}' \in \mathcal{W}_c$ **do**
14: $\Delta(\mathbf{w}') \leftarrow \text{EstimateImprovement}(\mathbf{w}', \Psi, \mathcal{W}_{exp})$
15: Add $\mathbf{w}'$ to Q with priority $\Delta(\mathbf{w}')$
16: **end for**
17: **end if**
18: **until** Q is empty
19: **return** Π, Ψ

The algorithm starts by inserting into a priority queue, Q, the weights in the extrema of the weight simplex $\mathcal{W}$ (i.e., weights in which one component is 1 and all others are 0), assigning them maximum priority. SFOLS then iteratively pops the weight $\mathbf{w}$ with the largest priority and uses any RL algorithm to learn a policy, π, for solving task $\mathbf{w}$.[1] SFOLS also computes the SF, $\boldsymbol{\psi}^{\pi}$, induced by π. After computing such a new SF, SFOLS identifies novel weight vectors to add to Q via a procedure that computes *corner weights*, as described below.[2] SFOLS is guaranteed to stop after a finite number of iterations, when Q is empty. This guarantee follows from similar properties discussed by Roijers (2016). Thus, as SFOLS identifies more weight vectors (i.e., tasks) to be processed, it incrementally expands the set Π until this set converges to a complete CCS.

To determine which task SFOLS is going to solve at each iteration, it only examines tasks associated with *corner weights*. Consider the curve describing the value of the SMP policy, $v^{\text{SMP}}_{\mathbf{w}} \equiv \max_{\pi\in\Pi} \boldsymbol{\psi}^{\pi}\cdot\mathbf{w}$, as a function of the task $\mathbf{w}$. Such a curve forms a *piecewise-linear and convex* (PWLC) surface (Roijers, 2016). Corner weights are defined as the points along this surface where it changes

[1] We assume that the agent can *observe* the features $\boldsymbol{\phi}_t$ at the current time step t. Notice that this is different than assuming prior knowledge of the analytic reward feature function $\boldsymbol{\phi}(s,a,s')$. Similar (or more restrictive) assumptions are made in related works (Zahavy et al., 2021a; Alver & Precup, 2022).

[2] In Appendix B.2 we detail how corner weights can be computed given any set of SFs (line 11 of Algorithm 1).

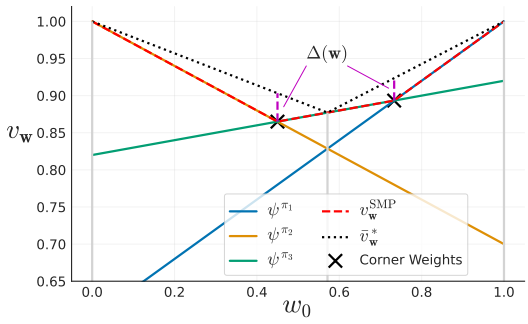

Figure 1. Example of two corner weights for the SF set $\Psi = \{\boldsymbol{\psi}^{\pi_1}, \boldsymbol{\psi}^{\pi_2}, \boldsymbol{\psi}^{\pi_3}\}$. Each curve shows how the value of a given policy changes as a function of the task, $\mathbf{w}$. Here, there are two reward features and thus two weights. We omit $w_1 = 1 - w_0$. Corner weights are the weights in which the value of the SMP policy ($v_{\mathbf{w}}^{\text{SMP}} = \max_{\pi \in \Pi} \boldsymbol{\psi}^{\pi} \cdot \mathbf{w}$) changes slope.

slope. These points can be observed in Figure 1 by analyzing the dashed red curve. Corner weights, here, are denoted by black crosses. SFOLS can safely consider only corner weights due to the following theorem by Cheng (1988):

Theorem 4.1. *(Cheng, 1988) The maximum value of*

$$\max_{\mathbf{w} \in \mathcal{W}, \boldsymbol{\psi}^{\pi} \in \text{CCS}} \min_{\pi' \in \Pi} \boldsymbol{\psi}^{\pi} \cdot \mathbf{w} - \boldsymbol{\psi}^{\pi'} \cdot \mathbf{w} \tag{16}$$

$$= \max_{\mathbf{w} \in \mathcal{W}} v_{\mathbf{w}}^{*} - v_{\mathbf{w}}^{SMP}, \tag{17}$$

is at one of the corner weights of $v_{\mathbf{w}}^{SMP} = \max_{\pi \in \Pi} \boldsymbol{\psi}^{\pi} \cdot \mathbf{w}$.

As a result of Theorem 4.1, corner weights represent tasks whose values under the SMP policy are maximally incorrect with respect to their optimal values. For this reason, they are the best candidate tasks to learn next—intuitively, they are the tasks that the agent knows the least about. Theorem 4.1 guarantees the correctness of OLS and SFOLS: if at a given iteration all corner weights have a maximal improvement of zero, then the algorithm has identified the complete CCS. Furthermore, Theorem 8 of (Roijers, 2016) extends such property to the case when the underlying RL algorithm learns ϵ-optimal policies. In this case, both OLS and SFOLS are guaranteed to produce an ϵ-CCS. Importantly, the number of iterations until SFOLS converges is bounded by $\mathcal{O}(|\text{CCS}| + |\mathcal{W}_{\text{CCS}}|)$, where $|\text{CCS}|$ is the size of the CCS and $|\mathcal{W}_{\text{CCS}}|$ is the number of corner weights.[3]

Theorem 4.1 ensures that the most promising task to practice is located at *one* of the (possibly many) corner weights. A simple heuristic for exploring these tasks is to prioritize them by estimating the difference between $v_{\mathbf{w}}^{\text{SMP}}$, and an optimistic upper bound on that task's optimal value, $\bar{v}_{\mathbf{w}}^{*}$. This is known as the *optimistic maximal improvement*, $\Delta(\mathbf{w}) = \bar{v}_{\mathbf{w}}^{*} - v_{\mathbf{w}}^{\text{SMP}}$, where $\bar{v}_{\mathbf{w}}^{*}$ is an optimistic upper-bound for $v_{\mathbf{w}}^{*}$ (see the black dotted line in Figure 1). Notice that this heuristic only changes the order in which corner weights are explored, but the algorithm still explores all of them in the limit. Therefore, this heuristic does not affect the optimality of SFOLS. The value of $\bar{v}_{\mathbf{w}}^{*}$ can be efficiently computed by the linear program in Algorithm 2 with an off-the-shelf solver (Diamond & Boyd, 2016). Interestingly, the upper-bound $\bar{v}_{\mathbf{w}}^{*}$ is equivalent to the one introduced by Nemecek & Parr (2021), which computes the dual version of the linear program. In Appendix A.4 we prove this equivalence.

[3]Notice, however, that one can also stop the algorithm earlier, when no weights in the priority queue Q have priority greater than a desired optimality threshold ϵ. This results in an ϵ-CCS.

Algorithm 2 Estimate Improvement

1: **Input:** New weight vector $\mathbf{w}$, SFs set Ψ, set of weights, $\mathcal{W}_{exp}$, for which optimal policies are already known.
2: Let $\bar{v}_{\mathbf{w}}^{*}$ be the optimistic upper bound on $v_{\mathbf{w}}^{*}$, computed by the following linear program:

$$\max \boldsymbol{\psi} \cdot \mathbf{w}$$
$$\text{subject to } \boldsymbol{\psi} \cdot \mathbf{w}' \leq v_{\mathbf{w}'}^{\text{SMP}}, \text{ for all } \mathbf{w}' \in \mathcal{W}_{exp}$$

3: $\Delta(\mathbf{w}) \leftarrow \bar{v}_{\mathbf{w}}^{*} - v_{\mathbf{w}}^{\text{SMP}}$
4: **return** $\Delta(\mathbf{w})$

5. Experiments

We compare SFOLS with the Worst Case Policy Iteration (WCPI) algorithm (Zahavy et al., 2021a) and other baselines. WCPI works by iteratively learning a new policy that is optimal for the reward function under which the previously-learned policies perform the worst. This worst-case reward function is defined as $\bar{\mathbf{w}} = \arg\min_{\mathbf{w} \in \mathcal{W}} \max_{\pi \in \Pi} \boldsymbol{\psi}^{\pi} \cdot \mathbf{w}$. WCPI stops when the value of $v_{\bar{\mathbf{w}}}^{\text{SMP}}$ no longer improves between successive iterations. The set of policies learned with WCPI is provably optimal with respect to the worst-case reward of the MDP. Empirically, it has been shown to produce a diverse set of policies with good performance over randomly selected test tasks. Additionally, we compare with a baseline that, at each iteration, learns a policy to solve a randomly-selected task. This is also the approach used in (Nemecek & Parr, 2021). We perform these comparisons in three scenarios: a classic MORL environment and two well-known benchmark environments used in the SFs literature. Additional details can be found in Appendix B.

