Utilising the Parameter-Performance Relationship for Efficient Multi-Objective Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Multi-objective reinforcement learning (MORL) aims to identify diverse optimal policies forming a Pareto front to balance different, often conflicting objectives. The complex mapping between the policy parameter space and the multi-objective performance space poses significant challenges for efficient exploration. This work formally introduces and exploits the Parameter-Performance Relationship (PPR), proposing that an understanding of its local structure enables more efficient MORL. We present an algorithm that realises the PPR through locally linear extensions, called LLE-MORL. By using a few initial policies and their briefly retrained variants to define extension directions, our method efficiently generates candidate policies along the Pareto front with minimal additional training. Experiments on continuous control benchmarks show our approach discovers high-quality, comprehensive Pareto fronts efficiently than existing methods. This demonstrates that systematically leveraging the PPR provides a powerful strategy for advancing MORL.

1 Introduction

Reinforcement Learning (RL) has shown great promise in complex decision-making problems, enabling significant advancements in a wide range [Silver et al., 2016, Levine et al., 2016]. In real-world scenarios, however, problems often feature multiple, often conflicting, objectives. Under this circumstance, multi-objective approaches provide flexibility in practical applications of reinforcement learning by providing a modifiable policy that can be adjusted according to changes of preference among a set of objectives [Roijers et al., 2013, Hayes et al., 2022]. This has fostered the development of the field known as multi-objective reinforcement learning (MORL). Ideally, the modifiable policies developed within MORL allow for efficient adaptation, ensuring that a policy optimal for one set of preferences can be readily transformed to be optimal for a new set when those preferences change. To prepare such a modifiable policy for application, three problems have to be solved: (i) The *learning* problem involves the solution of an RL problem for each combination of preference parameters or at least for a representative subset of preferences. (ii) The *representation* problem requires a parametrization of the policies, which typically results in either a discrete set of individual policies (common in population-based methods) or a single, continuously adaptable policy (prevalent in deep reinforcement learning approaches). (iii) The *selection* problem is to identify a suitable policy in the application which includes dynamic adjustments to preference drifts and possibly the decision whether a different policy should be invoked or whether further training is required to respond to a temporary detection of suboptimality.

We propose to consider these problems as a coherent task, in order to reduce the computational burden of the learning problem and improve the interpretability of the policy representation. We hypothesise that if a continuous representation of policies can be found where similar preferences correspond to similar policy parameters, then small performance differences might be compensable with brief, targeted retraining. It is also anticipated that such a structured and interpretable policy representation would benefit the selection problem, though this aspect is not the primary focus of our current study.

While a globally continuous mapping is an ideal, we notice that in non-trivial problems, the relationship between the performance space and parameter space of policies is not a simple, single continuous mapping but can be described by a family of locally continuous components [Xu et al., 2020, Li et al., 2024]. Our findings suggest that effectively exploring just a few of these components can be sufficient to achieve competitive performance in typical benchmark problems. This understanding forms the basis of our core concept: the Parameter-Performance Relationship (PPR). The process is seeded by an RL task that finds a good but not necessarily optimal parameter vector for an initial policy. Then a second policy is obtained by retraining with different preferences, and from there, additional policies are efficiently generated by a locally linear extrapolation, which led to the name LLE-MORL for the approach that we present in the following. If the policies obtained by the extension process are briefly retrained, they can improve with further extension, although eventually they may become dominated by earlier solutions which would indicate the need for a restart with a different initial policy. Within each solution component, the policy representation is easily interpretable in terms of the continuous PPR, but also the boundaries where the policies depart from optimality are interesting. They indicate that a discontinuous reparametrisation takes place and that thus policies of a potentially qualitatively different type are optimal on either side of the boundary.

Building upon the notions of PPR and locally linear extension, in this paper, we introduce LLE-MORL, a MORL algorithm that is designed to efficiently trace the Pareto front (see Sect. 2.2) by systematically exploring these identified local structures. Our experiments demonstrate that the proposed algorithm can achieve high-quality Pareto front approximations with notable sample efficiency. This strong performance is primarily attributed to its simple yet effective locally linear extension method, which significantly reduces the need for extensive retraining along the Pareto front. Such efficiency is made by exploiting the locally continuous nature of the parameter-performance relationship, a characteristic that also enhances the overall interpretability of our approach.

2 Background

2.1 Multi-Objective Reinforcement Learning

Multi-Objective Reinforcement Learning (MORL) extends the traditional RL framework to scenarios where agents must consider multiple, often conflicting objectives. This extension allows for more sophisticated decision-making models that mirror real-world complexities where trade-offs between competing goals, such as cost versus quality or speed versus safety, are common. To ground this notion formally, we represent a MORL problem as a Multi-Objective Markov Decision Process (MOMDP) which generalises the standard MDP framework to accommodate multiple reward functions, each corresponding to a different objective.

Definition 1 *Multi-Objective Markov Decision Process (MOMDP).* A MOMDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \{\mathcal{R}^d\}, \gamma, \Omega, f_\Omega)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(s'|s,a)$ is the state transition probability, $\mathcal{R}^d$ is a vector-valued reward function with d as the number of objectives, specifying the immediate reward for each of the considered objectives, γ is the discount factor, Ω is the preferences space, $f_\Omega : \mathbb{R}^d \to \mathbb{R}$ is the scalarisation function.

The crucial difference between MOMDPs and traditional single-objective MDPs is the reward structure. While single-objective MDPs use a scalar reward function $\mathcal{R}$, MOMDPs feature a vector-valued reward function $\mathcal{R}^d$ that delivers distinct numeric feedback for each objective, directly correlating the length of the reward vector with the number of objectives. At each timestep t, the agent in state $s_t \in \mathcal{S}$ selects an action $a_t \sim \pi(\cdot \mid s_t)$, transitions to a new state s_{t+1} with probability $P(s_{t+1} \mid s_t, a_t)$, and receives a reward vector $\mathbf{r}_t = \big[(R_1(s_t,a_t),\, R_2(s_t,a_t),\, \ldots,\, R_d(s_t,a_t))\big]$. We define the discounted return vector by $\mathbf{G}_t = \sum_{k=0}^{\infty} \gamma^k\, \mathbf{r}_{t+k}$, and the multi-objective action-value function of a policy π for a given state-action pair (s,a) by $\mathbf{Q}^\pi(s,a) = \mathbb{E}_\pi\big[\mathbf{G}_t \mid s_t = s,\, a_t = a\big]$. The goal of MORL is to find a policy π such that the expected return of each objective can be optimised. In practice, we trade off objectives via a scalarisation function $f_\omega(\mathbf{r})$, which produces a scalar utility using preference vector $\omega \in \Omega$. The scalarisation function $f_\omega(\mathbf{r})$ is used for mapping the multi-objective reward vector $\mathbf{r}(s, a)$ to a single scalar. In this paper, we consider the linear scalarisation function $f_\omega(\mathbf{r}(s, a)) = \omega^{\mathbf{T}}\mathbf{r}(s, a)$, which is commonly used in MORL literature [Yang et al., 2019, Felten et al., 2024]. When the preference dimension $d = 1$ (so that the return vector is one-dimensional), the MOMDP collapses to a standard single-objective MDP, since the reward vector reduces to a scalar and f_Ω becomes the identity mapping.

2.2 Pareto Optimality

In multi-objective optimisation, the concept of optimality differs from the single-objective case. Typically, no single policy simultaneously maximises all objectives, due to inherent trade-offs. Without any additional information about the user's preference, there can now be multiple possibly optimal solutions. In the following, we introduce several useful definitions for possibly optimal policies.

