ON GENERALIZATION WITHIN MULTI-OBJECTIVE REINFORCEMENT LEARNING ALGORITHMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world sequential decision-making tasks often require balancing trade-offs between multiple conflicting objectives, making Multi-Objective Reinforcement Learning (MORL) an increasingly prominent field of research. Despite recent advances, existing MORL literature has narrowly focused on performance within static environments, neglecting the importance of generalizing across diverse settings. Conversely, existing research on generalization in RL has always assumed scalar rewards, overlooking the inherent multi-objectivity of real-world problems. Generalization in the multi-objective context is fundamentally more challenging, as it requires learning a Pareto set of policies addressing varying preferences across multiple objectives. In this paper, we formalize the concept of generalization in MORL and how it can be evaluated. We then contribute a novel benchmark featuring diverse multi-objective domains with parameterized environment configurations to facilitate future studies in this area. Our baseline evaluations of state-of-the-art MORL algorithms on this benchmark reveals limited generalization capabilities, suggesting significant room for improvement. Our empirical findings also expose limitations in the expressivity of scalar rewards, emphasizing the need for multi-objective specifications to achieve effective generalization. We further analyzed the algorithmic complexities within current MORL approaches that could impede the transfer in performance from the single- to multiple-environment settings. This work fills a critical gap and lays the groundwork for future research that brings together two key areas in reinforcement learning: solving multi-objective decision-making problems and generalizing across diverse environments.

1 INTRODUCTION

Developing agents capable of generalizing across diverse environments is a central challenge in reinforcement learning (RL) research. While significant progress has been made in studying the generalizability of RL algorithms, these efforts predominantly focus on optimizing a single scalar reward signal (Zhang et al., 2018; Cobbe et al., 2019; Irpan & Song, 2019; Packer et al., 2019; Kirk et al., 2023). Single-objective RL (SORL) overlooks the complexity of real-world problems, which often necessitate trade-offs to be made between multiple conflicting objectives. Reducing these multifaceted considerations to a single scalar reward (objective) obscures critical interactions between the objectives and limits the agent's utility (Vamplew et al., 2022). The field of Multi-Objective Reinforcement Learning (MORL) has sought to address the inherent multi-objective nature of sequential decision-making tasks (Roijers et al., 2013; Hayes et al., 2022). However, the existing body of MORL research has concentrated on optimizing agent performance within *static environments*, neglecting the dimension of generalization across varying situations. Consequently, there exists a significant gap in the RL literature: the intersection of generalization and MORL.

Generalising over multiple scenarios and objectives simultaneously is routinely demanded in many real-world applications, such as healthcare management, autonomous driving, and recommendation systems. Consider an autonomous vehicle, which must not only generalize across varied environmental conditions—different weather patterns, lighting, and road surfaces—but also learn optimal trade-offs between competing objectives such as fuel consumption, travel time, passenger's comfort, and safety. Failure to effectively generalize across these environments and objectives would lead to

inefficient operation or even catastrophic outcomes. The real world's dynamic nature extends beyond just environmental variability, but also includes evolving goals and utility preferences. An agent optimizing a single scalar reward may exhibit some level of generalization, such as adapting to state variations, but it will struggle to generalize when faced with new goals or reward structures. This is because the agent has only observed its current reward signal, and lacks the basis for adapting its behaviour should the reward signal change. In contrast, a MORL agent learns to consider all dimensions of a vector reward, even those that are not immediately relevant to current goals. This holistic approach to learning allows the agent to adapt swiftly when its utility landscape evolves or when stakeholders' prioritisation over the different objectives shifts. For example, in autonomous driving, a generally capable MORL agent can satisfy unique preferences over objectives for different passengers without the need for retraining. Therefore, developing generally capable multi-objective agents enables not only generalization across *diverse environments*, but also across *dynamic goals and utility functions*—an overlooked aspect in current single-objective RL generalization literature, yet one that is arguably essential for real-world applicability.

The main contributions of this paper are threefold. First, we formalize a general framework for discussing and evaluating generalization in MORL. Second, we introduce a novel benchmark comprising diverse multi-objective domains with parameterized environment configurations, designed to facilitate rigorous investigations into the generalizability of MORL algorithms. Finally, we provide extensive baseline evaluations of state-of-the-art MORL algorithms using this benchmark. The empirical results highlight significant gaps in generalization capabilities of current MORL approaches. Additionally, the findings reveal limitations in using scalar specifications of rewards in RL generalization, underscoring the need for multi-objective rewards due to their better expressiveness. This paper lays the groundwork for future research on generalization in multi-objective domains, ultimately pushing the boundaries of what RL agents can accomplish in complex, real-world scenarios.

2 Background

In this section, we introduce MORL and establish the formal notations referenced throughout this paper. A multi-objective sequential decision-making problem can be modeled by a *Multi-Objective Markov Decision Process* (MOMDP; White (1982)) represented by the tuple: $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathbf{R}, \mu, \gamma \rangle$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, initial state distribution μ, and discount factor $\gamma \in [0, 1)$. The key distinction between MOMDPs and standard MDPs lies in the vector-valued reward function $\mathbf{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^k$, where k is the number of objectives. The goal of a standard RL agent is to maximize its expected long-term discounted sum of rewards, i.e. value function. For a stationary policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, the *multi-objective state value function* at state $s \in \mathcal{S}$ is given by

$$\mathbf{V}^{\pi}(s) := \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t \mathbf{R}(s_t, a_t, s_{t+1}) | s_0 = s\Big],$$

where $\mathbf{R}(s_t, a_t, s_{t+1})$ is the k-dimensional reward vector for the transition s_t, a_t, s_{t+1}. The expected value vector of π under the initial state distribution μ is defined as $\mathbf{v}^{\pi} = \mathbb{E}_{s_0 \sim \mu}[\mathbf{V}^{\pi}(s_0)]$. In MORL, each user expresses varying preferences over the objectives, which are modeled by a *utility (scalarization) function* $u : \mathbb{R}^k \to \mathbb{R}$ translating the expected vector reward $\mathbf{v}^{\pi}$ into a scalar utility, i.e. $\mathbf{v}_u^{\pi} = u(\mathbf{v}^{\pi})$. These utility functions are assumed to be *monotonically increasing* in every objective. This is a natural assumption in accordance with notions of reward – getting more reward for an objective should not decrease a user's utility as long as it does not result in a decrease in reward for another. Since there may be no single policy that satisfies every user's preference, unlike single-objective RL, MORL requires the agent to learn a *solution set* of policies, each reflecting different trade-offs across objectives. This leads to the concept of *Pareto dominance*. A policy π Pareto dominates (denoted by $\succ_P$) another policy π' if its expected value vector is higher or equal across all objectives, that is: $\mathbf{v}^{\pi} \succ_P \mathbf{v}^{\pi'} \iff (\forall i : v_i^{\pi} \geq v_i^{\pi'}) \wedge (\exists i : v_i^{\pi} > v_i^{\pi'})$. The *Pareto Set* consists of all nondominated (Pareto optimal) policies:

$$\mathcal{PS}(\Pi) = \{\pi \in \Pi \mid \nexists \pi' \in \Pi, \mathbf{v}^{\pi'} \succ_P \mathbf{v}^{\pi}\},$$

where Π is the set of all possible policies. The image of the Pareto set under the expected value function mapping is known as the *Pareto Front*. In MORL, there are two primary approaches: the

axiomatic approach and the *utility-based approach* (Roijers et al., 2013). The axiomatic approach operates on the axiom that the Pareto set must contain an optimal policy for any possible monotonically increasing utility function. Hence, they seek to derive the entire Pareto set without explicitly considering specific utility functions. On the other hand, the more prevalent utility-based approach considers classes of parameterized utility functions that can be expressed as $u_{\mathbf{w}}(\mathbf{v}^{\pi})$, where $\mathbf{w}$ is a weight vector parameterizing u. During training, utility-based agents learn optimal policies π^* that maximize the scalar utility for different $\mathbf{w}$ (or neighborhood of $\mathbf{w}$) across the weight space, i.e. $\pi^* = \arg\max_{\pi \in \Pi} u_{\mathbf{w}}(\mathbf{v}^{\pi})$. Linear scalarization functions with weights summing to unity, i.e. $u_{\mathbf{w}}(\mathbf{v}^{\pi}) = \mathbf{w}^{\intercal}\mathbf{v}^{\pi}$ where $\sum_i w_i = 1,\ w_i \geq 0,\ i = 1, \ldots, k$, are commonly employed. Axiomatic approaches and utility-based approaches are two sides of the same coin: both perspectives seek to derive optimal trade-offs across the entire Pareto front. However, the utility-based approach puts the user preferences at the forefront, making it easier for users to select policies based on their preferences and allowing constraints to be placed on the solution set. We will use the terms "utility function" and "scalarization function" interchangeably throughout this paper.

3 Multi-Objective Contextual Markov Decision Process

To formalize the notion of generalization in the context of MORL, we need to start with a way to reason about a collection of multi-objective environments. In single-objective RL (SORL), this is often done using the Contextual MDP (CMDP; Hallak et al. (2015)) [1] framework. As such, we formally define a *Multi-Objective Contextual Markov Decision Process* (MOC-MDP) – an adaptation of the CMDP framework to the multi-objective setting.

Definition 1 (Multi-Objective Contextual MDP)**.** A MOC-MDP is defined by the tuple

$$\langle \mathcal{C}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mu, \gamma, \mathcal{M} \rangle$$

where $\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}$ are as in the definition of the MOMDP. $\mathcal{C}$ is the *context space* and $\mathcal{M}$ is a function mapping any $c \in \mathcal{C}$ to a MOMDP, i.e. $\mathcal{M}(c) = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}^c, R^c, \mu^c, \gamma \rangle$.

The context space, $\mathcal{C}$ defines a set of *static* parameters, each representing a different MOMDP. Intuitively, the context can be viewed as a discrete or continuous parameter specifying the multi-objective environment configuration, such as a seed or a vector controlling the environment dynamics. Each configuration varies in its initial state distribution, transition functions and multi-objective rewards, but share enough common structure across which the MORL agent can generalize. MOC-MDP describes a model where for each context there is a potentially distinct optimal Pareto front. Throughout this paper, we will also refer to a particular MOC-MDP as a "domain", and its associated "contexts" as "environments", interchangeably. For a given MOC-MDP M, the expected multi-objective value vector of a policy π across all contexts is

$$\mathbf{v}_{\mathcal{C}}^{\pi} = \mathbb{E}_{c \sim p(c)}\left[\mathbf{v}_c^{\pi}\right] = \mathbb{E}_{c \sim p(c)}\Big[\mathbb{E}_{s_0 \sim \mu^c}\left[\mathbf{V}_c^{\pi}(s_0)\right]\Big],$$

where $p(c)$ is the context distribution, $\mathbf{v}_c^{\pi}$ denotes the expected value vector under μ^c, and $\mathbf{V}_c^{\pi}(s)$ represents the multi-objective state value function for the MOMDP mapped by context c. We begin by formalizing the generalization objective for axiomatic MORL approaches. The main objective of the axiomatic approach lies in identifying all nondominated vectors across the Pareto front that is optimal for any monotonically increasing scalarization function. In the case of a MOC-MDP, since there are different Pareto fronts for each context, to attain optimality in any scalarization function for any context, it would involve a union of policies from Pareto sets across contexts. Collectively, these policies form a *global Pareto set*.

Definition 2 (Generalization in Axiomatic MORL Approaches)**.** Given a MOC-MDP M with policy space Π, the generalization problem for axiomatic approaches is to learn a *global Pareto set*:

$$Global\ Pareto\ Set = \{\pi \in \Pi \mid \exists c \in \mathcal{C}, \nexists \pi' \in \Pi,\ \mathbf{v}_c^{\pi'} \succ_P \mathbf{v}_c^{\pi}\},$$

where $\mathbf{v}_c^{\pi}$ is the expected value vector in a context c. Thus, the global Pareto set comprises of policies that are nondominated in at least one context, ensuring that all necessary policies for constructing the Pareto Fronts in every context are captured.

[1] Not to be confused with Constrained MDPs.