Deep Sea Treasure (DST). We first compare all methods in the DST environment, a classic MORL domain (Abels et al., 2019; Yang et al., 2019). Here, the agent is a submarine in a 10×11-grid (left panel of Figure 2) that must collect a treasure under a time penalty. The first component of the reward feature vector, $\boldsymbol{\phi}(s, a, s') \in \mathbb{R}^2$, is the treasure value (or zero when in a blank cell), and the second component is always -1. In Figure 2's middle panel, we

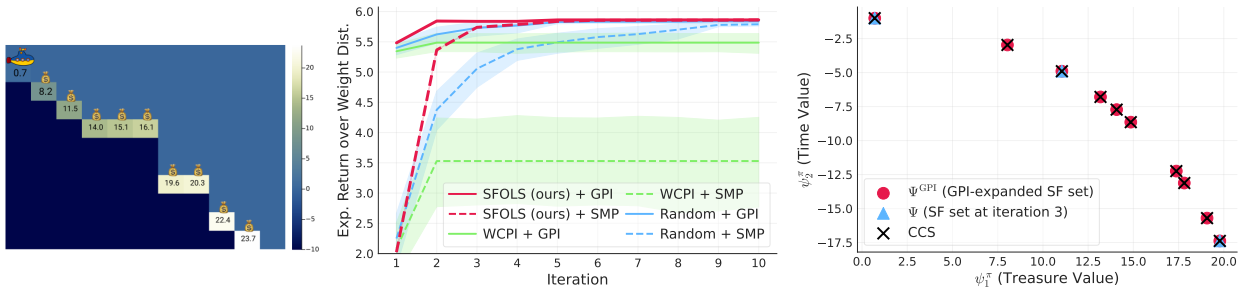

Figure 2. **Left:** DST domain. **Middle:** Expected return of each algorithm over the task/reward weight distribution, $\mathcal{W}$, when evaluated using either GPI or SMP. **Right:** SFOLS recovers the complete CCS (black crosses) by performing GPI over only *three* policies (blue triangles), thereby identifying all other policies in the CSS (red circles) after only three iterations.

show the expected return (value) achieved by each method when evaluated over a test set of 64 tasks uniformly sampled from $\mathcal{W}$. We report the mean value and its 95% confidence interval over 30 random seeds. First, notice that the WCPI algorithm converges to a sub-optimal policy set, as it does not learn any new policies after it solves the task with the lowest optimal value. SFOLS, by contrast, and the random baseline, continuously improve their expected performances by learning new policies.

The rightmost plot of Figure 2 shows how SFOLS is capable of rapidly identifying the complete CCS, whose policies are shown as black crosses. Notice that after three iterations, SFOLS's SF set Ψ contains only three policies (depicted as blue triangles). Even though SFOLS is forced, here, to operated over a reduced amount of experiences/policies, it is already capable of identifying and recovering *all* policies in the CSS. In particular, the policies that SFOLS identifies by performing GPI over the three policies in Ψ are shown as red circles. Notice, then, that our approach succeeds in its main goal of efficiently selecting tasks to practice in a way that (by combining their corresponding policies via GPI) allows the agent to rapidly reconstruct the entire CCS. This emphasizes the potential of using GPI as a method to avoid the costs of explicitly learning a complete CCS—which often contains a large number of policies. Plots depicting SFOLS' performance after different number of iterations can be found in Appendix B.3.

Four Room. Next, we evaluate SFOLS in the Four Room domain (Barreto et al., 2017; Gimelfarb et al., 2021). The Four Room domain has a significantly larger state space than the DST domain. In this task (depicted in the leftmost panel of Figure 3), the reward features are one-hot encoded vectors $\boldsymbol{\phi}(s, a, s') \in \{0, 1\}^3$ indicating the presence of one of three different classes of objects. Since this domain satisfies the *independent features* assumption, as defined by Alver & Precup (2022), we also compare SFOLS with their method: *set of independent policies* (SIP) (Alver & Precup, 2022). The SIP algorithm learns d policies, where each policy is optimal with respect to one task defined by a weight vector in which only one of its d components is a positive value. All others are negative values.

We can observe (in the middle panel of Figure 3) that WCPI constructs a better policy set than the competing methods in the first five iterations. However, after five iterations, it converges to a sub-optimal policy set. The SIP algorithm failed to present good performance when evaluated in the test tasks. We believe this occurs because SIP was designed to maximize the undiscounted total reward, and its performance guarantees do not extend to the discounted return setting. SFOLS, by contrast, keeps improving its expected return over the task distribution after every iteration. Finally, in the rightmost panel of Figure 3 we show the volume under the CCS frontier discovered by SFOLS (its *hypervolume*) as a function of the number of iterations. The hypervolume is a widely-used metric deployed to measure the coverage of the set of solutions over the objectives space (Hayes et al., 2022). In particular, notice that the red curve indicates that the volume of the CCS frontier identified by SFOLS grows rapidly, thus indicating that after only a few iterations, our method is capable of quickly converging to an almost-complete CCS. This emphasizes how GPI, when performed over the solutions identified by SFOLS, can efficiently recover novel policies in settings where a computational budget may restrict the agent's capability of learning many policies.

Reacher. Lastly, we evaluate all algorithms in a setting with a continuous state space that requires using function approximation techniques. We modify the Reacher environment from PyBullet (Ellenberger, 2018–2019), similarly as done in (Barreto et al., 2017; Gimelfarb et al., 2021; Nemecek & Parr, 2021). In this domain, the agent is a robotic arm composed of two segments and that can apply torque to each of its two joints. The reward features $\boldsymbol{\phi}(s, a, s') \in \mathbb{R}^4$ are defined as 1 minus the Euclidean distance from the tip of the

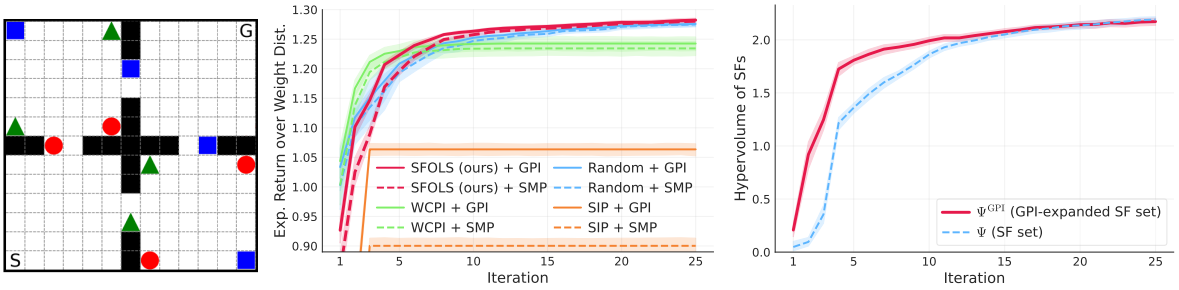

Figure 3. **Left:** Four Room domain. **Middle:** Expected return of each algorithm over the task/reward weight distribution, $\mathcal{W}$, when evaluated using either GPI or SMP. **Right:** Volume under the CCS frontier (*hypervolume*) discovered by SFOLS as a function of iterations. The red curve indicates that SFOLS' hypervolume grows rapidly and that it quickly converges to an almost-complete CCS.

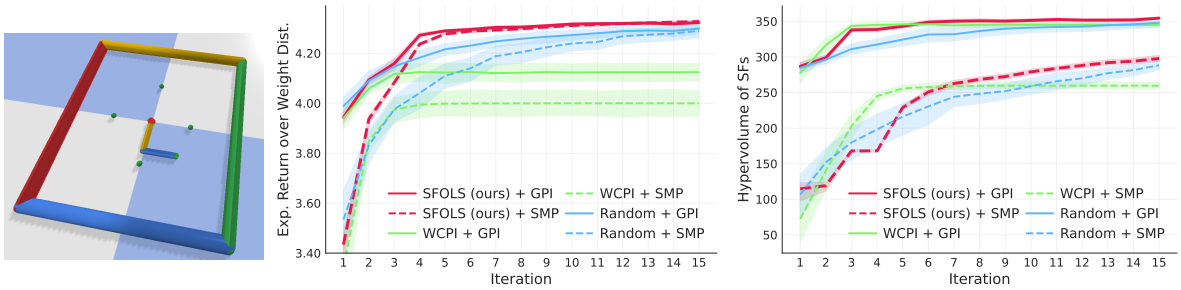

Figure 4. **Left:** Reacher environment. **Middle:** Expected return of each algorithm over the task/reward weight distribution, $\mathcal{W}$, when evaluated using either GPI or SMP. **Right:** Hypervolume identified by each method (SFOLS, WCPI, Random) at each iteration.

arm to four different targets (Figure 4). In this domain, we use neural networks to learn the policies' successor features.