Definition 2 *Pareto optimality* A policy π is said to *dominate* another policy π' if and only if: $\forall i \in \{1, \ldots, d\}$, $V_i^\pi(s) \geq V_i^{\pi'}(s)$, and $\exists j, V_j^\pi(s) > V_j^{\pi'}(s)$, where $V_i^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, a_t) \mid s_0 = s]$ denotes the expected discounted return for objective i under policy π. A policy π^* is *Pareto optimal* if and only if it is not dominated by another policy. The set of all Pareto optimal policies forms the *Pareto set*: $\mathcal{P} = \{\pi \mid \pi \text{ is Pareto optimal}\}$. The corresponding set of expected returns incured by policies in the Pareto set is termed *Pareto front*: $\mathcal{F} = \{V^\pi(s) \mid \pi \in \mathcal{P}\}$.

Since obtaining the true Pareto set is intractable in complex problems, the practical aim of multi-objective optimisation is to construct a finite set of policies that closely approximate the true Pareto front. So that practitioners can select the policy based on their preferred trade-off among objectives.

2.3 Related Work

Prior work in Multi-Objective Reinforcement Learning (MORL) offers various strategies for handling conflicting objectives. These can be broadly grouped into single-policy methods and multi-policy methods for approximating the Pareto front. Single-policy approaches, a foundational strategy in MORL, typically convert the multi-objective problem into a single-objective task using a predefined preference or weighting scheme to find a policy optimal for that specific trade-off. A common instance of such a weighting scheme is linear scalarization [Van Moffaert et al., 2013]. Limitations of linear scalarization, particularly in capturing non-convex Pareto fronts, have been addressed by more advanced scalarization functions such as Chebyshev methods [Van Moffaert et al., 2013] and hypervolume-based approaches [Zhang and Golovin, 2020]. Further theoretical work has aimed at enhancing scalarization robustness and performance, for instance, by proposing the addition of concave terms to rewards [Lu et al., 2023]. Concurrently, significant efforts have developed generalised single-policy models conditioned on preference inputs to achieve adaptability across diverse objectives [Teh et al., 2017, Yang et al., 2019, Basaklar et al., 2022, Parisi et al., 2016], with subsequent extensions into offline learning contexts [Zhu et al., 2023, Lin et al., 2024] and methods to improve sample efficiency in these settings [Huang, 2022].

Multi-policy MORL strategies directly target the approximation of the entire Pareto front by learning a diverse collection of policies. One direction for generating diverse behaviours involves developing single, highly adaptable models conditioned on preferences, which generalise across various objectives using techniques like specialised experience replay or policy gradient methods that enforce Pareto stationarity [Abels et al., 2019, Friedman and Fontaine, 2018, Kyriakis and Deshmukh, 2022]. Other approaches explicitly learn a diverse set of policies or their value functions; this includes direct value-based methods like Pareto Q-learning [Van Moffaert and Nowé, 2014], and evolutionary algorithms often guided by prediction models to discover a dense Pareto set [Xu et al., 2020]. Further techniques for generating policy sets involve Generalised Policy Improvement (GPI) for sample-efficient learning [Alegre et al., 2023] or the development of transferable policy components using representations like successor features [Alegre et al., 2022]. The use of constrained optimisation to efficiently complete and refine the Pareto front is also explored in [Liu et al., 2024, He et al., 2024]. Furthermore, the principles of decomposition-based strategies, which find a set of solutions by solving multiple interrelated scalarised sub-problems, have been a significant focus, with recent work providing clarifying taxonomies and conceptual frameworks [Felten et al., 2024, Röpke et al., 2024].

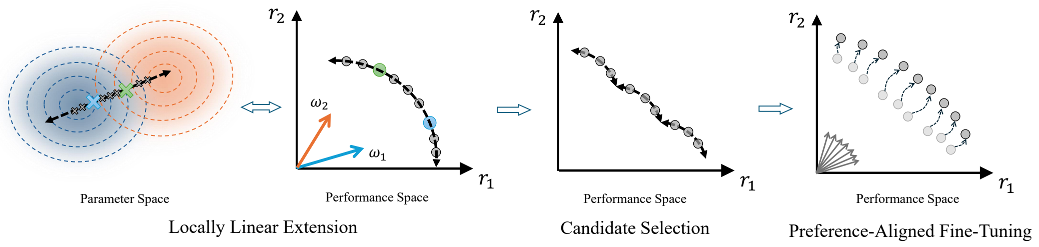


Figure 1: Algorithm Overview (see Sect. 3.5). (From left) **Locally Linear Extension:** After initialisation of a policy in the parameter space, brief retraining produces a second policy parameter vector. The corresponding two policies are linearly extended to produce a set of candidate policies which map to a trajectory of performance vectors (black). The green and the blue point represent, resp., the performance of the initial and the retrained policy. **Candidate Selection:** From this set of policies, the non-dominated candidate solutions are selected. **Preference-Aligned Fine-Tuning:** The selected non-dominated candidates undergo a brief fine-tuning phase. Policies are refined based on respective preference vectors (fanned arrows from origin) which tends to move them closer to the true Pareto front providing the final approximation of the front.

While these established single-policy and multi-policy paradigms have significantly advanced MORL, the explicit characterisation and systematic exploitation of the structural relationship between the learned policies' underlying parameter space and their resultant performance on the Pareto front remain largely underexplored. Although multi-objective optimization offers techniques for navigating Pareto sets [Ye and Liu, 2022], and some MORL studies have touched upon parameter space regularities [Xu et al., 2020], policy manifolds [Parisi et al., 2016], or front geometries [Li et al., 2024], these explorations typically do not formalise or exploit the parameter-to-performance mapping for systematic, guided Pareto front generation.

3 Methods

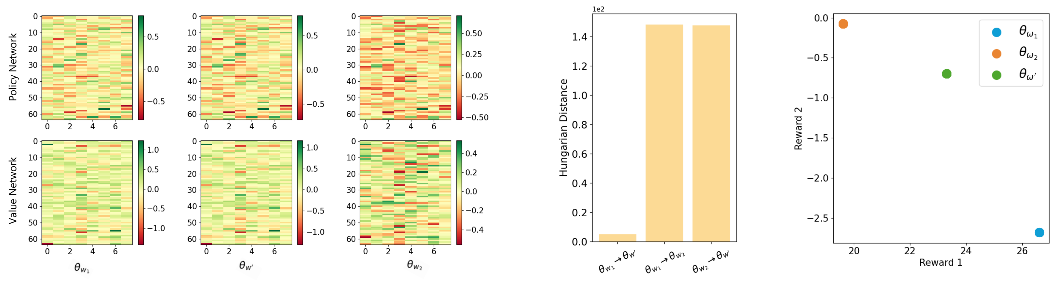
3.1 Overview

As shown in Fig. 1, the LLE-MORLalgorithm relies on the relationship between the parameter space and the performance space. We empirically find that a short retraining of a converged policy under a new preference induces a small, structured update in parameter space that corresponds to a predictable shift of the expected returns of the policy. This "model similarity"—the fact that the retrained policy stays close to the original parameters while already moving toward a different region of the Pareto front-—underpins our method for steering policies along the front. Building on this insight, we explore the possibility that using the parameter-space difference between two structurally similar policies—trained under different preferences—to guide directional updates that extend our approximation of the Pareto front.

Leveraging this property, we develop an efficient algorithm to approximate the Pareto set of policies. We start by initializing a small collection of base policies, each trained to converge under a distinct scalarization weight chosen to span the preference evenly. Next, for each base policy, we perform a short retraining under a different preference weight, capturing the small parameter update that shifts the policy toward a new trade-off. These updates serve as directional moving vectors: we move from each base policy along its vector by a tunable step size to generate intermediate policies. Finally, we apply a brief fine-tuning to each intermediate policy under its corresponding preference, i.e. the scalarization weight shifted by the same fraction as the parameter updated, nudging it onto the true Pareto front.