However, learning the global Pareto set in axiomatic approaches can pose challenges. First, the axiomatic goal lacks specifications on the properties of the policies that is learned in the global Pareto set. This permits the learning of disjoint sets of Markovian policies optimized for individual contexts, which resembles "memorization" rather than generalization. Moreover, when the context is partially observable at test time, it is unclear how the agent should select the subset of policies corresponding to the Pareto front of the context. Evaluating all policies within the global Pareto set would be intractable. The utility-based approach offers a more structured path for formalizing generalization in MORL. Recall in utility-based approaches, each user's preference is modeled by a utility function $u_{\mathbf{w}}(\cdot)$ parameterized by a weight vector $\mathbf{w}$. In a MOC-MDP, the optimal policy for a given $\mathbf{w}$ may vary across contexts. Thus, utility-based approaches would aim to find policies that maximize the expected utility across the context distribution for each $\mathbf{w}$.

Definition 3 (Generalization in Utility-based MORL Approaches)**.** For each weight $\mathbf{w}$, the generalization problem for utility-based approaches is to find an optimal policy $\pi^*_{\mathbf{w}}$ such that

$$\pi^*_{\mathbf{w}} = \arg\max_{\pi \in \Pi} \mathbb{E}_{c \sim p(c)} \left[u_{\mathbf{w}}(\mathbf{v}^{\pi}_c) \right]$$

In this framework, each policy must generalize across contexts for a specific utility function parameterized by $\mathbf{w}$, keeping the size of the agent's policy set bounded by the weight space. This structure also permits for clear specification of policies by users via $\mathbf{w}$ during test time, regardless of the context. Besides it practical benefits, the utility-based approach aligns more closely with the concept of generalization. Since the agent only learns a single policy per preference weight, for each policy to maximize its corresponding utility function in any context, even those unseen from its training distribution, the agent would likely have to find a way to learn and leverage the shared structure across contexts in the MOC-MDP.

4 Empirical Evaluation of MORL Generalization Performance

In this section, we propose an evaluation protocol for generalization performance in MORL and discuss important considerations. Let $\mathcal{C}_{\text{eval}} = \{c_1, c_2, \ldots, c_n\}$ represent a set of independent evaluation contexts. Measuring an agent's generalization performance in SORL is straightforward: the larger the reward value across $\mathcal{C}_{\text{eval}}$, the better. In MORL, however, agents produce an approximate Pareto front comprising multiple value vectors for each $c \in \mathcal{C}_{\text{eval}}$, and translating the quality of this Pareto front into a scalar metric that captures generalization performance is non-trivial. The *Hypervolume* indicator (Zitzler & Thiele, 1998) is widely used to evaluate the quality of approximate Pareto fronts in single-environment MORL. It measures the volume in the objective space occupied by a Pareto front, relative to a reference point. However, the Hypervolume indicator is inherently not scale-invariant, and biases evaluations towards objectives with larger magnitudes. While normalization has been discussed in multi-objective optimization (MOO) (Deb & Kalyanmoy, 2001), it has been overlooked in the MORL literature. To ensure fair and reliable generalization evaluations, we first propose the *Normalized Hypervolume* (HV_{norm}).

Definition 4 (Normalized Hypervolume)**.** Let $\tilde{\mathcal{F}}_c \subset \mathbb{R}^k$ be an approximate Pareto front in a k-dimensional objective space for context c. The Normalized Hypervolume (HV_{norm}) is defined as:

$$\text{HV}_{\text{norm}}(\tilde{\mathcal{F}}_c) = \lambda_k\Big(\bigcup_{\mathbf{v} \in \mathcal{N}_c} [\mathbf{v}, \mathbf{0}] \Big),$$

where λ_k is the k-dimensional Lebesgue measure (Lebesgue, 1902) and $\mathcal{N}_c$ is the normalized Pareto front. The Pareto front $\tilde{\mathcal{F}}_c$ is normalized to $\mathcal{N}_c$ by linearly mapping each objective dimension to the range $[0, 1]$, using the minimum and maximum achievable values for that objective in the corresponding objective space. Since the objectives are normalized, we can use the origin $\mathbf{0}$ as the reference point, eliminating the need for a priori knowledge of an appropriate reference point.

The use of HV_{norm} also enhances interpretability as it is bounded within 0 and the hypervolume of the unit hypercube (which is 1). However, in practice, determining the true optimal Pareto front and its boundary values for normalization is difficult, especially in continuous domains. To approximate the optimal Pareto front for each context, we can combine the nondominated value vectors from a set of specialist agents trained independently on that specific context, forming a *combined specialist Pareto front*. The approximate normalization vectors for a context c, denoted as $\mathbf{v}^c_{\min}$ and $\mathbf{v}^c_{\max}$, are then

derived from the boundary values of the combined specialist Pareto front for c. For each objective $i \in \{1, \cdots, k\}$ and $c_j \in \mathcal{C}_{\text{eval}}$, we measure: $v^{c_j}_{\text{min},i} = \min_{\pi \in \Pi_{\text{ind}}} V^{\pi}_i(c_j)$, $\quad v^{c_j}_{\text{max},i} = \max_{\pi \in \Pi_{\text{ind}}} V^{\pi}_i(c_j)$, where $V^{\pi}_i(c_j)$ is the discounted return of a policy π on objective i in context c_j, and Π_{ind} is the set of Pareto optimal policies obtained by specialist agents trained on c_j. With these normalization bounds for calculating HV_{norm} established, we introduce a novel metric to evaluate the generalization performance of MORL agents called the *Normalized Hypervolume Generalization Ratio* (NHGR):

Definition 5 (Normalized Hypervolume Generalization Ratio)**.** Let $\mathcal{N}^{\text{gen}}_{c_j}$ and $\mathcal{N}^{\text{spec}}_{c_j}$ be the normalized Pareto front obtained by the generalist MORL agent and the normalized combined specialist Pareto front on context c_j respectively. The NHGR for c_j is defined as:

$$\text{NHGR}(\mathcal{N}^{\text{gen}}_{c_j}, \mathcal{N}^{\text{spec}}_{c_j}) = \frac{\text{HV}_{\text{norm}}(\mathcal{N}^{\text{gen}}_{c_j})}{\text{HV}_{\text{norm}}(\mathcal{N}^{\text{spec}}_{c_j})}.$$

NHGR measures the ratio of normalized hypervolume between the generalist and specialist agents. When a generalist agent attains NHGR=1 across all $c_j \in \mathcal{C}_{\text{eval}}$, it has recovered the performances of specialists optimized for each context. NHGR draws similarities to the *Hyperarea Ratio* (Veldhuizen & Allen, 1999) metric in MOO literature but additionally employs normalization before calculating hypervolume to ensure scale-invariance.

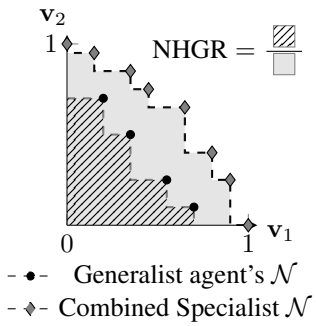


Figure 1: NHGR visualized as the ratio between hypervolume of the normalized generalist and combined specialist Pareto fronts (dashed and shaded areas).

Fig. 1 illustrates the NHGR metric for a biobjective domain. The NHGR metric is intuitive because a generally-capable MORL agent should ideally achieve a Pareto front of comparable quality to that of agents specialized for each context. While collecting specialist performances is tedious, it is essential for accurate generalization evaluations. Without the combined specialist Pareto front as a reference, evaluations would be biased towards agents that excel in contexts with convex-like Pareto fronts and higher hypervolume, while penalizing those that divide their learning across all contexts, even those with concave Pareto fronts. That contradicts the motivations of generalization. NHGR resolves this issue by evaluating generalist performance as a ratio of an approximated maximal achievable one, ensuring every evaluation context is fairly considered. Table 1 shows the mean performance of the GPI-PD (Alegre et al., 2023) algorithm on 3 environments in the MO-HalfCheetah domain (discussed more in Section 5). As shown, the raw hypervolume metric results in differences in performance between environments in the magnitude of $10e^4$. This is because slight differences in reward ranges compounds in hypervolume with every added dimension, resulting in lack of interpretability. Meanwhile, the HV_{norm} score heavily penalizes the performance of the agent in the *Hard* environment, and the NHGR metric offers the most balanced assessment by taking the ratio over the specialists' achieved scores.

	Default	Hard	Slippery
Hypervolume	$1.7e^5$	$6.1e^4$	$1.3e^5$
HV_{norm}	0.25	0.048	0.19
NHGR	0.40	0.11	0.34

Table 1: Illustration of different metrics on 3 MO-HalfCheetah environments.

5 MORL Generalization Benchmark

In this section, we introduce a novel benchmark featuring a diverse set of multi-objective domains with rich environmental variations to facilitate the future study of generalization in MORL algorithms. We adapted existing domains from MO-Gymnasium (Felten et al., 2023), a multi-objective extension of the Gymansium (Towers et al., 2024; Brockman et al., 2016) library, and introduced new ones, each with expressive parameters controlling environmental variations. Kirk et al. (2023) identified four key types of domain variations for studying generalization: 1) state-space variation (S), which alters the initial state distribution, 2) dynamics variation (D), which alters the transition function, 3) visual variation (O), which impacts the observation function, and 4) reward function variation (R). This benchmark primarily focuses on state-space and dynamics variations. Observation variations do not alter the underlying MOMDP structure (Du et al., 2019). Hence, they provide

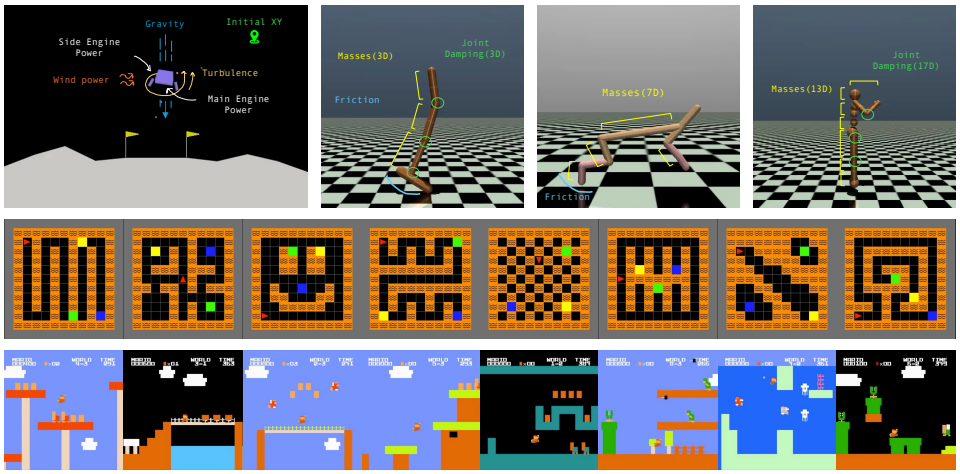


Figure 2: Domains in the MORL Generalization benchmark. Top row from left to right: 1) MO-LunarLander, 2) MO-Hopper, 3) MO-Cheetah, 4) MO-Humanoid. Middle row: MO-LavaGrid (8 handcrafted evaluation environments). Bottom row: MO-SuperMarioBros (8 out of 32 stages).

limited insights into the multi-objective decision-making capabilities of the agent since the optimal Pareto front across variations remain isomorphic. Reward variations are often introduced through multiple goals. Multiple goals can naturally be modeled as multiple objectives by treating each goal as a conflicting objective (Sener & Koltun, 2018), which means MORL inherently involves learning to adapt to reward function variations. Nevertheless, we provided a novel maze domain that explicitly segregates the goals and multiple objectives, for posterity. Fig. 2 visualizes the domains provided in the benchmark with annotations for their environment parameters, where applicable. We provide brief descriptions of the domains below, and leave the rest to Section F of the appendix:

MO-LunarLander (D+S) This is a multi-objective adaptation of Gymnasium's *LunarLander* domain where the agent has to balance between successfully landing on the moon surface, the stability of the spacecraft, the fuel cost of the main engine, and the fuel cost of the side engine. The agent operates over discrete-action and continuous-observation spaces. We introduce a 7-dimensional parameter that varies the environment's gravity, wind power, turbulence, and the lander's main engine power, side engine power, and initial x, y coordinates.

MO-Hopper (D) This is a multi-objective adaptation of Gymnasium's *Hopper* domain. The one-legged agent must balance optimizing for its forward velocity, torso height, and energy cost. The agent operates over continuous action and observation spaces. We introduce an 8-dimensional parameter that varies the hopper's body masses (4D), joint damping (3D), and the floor's friction.

MO-HalfCheetah (D) This is a multi-objective adaptation of Gymnasium's *HalfCheetah* domain. The 2-dimensional cheetah robot must balance optimizing for its forward velocity and energy cost. The agent operates over continuous action and observation spaces. We introduce an 8-dimensional parameter that varies the cheetah's body masses (7D) and the floor's friction.