The results in the middle and rightmost panels of Figure 4 show that SFOLS solves the problem after five iterations, while competing methods only approximate (but never reach) the performance of solution after three times more iterations. In this domain, the GPI policy significantly outperforms the SMP policy up until the fifth iteration of SFOLS. WCPI converges to a sub-optimal policy set (with respect to the test weights), while the random baseline requires more iterations than SFOLS to produce a good policy set. The hypervolume metric (shown in the rightmost panel of Figure 4) reveals how GPI generates novel solutions that more efficiently cover the space of solutions. Finally, notice that WCPI has a higher hypervolume during the first six iterations, which shows its capability to quickly create a diverse behavior basis. SFOLS, by contrast, learns policies for the weights/tasks where the optimistic maximal improvement is higher, allowing it to continuously improve its performance over the task distribution.

Lifelong RL. Finally, we consider a lifelong setting (similar to the one described in Alver & Precup (2022)) in which the task being solved by the agent changes with time in an unpredictable manner. In particular, after every 10,000 steps, a new (previously unknown) task $\mathbf{w}$ is uniformly sampled from the task space $\mathcal{W}$. The goal of this experiment is to evaluate the *zero-shot performance* of the GPI solution/behavior resulting from the policy sets constructed by each algorithm, after a pre-training period corresponding to the learning process described previously.[4] Figure 5 shows the zero-shot performance of SFOLS and competing methods, including a baseline a DQN agent (Mnih et al., 2015) which has to continuously re-adapt its action-value function. In this zero-shot learning setting, algorithms that can adapt

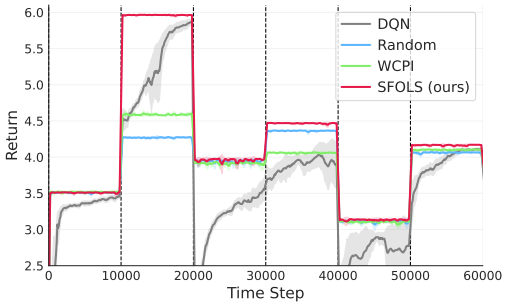

Figure 5. Episodic return of each algorithm in a lifelong setting in which the reward vector $\mathbf{w}$ changes every 10,000 time steps.

[4] That is, after executing each algorithm for 15 iterations in the setting corresponding to the previous experiment; see Figure 4.

more rapidly to novel tasks have better performance (higher return). As can be seen, the GPI solution/behavior obtained by combining the policies learned by SFOLS *always* results in higher or similar returns than the competing algorithms. This is because the policies associated with the CCS constructed by SFOLS better cover the space of tasks, $\mathcal{W}$. As expected, the DQN baseline struggles to adapt to changes to the task being tackled by the agent.

6. Discussion and Related Work

To the best of our knowledge, the survey of Hayes et al. (2022) is the only work that briefly discusses similarities between MORL and SFs. In this paper, we first introduced theoretical results capable of successfully (and formally) combining ideas from both frameworks to derive a novel method that outperforms the state-of-the-art.

The work of Zahavy et al. (2021a) is the closest to ours. They proposed the WCPI algorithm to learn a diverse set of policies. While this algorithm produces a set of policies guaranteed to be optimal with respect to the worst-case reward, there are no guarantees regarding its performance in relation to the entire space of linear rewards. We, on the other hand, show theoretically and empirically that SFOLS *can* construct a set of policies optimal to any linear reward. Zahavy et al. (2021b) studied the problem of learning a set of policies that maximizes a diversity metric while maintaining good performance in a task given by a fixed extrinsic reward. We, by contrast, consider the problem of constructing a policy set that allows for optimal (or near-optimal) behavior in any linearly-expressed task. More recently, Alver & Precup (2022) proposed to construct a set of independent policies as a basis to be used with generalized policy updates. Their method is guaranteed to be optimal only in the undiscounted case. Additionally, it assumes that the MDP's transition function is deterministic, while SFOLS does not have this restriction. Nemecek & Parr (2021) introduced a method that decides whether a policy should be added to the agent's policy library based on an upper-bound computed using the policies' SFs. However, they do not provide a way to decide which tasks the agent should learn at each time, and instead train on randomly sampled ones.

Although prior policy transfer methods, which were not based on SFs, have been proposed (Pickett & Barto, 2002; Fernández & Veloso, 2006; Taylor et al., 2007; Abel et al., 2018), they do not address the problem of constructing a set of policies that allows for optimal behaviors (over arbitrary tasks) to be identified. An exception is the work of Tasse et al. (2020; 2022), which can recover optimal policies for tasks expressible under their proposed Boolean Task Algebra framework. Notice, however, that unlike SFOLS their optimality guarantees only hold for goal-based tasks.

The problem we address in this paper is also related to the one studied in the unsupervised skill discovery literature (Gregor et al., 2016; Eysenbach et al., 2019; Hansen et al., 2020; Liu & Abbeel, 2021). Typically, methods studied in this area use information-theoretic objectives to identify (in an unsupervised manner) policy sets that can be used to enable faster transfer to novel tasks. Recently, Eysenbach et al. (2022) studied this problem under a geometric perspective in which policies are defined by their discounted state occupancy. In this case, each policy is associated with an $|\mathcal{S}|$-dimensional point on the probability simplex. Eysenbach et al. (2022) define the open problem of skill discovery as the problem of identifying the smallest set of policies such that every vertex of the discounted state occupancy polytope contains at least one policy. In the discrete state case, in which reward features $\boldsymbol{\phi}_t$ are a one-hot representation of the agent's state in $\mathbb{R}^{|\mathcal{S}|}$, SFs correspond to the *successor representation* (SR) (Dayan, 1993). Notably, successor representations are also discounted state occupancy measures. In Appendix A.5 we show that SFOLS effectively constructs a CCS over successor representations and thus solves the open problem of skill discovery proposed by Eysenbach et al. (2022).

7. Conclusions

We showed that any transfer learning problem within the SF framework can be mapped into an equivalent problem of learning multiple policies in MORL under linear preferences. We then introduced a novel SF-based extension of the OLS algorithm (SFOLS) to iteratively construct a set of policies whose SFs form a CCS. Additionally, *we showed that these policies can be combined via GPI to directly identify optimal solutions for any novel linearly-expressible tasks.* To the best of our knowledge, ours is the only method capable of formally guaranteeing such a capability. We empirically showed that SFOLS outperforms state-of-the-art competing algorithms both in classic MORL and SF domains. We believe that our theoretical and empirical findings are relevant to both the MORL and transfer learning/SFs communities. As future work, we would like to combine SFOLS with *universal successor features approximators* (USFAs) (Borsa et al., 2019). USFAs may allow us to learn a single model that generalizes over multiple tasks and that better scales to high-dimensional problems.

Acknowledgements

We thank Diederik Roijers for insightful discussions, and the anonymous reviewers for their valuable feedback. This study was financed in part by the following Brazilian agencies: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001; CNPq (grants 140500/2021-9 and 304932/2021-3); and FAPESP/MCTI/CGI (grant number 2020/05165-1).

References

Abel, D., Jinnai, Y., Guo, S. Y., Konidaris, G., and Littman, M. Policy and value transfer in lifelong reinforcement learning. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 20–29. PMLR, 10–15 Jul 2018.

Abels, A., Roijers, D. M., Lenaerts, T., Nowé, A., and Steckelmacher, D. Dynamic weights in multi-objective deep reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *36th International Conference on Machine Learning, ICML 2019*, pp. 11–20. International Machine Learning Society (IMLS), 2019.

Alegre, L. N. MO-Gym: Multi-objective reinforcement learning environments. https://github.com/LucasAlegre/mo-gym, 2022.

Alver, S. and Precup, D. Constructing a good behavior basis for transfer using generalized policy updates. In *Proceedings of the Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=7IWGzQ6gZ1D.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. Successor features for transfer in reinforcement learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D., Zidek, A., and Munos, R. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 501–510, Stockholm, Sweden, jul 2018.