3.2 Parameter-Performance Relationship

Recent work in multi-objective reinforcement learning has implicitly suggested a relationship between the parameter space of the policy network and the Pareto front in the performance space. [Xu et al., 2020] empirically show for PGMORL that each disjoint policy family occupies a continuous region



(a) Policy-net and value-net parameter heatmap to illustrate the retraining effect that is measured in (b). (b) Combined Hungarian model distance. (c) Respective positions in performance space.

Figure 2: Comparing independently trained policy θ_{w_2} versus retrained policy $\theta_{w'}$ based on θ_{w_1}, for details see Sect. 3.3. The environment used here is the multi-objective SWIMMER problem.

in parameter space and maps to a contiguous segment of the Pareto front, while MORL/D [Felten et al., 2024] assume that policies with similar parameters should lead to close evaluations. Motivated by these implicit observations, we introduce the a *parameter–performance relationship* and proceed to explain and empirically validate this property.

Definition 3 *Parameter-Performance Relationship (PPR).* Let $\Theta \subseteq \mathbb{R}^n$ be the policy parameter space and $V : \Theta \to \mathbb{R}^d$ the mapping from parameter vectors θ to the expected return vectors $V(\theta)$. We say V exhibits a *continuous parameter–performance relationship* on a region $U \subseteq \Theta$ if there exists a function $h : \mathbb{R}^n \to \mathbb{R}^d$ and a radius $\delta > 0$ such that, for any $\theta \in U$ and any parameter perturbation $\Delta\theta$ with $\|\Delta\theta\| < \delta$ and $\theta + \Delta\theta \in U$, $V(\theta + \Delta\theta) - V(\theta) = h\big(\Delta\theta\big)$.

To study this relationship, we first need a metric for policy closeness in parameter space. We adopt the Hungarian matching distance [Kuhn, 1955, Munkres, 1957] to measure model distance and thereby quantify structural similarity between policies. Our policies are represented by feedforward neural networks with multiple layers. For each layer, we view the neurons of the two networks as two point-sets and compute a minimum-cost perfect matching between them, where the cost of matching neuron i to neuron j is the ℓ_2 norm of the difference between their incoming weight vectors. Summing these layerwise minimal costs yields the total distance. This metric naturally handles the permutation invariance of hidden units [Goodfellow et al., 2016] and measures the smallest structural change needed to align one model to another—lower Hungarian distance indicates greater model similarity.

3.3 Sanity Check

To get a first idea about the PPR, we compare policies trained independently with those obtained by short retraining. We first train two policies to convergence using a multi-objective PPO-based [Schulman et al., 2017] algorithm with scalarization vectors w_1 and w_2, yielding model parameters θ_{w_1} and θ_{w_2}. Starting from θ_{w_1}, we then perform one short additional training step with w_2 to obtain $\theta_{w'}$. To quantify how "close" these policy variants are, we show neuron heatmaps for each model both at the policy-network and value-network level in Figure 2a, and visualise the Hungarian matching distances between those models in Figure 2b. We also plot the rewards for three policies in the two-objective performance space (Figure 2c) for the multi-objective SWIMMER problem.

We compare three pairs of models: (1) θ_{w_1} and θ_{w_2}, capturing differences between independently trained policies in both parameter space and performance space; (2) θ_{w_1} and $\theta_{w'}$, showing that brief retraining yields a structurally similar model and a low Hungarian matching distance, yet already shifted toward w_2 in reward space; and (3) $\theta_{w'}$ and θ_{w_2}, illustrating that although their parameters remain distinct, their rewards lie much closer on the performance space.

These empirical observations show that a short retraining step under a new preference produces a small, structured parameter update that directly maps to a predictable shift in performance, validating the local PPR.

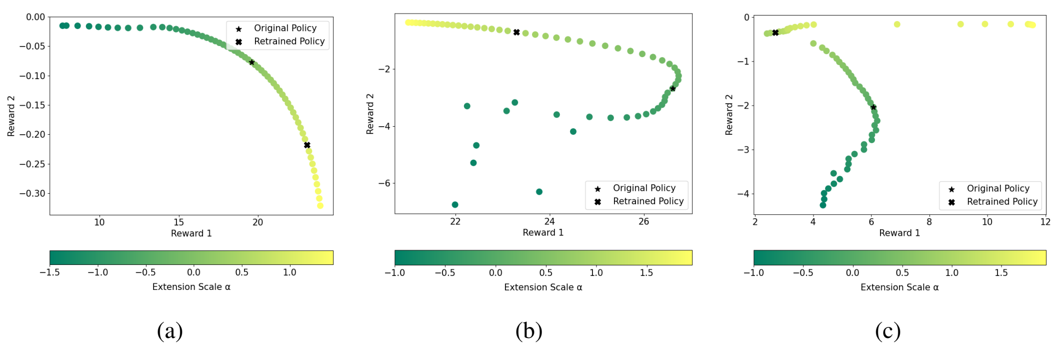


Figure 3: Visualisation of the process of applying the parameter difference $\Delta\theta = \theta_{w'} - \theta_w$ between two related policies. The policies are obtained by first training a policy θ_w to convergence using scalarization vector w and then find policy $\theta_{w'}$ by a brief additional training period with a different scalarization vector w'. Iterating the shift $\Delta\theta$ in the policy space induces a sequence of shifts also in the multi-objective reward space. The subfigures show results for different initial preferences: (a) A convex front is found from the two policies. (b) Although the original policy turns out to be Pareto suboptimal, the solution manifold extends into a Pareto optimal component. (c) Retraining can cause the (Pareto-suboptimal) original solution to jump to a different branch so that the corresponding solution consists of two components one of which can be ignored because of Pareto suboptimality.

3.4 Locally Linear Extension

Based on the PPR definition, a natural question is whether the parameter-space difference between two structurally similar policies—trained under different preferences—can serve as a directional update to extend our approximate Pareto front. To explore this, we consider two policies, a base policy θ_w and a retrained policy $\theta_{w'}$, which exhibits a parameter-performance relationship. Crucially, for this directional information to be meaningful for Pareto front exploration, both θ_w and $\theta_{w'}$ should ideally be non-dominated solutions, at least with respect to each other. Given such a pair, we compute the parameter update vector $\Delta\theta = \theta_{w'} - \theta_w$ and generate a set of intermediate policies by moving from the base policy θ_w along the parameter displacement $\Delta\theta$ in scaled steps. Concretely, for each scale α, we form $\theta_\alpha = \theta_w + \alpha\,\Delta\theta$ and evaluate its reward vectors in preference space.

Figure 3 visualises the resulting trajectory of reward vectors in the two-dimensional objective space: as α grows, the trajectory passes through the region around $\theta_{w'}$ and can extend beyond both the base and retrained endpoints, demonstrating how simple parameter-space moves can traverse broad trade-off regions, which offers a cost-effective strategy for efficiently expanding an approximate Pareto front without training each point from scratch.