MO-Humanoid (D) This is a multi-objective adaptation of Gymnasium's *Humanoid* domain. The humanoid robot must balance between optimizing for its forward velocity and its energy cost. The agent operates over continuous action and observation spaces. We introduce a 30-dimensional environment parameter that varies the humanoid's body masses (13D) and joint damping (17D).

MO-SuperMarioBros (S) This is a multi-objective adaptation of the *Gym Super Mario Bros* (Kauten, 2018) domain based on the popular Super Mario Bros video game. The agent has to balance between moving forward, collecting coins, and increasing the game score (by stomping enemies, breaking bricks, etc.). The agent operates over discrete-action and discrete-observation (pixel images) spaces. We introduce a 2-dimensional parameter that controls which stage of the Super Mario Bros game to place the agent in. There are a total of 32 possible stages.

MO-LavaGrid (S+R) This is a novel multi-objective domain based on *MiniGrid* (Chevalier-Boisvert et al., 2023). The agent (red triangle) has to navigate a 11 x 11 grid, incurring a penalty each

time it touches lava and another for every step it takes to collect all 3 goals (blue, green, and yellow blocks), after which the episode terminates. The placements of the agent, goals and lava blocks are fully controllable. We also introduce a 3-dimensional parameter that controls the reward weight of each goal. These weights are concatenated to the state space, allowing the agent to optimise its trajectory, while balancing between lava damage and collecting all goals. The agent operates over discrete-action and mixed continuous-discrete (because of the reward weights) observation spaces.

6 EXPERIMENTS

In this section, we evaluate state-of-the-art MORL algorithms on the newly-developed benchmark to establish baseline expectations for their generalization capabilities. The implementations of these algorithms are adapted from Felten et al. (2023). Specifically, the algorithms evaluated are CAPQL (Lu et al., 2023), Envelope (Yang et al., 2019), GPI-LS (Alegre et al., 2023), PCN (Reymond et al., 2022), PGMORL (Xu et al., 2020), and MORL/D SB (Felten et al., 2024). We also include the model-based extension of GPI-LS, i.e. GPI-PD, and the weight adaptation variant of MORL/D SB, i.e. MORL/D SB+PSA. Note that Envelope is limited to discrete-action domains, and both CAPQL and PGMORL are limited to continuous-action domains. Additionally, we include the SAC (Haarnoja et al., 2018) algorithm trained with a single objective/utility function in our evaluations to verify that the objectives are not so highly correlated that a single-objective agent could also achieve high performance across multiple objectives. In total, we evaluate 8 MORL algorithms across 6 domains using 5 seeds each, requiring approximately 933 GPU hours. These established baseline performances will be open-sourced via *Weights and Biases* (Biewald, 2020), facilitating future research and saving computational resources.

Domain Randomization (DR) is an efficient method to expose the agent to a wide range of environments during training by uniformly sampling from the environment parameter space. It has found success in deep RL even for complex visual domains and real-world robotic control (Tobin et al., 2017; Peng et al., 2018). We utilise DR for all our experiments by randomizing the environment parameters after every training episode. This also enables us to standardise the presentation of environments across algorithms via the RNG seed, and evaluate the algorithms solely for their generalization capabilities. At each evaluation time step, each algorithm is assessed over 100 episodes [2] across a set of environment configurations. Whenever possible, these configurations are chosen using the boundary values of environment parameter ranges to ensure diverse evaluation environments and behaviors. For MORL algorithms using linear scalarization, weights are sampled equally across the unit simplex during each evaluation episode. We aggregate the NHGR performance across all evaluation environments for each domain and report results in terms of inter-quartile mean (IQM) and optimality gap using the `rliable` library (Agarwal et al., 2021), which helps account for statistical uncertainty prevalent in deep RL. IQM focuses on the middle 50% of combined runs, discarding the bottom and top 25%. Optimality gap captures the amount by which the algorithm fails to meet a desirable target, i.e. when NHGR=1. The evaluation environment configurations, hyperparameters and other experiment setups are detailed in Section F of the appendix for reproducibility.

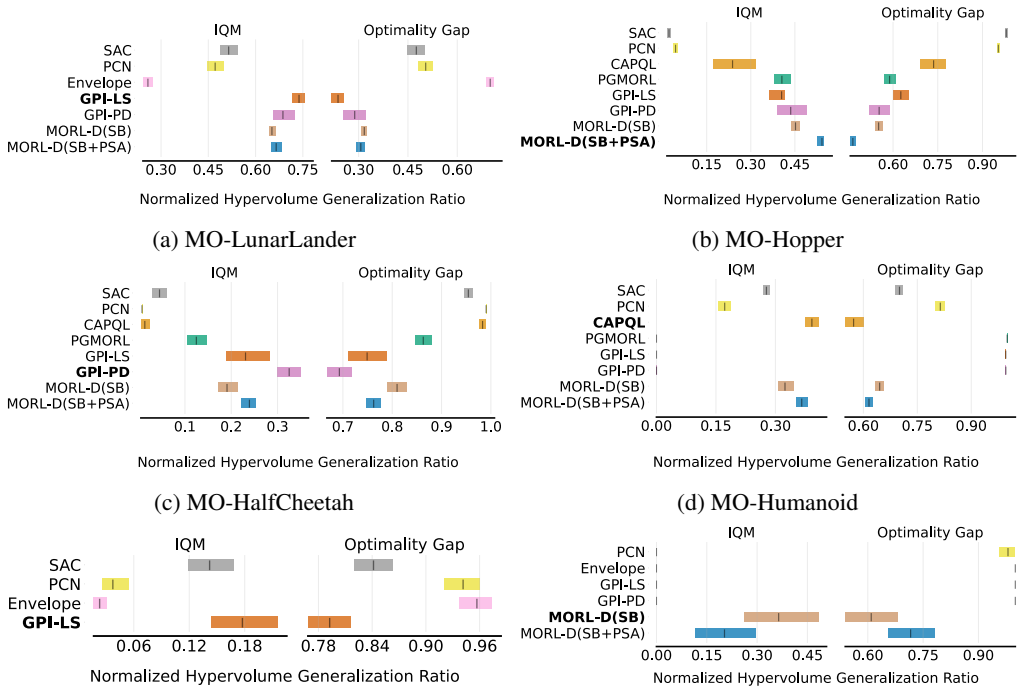
6.1 MORL GENERALIZATION RESULTS

The baseline results reveal significant performance gaps between specialist and generalist agents (via NHGR) across various domains, highlighting the benchmark's potential to serve as a foundational benchmark for future research in MORL generalization.

MO-LunarLander In MO-LunarLander (see Fig. 3a), most algorithms achieved a mean IQM NHGR score within the range of $\sim$0.65-0.75. Given this performance saturation, we recommend using this domain only as a starting point for testing and rapid iteration when developing new approaches, as its low-dimensional observation and discrete action spaces enable a relatively shorter training horizon compared to other domains. For testing more significant algorithmic advancements, exploring the continuous-action variant of this domain may reveal larger NHGR optimality gaps and greater opportunities for improvement.

Mujoco-based Domains The challenge of MORL generalization becomes more pronounced in the Mujoco-based (Todorov et al., 2012) domains, as shown in Figures 3b, 3c and 3d. Across the 3

[2] In MO-SuperMarioBros, 32 evaluation episodes are used to keep runtime under 120 hours.



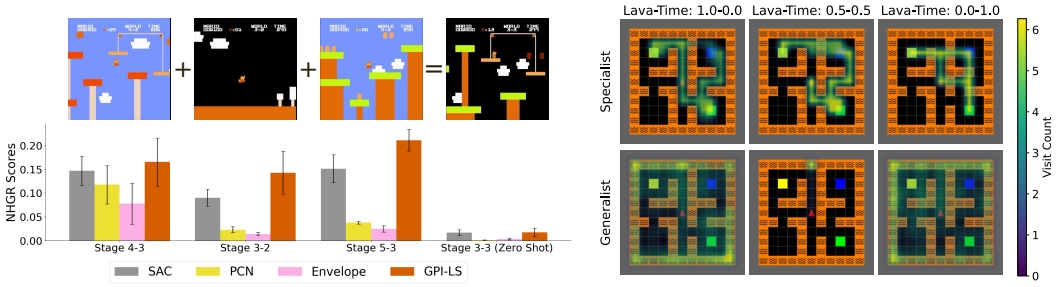
(a) MO-LunarLander (b) MO-Hopper

(c) MO-HalfCheetah (d) MO-Humanoid

(e) MO-SuperMarioBros (f) MO-LavaGrid

Figure 3: Aggregate NHGR performance in all domains of the benchmark. Each algorithm is evaluated across 5 independent seeds and several evaluation environment configurations. Higher IQM and lower optimality gap scores are better. The best algorithm for each domain is bolded.

domains, a wider spread in performances and noticeably lower performance ceilings are observed. In the MO-Hopper domain, the leading algorithm, MORL-D(SB+PSA), managed to reach a mean IQM NHGR score of only $\sim$0.53, resulting in a substantial 68% optimality gap. This gap further intensifies in the higher-dimensional domains, i.e. MO-HalfCheetah (Fig. 3c) and MO-Humanoid (Fig. 3d), where the leading algorithms, GPI-PD and CAPQL, achieved mean IQM NHGR scores of only $\sim$0.32 and $\sim$0.35, respectively. These low performance ceilings are expected, given the notorious difficulty of generalization in continuous control tasks already established in SORL. These wide optimality gaps, combined with the strong relevance to real-world robotic control tasks, suggest that these domains may serve as enduring benchmarks for studying generalization in MORL.



(a) MO-SuperMarioBros Performances (b) Heatmap of visited tiles

Figure 4: (a) MO-SuperMarioBros performances on 4 stages. Stage 3-3 in the rightmost column shares visual similarities with the other stages so it is excluded from training to evaluate for ZSG. (b) Heatmap of visited tiles for a specialist and generalist in the MO-LavaGrid "Room" environment. Each column's title shows the conditioned linear weights for the lava and time penalty objectives.

MO-SuperMarioBros Fig. 3e presents the NHGR performance of 3 discrete MORL algorithms and SAC on MO-SuperMarioBros. MORL/D SB and MORL/D SB+PSA were excluded due to the high GPU memory demands of using convolutional neural networks in this domain, which is incompatible with their evolutionary approach. The leading algorithm, GPI-LS, achieved a mean IQM NHGR score of only $\sim$0.18. We also conducted a *zero-shot generalization* (ZSG) experiment by excluding Stage 3-3 from the training distribution. This stage shares a combination of visual features with Stages 3-2, 4-3, and 5-3, allowing us to test if an agent has learned generalizable behaviors over the pixel space or merely memorized stage-specific sequences. The results in Fig. 4a show a steep decline in NHGR performance in the zero-shot environment, suggesting the latter.

MO-LavaGrid Fig. 3f shows the evaluation performance of 5 discrete MORL algorithms on MO-LavaGrid, with MORL/D SB achieving the highest mean IQM NHGR score, albeit still far from optimality. We recorded multiple trajectories for a generalist agent (MORL/D SB) and a specialist agent (GPI-LS) in the "Room" environment of MO-LavaGrid, both of which uses linear scalarization. Fig. 4b displays heatmaps of visit counts for each tile when the specialist and generalist were conditioned on three different linear weights (for lava and time penalty objectives). The specialist consistently takes optimal routes for each weight, while the generalist exhibits random walks overlapping with the three goals. This likely explains MORL/D SB's nonzero NHGR performance across environments but significant optimality gap, as it incurs high penalties from inefficient navigation.

In summary, the generalization performance of the current MORL algorithms leaves much to be desired. This outcome is not surprising, as these experiments were aimed to provide a baseline understanding of existing methods without any tailored interventions to enhance generalization yet. Despite not attaining the top performance in every domain, MORL/D SB and MORL/D SB+PSA, demonstrated the most consistent results overall. Future research aiming to improve MORL generalizability can consider building upon these algorithms for more reliable testing.

6.2 Scalar Reward is Not Enough for RL Generalization

Real-world problems are often multi-objective. In fact, many popular SORL benchmarks are inherently multi-objective but are simplified with hidden scalarization functions. For example, the original Hopper domain's combines forward velocity (v_x), control cost (c), and a bonus for not falling (h) into a scalar reward: $1.5v_x + 0.001c + h$. In contrast, MORL treats these as independent objectives, and occasionally adds objectives like torso height that are superfluous to the goal of the SORL agent. One might argue that if a stakeholder's sole goal is for the agent to move forward, utility-based MORL approaches that seek to maximise multiple utility functions might be redundant in RL generalization. However, our empirical results challenge this assumption.