Barreto, A., Hou, S., Borsa, D., Silver, D., and Precup, D. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48):30079–30087, 2020. ISSN 0027-8424. doi: 10.1073/pnas.1907370117.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, Dec 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-2939-8. URL https://doi.org/10.1038/s41586-020-2939-8.

Borsa, D., Barreto, A., Quan, J., Mankowitz, D. J., Munos, R., Hasselt, H. V., Silver, D., and Schaul, T. Universal successor features approximators. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Cheng, H.-T. *Algorithms for partially observable Markov decision processes*. PhD thesis, University of British Columbia, 1988. URL https://open.library.ubc.ca/collections/ubctheses/831/items/1.0098252.

Dayan, P. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993. doi: 10.1162/neco.1993.5.4.613.

Diamond, S. and Boyd, S. CVXPY: A python-embedded modeling language for convex optimization. *J. Mach. Learn. Res.*, 17(1):2909—-2913, jan 2016. ISSN 1532-4435.

Ellenberger, B. PyBullet Gymperium. https://github.com/benelot/pybullet-gym, 2018–2019.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019. OpenReview.net. URL https://openreview.net/forum?id=SJx63jRqFm.

Eysenbach, B., Salakhutdinov, R., and Levine, S. The information geometry of unsupervised reinforcement learning. In *Proceedings of the Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=3wU2UX0voE.

Fernández, F. and Veloso, M. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pp. 720—-727, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933034. doi: 10.1145/1160633.1160762.

Fujimoto, S., Meger, D., and Precup, D. An equivalence between loss functions and non-uniform sampling in experience replay. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 14219–14230. Curran Associates, Inc., 2020.

Gimelfarb, M., Barreto, A., Sanner, S., and Lee, C.-G. Risk-aware transfer in reinforcement learning using successor features. In *Proceedings of the 35th Annual Conference*

on Advances in Neural Information Processing Systems, Online, 2021.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. *CoRR*, abs/1611.07507, 2016. URL http://arxiv.org/abs/1611.07507.

Hansen, S., Dabney, W., Barreto, A., Warde-Farley, D., de Wiele, T. V., and Mnih, V. Fast task inference with variational intrinsic successor features. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Hasselt, H. v., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 2094—-2100. AAAI Press, 2016.

Hayes, C. F., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., Verstraeten, T., Zintgraf, L. M., Dazeley, R., Heintz, F., Howley, E., Irissappane, A. A., Mannion, P., Nowé, A., Ramos, G., Restelli, M., Vamplew, P., and Roijers, D. M. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, Apr 2022. ISSN 1573-7454. doi: 10.1007/s10458-022-09552-y. URL https://doi.org/10.1007/s10458-022-09552-y.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *Proceeding of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA., 2015.

Liu, H. and Abbeel, P. Aps: Active pretraining with successor features. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6736–6747. PMLR, 18–24 Jul 2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, February 2015.

Mossalam, H., Assael, Y. M., Roijers, D. M., and Whiteson, S. Multi-objective deep reinforcement learning. *CoRR*, abs/1610.02707, 2016. URL http://arxiv.org/abs/1610.02707.

Nemecek, M. and Parr, R. Policy caches with successor features. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Pickett, M. and Barto, A. G. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 506–513. Morgan Kaufmann, 2002.

Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley–Interscience, New York, NY, USA, 2005.

Roijers, D. *Multi-Objective Decision-Theoretic Planning*. PhD thesis, University of Amsterdam, 2016.

Roijers, D., Whiteson, S., and Oliehoek, F. Computing convex coverage sets for faster multi-objective coordination. *Journal of Artificial Intelligence Research*, 52:399–443, 3 2015. doi: 10.1613/jair.4550.

Roijers, D. M., Vamplew, P., Whiteson, S., and Dazeley, R. A survey of multi-objective sequential decision-making. *J. Artificial Intelligence Research*, 48(1):67–113, October 2013. ISSN 1076-9757.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *Proceedings of the 33rd International Conference on Learning Representations*, Puerto Rico, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, Oct 2017. ISSN 1476-4687. doi: 10.1038/nature24270. URL https://doi.org/10.1038/nature24270.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. The MIT Press, second edition, 2018.

Tasse, G. N., James, S., and Rosman, B. A boolean task algebra for reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

Tasse, G. N., James, S., and Rosman, B. Generalisation in lifelong reinforcement learning through logical composition. In *Proceedings of the Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ZOcX-eybqoL.

Taylor, M. E. and Stone, P. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(56):1633–1685, 2009.

Taylor, M. E., Whiteson, S., and Stone, P. Transfer via intertask mappings in policy search reinforcement learning. In Durfee, E. H., Yokoo, M., Huhns, M. N., and Shehory, O. (eds.), *6th International Joint Conference on Autonomous*

Agents and Multiagent Systems, pp. 37. IFAAMAS, 2007. doi: 10.1145/1329125.1329170. URL https://doi.org/10.1145/1329125.1329170.

Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., and Dekker, E. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Mach. Learn.*, 84(1–2):51—80, July 2011. ISSN 0885-6125. doi: 10.1007/s10994-010-5232-5.

Vamplew, P., Smith, B. J., Källström, J., de Oliveira Ramos, G., Radulescu, R., Roijers, D. M., Hayes, C. F., Heintz, F., Mannion, P., Libin, P. J. K., Dazeley, R., and Foale, C. Scalar reward is not enough: A response to silver, singh, precup and sutton (2021). *CoRR*, abs/2112.15422, 2021. URL https://arxiv.org/abs/2112.15422.

Van Moffaert, K. and Nowé, A. Multi-objective reinforcement learning using sets of pareto dominating policies. *J. Mach. Learn. Res.*, 15(1):3483–3512, January 2014. ISSN 1532-4435.

Vinyals, O., Babuschkin, I., Czarnecki, W., Mathieu, M., Dudzik, A., Chung, J., Choi, D., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J., Jaderberg, M., and Silver, D. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575:350—-354, November 2019. doi: 10.1038/s41586-019-1724-z.

Yang, R., Sun, X., and Narasimhan, K. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 14610–14621, 2019.

Zahavy, T., Barreto, A., Mankowitz, D. J., Hou, S., O'Donoghue, B., Kemaev, I., and Singh, S. Discovering a set of policies for the worst case reward. In *Proceedings of the 9th International Conference on Learning Representations*, 2021a.

Zahavy, T., O'Donoghue, B., Barreto, A., Mnih, V., Flennerhag, S., and Singh, S. Discovering diverse nearly optimal policies with successor features. *CoRR*, abs/2106.00669, 2021b. URL https://arxiv.org/abs/2106.00669.

A. Proofs

A.1. Proof of Lemma 3.1

Lemma. Let Π be a set of policies and $\mathbf{w}$ an arbitrary weight vector. If an optimal policy for the reward $r_{\mathbf{w}}$ is in Π, then $v^{\text{SMP}}_{\mathbf{w}} = v^*_{\mathbf{w}}$.

Proof.

$$\begin{aligned} v^{\text{SMP}}_{\mathbf{w}} &= \max_{\pi \in \Pi} \boldsymbol{\psi}^{\pi} \cdot \mathbf{w} \\ &= \boldsymbol{\psi}^{\pi^*_{\mathbf{w}}} \cdot \mathbf{w} \\ &= v^*_{\mathbf{w}} \quad \text{(Definition of Optimal Value).} \end{aligned}$$

□

A.2. Proof of Theorem 3.2

Theorem. Let $\Pi \equiv \{\pi_i\}_{i=1}^n$ be a set of policies such that the set of their expected SFs, $\Psi = \{\boldsymbol{\psi}^{\pi_i}\}_{i=1}^n$, constitute a CCS (Eq. (14)). Then, given any weight vector $\mathbf{w} \in \mathcal{W}$, the GPI policy $\pi^{\text{GPI}}(s; \mathbf{w}) \in \arg\max_{a \in \mathcal{A}} \max_{\pi \in \Pi} q^{\pi}_{\mathbf{w}}(s, a)$ is optimal with respect to $\mathbf{w}$: $v^{\text{GPI}}_{\mathbf{w}} = v^*_{\mathbf{w}}$.