3.5 The LLE-MORL Algorithm

The locally linear extension process is critical for the LLE-MORL algorithm in tracing an approximate Pareto front efficiently. The full algorithm (see also App. A) consists of five stages (compare Fig. 1): **(1) Initialization:** We train a set of K base policies $\{\theta_{w_i}\}_{i=1}^{K}$ to convergence using PPO [Schulman et al., 2017]. Each policy is trained under a distinct scalarization weight $w_i \in \Omega$, where these weights are chosen to be evenly distributed across the preference space. **(2) Directional Retraining:** For each $i = 1, \ldots, K-1$, continue train based on θ_{w_i} under a new preference $w_{i'}$ for T_{dir} steps to obtain $\theta_{w'_i}$, where θ_{w_i} and $\theta_{w_{i'}}$ should be both non-dominated points. Record the parameter update vector $\Delta\theta_i = \theta_{w'_i} - \theta_{w_i}$ and weight shift $\Delta w_i = w_{i+1} - w_i$. **(3) Locally Linear Extension:** For each base policy θ_{w_i}, we generate a set of intermediate policies by applying each step-scale factor α_j to the parameter update vector $\Delta\theta_i$. Concretely, each candidate is $\theta_{i,j} = \theta_{w_i} + \alpha_j\,\Delta\theta_i$, allowing negative and positive moves along the local direction in parameter space. Simultaneously, we adjust the preference weight by Δw_i scaled by α_j to obtain $w_{i,j}$. These step-scale factors control how far along the local direction each intermediate policy moves. **(4) Candidate Selection:** All candidate policies $\theta_{i,j}$ generated in the locally linear extension stage are evaluated to obtain their respective performance vectors. From this set of extended policies, we identify and select the subset of non-dominated solutions. These selected non-dominated candidates are then advanced to the fine-tuning stage. **(5)**

Preference-Aligned Fine-Tuning: from each candidate θ and its matched weight w, perform a short PPO fine-tuning of T_{ref} steps under w to push the generated policy closer to the true Pareto front.

4 Experiments

4.1 Benchmark Problems and Evaluation Metrics

In this section, we evaluate the LLE-MORL algorithm using popular continuous MORL benchmark problems from the MO-Gymnasium [Felten et al., 2023]. Our benchmark problems include three two-objective continuous environments: MO-Swimmer-v5, MO-Hopper-2d-v5, and MO-Ant-2d-v5.

We evaluate the quality of the approximate Pareto front using three standard metrics, hypervolume (HV), expected utility (EU), and sparsity (SP), following the formalism in [Zitzler and Thiele, 2002, Zintgraf et al., 2015, Hayes et al., 2022]. A higher hypervolume implies a front closer to and more extensive with respect to the true Pareto front. A higher EU denotes better average performance over preferences. Lower sparsity indicates a more uniform distribution of solutions along each objective.

Details of benchmarks and evaluation metrics can be found in App. B.1.

4.2 Baselines

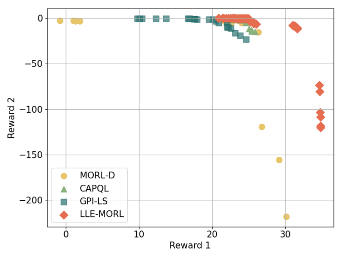
We compare our LLE-MORL against the following state-of-the-art MORL algorithms: (i) **GPI-LS** [Alegre et al., 2023] applies Generalised Policy Improvement over a discretised set of preference weights and uses linear scalarization to construct a diverse Pareto set. (ii) A Conservative Approximate Pareto Q-Learning method (**CAPQL**) [Lu et al., 2023]: learns an ensemble of Q-functions under different preferences and selects actions via conservative aggregation to improve front coverage. (iii) **MORL/D** [Felten et al., 2024] is a deep-RL analogue of decomposition-based multi-objective optimisation that trains subpolicies under scalarised objectives and recombines them via weight decompositions to approximate the Pareto front.

4.3 Results and Analysis

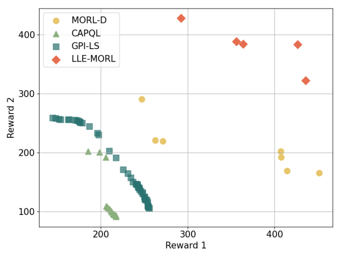
To assess the performance of MORL, we now present quantitative results evaluating the quality of the approximated Pareto fronts. We conduct experiments under two distinct settings to provide a comprehensive understanding of algorithms' capabilities: **(1) Sample-Efficient Setting:** All methods, including our LLE-MORL approach, were trained for 1.5×10^5 timesteps. Given the complexity of continuous control benchmarks, this relatively limited interaction budget serves as a critical testbed for evaluating how rapidly different MORL strategies can discover effective Pareto front approximations. **(2) Standard-Training Setting:** To assess performance under more common training conditions for these continuous control benchmarks, most methods, including our LLE-MORL approach, were trained for 1×10^6 timesteps. An exception was made for the CAPQL baseline, which, due to its significant computational demands, was trained for 5×10^5 timesteps. This setting aligns with common practices for benchmarking in continuous control and allows us to assess the final quality of the Pareto fronts achieved by each algorithm after a more thorough learning process. Detailed training setups can be found in App. B.3.

First, we analyse performance in the sample-efficient setting, with results using Hypervolume (HV), Expected Utility (EU), and Sparsity (SP) metrics presented in Table 1 and the corresponding Pareto front visualisations in Figure 4. In this limited-interaction scenario, LLE-MORL achieves the highest HV and EU in all benchmarks, demonstrating strong capabilities in rapidly achieving high-quality Pareto fronts. Regarding SP, while LLE-MORL does not consistently achieve the leading scores on this metric, its performance generally reflects a good and effective distribution of solutions along the high-quality Pareto fronts it identifies. It should be noted that SP results can be confounded by a fragmentary recovery of the Pareto front. For instance, if CAPQL discovers only two close points of the Pareto front for the MO-Ant problem, the sparsity rating is nearly perfect. Conversely, a low SP might also arise from solutions being overly clustered in a small region, as potentially seen with GPI-LS in MO-Swimmer shown in Figure 4a.

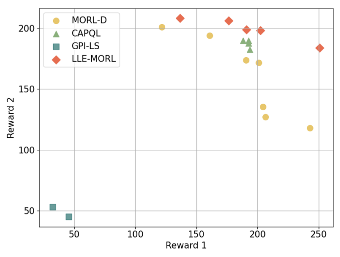
Transitioning to the standard-training setting, where methods were trained for a more extensive duration, the evaluation results are presented in Table 2, and the corresponding Pareto front visualisation can be found in Figure 5. Across all benchmarks, LLE-MORL typically achieves the highest HV and



(a) MO-Swimmer

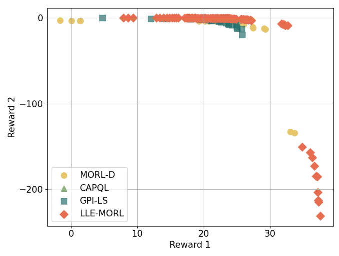


(b) MO-Hopper-2d

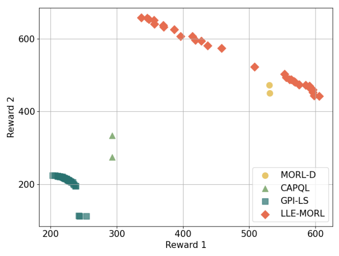


(c) MO-Ant-2d

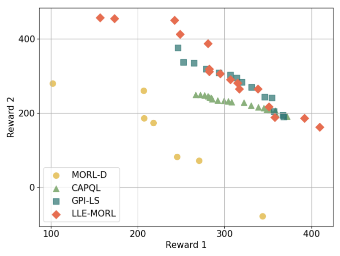
Figure 4: Pareto fronts from the sample-efficient training setting, comparing our LLE-MORL method with baselines on three continuous-control benchmarks. LLE-MORL demonstrates more comprehensive Pareto fronts across all benchmarks.