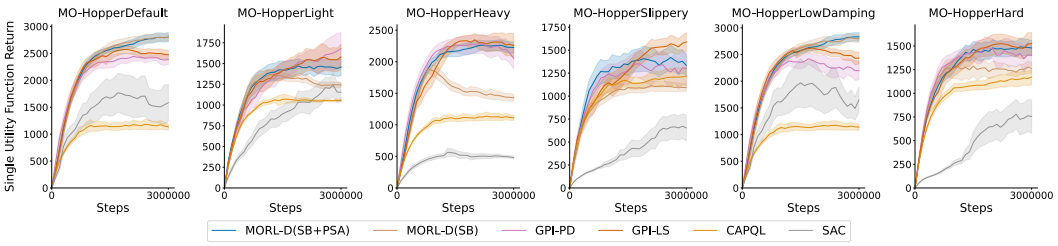
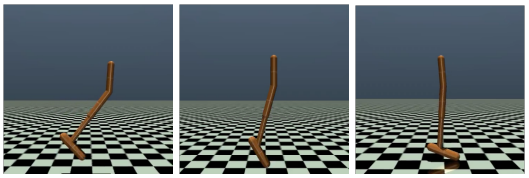


Figure 5: Single-objective return on 6 MO-Hopper testing environments during training. Each curve is measured across 5 seeds (mean and standard error).

Let f_{SORL} denote the fixed scalar utility function that SORL agents are trained to optimise during generalization. Throughout the generalization training horizon of the MORL algorithms in Section 6.1, we sampled solution vectors across their Pareto front and scalarized them using f_{SORL}, and recorded the highest scalar utility for each evaluation environment. For the SORL agent, which already specializes on f_{SORL}, we allow it as many runs as solution vectors sampled from the MORL agents and take the best result. Our results reveal that when trained on the same generalization procedure, leading MORL algorithms can actually outperform the SORL agent on its specialized utility function, i.e. f_{SORL}. Fig. 5 shows several MORL algorithms surpassing the SAC agent on f_{SORL} return by large margins across six distinct environment variations during generalization training in the MO-Hopper domain. Note that CAPQL is a multi-objective variant of SAC, while MORL/D

SB and MORL/D SB+PSA is population-based approach of SAC. All SAC-based implementations are from the same library, CleanRL (Huang et al., 2022), making these results fair. Similar findings are observed in other domains (see Section D.3 in appendix), where leading MORL algorithms consistently outperform or achieved parity with SAC on f_{SORL} performance. Sl

Fig. 6 presents snapshots from the highest f_{SORL} episodes of the MORL/D SB+PSA agent on three MO-Hopper environments. Since MORL/D SB+PSA is a linear utility approach, the table in Fig. 6 provides the weight vectors which the agent conditioned on. In the *Default* environment, the agent placed a higher weight on forward velocity, causing it to lean forward and cover more distance. In the *Slippery* environment, where the friction coefficient is minimal, leaning forward would make the agent topple over. Instead, MORL/D SB+PSA maximised f_{SORL} when balancing between the forward velocity and torso height, thereby maintaining an upright posture to prevent slipping. In the *Hard* environment, which features a slippery floor, unbalanced body masses, and low joint damping, the agent maximized f_{SORL} by prioritizing torso height and minimizing control cost. This allows the agent to maintain stability and reduce abrupt movements. In contrast, the single-objective SAC agent only learnt a single behavior to generalise across all environments, causing it to fail in dire conditions.



(a) Default (b) Slippery (c) Hard

	Default	Slippery	Hard
forward velocity	**0.84**	0.52	0.3
torso height	0.09	**0.4**	0.23
control cost	0.07	0.08	**0.47**

Figure 6: Screenshots of MORL/D SB+PSA agent's behavior in different MO-Hopper environments and the corresponding linear weights.

The single-objective approach to RL generalization is heavily reliant on *reward engineering*, i.e. finding an optimal scalar reward signal through trial-and-error search of scalarization functions (Sutton & Barto (2018), Chapter 17.4). However, the observations above highlight that there may be no universal scalarization function which optimizes reward signal during generalization. Each environment demands distinct behaviors from the agent to maximize performance, even for a fixed goal like moving forward. Consequently, a priori scalarization in SORL limits the agent's flexibility to adapt its behavior to environmental changes. In contrast, generalization with MORL approaches circumvents the reward engineering problem by considering all dimensions of a vector reward independently, even those not immediately relevant to current goals. This allows agents to learn diverse behaviors for different tradeoffs among objectives. Stakeholders can then select policies from the agent's Pareto set that best maximize their utility function for any given environment, enhancing the adaptability of MORL agents in generalization tasks. These observations align with recent studies that challenge the expressivity of scalar rewards and advocate for the adoption of multi-objective reward formulations (Vamplew et al., 2022; Skalse & Abate, 2024; Subramani et al., 2024).

7 Discussion and Conclusion

Developing reinforcement learning agents for real-world tasks necessitates not only generalization across diverse environments, but also across multiple objectives. By formally introducing a framework for discussing and evaluating generalization in MORL, we bridge a crucial gap between RL generalization and multi-objective decision-making. A novel benchmark has been contributed, which would help kickstart rigorous investigations into MORL generalization. The extensive baseline evaluations of state-of-the-art MORL algorithms on the benchmark also highlight significant room for future research to improve upon. We encourage readers to look at Section B of the appendix, where we analyzed algorithmic failure modes in current MORL approaches which possibly explains the poor generalization performance. Related works and discussions on future research directions are also provided in the appendix. We hope this paper spurs greater recognition of the importance of multi-objective reward structures for RL generalization and provide a foundation for future work unifying these two fields. Ultimately, this paper lays the groundwork for advancing RL agents capable of tackling the complexities of real-world, multi-objective scenarios.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization, 2019. URL https://arxiv.org/abs/1811.11214.

Lucas N. Alegre, Ana L. C. 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*, AAMAS '23, pp. 2003–2012, Richland, SC, 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450394321.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Rich Caruana. *Multitask Learning*, pp. 95–133. Springer US, Boston, MA, 1998. ISBN 978-1-4615-5529-2. doi: 10.1007/978-1-4615-5529-2_5. URL https://doi.org/10.1007/978-1-4615-5529-2_5.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in Neural Information Processing Systems 36, New Orleans, LA, USA*, December 2023.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning, 2019. URL https://arxiv.org/abs/1812.02341.

Kalyanmoy Deb and Deb Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., USA, 2001. ISBN 047187339X.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design, 2021. URL https://arxiv.org/abs/2012.02096.

Simon Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudik, and John Langford. Provably efficient RL with rich observations via latent state decoding. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1665–1674. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/du19b.html.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Robust predictable control, 2021. URL https://arxiv.org/abs/2109.03214.

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 *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023.

Florian Felten, El-Ghazali Talbi, and Grégoire Danoy. Multi-objective reinforcement learning based on decomposition: A taxonomy and framework. *J. Artif. Int. Res.*, 79, feb 2024. ISSN 1076-9757. doi: 10.1613/jair.1.15702. URL https://doi.org/10.1613/jair.1.15702.

Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P. Adams, and Sergey Levine. Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability, 2021. URL https://arxiv.org/abs/2107.06277.

Abhishek Gupta, Yew-Soon Ong, Liang Feng, and Kay Chen Tan. Multiobjective multifactorial optimization in evolutionary multitasking. *IEEE Transactions on Cybernetics*, 47(7):1652–1665, 2017. doi: 10.1109/TCYB.2016.2554622.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes, 2015. URL https://arxiv.org/abs/1502.02259.

Conor F. Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1), apr 2022. ISSN 1387-2532. doi: 10.1007/s10458-022-09552-y. URL https://doi.org/10.1007/s10458-022-09552-y.

Irina Higgins, Arka Pal, Andrei A. Rusu, Loic Matthey, Christopher P Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning, 2018. URL https://arxiv.org/abs/1707.08475.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. In *Neural Information Processing Systems*, 2019.

Alex Irpan and Xingyou Song. The principle of unchanged optimality in reinforcement learning generalization, 2019. URL https://arxiv.org/abs/1906.00336.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay, 2021. URL https://arxiv.org/abs/2010.03934.

Yiding Jiang, J. Zico Kolter, and Roberta Raileanu. On the importance of exploration for generalization in reinforcement learning, 2023. URL https://arxiv.org/abs/2306.05483.

Christian Kauten. Super Mario Bros for OpenAI Gym. GitHub, 2018. URL https://github.com/Kautenja/gym-super-mario-bros.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *J. Artif. Int. Res.*, 76, May 2023. ISSN 1076-9757. doi: 10.1613/jair.1.14174. URL https://doi.org/10.1613/jair.1.14174.

Henri Lebesgue. *Integrale, Longueur, Aire*. PhD thesis, PhD Thesis. Universite de Paris, 1902.

Ke Li, Kalyanmoy Deb, and Xin Yao. R-metric: Evaluating the performance of preference-based evolutionary multiobjective optimization using reference points. *IEEE Transactions on Evolutionary Computation*, 22(6):821–835, 2018. doi: 10.1109/TEVC.2017.2737781.

Xiang Li, Wenhao Yang, and Zhihua Zhang. A regularized approach to sparse optimal policy in reinforcement learning, 2019. URL https://arxiv.org/abs/1903.00725.

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.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 1476-4687. doi: 10.1038/nature14236.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey, 2020. URL https://arxiv.org/abs/2003.04960.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning, 2019. URL https://arxiv.org/abs/1810.12282.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3803–3810, 2018. doi: 10.1109/ICRA.2018.8460528.

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning, 2021. URL https://arxiv.org/abs/2102.10330.

Mathieu Reymond, Eugenio Bargiacchi, and Ann Nowé. Pareto conditioned networks. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, pp. 1110–1118, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450392136.

Diederik M. Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *J. Artif. Int. Res.*, 48(1):67–113, oct 2013. ISSN 1076-9757.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 525–536, Red Hook, NY, USA, 2018. Curran Associates Inc.

Joar Skalse and Alessandro Abate. On the limitations of markovian rewards to express multi-objective, risk-sensitive, and modal tasks, 2024. URL https://arxiv.org/abs/2401.14811.

Rohan Subramani, Marcus Williams, Max Heitmann, Halfdan Holm, Charlie Griffin, and Joar Skalse. On the expressivity of objective-specification formalisms in reinforcement learning, 2024. URL https://arxiv.org/abs/2310.11840.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

F. Tanaka and M. Yamamura. Multitask reinforcement learning on the distribution of mdps. In *Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation. Computational Intelligence in Robotics and Automation for the New Millennium (Cat. No.03EX694)*, volume 3, pp. 1108–1113 vol.3, 2003. doi: 10.1109/CIRA.2003.1222152.

Yujin Tang, Duong Nguyen, and David Ha. Neuroevolution of self-interpretable agents. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20. ACM, June 2020. doi: 10.1145/3377930.3389847. URL http://dx.doi.org/10.1145/3377930.3389847.

Cem Tekin and Eralp Turgay. Multi-objective contextual bandits with a dominant objective. In *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2017. doi: 10.1109/MLSP.2017.8168123.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017. doi: 10.1109/IROS.2017.8202133.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Eralp Turgay, Doruk Oner, and Cem Tekin. Multi-objective contextual bandit problem with similarity information. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1673–1681. PMLR, 09–11 Apr 2018. URL https://proceedings.mlr.press/v84/turgay18a.html.

Peter Vamplew, John Yearwood, Richard Dazeley, and Adam Berry. On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts. In Wayne Wobcke and Mengjie Zhang (eds.), *AI 2008: Advances in Artificial Intelligence*, pp. 372–378, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-89378-3.

Peter Vamplew, Richard Dazeley, and Cameron Foale. Softmax exploration strategies for multiobjective reinforcement learning. *Neurocomputing*, 263:74–86, 2017. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2016.09.141. URL https://www.sciencedirect.com/science/article/pii/S0925231217310974. Multiobjective Reinforcement Learning: Theory and Applications.

Peter Vamplew, Benjamin J. Smith, Johan Källström, Gabriel Ramos, Roxana Rădulescu, Diederik M. Roijers, Conor F. Hayes, Fredrik Heintz, Patrick Mannion, Pieter J. K. Libin, Richard Dazeley, and Cameron Foale. Scalar reward is not enough: A response to Silver, Singh, Precup and Sutton (2021). *Autonomous Agents and Multi-Agent Systems*, 36(2):41, July 2022. ISSN 1573-7454. doi: 10.1007/s10458-022-09575-5.

