MULTI-POLICY PARETO FRONT TRACKING BASED MULTI-OBJECTIVE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-objective reinforcement learning (MORL) plays a pivotal role in addressing multi-criteria decision-making problems in the real world. The multi-policy (MP)-based approaches are widely used to obtain high-quality Pareto front approximations for the MORL problems. Relying primarily on the online reinforcement learning (RL), the traditional MP approaches usually adopt the evolutionary framework that requires maintaining a large policy population. In practice, however, this often leads to sample inefficiency and/or excessive agent-environment interactions. To address these issues, we propose the novel Multi-policy Pareto Front Tracking (MPFT) framework that eliminates the need to maintain any policy population, compatible with both online and offline MORL algorithms. The proposed MPFT framework comprises four stages: Stage 1 approximates all the Pareto-vertex policies whose mappings to the objective space lie on the vertices of the Pareto front; Stage 2 proposes a new Pareto tracking mechanism that starts from each Pareto-vertex policy to track the Pareto front, where a proof of its exponential convergence is provided; Stage 3 identifies the sparse regions in the tracked Pareto front, and then newly designs an objective weight adjustment method to facilitate the policy tracking for filling these regions; Finally, by combining all the policies tracked in Stages 2 and 3, Stage 4 approximates the complete Pareto front. Experiments are conducted on seven continuous-action robotic control tasks using both online and offline MORL algorithms. Results demonstrate that our proposed MPFT approach outperforms state-of-the-art benchmarks in terms of hypervolume and expected utility performances, while significantly reducing the agent-environment interactions.

1 INTRODUCTION

Deep Reinforcement Learning (RL) is a promising approach for solving decision-making problems. It has been widely used to a variety of fields such as robot control Duan et al. (2016), autonomous driving Feng et al. (2023), and wireless communications Zhao et al. (2024). In deep RL, the behavior of the agent is guided by a reward function that defines the optimization objective. However, decision-making problems may involve multiple conflicting objectives, which pose a severe challenge for RL applications in practice Dulac-Arnold et al. (2021). For example, the bipedal robot motion control may require optimizing two objectives: running speed and energy usage efficiency. However, while increasing the robot's speed may raise its energy consumption and thus decrease the energy usage efficiency, maximizing the energy usage efficiency could depress the robot's speed. In such multi-objective decision-making problems, a single optimal policy does not exist in general, because conflicting objectives cannot be simultaneously optimized. Instead, there exists a Pareto policy set. Each policy in this set corresponds to a distinct objective trade-off, and the final policy selection depends on the user preferences.

In this context, multi-objective reinforcement learning (MORL) has attracted the increasing attention. The single-policy (SP) based MORL was proposed to use one consistent policy to address the user's needs under a given user preference Yang et al. (2019). However, since the user preferences are very difficult to quantify Lin et al. (2024a), the SP-MORL policy generally cannot align with varied user preferences in practice. Unlike the SP-MORL, the multi-policy (MP) based MORL generates a high-quality Pareto-approximation policy set, where the user can freely select the policies that align with his/her preferences even in the real-time tasks Hu & Luo (2024).

It is noted that almost all of the existing MP-MORL approaches rely on the evolutionary framework, to simultaneously train a diverse population of policies that approximate the Pareto front for varied objective preferences. To let that population evolve in real time through massively parallel interactions and updates, the online RL algorithms like PPO Schulman et al. (2017) are usually employed. However, the evolutionary framework requires many agent–environment interactions, and online RL has low sample efficiency. As a result, MP-MORL methods built on evolutionary frameworks are hard to deploy in real-world settings where interactions and sampling are expensive.

This paper focuses on efficient MP-MORL for real-world applications. In contrast to the evolutionary framework, we propose a novel multi-policy Pareto front tracking (MPFT) framework that eliminates the need to maintain any policy population. This framework can incorporate both online and offline RL algorithms, achieving a tight approximation to the Pareto front with significantly reduced agent-environment interactions. In particular, we divide the Pareto-optimal policies into the Pareto-vertex policies and the Pareto-interior policies, whose mappings to the objective space lie on the vertices and the interior of the Pareto front, respectively. We find the approximate Pareto-vertex policies and the approximate Pareto-interior policies, and use them as the starting points to track the Pareto front, under the guidance of the proposed novel Pareto tracking mechanism. The main contributions of this work are summarized as follows:

• We propose the MPFT framework to obtain the high-quality Pareto-approximation policy set, consisting of four stages: 1) Stage 1 approximates the Pareto-vertex policies by solving a single-objective optimization problem for each objective; 2) Stage 2 initializes the Pareto-approximation policy set by parallelly tracking the Pareto front from each approximate Pareto-vertex policy; 3) Stage 3 fills the top-K sparse regions, by tracking the Pareto front from the corresponding K approximate Pareto-interior policies; 4) and lastly, Stage 4 completes the Pareto-approximation policy set, by combining the tracked policies from Stages 2 and 3.