(a) MO-Swimmer



(b) MO-Hopper-2d



(c) MO-Ant-2d

Figure 5: Pareto fronts from the standard-training setting, comparing our LLE-MORL method with baselines on three continuous-control benchmarks. LLE-MORL consistently achieves wider coverage and closer proximity to the true Pareto front.

highly competitive EU. This superior performance indicates that LLE-MORL finds a more extensive and higher-quality set of solutions, which strongly suggests a better approximation of the Pareto front compared to the baselines. The evidence from Pareto front visualisation further corroborates LLE-MORL's advantages. In the MO-Swimmer environment, shown in Figure 5a, LLE-MORL more comprehensively explores the objective space, successfully identifying Pareto optimal solutions in the lower-right region consistently missed by baselines such as GPI-LS and CAPQL. Notably, when comparing the GPI-LS and CAPQL performance to those in the sample-efficient setting for the MO-Swimmer environment, these particular baselines appear to remain constrained by suboptimal solutions in this challenging region, indicating that simply extending training duration did not resolve their exploration deficiencies here. While LLE-MORL's thorough exploration to achieve this broader coverage means its SP may not be the numerically lowest, the result could be well-justified by the extensive nature of the front.

In summary, LLE-MORL consistently demonstrates superior Pareto front approximations across both sample-efficient and standard-training evaluations. This robust performance is significantly supported by its innovative extension process, which is largely training-free once core parameter-performance relationships are established, allowing for the efficient generation of diverse and high-quality solutions. Consequently, LLE-MORL excels at both rapid learning in data-limited scenarios and achieving comprehensive, high-fidelity fronts with extended training, highlighting its distinct advantages for multi-objective reinforcement learning. Additionally, we present the running time of each algorithm in App. B, which shows the high efficiency level achievable by LLE-MORL.

Environment	Metric	Method				
		GPI-LS	CAPQL	MORL/D	LLE-MORL-0	LLE-MORL
MO-Swimmer	HV(10^4)	4.92	5.16	5.95	6.68	**6.93**
	EU(10^1)	0.99	**1.09**	1.01	0.84	**1.09**
	SP(10^2)	**0.05**	0.97	21.99	0.88	0.66
MO-Hopper	HV(10^5)	1.22	0.95	1.96	2.68	**2.77**
	EU(10^2)	2.31	2.02	3.33	4.05	**4.13**
	SP(10^2)	**0.57**	4.48	43.10	16.56	33.85
MO-Ant	HV(10^4)	2.22	8.52	9.65	10.44	**10.67**
	EU(10^2)	0.46	1.91	1.99	2.25	**2.32**
	SP(10^3)	2.46	**0.27**	1.11	1.45	1.14

Table 1: Sample-efficient evaluation of the quality of the Pareto front by hypervolume (HV), expected utility (EU) and sparsity (SP).

Environment	Metric	Method				
		GPI-LS	CAPQL	MORL/D	LLE-MORL-0	LLE-MORL
MO-Swimmer	HV(10^4)	5.56	4.86	6.68	7.37	**7.44**
	EU(10^1)	1.09	**1.10**	**1.10**	1.03	1.03
	SP(10^2)	0.07	**0.02**	6.38	2.10	2.28
MO-Hopper	HV(10^5)	1.13	1.70	3.62	4.88	**5.05**
	EU(10^2)	2.26	3.14	5.02	5.65	**5.77**
	SP(10^2)	**1.91**	35.94	4.93	7.20	6.56
MO-Ant	HV(10^5)	2.44	1.62	1.28	2.43	**2.51**
	EU(10^2)	3.55	2.93	2.56	3.44	**3.61**
	SP(10^3)	0.44	**0.06**	9.19	1.76	1.33

Table 2: Standard-training evaluation of the quality of Pareto front by hypervolume (HV), expected utility (EU) and sparsity (SP).

4.4 Ablation Study

The LLE-MORL integrates a locally linear extension process with a subsequent fine-tuning stage. To understand the distinct contributions of these components to the overall performance, our ablation study separates them. We first evaluate LLE-MORL-0, which solely employs the extension process without fine-tuning. As detailed in Table 1 and Table 2, LLE-MORL-0 itself demonstrates competitiveness, achieving strong Hypervolume (HV) and Expected Utility (EU) scores that are often competitive with or superior to baselines. This emphasises the efficacy of our extension mechanism in rapidly discovering a high-quality approximation of the Pareto front.

Subsequently, we assess the improvement of the fine-tuning stage by comparing LLE-MORL (which includes fine-tuning) to LLE-MORL-0. This comparison reveals that the inclusion of fine-tuning consistently yields further improvements in Hypervolume (HV) and Expected Utility (EU) across both sample-efficient and standard-training settings. The impact on Sparsity (SP) is less uniform, which is an expected outcome, as refining solutions towards a more optimal Pareto front can alter their relative spacing. Nevertheless, the consistent enhancements in HV and EU prove the value of fine-tuning for improving the overall quality of the approximate Pareto front and its coverage by diverse solutions. This demonstrates that the extension process provides a strong foundation for the fine-tuning stage that enables LLE-MORL to outperform other algorithms.

5 Conclusion

In this paper, we have introduced LLE-MORL, an algorithm that identifies solution components in multi-objective reinforcement learning. The main benefit of LLE-MORL is increased efficiency which is enabled by maintaining a direct relation between the multi-objective performance and the representation of the policy in the parameter space. Preference values of the objectives are used to generate a relatively small but diverse initial set of starting points in the weight space, but Pareto-optimality is not required for this initialisation, so that learning times can remain comparatively small. Likewise, brief retraining after linear shifts in the weight space compensates for non-linearities that are assumed to be small locally, so the current component of the Pareto front can be traced out efficiently. We have shown that this simple set-up is sufficient to obtain highly efficient coverage of a Pareto front which is superior to recent MORL algorithms. Although not shown here, it can be expected that the approach can be easily extended to more than $d = 2$ objectives as the number of initial policy pairs (see Fig. 3) increases only linearly with d, although the representation of the policies set and of the Pareto front requires exponentially many points. An implicit representation of the Pareto front may seem to become preferable for $d > 2$, although this would reduce the interpretability in terms of an accessible PPR as featured here.

References

Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. Dynamic weights in multi-objective deep reinforcement learning. In *International Conference on Machine Learning*, pages 11–20. PMLR, 2019.

Lucas N Alegre, Ana LC Bazzan, Diederik M Roijers, Ann Nowé, and Bruno C da Silva. Sample-efficient multi-objective learning via generalized policy improvement prioritization. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 2003–2012, 2023.

Lucas Nunes Alegre, Ana Bazzan, and Bruno C Da Silva. Optimistic linear support and successor features as a basis for optimal policy transfer. In *International Conference on Machine Learning*, pages 394–413. PMLR, 2022.

Toygun Basaklar, Suat Gumussoy, and Umit Ogras. PD-MORL: Preference-driven multi-objective reinforcement learning algorithm. *arXiv preprint arXiv:2208.07914*, 2022.

Florian Felten, Lucas N. Alegre, Ann Nowé, Ana L. C. Bazzan, El Ghazali Talbi, Grégoire Danoy, and Bruno C. da Silva. A toolkit for reliable benchmarking and research in multi-objective reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 23671–23700, 2023.

Florian Felten, El-Ghazali Talbi, and Grégoire Danoy. Multi-objective reinforcement learning based on decomposition: A taxonomy and framework. *Journal of Artificial Intelligence Research*, 79: 679–723, 2024.

Eli Friedman and Fred Fontaine. Generalizing across multi-objective reward functions in deep reinforcement learning. *arXiv preprint arXiv:1809.06364*, 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, Cambridge, 2016.

Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, 2022.

Xiangkun He, Zhongxu Hu, Haohan Yang, and Chen Lv. Personalized robotic control via constrained multi-objective reinforcement learning. *Neurocomputing*, 565:126986, 2024.

Bo-Kai Huang. Q-pensieve: Boosting sample efficiency of multi-objective rl through memory sharing of Q-snapshots. Master's thesis, National Yang Ming Chiao Tung University, 2022.

Harold W Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

Panagiotis Kyriakis and Jyotirmoy Deshmukh. Pareto policy adaptation. In *International Conference on Learning Representations*, volume 2022, 2022.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Yining Li, Peizhong Ju, and Ness B Shroff. How to find the exact pareto front for multi-objective mdps? *arXiv preprint arXiv:2410.15557*, 2024.

Qian Lin, Chao Yu, Zongkai Liu, and Zifan Wu. Policy-regularized offline multi-objective reinforcement learning. In *AAMAS*, pages 1201–1209, 2024. URL `https://dl.acm.org/doi/10.5555/3635637.3662977`.

Ruohong Liu, Yuxin Pan, Linjie Xu, Lei Song, Pengcheng You, Yize Chen, and Jiang Bian. C-morl: Multi-objective reinforcement learning through efficient discovery of Pareto front. *arXiv preprint arXiv:2410.02236*, 2024.

Haoye Lu, Daniel Herman, and Yaoliang Yu. Multi-objective reinforcement learning: Convexity, stationarity and pareto optimality. In *The Eleventh International Conference on Learning Representations*, 2023.

James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

Simone Parisi, Matteo Pirotta, and Marcello Restelli. Multi-objective reinforcement learning through continuous pareto manifold approximation. *Journal of Artificial Intelligence Research*, 57:187–227, 2016.

Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3. `https://github.com/DLR-RM/stable-baselines3`, 2019.

Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

Willem Röpke, Mathieu Reymond, Patrick Mannion, Diederik M Roijers, Ann Nowé, and Roxana Rădulescu. Divide and conquer: Provably unveiling the pareto front with multi-objective reinforcement learning. *arXiv preprint arXiv:2402.07182*, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Peter Vamplew, John Yearwood, Richard Dazeley, and Adam Berry. On the limitations of scalarisation for multi-objective reinforcement learning of Pareto fronts. In *AI 2008: Advances in Artificial Intelligence: 21st Australasian Joint Conference on Artificial Intelligence Auckland, New Zealand, December 1-5, 2008. Proceedings 21*, pages 372–378. Springer, 2008.

Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

Kristof Van Moffaert, Madalina M Drugan, and Ann Nowé. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*, pages 191–199. IEEE, 2013.

Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *International Conference on Machine Learning*, pages 10607–10616. PMLR, 2020.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *Advances in Neural Information Processing Systems*, 32, 2019.

Mao Ye and Qiang Liu. Pareto navigation gradient descent: a first-order algorithm for optimization in pareto set. In *Uncertainty in Artificial Intelligence*, pages 2246–2255. PMLR, 2022.

Richard Zhang and Daniel Golovin. Random hypervolume scalarizations for provable multi-objective black box optimization. In *International conference on machine learning*, pages 11096–11105. PMLR, 2020.

Baiting Zhu, Meihua Dang, and Aditya Grover. Scaling pareto-efficient decision making via offline multi-objective RL. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=Ki4ocDm364`.

Luisa M Zintgraf, Timon V Kanters, Diederik M Roijers, Frans Oliehoek, and Philipp Beau. Quality assessment of morl algorithms: A utility-based approach. In *Benelearn 2015: Proceedings of the 24th Annual Machine Learning Conference of Belgium and the Netherlands*, 2015.

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 2002.

A Algorithm

In this section, we present a complete description of LLE-MORL, an efficient procedure for tracing an approximate Pareto front. Algorithm 1 details this process.

Algorithm 1 LLE-MORL

```
Require: Initial scalarization weights $\{w_i\}_{i=1}^K$ evenly spanning preference space, Target scalarization weights for directional retraining $\{w'_i\}_{i=1}^K$, Initialization training length $T_{\text{init}}$, Directional retraining length $T_{\text{dir}}$, Fine-tuning length $T_{\text{ref}}$, Step-scale factors $\{\alpha_j\}_{j=1}^M$
Ensure: Approximate Pareto-optimal policy set $\Pi$
 1: Initialization:
 2: for $i = 1$ to $K$ do
 3:     Train base policy $\theta_{w_i}$ with PPO under weight $w_i$ for $T_{\text{init}}$ steps
 4: end for
 5:
 6: Directional Retraining:
 7: for $i = 1$ to $K - 1$ do
 8:     $\theta_{w'_i} \leftarrow$ continue training $\theta_{w_i}$ for $T_{\text{dir}}$ steps under $w'_i$
 9:     $\Delta\theta_i \leftarrow \theta_{w'_i} - \theta_{w_i}$
10:     $\Delta w_i \leftarrow w'_i - w_i$
11: end for
12:
13: Locally Linear Extension:
14: $\mathcal{C} \leftarrow \emptyset$
15: for $i = 1$ to $K - 1$ do
16:     for $j = 1$ to $M$ do
17:         $\theta_{i,j} \leftarrow \theta_{w_i} + \alpha_j \, \Delta\theta_i$
18:         $w_{i,j} \leftarrow w_i + \alpha_j \, \Delta w_i$
19:         Evaluate performance $V(\theta_{i,j})$ under weight $w_{i,j}$
20:         $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\theta_{i,j}, w_{i,j})\}$
21:     end for
22: end for
23:
24: Candidate Selection:
25: $\mathcal{N} \leftarrow$ non-dominated subset of $\mathcal{C}$
26:
27: Preference-Aligned Fine-Tuning:
28: $\mathcal{F} \leftarrow \emptyset$
29: for all $(\theta, w) \in \mathcal{N}$ do
30:     fine-tune $\theta$ for $T_{\text{ref}}$ steps under $w$, yielding $\theta'$
31:     add $(\theta', w)$ to $\mathcal{F}$
32: end for
33: $\mathcal{C}_{\text{all}} \leftarrow \mathcal{N} \cup \mathcal{F}$
34: $\mathcal{N}_{\text{final}} \leftarrow$ non-dominated subset of $\mathcal{C}_{\text{all}}$
35: $\Pi \leftarrow \Pi \cup \{\theta \mid (\theta, \cdot) \in \mathcal{N}_{\text{final}}\}$
36: return $\Pi$
```

The core pipeline involves several stages. First, K base policies $\{\theta_{w_i}\}$ are trained, each under its respective initial weight w_i for T_{init} steps. Next, for the first $K-1$ base policies, a short directional retraining is performed: each θ_{w_i} (for $i = 1 \dots K-1$) is further trained for T_{dir} steps under its corresponding target weight w'_i to yield $\theta_{w'_i}$. This allows the calculation of a parameter-space update vector $\Delta\theta_i = \theta_{w'_i} - \theta_{w_i}$ and the associated preference shift $\Delta w_i = w'_i - w_i$.

Using these $K-1$ pairs of delta vectors, the Locally Linear Extension stage generates a set of candidate policies $\mathcal{C}$. For each original base policy θ_{w_i} (that had a corresponding $\Delta\theta_i$), intermediate candidates are formed by applying the scale factors α_j to $\Delta\theta_i$, also determining matched weights $w_{i,j}$. From this pool of generated candidates $\mathcal{C}$, a non-dominated subset $\mathcal{N}$ is selected. Policies in $\mathcal{N}$ then undergo Preference-Aligned Fine-Tuning for T_{ref} steps under their matched weights, resulting in a set of fine-tuned policies $\mathcal{F}$. Finally, the algorithm returns Π, which is the set of non-dominated policies selected from the combined pool of the initially selected non-dominated candidates $\mathcal{N}$ and their fine-tuned versions $\mathcal{F}$. The set Π constitutes the approximated Pareto front.