Peter Vamplew, Cameron Foale, and Richard Dazeley. Value function interference and greedy action selection in value-based multi-objective reinforcement learning. *arXiv preprint arXiv:2402.06266*, 2024.

Van Veldhuizen and David Allen. *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, 1999.

Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization, 2020. URL https://arxiv.org/abs/2010.10814.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions, 2019. URL https://arxiv.org/abs/1901.01753.

D.J White. Multi-objective infinite-horizon discounted markov decision processes. *Journal of Mathematical Analysis and Applications*, 89(2):639–647, 1982. ISSN 0022-247X. doi: https://doi.org/10.1016/0022-247X(82)90122-6. URL https://www.sciencedirect.com/science/article/pii/0022247X82901226.

Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. Prediction-guided multi-objective reinforcement learning for continuous robot control. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10607–10616. PMLR, 13–18 Jul 2020.

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

Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification, 2017. URL https://arxiv.org/abs/1702.02453.

Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning, 2018. URL https://arxiv.org/abs/1806.07937.

Luisa Zintgraf, Timon Kanters, Diederik 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 optimization using evolutionary algorithms — a comparative case study. In Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel (eds.), *Parallel Problem Solving from Nature — PPSN V*, pp. 292–301, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-49672-4.

A Related Work

Multi-Objective Contextual Multi-Armed Bandits Multi-Objective Contextual Multi-Armed Bandits (MOC-MAB; Tekin & Turgay (2017); Turgay et al. (2018)) are a context-dependent, multi-objective extension of the Multi-Arm Bandit (MAB) problem. In MOC-MAB, at each decision point, the agent observes a context and selects an action (arm) to maximize a vector of immediate rewards corresponding to different objectives. While MOC-MAB provides valuable insights into handling contexts and balancing multiple objectives simultaneously, it fundamentally differs from the MOC-MDP framework. Specifically, MOC-MAB does not address the state-transitions and sequential decision-making inherent in MORL. Our work extends beyond the MOC-MAB setting by focusing on the generalization of RL agents in a context-dependent, multi-objective environment—a problem that, to our knowledge, has not been previously explored in the literature. However, bandit analysis often forms the foundations of progress in RL, so we implore future work to look into the MOC-MAB framework for inspiration on improving generalization in MORL.

Multi-Tasking and Meta-Learning Multi-Task Learning (MTL; Caruana (1998)) and Multi-Task Reinforcement Learning (MTRL; Tanaka & Yamamura (2003)) aim to improve learning efficiency and performance by leveraging shared representations across multiple tasks. Reinforcement Learning based on CMDPs is closely related to MTRL but involves a parameterized variable, termed the context, which allows for a more unified modeling of tasks within a single framework. However, both MTRL and CMDPs have predominantly been studied in the single-objective setting, focusing on maximizing a scalar reward function. Sener & Koltun (2018) framed MTL as a MOO problem by treating different tasks as conflicting objectives. While this perspective introduces multi-objectivity into MTL, it primarily addresses trade-offs between tasks rather than scenarios where each task involves multiple objectives. In the optimization domain, the Multi-Objective Multifactorial Optimization (MO-MFO) paradigm (Gupta et al., 2017) considers multitasking across multiple multi-objective problems by leveraging shared evolutionary operators to solve them simultaneously. Despite these advancements, there is a notable gap in the literature regarding the simultaneous consideration of multi-objectivity and generalization across contexts (or tasks) in reinforcement learning. To the best of our knowledge, this is the first study to formalise and systematically explore generalization in MORL, highlighting unique difficulties within this combined setting.

B Analysis of Failure Modes in MORL Approaches

In this section, we seek a deeper understanding on failure modes within the current MORL algorithms that can hinder generalization. We caution readers looking to further MORL generalization to be wary of them and encourage exploration to solve these failure modes. Note that we will only discuss challenges that are unique to generalization within MORL, and problems pertaining to the broader RL generalization literature is excluded.

Pareto Archival Methods MORL methods often maintain a *Pareto archive*—a set of nondominated policies discovered during training. This archive is constantly updated by comparing the value vector of new policies with old ones, and discarding the dominated ones. This archive can then aid the agent's search process within the objective space, or be used as solutions during test time. This technique is commonly used in multi-objective evolutionary algorithms like PGMORL and MORL/D. Similarly, GPI-LS and GPI-PD track a finite convex subset of the PF where dominance is defined only for linear utility functions. However, when extending these methods to a MOC-MDP—where each context has its own optimal PF—current archiving mechanisms can lead to suboptimal outcomes. Most MORL literature assumes a static environment, so existing Pareto archival mechanisms are not designed to handle context variability in MOC-MDPs. As a result, the archive overrepresents policies that perform well in a narrow set of contexts with higher reward scales or lower difficulty, while discarding those optimal for more challenging or less rewarding contexts. This has severe implications as it will cause the agent to converge to a *maximax strategy*, adopting policies that are only optimized to yield the best of the best possible outcomes during test time, and results in poor generalization across the entire range of contexts in the MOC-MDP.

Reliance on Linear Scalarisation The convexity of the induced value functions' range determines if MORL algorithms relying on linear scalarization (LS), are capable of finding all policies

corresponding to the optimal PF (Vamplew et al., 2008; Roijers et al., 2013). Lu et al. (2023) showed that in the static-environment setting, the induced value functions' range of stochastic stationary policies in a MOMDP is convex, which means LS is not a bottleneck for approximating the PF. If we consider maximizing the expected multi-objective value function (across contexts) as the learning objective, the same results by Lu et al. (2023) can be applied directly to show the convexity of the range of expected value vectors in a MOC-MDP. This is a trivial result of the linearity of the expectation operator, which preserves convexity (Boyd & Vandenberghe, 2004). However, if our maximization objective is the recovery of the globally optimal PF across contexts, the policies that the agent learn may need to be non-stationary and/or non-Markovian (further discussion in C.1). Prima facie, in cases where the policies exhibits nonlinear dependence on state-action history, methods relying solely on LS would be insufficient to identify all globally Pareto-optimal policies in a MOC-MDP. Currently, the MORL field lags significantly behind MOO, particularly in methodological advancements. The prominence of LS-reliant methods in state-of-the-art MORL algorithms underscores this gap. We encourage future exploration of approaches established in MOO which can approximate the PF without relying on convexity assumptions, such as those based on nonlinear scalarization or evolutionary algorithms.

Value Function Interference Within state-of-the-art MORL, many approaches extend value-based scalar RL algorithms such as Q-learning or Deep Q-Networks to handle vector rewards. If the utility function allows actions with widely differing vector outcomes to map to the same utility value, then the vector value function learned for earlier states may be inconsistent with the actual optimal policy (Vamplew et al., 2024). This problem is particularly likely to arise in environments which are stochastic or partially-observable. We note that for MOC-MDPs, the dynamics and rewards observed by the agent may appear to be stochastic even if the underlying MDPs are deterministic, due to the influence of the hidden context variables. In fact, Ghosh et al. (2021) showed that generalization, even in fully-observable RL tasks, requires solving an implicitly partially-observable RL problem they term as *epistemic POMDP*. Therefore, value function interference may pose a particular problem when naively applying value-based MORL algorithms to MOC-MDPs. We note that if the utility function is linear then value interference does not impact on selecting the optimal action, hence there is an implicit tension between this failure mode, and the issues of reliance on linear scalarisation raised in the previous paragraph.

C Extended Discussions

C.1 Principle of Unchanged Pareto Optimality

When constructing a benchmark in reinforcement learning, it is important to ensure that the domain satisfies *The Principle of Unchanged Optimality* (Irpan & Song, 2019), an underappreciated yet fundamentally important principle. This principle asserts that, for a domain to support generalization, it should provide all necessary information such that a policy optimal in every context can exist. In the MOC-MDP framework, *The Principle of Unchanged (Pareto) Optimality* implies the existence of globally Pareto optimal policies, π^*, such that:

$$\forall c \in \mathcal{C}: \quad \pi^* \in \mathcal{PS}(\Pi_c),$$

where Π_c is the set of feasible policies, and $\mathcal{PS}(\Pi_c)$ denotes the Pareto set containing nondominated policies, for a given $c \in \mathcal{C}$. This principle has significant theoretical implications. When the unchanged optimality principle is disregarded, the benchmark can become a proxy measure of the memorization capability (Zhang et al., 2018) of the MORL agents, instead of generalization.

The Principle of Unchanged Pareto Optimality is also important for our generalization evaluations. If this principle is violated, generalist agents would be fundamentally unable to achieve an NHGR score of 1, as they could never match the performance of specialists across all contexts. This section examines how The Principle of Unchanged Pareto Optimality is upheld in the domains of our MORL generalization benchmark, thereby supporting its validity for measuring MORL generalization. It is important to note that each context or environment in the benchmark varies in its initial state distribution, transition dynamics, and multi-objective reward function. When the context is fully observable, the agent can include the context as part of its state representation, enabling the learning of "universal" policies that adapt to variations across contexts. However, if the context is hidden, the agent must infer it during deployment to recover optimality. Therefore, for our benchmark to respect

The Principle of Unchanged Pareto Optimality, we need to ensure that **sufficient information about the context** can be inferred from the agent's observations within our proposed domains.

For MO-SuperMarioBros, although visual similarities exist between levels, each observation provides sufficient information for determining the optimal action at every time step (e.g. immediate coins, enemies, bricks are visible). Additionally, given the finite number of stages (32), the agent can easily deduce its current stage from its observations. Similarly, in MO-LavaGrid, the placement of lava and goals is fully observable in each observation. Furthermore, as described in Section 5, we concatenate the reward weights of each goal with the agent's observations, a deliberate choice to ensure that necessary information about the context, specifically the current reward function, is provided to the agent for optimal planning.

For the continuous control domains–MO-Hopper, MO-HalfCheetah, and MO-Humanoid–each context varies in environment dynamics, such as gravity and friction. However, the agent's observations only include the positions and velocities of the robot's joints, making it impossible to infer the context or determine optimal actions from a single time step. A similar situation occurs in MO-LunarLander, where the agent's observations are limited to its orientation and velocity. The environment dynamics is, however, inferrable when the agent considers its state-action history. Consequently, the optimal policies in these domains are inherently non-Markovian, requiring either recurrent policies (e.g., recurrent neural networks) or regression over state-action history buffers. We must therefore note that **The Principle of Unchanged Pareto Optimality can indeed be upheld in our proposed domains**, but current MORL algorithms, which typically assume Markovian policies, fail to achieve this. This is expected since we are the first work to consider MORL outside of static environments. This also largely explains the poor NHGR performances observed for existing MORL algorithms, as shown in Section 6.

C.2 Future Work and Limitations

In this paper, we conducted extensive evaluations of current Multi-Objective Reinforcement Learning (MORL) algorithms on the benchmark we introduced, utilizing domain randomization techniques. However, the poor generalization results indicate a clear need for more innovative approaches. A promising starting point for future research would be to leverage insights from the single-objective RL generalization literature, where several established methods could be adapted to enhance MORL generalization. These approaches include regularization techniques (Cobbe et al., 2019; Ahmed et al., 2019; Li et al., 2019; Igl et al., 2019; Eysenbach et al., 2021; Wang et al., 2020), incorporating inductive biases (Tang et al., 2020; Raileanu & Fergus, 2021; Higgins et al., 2018), and curriculum learning methods (Wang et al., 2019; Narvekar et al., 2020; Dennis et al., 2021; Jiang et al., 2021).

For more specialized techniques that target MORL generalization specifically, there are several possible avenues. As highlighted in Section B, many current methods rely heavily on linear scalarization, which may constrain the generalization potential of MORL agents. Recently, evolutionary methods such as those proposed by Xu et al. (2020) and Felten et al. (2024) have been introduced, but they remain underexplored in the MORL context and warrant further investigation to boost generalization. Moreover, exploration has been shown to play a critical role in enhancing generalization in single-objective RL (Jiang et al., 2023). Thus, approaches like Vamplew et al. (2017), which incorporate exploration techniques from single-objective RL into the MORL framework would be of interest to future research. Finally, as just mentioned in Section C.1, implementing MORL algorithms which can recover The Principle of Unchanged Pareto Optimality in domains with partially-observable contexts would be important for generalization. Therefore, future work should definitely look into how leveraging of state-action history and recurrent policies have been implemented in SORL generalization literature (Yu et al., 2017; Peng et al., 2018; Packer et al., 2019), and adapt them to current MORL algorithms.