Proof. [5]

$$\begin{aligned} v^{\text{GPI}}_{\mathbf{w}}(s) &= q^{\text{GPI}}_{\mathbf{w}}(s, \pi^{\text{GPI}}(s)) \quad \forall s \\ &\geq \max_{\pi \in \Pi, a \in \mathcal{A}} q^{\pi}_{\mathbf{w}}(s, a) \quad \forall s \text{ (GPI Theorem)} \\ &\geq \max_{\pi \in \Pi} v^{\pi}_{\mathbf{w}}(s) \quad \forall s \,. \end{aligned}$$

Taking the expected value with respect to the initial state distribution, μ, on both sides, gives us:

$$\begin{aligned} \mathbb{E}_{S_0 \sim \mu}\left[v^{\text{GPI}}_{\mathbf{w}}(S_0)\right] &\geq \mathbb{E}_{S_0 \sim \mu}\left[\max_{\pi \in \Pi} v^{\pi}_{\mathbf{w}}(S_0)\right] \\ &\geq \max_{\pi \in \Pi} \mathbb{E}_{S_0 \sim \mu}\left[v^{\pi}_{\mathbf{w}}(S_0)\right], \\ v^{\text{GPI}}_{\mathbf{w}} &\geq \max_{\pi \in \Pi} v^{\pi}_{\mathbf{w}} \\ &= v^{\text{SMP}}_{\mathbf{w}} \quad \text{(SMP Definition)} \\ &= v^*_{\mathbf{w}} \quad \text{(Lemma 1 and CCS Definition).} \end{aligned}$$

The last equality comes from the fact that given any weight vector $\mathbf{w} \in \mathcal{W}$, there exists an optimal policy $\pi^*_{\mathbf{w}}$ such that $\boldsymbol{\psi}^{\pi^*_{\mathbf{w}}} \in \text{CCS}$. □

A.3. Proof of Theorem 3.5

Theorem. Let $\Pi = \{\pi^*_i\}_{i=1}^n$ be a set of optimal policies with respect to weights $\{\mathbf{w}_i\}_{i=1}^n$, such that their SF set $\Psi = \{\boldsymbol{\psi}^{\pi^*_i}\}_{i=1}^n$ is an ϵ_1-CCS according to Def. (3.3). Let $\boldsymbol{\phi}_{\max} = \max_{s,a} ||\boldsymbol{\phi}(s, a)||$. Then, the GPI-expanded SF set Ψ^{GPI} is an ϵ_2-CCS where:

$$\epsilon_2 \leq \min\{\epsilon_1, \frac{2}{1-\gamma} \boldsymbol{\phi}_{\max} \max_{\mathbf{w} \in \mathcal{W}} \min_i ||\mathbf{w} - \mathbf{w}_i||\}.$$

Proof. We start proving that $\epsilon_2 \leq \epsilon_1$. From Lemma 2 of Zahavy et al. (2021a), we have that:

$$\begin{aligned} \forall \mathbf{w} \in \mathcal{W}, v^{\text{GPI}}_{\mathbf{w}} \geq v^{\text{SMP}}_{\mathbf{w}}, \\ v^*_{\mathbf{w}} - v^{\text{GPI}}_{\mathbf{w}} \leq v^*_{\mathbf{w}} - v^{\text{SMP}}_{\mathbf{w}}. \end{aligned}$$

[5] This proof follows part of the proof of Lemma 2 of Zahavy et al. (2021a).

Because Ψ is an ϵ_1-CCS, then:

$$\forall \mathbf{w} \in \mathcal{W},\, v^*_{\mathbf{w}} - v^{\text{GPI}}_{\mathbf{w}} \leq v^*_{\mathbf{w}} - v^{\text{SMP}}_{\mathbf{w}} \leq \epsilon_1.$$

Therefore, it must exist an $\epsilon_2 \leq \epsilon_1$ such that:

$$\forall \mathbf{w} \in \mathcal{W},\, v^*_{\mathbf{w}} - v^{\text{GPI}}_{\mathbf{w}} \leq \epsilon_2 \leq \epsilon_1. \tag{18}$$

From Theorem 2 of Barreto et al. (2017), we have that $\forall \mathbf{w} \in \mathcal{W}$ and $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$:

$$q^*_{\mathbf{w}}(s,a) - q^{\text{GPI}}_{\mathbf{w}}(s,a) \leq \frac{2}{1-\gamma} \boldsymbol{\phi}_{\max} \min_i ||\mathbf{w} - \mathbf{w}_i||. \tag{19}$$

Without loss of generality, consider a new state space $\bar{\mathcal{S}} = \mathcal{S} \cup \{\bar{s}\}$, where $\bar{s}$ is a new dummy initial state in which only a single action $\bar{a}$ is available. Let $p(s_0|\bar{s},\bar{a}) = d_0(s)$ for all $s_0 \in \mu$, where $d_0(s_0)$ is the original probability of the initial state being s_0. Notice that this does not change the values of any policy for the states $s \in \mathcal{S}$. Hence:

$$\begin{aligned} q^*_{\mathbf{w}}(\bar{s},\bar{a}) - q^{\text{GPI}}_{\mathbf{w}}(\bar{s},\bar{a}) &= q^*_{\mathbf{w}}(\bar{s},\pi^*(\bar{s})) - q^{\text{GPI}}_{\mathbf{w}}(\bar{s},\pi^{\text{GPI}}(\bar{s})), \\ &= v^*_{\mathbf{w}} - v^{\text{GPI}}_{\mathbf{w}}. \end{aligned}$$

Because (19) holds for any $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$, it holds for $(\bar{s},\bar{a})$, and we have that:

$$\forall \mathbf{w} \in \mathcal{W},\; v^*_{\mathbf{w}} - v^{\text{GPI}}_{\mathbf{w}} \leq \frac{2}{1-\gamma} \boldsymbol{\phi}_{\max} \min_i ||\mathbf{w} - \mathbf{w}_i||.$$

Finally, in the worst case:

$$\forall \mathbf{w} \in \mathcal{W},\; v^*_{\mathbf{w}} - v^{\text{GPI}}_{\mathbf{w}} \leq \epsilon_2 \leq \frac{2}{1-\gamma} \boldsymbol{\phi}_{\max} \max_{\mathbf{w} \in \mathcal{W}} \min_i ||\mathbf{w} - \mathbf{w}_i||. \tag{20}$$

Combining upper-bounds (18) and (20) for ϵ_2 completes the proof. □

A.4. Equivalence of Nemecek & Parr (2021) and SFOLS Upper-Bounds

Given a set of policies $\Pi = \{\pi_i\}_{i=1}^n$ which are optimal, respectively, with respect to the tasks defined by weight vectors $\{\mathbf{w}'_i\}_{i=1}^n$, and given a novel weight vector $\mathbf{w} \in \mathbb{R}^d$, Nemecek & Parr (2021) proposed a method to compute an upper-bound for the optimal value $v^*_{\mathbf{w}}$ by solving the following linear program:

$$\begin{aligned} &\min \sum_{i=1}^n \alpha_i v^{\pi_i}_{\mathbf{w}'_i} \\ &\text{subject to} \sum_{i=1}^n \alpha_i w'_{i,j} = w_j,\; j = 1,...,d \\ &\alpha_i \geq 0,\; i = 1,...,n. \end{aligned}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^n$ is the vector of variables. Let $\boldsymbol{\psi} \in \mathbb{R}^m$ be the dual variables of this linear program. Then, the dual linear program can be expressed as:

$$\begin{aligned} &\max \sum_{j=1}^d \psi_j w_j \\ &\text{subject to} \sum_{j=1}^d \psi_j w'_{i,j} \leq v^{\pi_i}_{\mathbf{w}'_i},\; i = 1,...,n \\ &\psi_j \in \mathbb{R},\; j = 1,...,d. \end{aligned}$$

This is the same linear program used by SFOLS in Algorithm 2, which is adapted from the OLS algorithm (Roijers, 2016), to compute the optimistic upper-bound $\bar{v}^*_{\mathbf{w}}$. Hence, as the linear programs of Nemecek & Parr (2021) and of SFOLS are dual, they share the same optimal value.