• We propose the novel Pareto tracking mechanism to guide the Pareto front tracking in both Stages 2 and 3, and prove it converges exponentially toward a neighborhood of the Pareto-stationary set in any arbitrary objective dimension. Under this mechanism, the policies are updated continuously first along a newly defined Pareto-reverse direction and then along the Pareto-ascent direction, achieving tight tracking of the Pareto front with significantly reduced agent-environment interactions. To facilitate Stage 3, we also newly design an objective weight adjustment method to anchor the Pareto-interior policies.

• We incorporate both online RL and offline RL algorithms into the proposed MPFT framework. We extend the existing offline SAC and TD7 algorithms to their multi-objective versions, respectively, whose deployment under the proposed MPFT framework can further reduce the agent-environment interactions while increasing the sample efficiency.

• We evaluate the proposed MPFT framework using both online and offline algorithms under seven continuous-action robotic control tasks. As compared to state-of-the-art (SOTA) benchmarks, the proposed approach achieves superior hypervolume and expected utility performances, with up to a 77.72% reduction of the agent-environment interactions in our settings.

2 Related Works

Existing MORL approaches are broadly categorized into the SP-based and the MP-based approaches.

SP-MORL Approaches: To allow a single policy to cover all objective preferences, meta-RL is widely used in the design of the SP-MORL. Meta-RL-based approaches facilitate rapid online adaptation, making them particularly valuable in real-world settings where sampling is costly. Similarly, recent offline MORL algorithms, such as those in Yang et al. (2019), Zhu et al. (2023), and Lin et al. (2024b), leverage superior sample efficiency to swiftly derive policies aligned with user preferences. The framework proposed by Lin et al. (2024a) further extends these benefits by generating preference-aligned policies during deployment and by incorporating safety considerations.

Despite these strengths, several notable limitations persist. First, meta-RL approaches often exhibit suboptimal behaviors under a universal meta policy Chen et al. (2019). Second, the continual training of meta-RL agents may be unstable due to the issue of catastrophic forgetting Dohare et al. (2024). Third, the offline MORL algorithms generally depend heavily on users' prior preference in-

puts, restricting their applicability in domains where such inputs are unavailable (e.g., autonomous driving). It is noted that the paradigm in Lin et al. (2024a) obviates the need for pre-specified preferences during deployment. However, it is still fundamentally meta-RL–based, and cannot fully circumvent the issue of catastrophic forgetting. Another branch of SP-MORL is Pareto set learning, which indirectly realizes the effect of multi-policy by learning a "generalized policy generator" at once through conditional generation mechanisms (e.g., hypernetworks) Liu et al. (2025a); Nguyen et al. (2025), but it is still a SP-based approach, which is difficult to cover the entire Pareto front Lin et al. (2022). In addition the size of the hypernetwork will grow geometrically as the parameters of the policy network increase. Moreover, SP-based approaches suffer from gradient interference across objectives, which leads to performance imbalance over the preference space.

MP-MORL Approaches: Unlike the SP-MORL, the MP-MORL maintains a set of non-dominated policies (i.e., the Pareto-approximation policy set) to support flexible decision making Xu et al. (2020); Hayes et al. (2022); Alegre et al. (2022); Hu & Luo (2024). Hence, the MP-MORL allows users to select solutions that align with their preferences from the learned Pareto-approximation policy set Horie et al. (2019); Wen et al. (2020).