D EXTENDED METRIC DISCUSSION AND RESULTS

D.1 EXPECTED UTILITY METRIC

The *Expected Utility Metric* (EUM) proposed by Zintgraf et al. (2015) can be used when a prior over scalarisation functions is known. This metric calculates the expected utility of the agent's approximate Pareto front under some prior distribution over utility functions. It is similar to the R-Metric (Li et al., 2018) in MOO. It assesses the expected utility of policies on the normalized Pareto front across various weights $\mathbf{w}$ from a predefined weight space W. A higher EUM indicates that the policies offer a better trade-off between objectives and leads to more desirable outcomes **under the specified distribution of utility functions**. The EUM of a given approximate Pareto front $\tilde{\mathcal{F}}$ is given by:

$$\text{EUM}(\tilde{\mathcal{F}}) = \mathbb{E}_{w \sim W}\left[\max_{\mathbf{v}^\pi \in \tilde{\mathcal{F}}} u_w(\mathbf{v}^\pi)\right],$$

where u_w is the chosen utility function parameterized a weight w and $\mathbf{v}^\pi$ is the expected value vector of the policy π taken from $\tilde{\mathcal{F}}$.

There are several reasons as to why EUM is used in **single-environment** MORL evaluations. In practical applications, the utility function of the stakeholders might be known due to domain knowledge. Using EUM would therefore allow for more direct evaluations on how the solutions generated by the MORL agent corresponds to improving the utility of the stakeholders. Not every point on the Pareto front would contribute to an increase in the EUM for a given utility function. For example, with linear utility functions, adding solutions in concave regions of the Pareto front do not result in an increase of utility. Lastly, the hypervolume metric is known for its computational challenge especially in higher dimensions, although various approximation algorithms and heuristics have been developed to estimate the hypervolume more efficiently. The EUM, on the other hand, depends only on the number of solutions on the approximate Pareto front and the size of the weight space.

In the main body of this paper, we focused on hypervolume-based measures of MORL generalization. While EUM offers a meaningful way to evaluate solution sets, it does require a well-informed prior over possible scalarization functions to be effective. This dependency limits its generality, and our goal is to introduce a general metric that can be applied in any MORL problem. A Pareto front that maximizes hypervolume will also maximize the EUM for any monotonic utility function, but the reverse is not necessarily true. Using EUM requires assuming a specific utility function for the calculations, which restricts its applicability in cases where the true utility function is unknown. Hypervolume also possesses desirable mathematical properties and remains a popular metric in the multi-objective optimization (MOO) literature, and it is more commonly used than the R-metric.

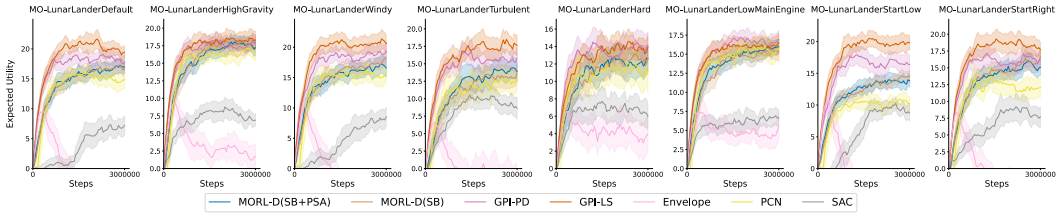
That said, in specific scenarios where a reliable prior over utility functions is available, it is important to explore how EUM can be incorporated into generalization evaluations. As mentioned in Section 4 of the main body, when aggregating performances across multiple contexts for measuring generalization, we must ensure that each context is equally attributed. Specifically, we can calculate a variant on the NHGR metric we call the *Expected Utility Generalization Ratio* (EUGR). Let $\tilde{\mathcal{F}}_{c_j}^{\text{gen}}$ and $\tilde{\mathcal{F}}_{c_j}^{\text{spec}}$ be the approximate Pareto front obtained by generalist MORL agent and the combined specialist Pareto front on context c_j. The EUGR for c_j is defined as:

$$\text{EUGR}(\tilde{\mathcal{F}}_{c_j}^{\text{gen}}, \tilde{\mathcal{F}}_{c_j}^{\text{spec}}) = \frac{\text{EUM}(\tilde{\mathcal{F}}_{c_j}^{\text{gen}})}{\text{EUM}(\tilde{\mathcal{F}}_{c_j}^{\text{spec}})}.$$

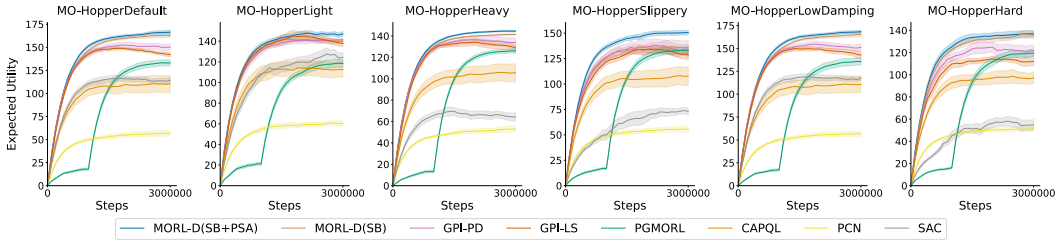
Unlike in NHGR, the Pareto front is not normalized here. This is because the utility functions used in the EUM should inherently reflect the stakeholders' preferences over objectives, including their relative importance. Normalizing the objective space would eliminate these preferences. However, obtaining the combined specialist Pareto front for the evaluation contexts remains essential, as each context presents a distinct shape of the optimal Pareto front. Without comparing against an approximation of the true maximally-achievable EUM in each context, evaluations risk introducing bias towards specific contexts under different utility functions.

Now that EUGR is established and every context can be fairly weighted in generalization evaluations, we are ready to present the aggregated EUGR performances for each domain. Note that, we

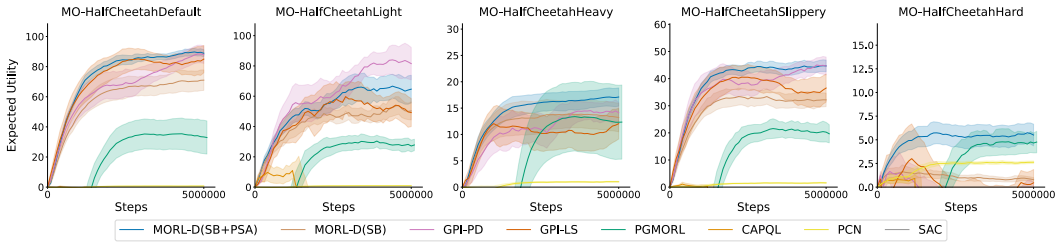
assume the utility functions are linear with weights summing to unity, and we sampled 100 weight vectors equally across the unit weight simplex during evaluations and calculated each algorithms' EUM under the EUM metric. We provide plots of our evaluated MORL algorithms and the SAC agent on individual evaluation contexts (without aggregation) across training in Fig. 7. We then present the aggregated IQM and optimality gap EUGR performances using the `rliable` library in Fig. 8. MORL/D SB and MORL/D SB+PSA again appears demonstrates the most consistent results in terms of leading EUGR performances.



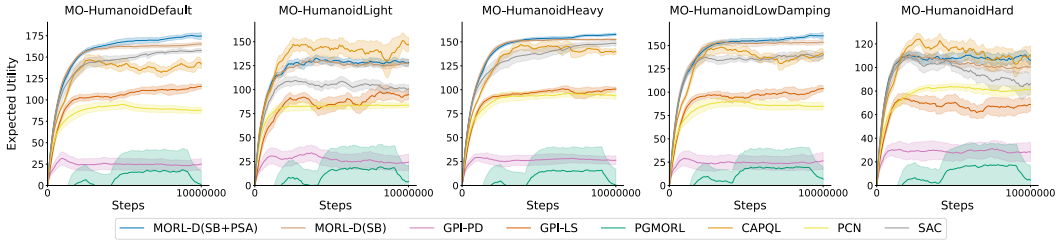
(a) MO-LunarLander



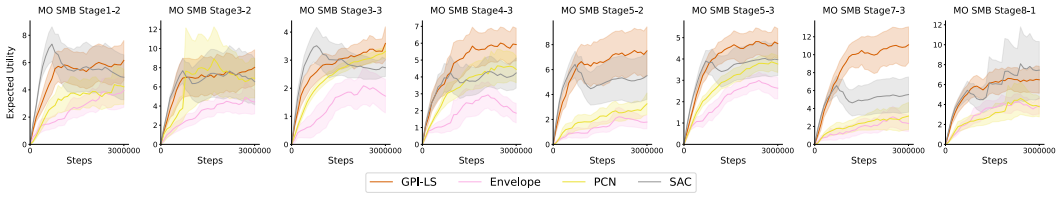
(b) MO-Hopper



(c) MO-HalfCheetah

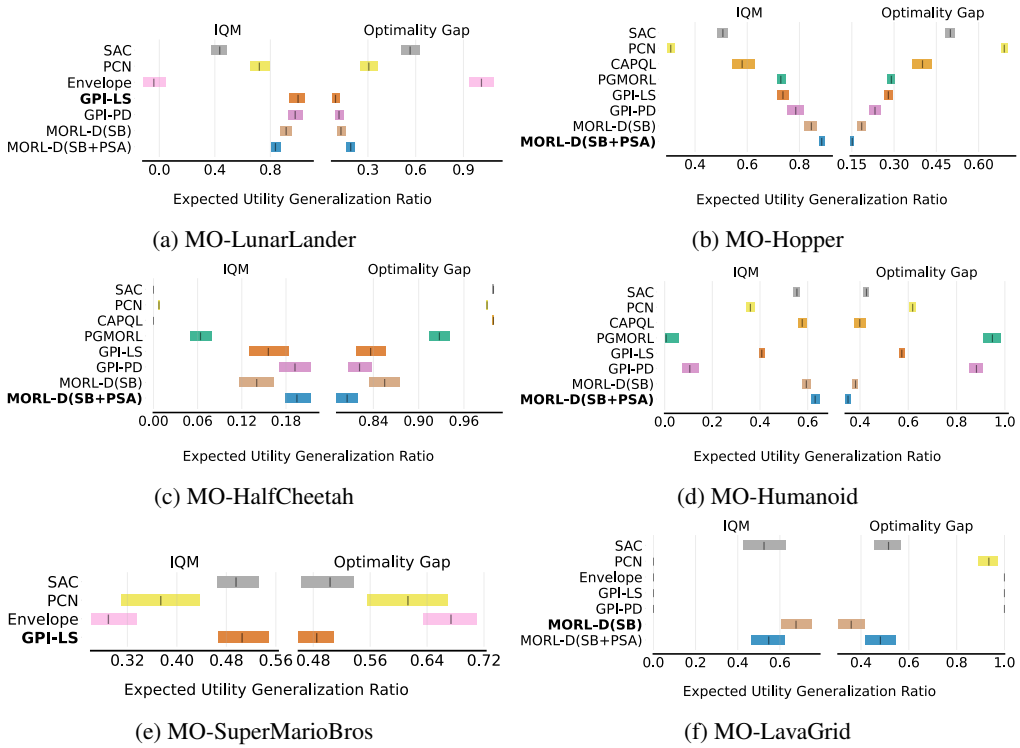


(d) MO-Humanoid



(e) MO-SuperMarioBros

Figure 7: Expected Utility Metric performance in individual evaluation context of each benchmark domain across training. Each algorithm is evaluated across 5 independent seeds.



(a) MO-LunarLander (b) MO-Hopper

(c) MO-HalfCheetah (d) MO-Humanoid

(e) MO-SuperMarioBros (f) MO-LavaGrid

Figure 8: Aggregate EUGR performance in all domains of the benchmark. Each algorithm is evaluated across 5 independent seeds and several evaluation environment configurations. Higher IQM and lower optimality gap scores are better. The best algorithm for each domain is bolded.

D.2 Cardinality

The *Cardinality metric* used in MORL measures the number of non-dominated points generated by a given algorithm. This evaluation mechanism closely relates to the cardinality indicator in multi-objective optimization. We provide calculations for cardinality in our codebase.