A.5. Optimal Unsupervised Skill Discovery

Recently, Eysenbach et al. (2022) studied the problem of unsupervised skill discovery (Gregor et al., 2016; Eysenbach et al., 2019; Hansen et al., 2020; Liu & Abbeel, 2021) under a geometric perspective. They characterize policies by their discounted state occupancy measure, a $|\mathcal{S}|$-dimensional point lying on the probability simplex. Formally, the discounted state occupancy measure of a policy π is defined as:

$$\rho^{\pi}(s) \equiv (1-\gamma)\sum_{t=0}^{\infty}\gamma^{t}P_{t}^{\pi}(s), \tag{21}$$

where $P_t^{\pi}(s)$ is the probability that policy π visits state s at time t. Then, each policy can be represented as an $|\mathcal{S}|$-dimensional point $\boldsymbol{\rho}^{\pi} = [\rho^{\pi}(s_1) \dots \rho^{\pi}(s_{|\mathcal{S}|})]^{\top}$. Since states are assumed to be discrete, any reward function, $r_{\mathbf{w}}$, can be represented as a vector $\mathbf{w} \in \mathbb{R}^{|\mathcal{S}|}$, where $\mathbf{w} = [r_{\mathbf{w}}(s_1) \dots r_{\mathbf{w}}(s_{|\mathcal{S}|})]^{\top}$. Based on this representation, the expected return of a policy, $v_{\mathbf{w}}^{\pi}$, can be expressed as the inner product between its state occupancy measure and the reward vector. That is, $v_{\mathbf{w}}^{\pi} = \boldsymbol{\rho}^{\pi} \cdot \mathbf{w}$. Eysenbach et al. (2022) then introduce two propositions, which we restate here:

Proposition A.1. *(Eysenbach et al., 2022). For every state-dependent reward function, at least one policy that maximizes that reward function lies at a vertex of the discounted state occupancy polytope. That is, given any reward function, at least one of the corresponding optimal policies lies at a vertex of the polytope.*

Proposition A.2. *(Eysenbach et al., 2022). For every vertex $\boldsymbol{\rho}^{\pi}$ of the state occupancy polytope, there exists a reward function for which π is optimal. That is, every vertex of the polytope is associated with the optimal policy of a reward function.*

Next, (Eysenbach et al., 2022) define the following open problem in the unsupervised skill discovery literature, which current state-of-the-art skill learning algorithms are unable to solve:

Definition A.3. *(Vertex discovery problem (Eysenbach et al., 2022)) Given a controlled Markov process (i.e., an MDP without a reward function), find the smallest set of policies such that every vertex of the discounted state occupancy polytope contains at least one policy.*

The importance of this open problem is the following. If solved, it would allow one to identify the smallest set of policies that solve *all* possible tasks that may be defined over a controlled MDP. [6] Intuitively, then, identifying the (finite) set policies at the vertices of the state occupancy polytope allows one to solve *all* possible tasks of interest.

Our key insight is that this problem is equivalent to the problem of finding a set of policies whose state occupancy measures form a CCS:

$$\text{CCS} = \{\boldsymbol{\rho}^{\pi} \mid \exists \mathbf{w} \text{ s.t. } \forall \boldsymbol{\rho}^{\pi'}, \boldsymbol{\rho}^{\pi} \cdot \mathbf{w} \geq \boldsymbol{\rho}^{\pi'} \cdot \mathbf{w}\}. \tag{22}$$

In the discrete state case, if the reward features $\boldsymbol{\phi}_t$ are a one-hot representation of the agent's state in $\mathbb{R}^{|\mathcal{S}|}$, then SFs correspond to the *successor representation* (SR) (Dayan, 1993). Notably, the SR $\boldsymbol{\psi}^{\pi} \in \mathbb{R}^{|\mathcal{S}|}$ is also the discounted state occupancy measure associated with policy π:[7]

$$\begin{aligned}
\boldsymbol{\psi}^{\pi} &\equiv \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}\boldsymbol{\phi}_{t}\right] && (23)\\
&= \sum_{t=0}^{\infty}\gamma^{t}\mathbb{E}_{\pi}\left[\boldsymbol{\phi}_{t}\right] && (24)\\
&= \sum_{t=0}^{\infty}\gamma^{t}[P_{t}^{\pi}(s_1) \dots P_{t}^{\pi}(s_{|\mathcal{S}|})]^{\top} && (25)\\
&= [\rho^{\pi}(s_1) \dots \rho^{\pi}(s_{|\mathcal{S}|})]^{\top} && (26)\\
&\equiv \boldsymbol{\rho}^{\pi}. && (27)
\end{aligned}$$

Hence, SFOLS can be employed to solve the open problem stated by Eysenbach et al. (2022) (Definition A.3).

[6] The need for identifying the smallest set of policies is due to the fact that many tasks may share the same optimal policy.

[7] We omit the normalizing $(1-\gamma)$ term for clarity.

Notice that although SFOLS solves this open problem, it may be challenging—in practice—to use it to identify the optimal set of skills. Recall that in the general case, the number of corner weights grows exponentially with d (the number of objectives) (Roijers, 2016). This is generally feasible since the number of objectives is typically significantly smaller than the number of states. If using SFOLS to tackle the skill-discovery problem, however, d is equal to the number of states. An interesting future direction is to exploit properties particular to the setting proposed by (Eysenbach et al., 2022) in order to reduce the complexity of SFOLS and employ it to identify optimal sets of skills.

B. Experiments Details

The code containing the algorithms and training scripts necessary to reproduce the results are available at `https://github.com/LucasAlegre/sfols`.

B.1. Environments

All environments used in the experiments can be found in the MO-Gym library (Alegre, 2022).

Deep Sea Treasure. The Deep Sea Treasure is a classic MORL environment (Vamplew et al., 2011; Van Moffaert & Nowé, 2014; Abels et al., 2019; Yang et al., 2019). The agent's state at a given time step t is its coordinates in the grid, $S_t = [x, y]$. The action space consists of four directions the agent can move to, $\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}\}$. The first component of the feature/reward vector $\boldsymbol{\phi}(s, a, s') \in \mathbb{R}^2$ is the treasure value[8] (or zero if the agent is in a blank cell), as shown in the left of Figure 2, and the second component is a time penalty of -1 in all states. The cells with treasures are also terminal states. We considered a discount factor of $\gamma = 0.99$ in this domain. There are ten different optimal values in this domain's CCS, each corresponding to a policy that reaches one of the ten treasures in the map.

Four Room. The Four Room domain (Barreto et al., 2017; Gimelfarb et al., 2021) is defined by a grid of dimensions 13×13 containing four rooms separated by walls. Each time step t, the agent occupies a cell and can move to one of the four directions $\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}\}$. If the destination cell is a wall, then the agent remains in its current cell. The grid contains 3 different types of objects the agent can pick up, as shown in the left of Figure 3. There are 4 instances of each type of object in the grid. The state space consists of the concatenation of the agent's current x-y coordinates and a set of binary variables indicating whether or not each object has already been picked up: $\mathcal{S} = \{0, .., 12\}^2 \times \{0, 1\}^{12}$. The features $\boldsymbol{\phi}(s, a, s') \in \{0, 1\}^3$ are one-hot encoded vectors indicating the type of object present in the current cell. If there are no objects in the cell, the features are zeroed. A special case is the goal cell at the upper-right of the map, which has all features activated and terminates the episode. We considered a discount factor of $\gamma = 0.95$ in this domain.

Reacher. We adapt the Reacher environment from PyBullet (Ellenberger, 2018–2019), as done in Barreto et al. (2017); Gimelfarb et al. (2021); Nemecek & Parr (2021).[9] The agent's state space $\mathcal{S} \subset \mathbb{R}^4$ consists of the angles and angular velocities of the robotic arm's two joints. The agent's initial state is the one shown in the left of Figure 4. The action space, originally continuous, is discretized using 3 values per dimension corresponding to maximum positive (+1), negative (-1), and zero torque for each actuator. This results in a total of 9 possible actions: $\mathcal{A} = \{-1, 0, +1\}^2$. Each feature $\boldsymbol{\phi}(s, a, s') \in \mathbb{R}^4$ is computed as $\phi_i(s, a, s') = 1 - 4\Delta(\text{target}_i), i = 1...4$, where $\Delta(\text{target}_i)$ is the Euclidean distance of the tip of the robotic arm to the i-th target position. We considered a discount factor of $\gamma = 0.9$ in this domain.

B.2. Computing Corner Weights

Computing the corner weights (line 11 of Algorithm 1) is an important step of SFOLS. In Algorithm B.2 we present the algorithm used to compute the corner weights efficiently in each iteration. For more details on each of these steps, see Chapter 3 of Roijers (2016).

B.3. Additional Results

In Figure 6, we show the SF set, Ψ, and the GPI-expanded SF set , Ψ^{GPI}, computed at each iteration of SFOLS in the DST environment. In the upper-left panel, we also show the variation of the hypervolume metric (Hayes et al., 2022) at each

[8] We adopted the treasures values as defined in Yang et al. (2019).

[9] The code for this environment and Figure 4 (left) were adapted from Gimelfarb et al. (2021).