However, almost all the existing MP-MORL approaches rely on the evolutionary framework with online RL, such as the SOTA algorithms PGMORL Xu et al. (2020), PA2D-MORL Hu & Luo (2024), and C-MORL Liu et al. (2025b). Hence, the existing MP-MORL approaches generally suffer from the low sample efficiency and large agent-environment interactions. Moreover, under the evolutionary framework, since the policy corresponding to the solution farthest from the reference point is selected in each generation, the PGMORL and PA2D-MORL also suffers from the over-optimization of frequently selected policies and stagnation of others. To address this limitation, C-MORL replaces population-based policy selection during the evolution phase with a crowding-distance-based selection over the non-dominated set, achieving strong scalability with high-dimensional objectives. However, as other evolutionary methods, its applicability is still constrained in scenarios with expensive interactions. To alleviate the sample inefficiency, Tran et al. (2023) introduced offline RL in the warm-up phase, but it still uses the online RL for the policy training during the evolutionary phase, and it does not fundamentally solve this problem. In addition, Alegre et al. (2022) proposed a non-evolutionary approach that learns a convex coverage set (CCS) of successor features for optimal zero-shot transfer to arbitrary linear tasks, though its worst-case cost grows exponentially with the number of objectives. Alegre et al. (2023) further introduced Dyna-style MORL method (GPI-LS) to alleviate this problem. However, constructing CCS in high-dimensional objective tasks remains extremely time-consuming.

To address the above issues, we propose the novel MPFT framework compatible with both online and offline RL algorithms in this work, to tightly and efficiently approximate the Pareto front, where the time complexity increases linearly with the number of objectives.

3 Preliminaries

3.1 Multi-Objective Markov Decision Process

The MORL problem is usually modeled as a multi-objective Markov decision process (MOMDP). An MOMDP is defined as the tuple ($\mathcal{S}$,$\mathcal{A}$,$\mathcal{P}$,$\boldsymbol{R}$, γ). As in the standard MDP, the agent under the state $s \in \mathcal{S}$, takes an action $a \in \mathcal{A}$, and transits into a new state $s^{'} \in \mathcal{S}$ with the state transition probability $\mathcal{P}(s^{'}|s,a)$. The agent then obtains a reward vector $\boldsymbol{R} = [R_1, ..., R_m]^\top$ for $m > 1$ objectives with the discount factor $\gamma \in [0, 1]$, where R_i is the reward of the i-th objective, $i \in \{1, ..., m\}$, and $\top$ is the transposed operation. Denote the policy parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$ as $\pi_{\boldsymbol{\theta}} : \mathcal{S} \to \mathcal{A}$, where d is the dimension of $\boldsymbol{\theta}$. Since $\pi_{\boldsymbol{\theta}}$ can be completely represented by $\boldsymbol{\theta}$, we will use $\pi_{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$ interchangeably in the rest of this paper. In addition, Appendix A lists the key notations and definitions.

Denote the vector of expected return of the policy $\pi_{\boldsymbol{\theta}}$ as $\boldsymbol{J}(\boldsymbol{\theta}) = [J_1(\boldsymbol{\theta}), J_2(\boldsymbol{\theta}), ..., J_m(\boldsymbol{\theta})]^\top$, where $J_i(\boldsymbol{\theta})$ is the expected return of the i-th objective. As in Hu & Luo (2024), $J_i(\boldsymbol{\theta})$ is defined as:

$$J_i(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R_i(s_t, a_t) \middle| a_t \sim \pi_{\boldsymbol{\theta}}(s_t), s_0 = s\right], \tag{1}$$

where $\sim$ represents the sampling operation, $s_0 = s$ denotes the initial state, and $t \in \{0, \ldots, T\}$ is the timestep. The goal of the MOMDP problem is to find the optimal policy, that maximizes the

expected return $\boldsymbol{J}(\boldsymbol{\theta})$ of all the objectives. This can be achieved by using the policy gradient descent method. The policy gradient of the objective $i \in \{1, \ldots, m\}$ is given by:

$$\nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}) = -\nabla \mathbb{E}\left[\mathbf{Loss}_i(s, a)\right], \tag{2}$$

where $\mathbf{Loss}_i(s, a)$ is the loss function with respect to state s and action a. Let $\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta}) = [\nabla_{\boldsymbol{\theta}} J_1(\boldsymbol{\theta}), \ldots, \nabla_{\boldsymbol{\theta}} J_m(\boldsymbol{\theta})]^\top \in \mathbb{R}^{m \times d}$ represent the policy gradient matrix of the m objectives.