D.3 Single Objective Utility Function Plots

As mentioned in Section 6.2, majority of the classic SORL domains from Gymnasium involves implicit scalarization of multiple objectives when crafting the scalar reward. As such, for every domain, we record and plot the single utility function (scalarization function), f_{SORL} return for all the MORL algorithms and SAC for all evaluations throughout training. Across all domains, we observe that the leading MORL algorithms often outperforms or achieves parity with the single-objective SAC algorithm in terms of maximum f_{SORL} return across the evaluation episodes. Here are all the f_{SORL} equations for each environment and their corresponding plots.

D.3.1 MO-Hopper

The default single objective utility function of the MO-Hopper domain is same as the one used in Gymnasium's Hopper, which is

$$f_{\text{SORL}} = 1.5v_x + 0.001c + h$$

where v_x is the forward speed, c is the control cost and h is the reward for staying alive.

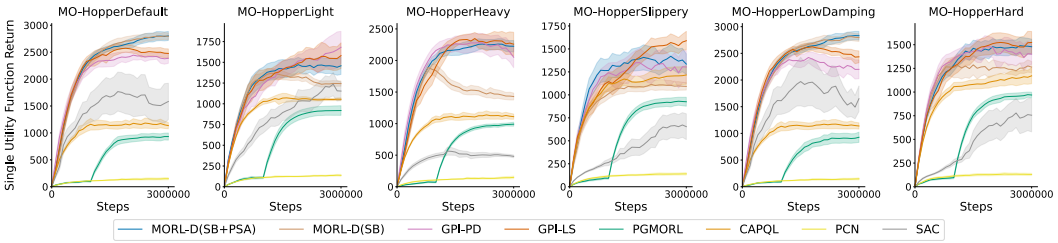


Figure 9: Single-objective return on 6 MO-Hopper testing environments during training. Each curve is measured across 5 seeds (mean and standard error).

D.4 MO-HALFCHEETAH

The default single objective utility function of the MO-HalfCheetah domain is same as the one used in Gymnasium's HalfCheetah, which is

$$f_{\text{SORL}} = 1.0v_x + 0.1c$$

where v_x is the forward reward and c is the control cost. The HalfCheetah is always alive so it has no alive reward.

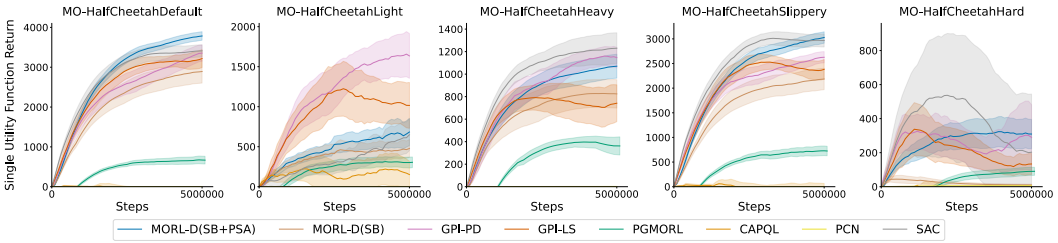


Figure 10: Single-objective return on 5 MO-HalfCheetah testing environments during training. Each curve is measured across 5 seeds (mean and standard error).

D.4.1 MO-HUMANOID

The single objective utility function of the MO-Humanoid domain is

$$f_{\text{SORL}} = 1.25v_x + 0.001c + 2.0h$$

where v_x is the forward speed, c is the control cost and h is the reward for staying alive. The original Gymansium's Humanoid domain uses a 5.0 coefficient for the alive reward but we tuned it down to because it dominating all the other objectives in terms of magnitude.

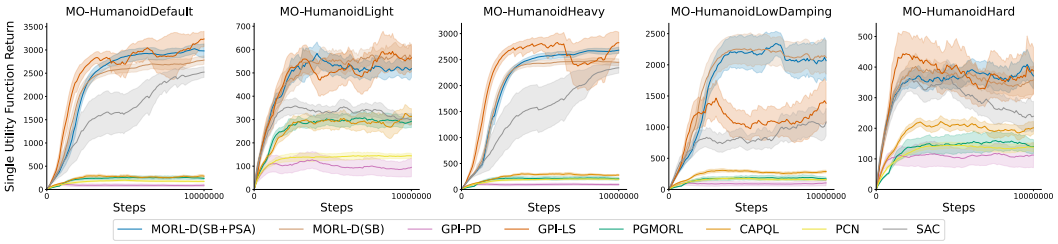


Figure 11: Single-objective return on 5 MO-Humanoid testing environments during training. Each curve is measured across 5 seeds (mean and standard error).

D.4.2 MO-LUNARLANDER

The default single objective utility function of the MO-LunarLander domain is same as the one used in Gymnasium's LunarLander, which is

$$f_{\text{SORL}} = l + s + 0.3mc + 0.03sc$$

where l is a -100/+100 one-time reward if the lander lands successfully or crashes, s is the shaping reward, mc is the main engine cost and sc is the side engine cost.

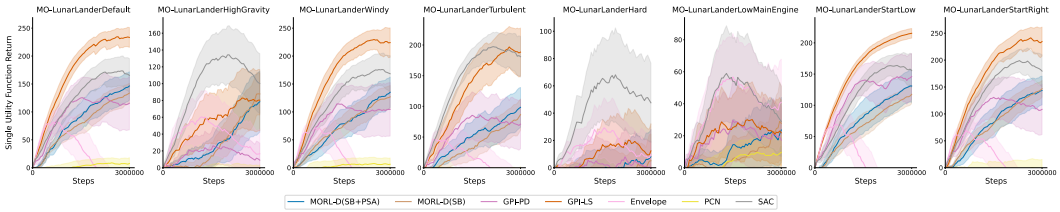


Figure 12: Single-objective return on 8 MO-LunarLander testing environments during training. Each curve is measured across 5 seeds (mean and standard error).

D.4.3 MO-SUPERMARIOBROS

The default single objective utility function of the MO-SuperMarioBros domain is same as the one used in Gym Super Mario Bros (Kauten, 2018), which is

$$f_{\text{SORL}} = f + t + d$$

where f is a forward reward, t is the time penalty, d is the death penalty.

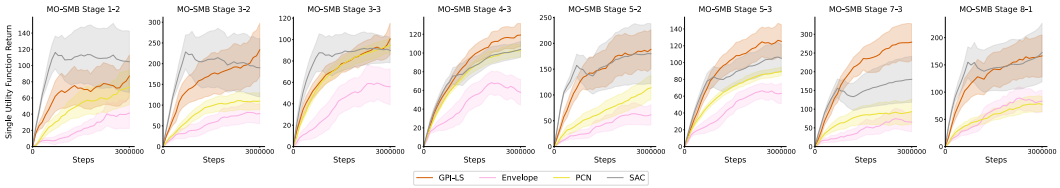


Figure 13: Single-objective return on 5 MO-SuperMarioBros (abbreviated as MO-SMB) testing environments during training. Each curve is measured across 5 seeds (mean and standard error).

E MORL GENERALIZATION TRAINING DETAILS

Table 2 shows general training hyperparameters for each domain in the MORL generalization benchmark. The scripts to reproduce the results in this paper are provided in the codebase, alongside with more specific hyperparameters for the different algorithms. To have fair evaluations, we utilize the same architectures for the policy and value functions across all algorithms for each domain. Specifically, for MO-LavaGrid, MO-LunarLander, MO-Hopper, MO-HalfCheetah, and MO-Humanoid, the policy and value functions are multi-layer perceptrons (MLPs) with four hidden layers of 256 units each. For MO-SuperMarioBros which has image observations, the policy and value functions consist of a NatureCNN Mnih et al. (2015) followed by a MLP with two hidden layers of 512 units each. For off-policy algorithms that depend on experience replay, we ensure the same replay buffer size is used.

Parameter	MO-LavaGrid	MO-Lunar-Lander	MO-Super-MarioBros	MO-Hopper	MO-HalfCheetah	MO-Humanoid
Discount γ	0.995	0.99	0.99	0.99	0.99	0.99
Adam learning rate	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$
Adam ϵ	$1e^{-8}$	$1e^{-8}$	$1e^{-8}$	$1e^{-8}$	$1e^{-8}$	$1e^{-8}$
Batch Size	128	128	64	256	256	256
Replay buffer size	$1e^{6}$	$1e^{6}$	$^{1}e5$	$1e^{6}$	$1e^{6}$	$1e^{6}$
Max episode steps	256	1000	2000	1000	1000	1000
Env Steps	$5e^{6}$	$3e^{6}$	$3e^{6}$	$3e^{6}$	$5e^{6}$	$1e^{7}$

Table 2: Hyperparameters used for training on MORL generalization benchmark.

F MORL Generalization Evaluation Details

In this section we list the environment parameters used creating our evaluation environments in each the benchmark domain of the main body. The code commands to initialize these environments using Gymnasium are also be included within our codebase. We also detail the normalization ranges used for calculating the HV_{norm} and NHGR metrics within the evaluations of the main body. These normalization ranges are derived from the specialist and generalist agents performances with some minor adjustments and rounding.

F.1 MO-LavaGrid

The environment parameters for the MO-LavaGrid domain are represented using bit maps, which we are unable to directly translate into this paper. Instead, the evaluation environments are visually shown in Fig. 14. Also, as mentioned in 5, the MO-LavaGrid environment has a 3-dimensional parameter controlling the reward weightages of each goal square (green, blue, yellow). The reward weights for each goal are concatenated to the state space of the agent, and the weights sum to unity. The reward weightages for each goal in each evaluation environment are shown in Table 3.

During training using domain randomization, after each episode concludes, the agent's start position and orientation, the number of lava blocks, the placement of the goals and lava blocks, and the reward weightages of the goals are all randomly set. When an agent has collected/visited a goal, the weightage of the goal in the state space is set to 0, to indicate that the reward corresponding to that goal has already been awarded. The normalization ranges used for calculating the HV_{norm} and NHGR metrics for MO-LavaGrid evaluations are shown in Tables 4 and 5.

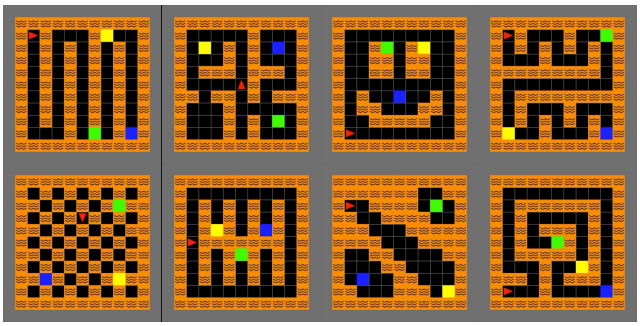


Figure 14: MO-LavaGrid Evaluation Environments. Top row (left to right): MO-LavaGridSnake, MO-LavaGridRoom, MO-LavaGridSmiley, MO-LavaGridMaze. Bottom row (left to right): MO-LavaGridCheckerBoard, MO-LavaGridCorridor, MO-LavaGridIslands, MO-LavaGridLabyrinth

Environment	**Green**	**Yellow**	**Blue**
MO-LavaGridSnake	0.20	0.30	0.50
MO-LavaGridRoom	0.50	0.30	0.20
MO-LavaGridSmiley	0.40	0.40	0.20
MO-LavaGridMaze	0.05	0.05	0.90
MO-LavaGridCheckerBoard	0.30	0.10	0.60
MO-LavaGridCorridor	0.60	0.10	0.30
MO-LavaGridIslands	$0.3\dot{3}$	$0.3\dot{3}$	$0.3\dot{3}$
MO-LavaGridLabyrinth	0.50	0.05	0.45

Table 3: Illustration of different metrics on 3 MO-HalfCheetah environments.

Objectives	**CheckerBoard**	**Smiley**	**Snake**	**Islands**
Lava Penalty	[0, 107.34]	[0, 270.69]	[0, 234.21]	[0, 124.23]
Time Penalty	[0, 218.76]	[0, 225.5]	[0, 220.54]	[0, 204.70]

Table 4: Normalization Ranges for MO-LavaGrid Environments (Part 1)

Objectives	**Labyrinth**	**Maze**	**Corridor**	**Room**
Lava Penalty	[0, 250.05]	[0, 237.33]	[0, 265.66]	[0, 263.86]
Time Penalty	[0, 226.95]	[0, 203.59]	[0, 240.21]	[0, 215.75]

Table 5: Normalization Ranges for MO-LavaGrid Environments (Part 2)

F.2 MO-SuperMarioBros

In MO-SuperMarioBros, each environment configuration is instantiated via a 2-dimensional parameter. The first dimension has discrete values $\{1, 2, 3, 4, 5, 6, 7, 8\}$, and indicates the SuperMarioBros world. The second dimension has discrete values $\{1, 2, 3, 4\}$, and indicates the level within the chosen world. Together, the parameters `<world>-<level>` defines the stage (configuration) of the environment.