B Experiment Setup Details

B.1 Benchmarks

To evaluate the performance of our proposed LLE-MORL method and compare it against existing baselines, we utilise a suite of continuous control benchmarks from the MO-Gymnasium library [Felten et al., 2023]. These environments are designed to test the ability of an agent to learn policies that effectively balance multiple, often conflicting objectives. The specific environments and their multi-objective reward formulations are detailed below:

MO-Swimmer-v5. A planar, three-link swimmer operating in a viscous fluid, utilising a 2D continuous action space to control its joint torques. The objectives are to maximise forward velocity along the x-axis and minimise the control cost.

The observation space $\mathcal{S} \subset \mathbb{R}^8$ includes joint angles and velocities, and the action space $\mathcal{A} \subset \mathbb{R}^2$ represents joint torques in $[-1, 1]$. Let x_{before} and x_{after} be the x-coordinates of the centre of mass of swimmer before and after an action, Δt be the time step, and a_j be the j-th component of the action vector.

The first objective is the forward speed

$$R_1 = \frac{x_{\text{after}} - x_{\text{before}}}{\Delta t},$$

and the second objective is the energy efficiency (negative control cost):

$$R_2 = -\sum_j a_j^2$$

MO-Hopper-2obj-v5. This environment features a 2D one-legged hopper with a 3-dimensional continuous action space controlling torques for its thigh, leg, and foot joints. Originally a 3-objective task (forward speed, jump height, control cost), we use the 2-objective variant, in which the separate control-cost objective is added to other objectives.

The observation space $\mathcal{S} \subset \mathbb{R}^{11}$ includes joint states and torso position, and the action space $\mathcal{A} \subset \mathbb{R}^3$ represents joint torques in $[-1, 1]$. Let $v_x = (x_{\text{after}} - x_{\text{before}})/\Delta t$ be the forward velocity of the agent along the x-axis, where x_{after} and x_{before} are x-positions of the torso. Let $h_{\text{jump}} = 10 \times (z_{\text{after}} - z_{\text{init}})$ be a measure of jumping height, where z_{after} is the current z-position of the torso and z_{init} is its initial z-position. Let c_{ctrl} be the positive control cost, computed as $w_{\text{env\_ctrl}} \sum_j (a_j)^2$, where $w_{\text{env\_ctrl}}$ is the environment control cost weight (typically 0.001). Let r_{healthy} be the health reward (typically $+1$ if the agent has not fallen). The reward vector $\mathbf{R} = [R_1, R_2]$ is defined as:

- R_1 (Adjusted Forward Performance):

$$R_1 = v_x + r_{\text{healthy}} - c_{\text{ctrl}}$$

- R_2 (Adjusted Height Performance):

$$R_2 = h_{\text{jump}} + r_{\text{healthy}} - c_{\text{ctrl}}$$

MO-Ant-2obj-v5. A quadrupedal "ant" robot in 2D with an eight-dimensional action space for joint torques. By default, the environment emits a three-dimensional reward vector: (1) x-velocity, (2) y-velocity, and (3) control cost. Here, we use the two-objective variant in which the separate control-cost objective is added to other objectives.

The observation space $\mathcal{S} \subset \mathbb{R}^{27}$ includes joint states, torso position, and contact forces, and the action space $\mathcal{A} \subset \mathbb{R}^8$ represents joint torques in $[-1, 1]$. Let $v_x = (x_{\text{after}} - x_{\text{before}})/\Delta t$ be the forward velocity of the agent along the x-axis, where x_{after} and x_{before} are x-positions of the torso. Let $v_y = (y_{\text{after}} - y_{\text{before}})/\Delta t$ be the forward velocity of the agent along the y-axis, where y_{after} and

y_{before} are y-positions of the torso. Let c_{ctrl} be the positive control cost, computed as $w_{\text{env\_ctrl}} \sum_j (a_j)^2$, where $w_{\text{env\_ctrl}}$ is the environment control cost weight (typically 0.05). Let r_{healthy} be the health reward (typically $+1$ if the Ant is healthy). Let p_{contact} be the positive contact penalty, which is used for penalising the Ant if the external contact forces are too large, computed as $w_{\text{env\_contact}} \sum_k (\text{force}_k)^2$, where $w_{\text{env\_contact}}$ is the environment contact cost weight (typically 5×10^{-4}). The reward vector $\mathbf{R} = [R_1, R_2]$ is defined as:

- R_1 (Adjusted x-Velocity Performance):
$$R_1 = v_x + r_{\text{healthy}} - c_{\text{ctrl}} - p_{\text{contact}}$$
- R_2 (Adjusted y-Velocity Performance):
$$R_2 = v_y + r_{\text{healthy}} - c_{\text{ctrl}} - p_{\text{contact}}$$

B.2 Evaluation Metrics

We evaluate the quality of the approximate Pareto front using three standard metrics, following the formalism in [Zitzler and Thiele, 2002, Zintgraf et al., 2015, Hayes et al., 2022].

Hypervolume (HV). Let P be an approximate Pareto front and r a reference point dominated by all $p \in P$. The hypervolume is $\mathcal{H}(P) = \int_{\mathbb{R}^d} \mathbb{1}_{H(P)}(z)\, dz$, where $H(P) = \{\, z \in \mathbb{R}^n \mid \exists\, j,\ 1 \leq j \leq |P| \ :\ G_0 \preceq z \preceq P(j) \}$. Here, $P(j)$ is the j^{th} solution in P, the symbol $\preceq$ denotes objective dominance, and $\mathbb{1}_{H(P)}$ is an indicator function that equals 1 if $z \in H(P)$ and 0 otherwise. A higher hypervolume implies a front closer to and more extensive with respect to the true Pareto front.

Expected Utility (EU). Let P be an approximate Pareto front and Π be the corresponding policy set. The expected utility metric is $\mathcal{U}(P) = \mathbb{E}_{\omega \sim \Omega}\big[\max_{\pi \in \Pi}\ \omega^\top G^\pi_\omega\big]$. A higher EU denotes better average performance over preferences.

Sparsity (SP). Let P be an approximate Pareto front in a d-dimensional objective space. The sparsity metric is $S(P) = \frac{1}{|P|-1} \sum_{i=1}^{d} \sum_{k=1}^{|P|-1} \big(\tilde{G}_i(k) - \tilde{G}_i(k+1)\big)^2$, where $\tilde{G}_i$ is the sorted list of the i^{th} objective values in P, and $\tilde{G}_i(k)$ is the k^{th} entry in this sorted list. Lower sparsity indicates a more uniform distribution of solutions along each objective.

B.3 Training Details

All learning phases within our LLE-MORL algorithm, including the initial training of base policies, the directional retraining, and the final preference-aligned fine-tuning, utilize the Proximal Policy Optimization (PPO) algorithm [Schulman et al., 2017]. We employed a standard PPO implementation from the Stable Baselines3 library [Raffin et al., 2019]. The PPO parameters used across all training stages and benchmarks are detailed in Table 3.