Algorithm 3 Corner Weights (Roijers, 2016)

Input: New SF vector $\boldsymbol{\psi}^{\pi}$, current weight vector $\mathbf{w}$, current SF set Ψ.
Let $\mathcal{W}_{del}$ be the set of obsolete weights removed from Q in line 10 of Algorithm 1
Add $\mathbf{w}$ to $\mathcal{W}_{del}$
$\mathcal{V}_{rel} \leftarrow \{\boldsymbol{\psi}^{\pi} | \boldsymbol{\psi}^{\pi} \in \arg\max_{\boldsymbol{\psi}^{\pi} \in \Psi} \boldsymbol{\psi}^{\pi} \cdot \mathbf{w}' \text{ for at least one } \mathbf{w}' \in \mathcal{W}_{del}\}$
$\mathcal{B}_{rel} \leftarrow$ the set of boundaries of the weight simplex $\mathcal{W}$ involved in any $\mathbf{w}' \in \mathcal{W}_{del}$
$\mathcal{W}_c \leftarrow \{\}$
for each subset $\mathcal{X}$ of $d-1$ elements from $\mathcal{V}_{rel} \cup \mathcal{B}_{rel}$ **do**
    $\mathbf{w}_c \leftarrow$ the weight in $\mathcal{W}$ where $\boldsymbol{\psi}^{\pi}$ intersects with the vectors/boundaries in $\mathcal{X}$
    Add $\mathbf{w}_c$ to $\mathcal{W}_c$
end for
return $\mathcal{W}_c$

iteration. Given a partial CCS, Ψ, and a reference point $\boldsymbol{\psi}_{\text{ref}}$, the hypervolume is defined as:

$$\text{hypervolume}(\Psi, \boldsymbol{\psi}_{\text{ref}}) = \bigcup_{\boldsymbol{\psi}^{\pi} \in \Psi} \text{volume}(\boldsymbol{\psi}_{\text{ref}}, \boldsymbol{\psi}^{\pi}), \tag{28}$$

where $\text{volume}(\boldsymbol{\psi}_{\text{ref}}, \boldsymbol{\psi}^{\pi})$ is the volume of the hypercube spanned by the reference vector, $\boldsymbol{\psi}_{\text{ref}}$, and the vector $\boldsymbol{\psi}^{\pi}$. Notice that the hypervolume does not necessarily correlate with the mean value achieved over the space of reward weights $\mathcal{W}$. For instance, a set of diverse sub-optimal policies with low mean value can still have a high value of the hypervolume metric.

Notice that, from iteration 3 onward, SFOLS+GPI reaches optimal performance for every weight vector, that is, $\Psi^{\text{GPI}} = \text{CCS}$. Meanwhile, the SF set only recovers the complete CCS at iteration 13.

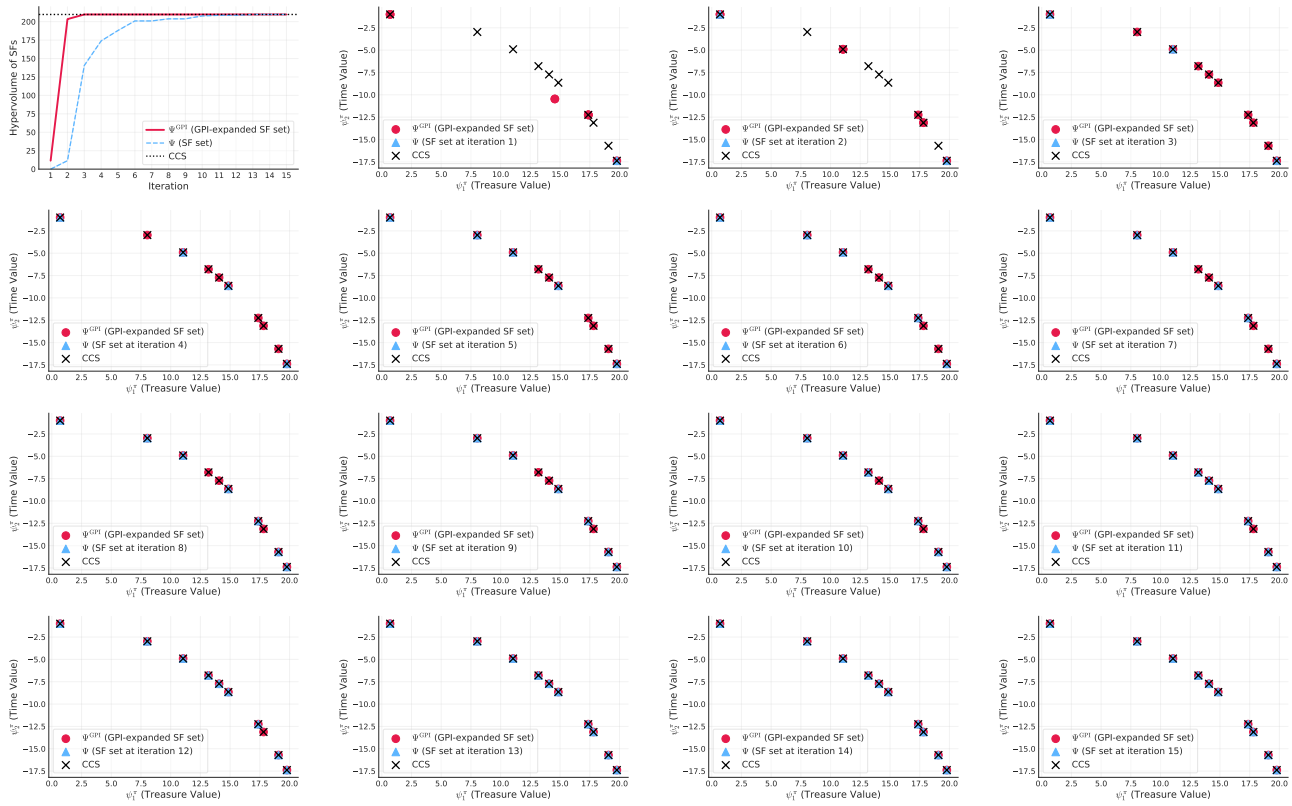

Figure 6. Hypervolume metric and the SF and GPI-expanded SF sets at each iteration of SFOLS in the DST domain. The reference point for the hypervolume calculation is $\boldsymbol{\psi}_{\text{ref}} = [0.0, -17.383]$.

In order to provide a clearer visualization of the performance differences between each algorithm, in Figure 7 we show the curves, previously depicted in Figure 2, separately for the SMP and GPI policies. Similarly, in Figure 8 we separately zoom in on the curves previously depicted in Figure 3. Notice that SFOLS performance has significantly lower variance than the competing algorithms. This is because WCPI and the random baseline depend on random initializations of the reward vectors, while SFOLS does not.

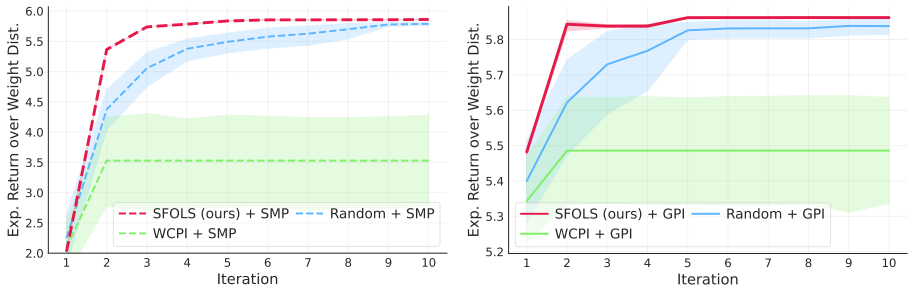

Figure 7. Expected return of each algorithm over the task distribution, $\mathcal{W}$, when evaluated using either SMP (**left**) or GPI (**right**) in the DST domain.

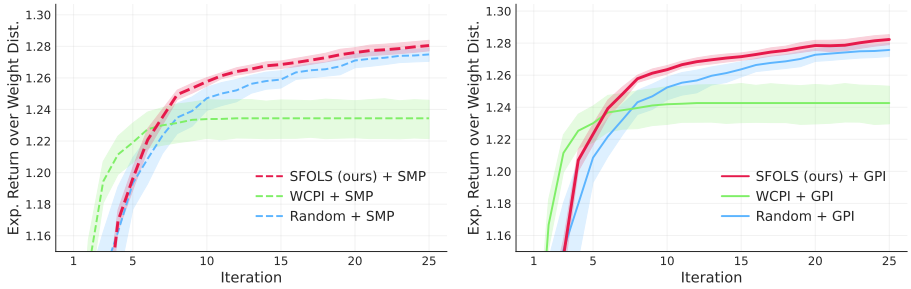

Figure 8. Expected return of each algorithm over the task distribution, $\mathcal{W}$, when evaluated using either SMP (**left**) or GPI (**right**) in the Four Room domain.