During training using domain randomization, an environment is randomly selected from the 32 possible stages, except Stage 3-3 which is reserved for zero-shot generalization evaluation. During evaluation, the agents are evaluated on only 8/32 stages to keep the runtime within reasonable limits. The evaluation stages are visually shown in Fig. 15. The evaluation stages are carefully selected to encompass a wide range of environment dynamics and visual renditions. Additionally, they are chosen to ensure that each stage offers non-zero rewards across all objective dimensions. This is crucial to prevent hypervolume evaluations from collapsing to zero, which would occur if any dimension of the objective space had a zero achievable range. The normalization ranges used for calculating the HV_{norm} and NHGR metrics for MO-SuperMarioBros evaluations are shown in Tables 6 and 7.

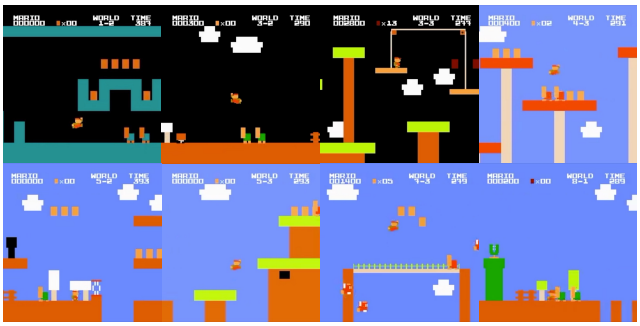


Figure 15: MO-SuperMarioBros Evaluation Environments. Top row (left to right): Stage 1-2, Stage 3-2, Stage 3-3 (zero shot), Stage 4-3. Bottom row (left to right): Stage 5-2, Stage 5-3, Stage 7-3, Stage 8-1

Objectives	Stage 1-2	Stage 2-3	Stage 3-2
Forward Reward	[0, 24.09]	[0, 30.7]	[0, 29.5]
Coin Reward	[-1, 6.52]	[-1, 26.7]	[-1, 3.73]
Points Reward	[-1, 85.78]	[-1, 40]	[-1, 102.4]

Table 6: Normalization Ranges for MO-SuperMarioBros Evaluation (Part 1)

Objectives	Stage 3-3	Stage 4-3	Stage 5-2	Stage 8-1
Forward Reward	[0, 29.1]	[0, 17.8]	[0, 26.1]	[0, 20.11]
Coin Reward	[-1, 22.3]	[-1, 16.3]	[-1, 8.6]	[-1, 0.58]
Points Reward	[-1, 20.5]	[-1, 44.9]	[-1, 31.5]	[-1, 158.17]

Table 7: Normalization Ranges for MO-SuperMarioBros Evaluation (Part 2)

F.3 MO-LunarLander

In MO-LunarLander, each environment configuration is instantiated via a 7-dimensional parameter. The dimensions of the environment parameter corresponds to the gravity coefficient, wind power, turbulence, the lander's main engine power, the lander's side engine power, the lander's initial x-coordinate, and the lander's initial y-coordinate.

During evaluation, we assess the agents performances on a predefined set of 8 environment configurations: Default, High Gravity, Windy, Turbulent, Low Main Engine, Start Low, Start Right, and Hard. The naming of the environment configurations are self-explanatory, and they are designed to test varying agent/environment dynamics corresponding to boundary values of the environment parameters. Table 8 displays the environment parameter values used for each environment configuration. The normalization ranges used for calculating the HV_{norm} and NHGR metrics for MO-LunarLander evaluations are shown in Tables 9 and 10.

Parameters	**Default**	**High Gravity**	**Windy**	**Turbulent**	**Low Main Engine**	**Start Low**	**Start Right**	**Hard**
Gravity	-10.0	-15.0	-10.0	-10.0	-10.0	-10.0	-10.0	-13.0
Wind Power	15.0	15.0	20.0	15.0	15.0	15.0	15.0	20.0
Turbulence Power	1.5	1.5	1.5	4.0	1.5	1.5	1.5	2.5
Main Engine Power	13.0	13.0	13.0	13.0	7.0	13.0	13.0	10.0
Side Engine Power	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.4
Initial X Coeff	0.5	0.5	0.5	0.5	0.5	0.5	0.75	0.7
Initial Y Coeff	1.0	1.0	1.0	1.0	1.0	0.7	1.0	1.0

Table 8: Environment parameters for MO-LunarLander

Objectives	**Default**	**High Gravity**	**Windy**	**Turbulent**	**Hard**
Landing Reward	[-60, 18.7]	[-59.3, 0]	[-59.3, 5.83]	[-60.2, -7.7]	[-62.7, -9.96]
Shaping Reward	[-175.9, 100]	[-174.0, 111.4]	[-174.0, 111.4]	[-175.4, 86.95]	[-200.2, 99.4]
Main Engine Cost	[-55.1, 0]	[-57.44, 0]	[-57.44, 0]	[-54.4, 0]	[-71.6, 0]
Side Engine Cost	[-47.4, 0]	[-45.7, 0]	[-45.8, 0]	[-60.3, 0]	[-43.6, 0]

Table 9: Normalization Ranges for MO-LunarLander Environments (Part 1)

Objectives	Start Low	Start Right	Low Main Engine
Landing Reward	[-67.3, 29.4]	[-57.1, 17.8]	[-56.5, -13.6]
Shaping Reward	[-218.9, 100]	[-223.1, 123.94]	[-196, 120.1]
Main Engine Cost	[-67.1, 0]	[-60.4, 0]	[-74.9, 0]
Side Engine Cost	[-41.6, 0]	[-47.63, 0]	[-39.4, 0]

Table 10: Normalization Ranges for MO-LunarLander Environments (Part 2)

F.4 MO-HOPPER

In MO-Hopper, each environment configuration are instantiated via a 8-dimensional parameter that varies the hopper's body masses (4D), joint damping (3D), and the floor's friction (1D).

During evaluation, we assess the agents performances on a predefined set of 6 environment configurations: Default, Light, Heavy, Slippery, Low Damping, and Hard. The naming of the environment configurations are self-explanatory, and they are designed to test varying agent/environment dynamics corresponding to boundary values of the environment parameters. Table 11 displays the environment parameter values used for each environment configuration. The normalization ranges used for calculating the HV_{norm} and NHGR metrics for MO-Hopper evaluations are shown in Table 12.

Parameters	Default	Light	Heavy	Slippery	Low Damping	Hard
Torso Mass	3.7	0.5	9.0	3.7	3.7	0.1
Thigh Mass	4.0	0.5	9.0	4.0	4.0	9.0
Leg Mass	2.8	0.3	8.5	2.8	2.8	9.0
Foot Mass	5.3	0.7	10.0	5.3	5.3	0.1
Damping 0	1.0	1.0	1.0	1.0	0.1	0.1
Damping 1	1.0	1.0	1.0	1.0	0.1	0.1
Friction	1.0	1.0	1.0	0.1	1.0	0.1

Table 11: Environment parameters for MO-Hopper

Objectives	Default	Light	Heavy	Slippery	Low Damping	Hard
Forward Reward	[0, 242]	[0, 252.2]	[0, 218]	[0, 175.3]	[0, 262.6]	[0, 174.8]
Upward Reward	[0, 321]	[0, 457.4]	[0, 247]	[-21, 324.1]	[0, 315]	[0, 334.2]
Control Cost	[-62.1, 97]	[-41.3, 100]	[-54, 96.5]	[-43.1, 96.1]	[-40, 96.5]	[-38, 97.3]

Table 12: Normalization Ranges for MOHopper Environments

F.5 MO-HalfCheetah

In MO-HalfCheetah, each environment configuration is instantiated via a 8-dimensional parameter that varies the cheetah's body masses (7D) and the floor's friction (1D).

During evaluation, we assess the agents performances on a predefined set of 5 environment configurations: Default, Light, Heavy, Slippery, and Hard. The naming of the environment configurations are self-explanatory, and they are designed to test varying agent/environment dynamics corresponding to boundary values of the environment parameters. Table 13 displays the environment parameter values used for each environment configuration. The normalization ranges used for calculating the HV_{norm} and NHGR metrics for MO-Hopper evaluations are shown in Table 14.

Parameters	Default	Light	Heavy	Slippery	Hard
Torso Mass	6.25	0.5	10.0	6.25	6.25
Back Thigh Mass	1.538	0.1	9.5	1.54	9.5
Back Shin Mass	1.441	0.1	9.5	1.59	9.5
Back Foot Mass	0.891	0.1	9.5	1.10	9.5
Front Thigh Mass	1.434	0.1	9.5	1.44	0.1
Front Shin Mass	1.198	0.1	9.5	1.20	0.1
Front Foot Mass	0.869	0.1	9.5	0.88	0.1
Friction	0.4	0.4	0.4	0.1	0.1

Table 13: Environment parameters for MO-HalfCheetah

Objectives	Default	Light	Heavy	Slippery	Hard
Forward Reward	[0, 836]	[0, 1061]	[0, 227]	[0, 791]	[0, 511]
Control Cost	[-353, -3.4]	[-345, -4.3]	[-379, -4.3]	[-411, -3.6]	[-417, -3.8]

Table 14: Normalization Ranges for MO-HalfCheetah Environments

F.6 MO-Humanoid

In MO-Humanoid, each environment configuration is instantiated via a 30-dimensional parameter that varies the humanoid's body masses (13D) and joint damping (17D).

During evaluation, we assess the agents performances on a predefined set of 5 environment configurations: Default, Light, Heavy, Low Damping, and Hard. The naming of the environment configurations are self-explanatory, and they are designed to test varying agent/environment dynamics corresponding to boundary values of the environment parameters. Table 15 displays the environment parameter values used for each environment configuration. The normalization ranges used for calculating the HV_{norm} and NHGR metrics for MO-Hopper evaluations are shown in Table 16.

Parameters	Default	Light	Heavy	Low Damping	Hard
Mass 1	8.91	1.7	10.0	8.91	8.91
Mass 2	2.26	0.5	7.0	2.26	2.26
Mass 3	6.62	1.3	9.0	6.62	6.62
Mass 4	4.75	0.7	8.0	4.75	0.7
Mass 5	2.76	0.6	7.0	2.76	0.6
Mass 6	1.77	0.5	6.0	1.77	0.5
Mass 7	4.75	0.7	8.0	4.75	8.0
Mass 8	2.76	0.5	7.0	2.76	7.0
Mass 9	1.77	0.3	6.0	1.77	6.0
Mass 10	1.66	0.3	6.0	1.66	0.1
Mass 11	1.23	0.1	5.5	1.23	0.1
Mass 12	1.66	0.3	6.0	1.66	5.0
Mass 13	1.23	0.1	5.5	1.23	5.0
Damp 1	1.0	5.0	5.0	1.0	1.0
Damp 2	1.0	5.0	5.0	1.0	1.0
Damp 3	1.0	5.0	5.0	1.0	1.0
Damp 4	1.0	5.0	5.0	1.0	1.0
Damp 5	1.0	5.0	5.0	1.0	1.0
Damp 6	1.0	5.0	5.0	1.0	1.0
Damp 7	0.2	1.0	1.0	0.2	0.2
Damp 8	1.0	5.0	5.0	1.0	1.0
Damp 9	1.0	5.0	5.0	1.0	1.0
Damp 10	1.0	5.0	5.0	1.0	1.0
Damp 11	0.2	1.0	1.0	0.2	0.2
Damp 12	0.2	1.0	1.0	0.2	0.2
Damp 13	0.2	1.0	1.0	0.2	0.2
Damp 14	0.2	1.0	1.0	0.2	0.2
Damp 15	0.2	1.0	1.0	0.2	0.2
Damp 16	0.2	1.0	1.0	0.2	0.2
Damp 17	0.2	1.0	1.0	0.2	0.2

Table 15: Environment parameters for MO-Humanoid

Objectives	Default	Light	Heavy	Low Damping	Hard
Forward Reward	[0, 395.4]	[0, 548]	[0, 232]	[0, 374]	[0, 330]
Control Cost	[-35, 188]	[0, 156.4]	[0, 135.4]	[0, 182.4]	[-9, 88.6]

Table 16: Normalization Ranges for MO-Humanoid Environments