The specific parameters for the LLE-MORL pipeline include:

- **Number of Initial Base Policies** (K): The total count of base policies θ_{w_i}, trained in the initialization stage. The corresponding K initial scalarization weights $\{w_i\}_{i=1}^{K}$ are generated by evenly distributing them across the preference space (e.g., for 2D objectives, from $[1, 0]$ to $[0, 1]$ in K steps).
- **Initialization Training Timesteps** (T_{init}): The number of environment interaction steps for which the initial base policy θ_{w_i} is trained under its weight w_i.
- **Retraining Preference Shift Strategy (controlled by shift magnitude δ_s)**: Target scalarization weights $\{w'_i\}$ for directional retraining are generated by shifting each initial weight w_i to a nearby, distinct point on the preference space. The extent of this shift is controlled by a hyperparameter δ_s. Conceptually, for d-dimensional preference spaces ($d > 2$), this shift could be defined as an angular displacement in the space. In our current two-dimensional objective experiments ($d = 2$), where $w_i = [w_{i,0}, w_{i,1}]$, this shift is implemented by moving the first component $w_{i,0}$ by the magnitude δ_s to obtain $w'_{i,0}$. The shift direction (decrease or increase) is chosen to keep the component within valid bounds (e.g., $[0, 1]$), and the default direction is decrease; then the second component $w_{i,1}$is adjusted accordingly (assuming all objective weights sum to 1).

- **Directional Retraining Timesteps** (T_{dir}): The number of environment interaction steps for which the base policy θ_{w_i} is retrained under its target weight w'_i to produce $\theta_{w'_i}$.
- **Step-Scale Factor Generation** ($\alpha_{\textbf{start}}, \alpha_{\textbf{end}}, \Delta\alpha$): The set of step-scale factors $\{\alpha_j\}$ used in Locally Linear Extension is generated based on a starting value (α_{start}), an ending value (α_{end}), and either a step increment ($\Delta\alpha$).
- **Fine-tuning Timesteps** (T_{ref}): The number of environment interaction steps for which the selected candidate policy from the extension phase is fine-tuned under its matched preference weight $w_{i,j}$.

The specific values for these LLE-MORL parameters, are provided in Table 4 and Table 5.

Parameter Name	MO-Swimmer	MO-Hopper-2d	MO-Ant-2d
steps per actor batch	512	512	512
learning rate ($\times 10^{-4}$)	3	3	3
learning rate decay ratio	1	1	1
γ	0.995	0.995	0.995
GAE lambda	0.95	0.95	0.95
number of mini batches	32	32	32
PPO epochs	10	10	10
entropy coefficient	0.0	0.0	0.0
value loss coefficient	0.5	0.5	0.5
maximum gradient norm	0.5	0.5	0.5
clip parameter	0.2	0.2	0.2

Table 3: PPO hyperparameters for benchmarks.

B.4 Experiments compute resources

All experiments were run on a workstation equipped with an AMD Ryzen Threadripper PRO 5975WX (32 cores), an NVIDIA GeForce RTX 3090 GPU (24 GB GDDR6X), and 256 GiB of RAM, running Ubuntu 24.04 LTS. The software stack included CUDA Toolkit 12.0 and the corresponding NVIDIA drivers. Approximate execution times for all methods and benchmarks are reported separately in Table 6.

C Limitations

Limitations of our approach are implied by inherent challenges in multi-objective optimisation, but we also note some limitations that are specific to our algorithm and require further study.

- We have restricted ourselves to problems with two objectives where the Pareto front is one-dimensional. A larger number of objectives is a problem for most of the existing MORL algorithms. Although usually some of the objectives are of different importance and can be lexicographically ranked, so that the complexity does not necessarily increase exponentially with the number of objectives. The high-dimensional case is nevertheless challenging, but our approach can be seen as promising: In higher dimensions, the number of solutions that are in the same local quasi-linear patch increases dramatically, so that the efficiency of the proposed local search will be even more beneficial. This benefit could be reduced by the potentially increasing complexity of the topological relation between the performance space and the parameter space which could be a fascinating subject for future work.
- We are assuming that the Pareto front consists of a relatively small number of connectivity components which have a manifold structure. While there is no theoretical bound to the complexity of the Pareto from, the idea of MORL implies that the objectives are at least in some sense comparable. For Pareto fronts that are fractal or of high genus, the result of multi-objective optimisation lacks robustness, although it will neither be possible to fix any limits for the complexity of the Pareto front. However, as long as there are only a limited number of manifold-like connectivity components, our algorithm will be applicable.
- Widely different scales and elasticities of the objectives can lead to problems as in optimisation in anisotropic error landscapes. Step size control that helps in gradient methods in

Parameter Name	**Symbol**	**MO-Swimmer**	**MO-Hopper-2d**	**MO-Ant-2d**
Number of base policies	K	6	6	6
Initialization timesteps	T_{init}	1×10^5	1×10^5	1×10^5
Preference shift magnitude	δ_s	0.1	0.1	0.1
Directional retraining timesteps	T_{dir}	1×10^4	1×10^4	1×10^4
Step-scale start	α_{start}	-1.5	-1.5	-1.5
Step-scale end	α_{end}	1.5	1.5	1.5
Step-scale increment	$\Delta\alpha$	0.05	0.05	0.05
Fine-tuning timesteps	T_{ref}	1×10^3	1×10^3	1×10^3

Table 4: Hyperparameters for the LLE-MORL across benchmarks under sample-efficient setting.

Parameter Name	**Symbol**	**MO-Swimmer**	**MO-Hopper-2d**	**MO-Ant-2d**
Number of base policies	K	6	6	6
Initialization timesteps	T_{init}	1×10^6	1×10^6	1×10^6
Preference shift magnitude	δ_s	0.1	0.1	0.1
Directional retraining timesteps	T_{dir}	1×10^4	1×10^4	1×10^4
Step-scale start	α_{start}	-1.5	-1.5	-1.5
Step-scale end	α_{end}	1.5	1.5	1.5
Step-scale increment	$\Delta\alpha$	0.05	0.05	0.05
Fine-tuning timesteps	T_{ref}	1×10^3	1×10^3	1×10^3

Table 5: Hyperparameters for the LLE-MORL across benchmarks under standard-training setting.

	Sample-efficient setting			Standard-training setting		
Method	MO-Swimmer	MO-Hopper-2d	MO-Ant-2d	MO-Swimmer	MO-Hopper-2d	MO-Ant-2d
GPI-LS	3	3	3	22	23	19
CAPQL	13	8	12	95	77	68
MORL/D	1	1	1	3	3	3
LLE-MORL	1	1	1	3	3	3

Table 6: Approximate execution times (hours) for each method and benchmark under sample-efficient and standard training setting.

optimisation, will also be useful here, but has not been studied yet, as the typical (benchmark) problems are sufficiently isotropic.

- The density of the identified solutions on the Pareto front is clearly a challenge which may be solved by step size control as mentioned in the previous point. This concerns higher-dimensional cases as well as extended one-dimensional trails as visible in the top trail in Figure 3c also a simple reduction of the parameter $\Delta\alpha$ at the observation of large steps in the performance space could have solved this issue already so that a more uniform covering of the Pareto from is not difficult to achieve in the present approach. See also App. E.2. In contrast to other approaches, linear regions of the Pareto can trivially be tracked by LLE-MORL. Concave regions connected to the Pareto front will be followed through without problem, but will need to be removed in a single postprocessing step as they are dominated by other solutions. Even full patches of solutions may turn out to be Pareto sub-optimal and require a similar treatment.
- We have made use of scalarisation to seed the solution domains, whereas the reconstruction of the Pareto front is done by a lateral process that does not depend on preference weights. It is in principle possible that a solution patch is not reachable by any scalarisation-based seeding attempt, see also the early discussion in [Vamplew et al., 2008]. In this case our approach might not find this patch, although it is still possible that it is found by retraining from a different solution domain as shown in Figure 3c.