B.4. Learning SFs

In Algorithm 4 we introduce an algorithm to learn SFs using GPI based on tabular Q-learning. We used a learning rate of $\alpha = 0.3$, and ε-greedy exploration with the value of ε linearly decaying from 1 to 0.05. At each iteration, the total number of time steps used to learn the SFs was set to num_steps $= 10^5$ for the DST domain and num_steps $= 10^6$ for the Four Room domain.

In the continuous state case, we learned SFs using neural networks as detailed in Algorithm 5. We employed a training scheme similar to DQN (Mnih et al., 2015). Each SF $\boldsymbol{\psi}(s, a)$ is a multi-layer perceptron (MLP) neural network with two layers of 256 neurons and ReLU non-linear activations. Each neural network outputs a matrix $\mathbb{R}^{|\mathcal{A}| \times d}$, consisting of the values for each one of the $|\mathcal{A}|$ actions and d features. We also adopted popular DQN extensions to speed up and stabilize learning, such as double Q-learning (Hasselt et al., 2016) and prioritized experience replay (Schaul et al., 2016; Fujimoto et al., 2020). We used Adam (Kingma & Ba, 2015) with a learning rate of 0.001 as the gradient-based optimizer, and a mini-batch of size $b = 256$. The value of ε was fixed to 0.05 for the ε-greedy exploration. We trained each SF for num_steps $= 2 \cdot 10^5$ time steps.

B.5. Worst-Case Reward Policy Iteration (Zahavy et al., 2021a)

In Algorithm B.5 we present the WCPI algorithm proposed by Zahavy et al. (2021a) to learn a diverse set of policies using SFs. For details on how to compute the worst-case reward (line 5) by solving linear programs, see Lemma 4 of Zahavy et al. (2021a).

Algorithm 4 Learn New Tabular Successor Features with GPI

Input: SF set $\Psi = \{\boldsymbol{\psi}^{\pi_1}, ..., \boldsymbol{\psi}^{\pi_{n-1}}\}$, learning rate α, exploration prob. ε, new preference weight $\mathbf{w}$, num_steps
Initialize new SF: $\boldsymbol{\psi}^{\pi_n}(s,a) \leftarrow \boldsymbol{\psi}^{\pi_i}(s,a) \forall (s,a)$, with $i = \arg\max_i \boldsymbol{\psi}^{\pi_i} \cdot \mathbf{w}$
new_episode $\leftarrow$ True
for $t = 0...$num_steps **do**
  if new_episode **then**
    $S_t \leftarrow$ initial state sampled from μ
    new_episode $\leftarrow$ False
  end if
  if Bernoulli$(1-\varepsilon)$ **then**
    $A_t \leftarrow \arg\max_a \max_i \boldsymbol{\psi}^{\pi_i}(S_t, a) \cdot \mathbf{w}$ {GPI}
  else
    $A_t \leftarrow$ random action sampled from Uniform$(\mathcal{A})$ {ε-greedy exploration}
  end if
  Execute A_t, observe $\boldsymbol{\phi}_t$ and S_{t+1}
  if S_{t+1} is terminal **then**
    new_episode $\leftarrow$ True
    $\boldsymbol{\delta}_t = \boldsymbol{\phi}_t - \boldsymbol{\psi}^{\pi_n}(S_t, A_t)$
  else
    $a' \leftarrow \arg\max_{a'} \max_i \boldsymbol{\psi}^{\pi_i}(S_{t+1}, a') \cdot \mathbf{w}$
    $\boldsymbol{\delta}_t = \boldsymbol{\phi}_t + \gamma \boldsymbol{\psi}^{\pi_n}(S_{t+1}, a') - \boldsymbol{\psi}^{\pi_n}(S_t, A_t)$
  end if
  $\boldsymbol{\psi}^{\pi_n}(S_t, A_t) \leftarrow \boldsymbol{\psi}^{\pi_n}(S_t, A_t) + \alpha \boldsymbol{\delta}_t$ {Update SFs}
end for
return $\boldsymbol{\psi}^{\pi_n}$

B.6. Set of Independent Policies (Alver & Precup, 2022)

In Algorithm B.6 we present the SIP algorithm proposed by Alver & Precup (2022) to learn a set of independent policies using SFs. Importantly, it requires that the features $\boldsymbol{\phi}(s,a,s') \in \mathbb{R}^d$ are *independent* (see Definition 1 of Alver & Precup (2022)) and that the MDP's transition function is deterministic.

Algorithm 5 Learn New DQN-based Successor Features with GPI

Input: SF set $\Psi = \{\boldsymbol{\psi}^{\pi_1}, ..., \boldsymbol{\psi}^{\pi_{n-1}}\}$, replay buffer $\mathcal{D}$, exploration prob. ε, new preference weight $\mathbf{w}$, num_steps
Initialize new SF $\boldsymbol{\psi}^{\pi_n}(s, a)$ as a neural network
new_episode $\leftarrow$ True
for $t = 0$...num_steps **do**
  if new_episode **then**
    $S_t \leftarrow$ initial state sampled from μ
    new_episode $\leftarrow$ False
  end if
  if Bernoulli$(1 - \varepsilon)$ **then**
    $A_t \leftarrow \arg\max_a \max_i \boldsymbol{\psi}^{\pi_i}(S_t, a) \cdot \mathbf{w}$ {GPI}
  else
    $A_t \leftarrow$ random action sampled from Uniform$(\mathcal{A})$ {ε-greedy exploration}
  end if
  Execute A_t, observe $\boldsymbol{\phi}_t$ and S_{t+1}
  Add $(S_t, A_t, \boldsymbol{\phi}_t, S_{t+1})$ to $\mathcal{D}$
  if S_{t+1} is terminal **then**
    new_episode $\leftarrow$ True
  end if
  Sample mini-batch $\{(s_i, a_i, \boldsymbol{\phi}_i, s'_i)\}_{i=1}^{b}$ from $\mathcal{D}$
  $a'_i = \arg\max_{a'} \max_i \boldsymbol{\psi}^{\pi_i}(s'_i, a')$
  $\mathbf{y}_i = \boldsymbol{\phi}_i + \gamma \boldsymbol{\psi}^{\pi_i}(s'_i, a'_i)$
  Update SF by minimizing the loss $\mathcal{L}(\boldsymbol{\psi}^{\pi_n}) = \frac{1}{b} \sum_{i=1}^{b} \left[(\mathbf{y}_i - \boldsymbol{\psi}^{\pi_n}(s_i, a_i))^2\right]$
end for
return $\boldsymbol{\psi}^{\pi_n}$

Algorithm 6 SMP Worst Case Policy Iteration (Zahavy et al., 2021a)

Initialize: $\Pi \leftarrow \{\}$; $\Psi \leftarrow \{\}$; $\bar{\mathbf{w}} \leftarrow$ random weight sampled from $\mathcal{W}$
$\pi, \boldsymbol{\psi}^{\pi} \leftarrow$ solution of the RL task $\bar{\mathbf{w}}$
Add π to Π and $\boldsymbol{\psi}^{\pi}$ to Ψ
repeat
  $\bar{\mathbf{w}} \leftarrow \arg\min_{\mathbf{w} \in \mathcal{W}} \max_{\pi \in \Pi} \boldsymbol{\psi}^{\pi} \cdot \mathbf{w}$
  $\pi, \boldsymbol{\psi}^{\pi} \leftarrow$ solution of the RL task $\bar{\mathbf{w}}$
  Add π to Π and $\boldsymbol{\psi}^{\pi}$ to Ψ
until $v_{\bar{\mathbf{w}}}^{\text{SMP}}$ does not improve
return Π, Ψ

Algorithm 7 Set of Independent Policies (Alver & Precup, 2022)

Initialize: $\Pi \leftarrow \{\}$; $\Psi \leftarrow \{\}$; $i \leftarrow 1$
while $i \leq d$ **do**
  $\mathbf{w}_i \leftarrow$ weight with a positive value in the i-th component and negative values elsewhere
  $\pi, \boldsymbol{\psi}^{\pi} \leftarrow$ solution of the RL task $\mathbf{w}_i$
  Add π to Π and $\boldsymbol{\psi}^{\pi}$ to Ψ
  $i \leftarrow i + 1$
end while
return Π, Ψ