3.2 Pareto-Optimal Policy

For two policies $\pi_{\boldsymbol{\theta}^1}$ and $\pi_{\boldsymbol{\theta}^2}$, where $\boldsymbol{J}(\boldsymbol{\theta}^1) \neq \boldsymbol{J}(\boldsymbol{\theta}^2)$, we say that the policy $\pi_{\boldsymbol{\theta}^1}$ dominates the policy $\pi_{\boldsymbol{\theta}^2}$, if $J_i(\boldsymbol{\theta}^1) \geq J_i(\boldsymbol{\theta}^2)$, $\forall i \in \{1, \ldots, m\}$. In the Pareto policy set, there is no policy dominated by any other policy, i.e., each policy is Pareto-optimal. In other words, in the Pareto policy set, no objective can be further improved without sacrificing another objective. The mapping from the Pareto policy set to the objective space is known as the Pareto front Xu et al. (2020); Hu & Luo (2024). However, it is usually very difficult to obtain the Pareto policy set in practice. The goal becomes to find the Pareto-approximation policy set that best approximates the Pareto policy set.

3.3 Pareto Stationarity

To find the Pareto-approximation policy set, the MOMDP problem is usually transformed into a series of single-objective Markov decision process (SOMDP) problems, where the m objectives are scalarized into a single objective under different objective weight vectors $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_m]^\top$ in different SOMDP problems. For a given $\boldsymbol{\omega}$, the SOMDP problem is formulated as follows:

$$\begin{aligned} \text{(P1)}: \ & \max_{\boldsymbol{\theta}} \ \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\omega}, \\ & \text{s.t. } \omega_i \geq 0, \|\boldsymbol{\omega}\|_1 = 1, \forall i \in \{1, \ldots, m\}, \end{aligned}$$

where $\|\cdot\|_1$ represents the L1-norm operation. The solutions to all the SOMDP problems form the Pareto-approximation policy set.

However, the above method requires an exhaustive search of the objective weight vectors, which is impractical. As in Sener & Koltun (2018); Désidéri (2012), a policy $\pi_{\boldsymbol{\theta}}$ is Pareto-stationary, if $\min_{\boldsymbol{\omega}} \left\|\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\omega}\right\|_2^2 = 0$ holds, where $\|\cdot\|_2$ represents the L2-norm operation. Pareto stationarity is a necessary condition for a policy to be Pareto-optimal Désidéri (2012). As will be introduced in the next subsection, the Pareto-ascent direction is exploited to find the Pareto-stationary policy.

3.4 Pareto-Ascent Direction

The Pareto-ascent direction can be derived by solving the following problem:

$$\begin{aligned} \text{(P2)}: \ & \min_{\boldsymbol{\alpha} \in \mathbb{R}^m} \ \left\|\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}\right\|_2^2, \\ & \text{s.t. } \alpha_i \geq 0, \|\boldsymbol{\alpha}\|_1 = 1, \forall i \in \{1, \ldots, m\}, \end{aligned}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]^\top$. If $\left\|\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}^*\right\|_2^2 > 0$, where $\boldsymbol{\alpha}^*$ is the optimal solution to problem (P2), the Pareto-ascent direction is obtained as $\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}^*$. When the policy $\pi_{\boldsymbol{\theta}}$ is updated along the Pareto-ascent direction, all the objectives obtain the same amount of improvement, since the gradient vectors of different objectives project equally in the Pareto-ascent direction Hu & Luo (2024). If $\left\|\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}^*\right\|_2^2 = 0$, the corresponding policy $\pi_{\boldsymbol{\theta}}$ is Pareto-stationary at $\boldsymbol{\omega} = \boldsymbol{\alpha}^*$. Further, based on Zhou et al. (2024), if problem (P1) is convex, the Pareto-stationary $\pi_{\boldsymbol{\theta}}$ is also a Pareto-optimal policy; and if problem (P1) is non-convex, we say that the Pareto-stationary $\pi_{\boldsymbol{\theta}}$ is a Pareto-approximation policy. Please refer to Appendix B.1 for solving problem (P2).

4 MPFT Framework

In this section, we propose the Multi-policy Pareto Front Tracking (MPFT) framework.

4.1 Pareto-Vertex and Pareto-Interior Policy

We divide the Pareto-optimal policies into the Pareto-vertex policies and the Pareto-interior policies. Their definitions are given as follows:

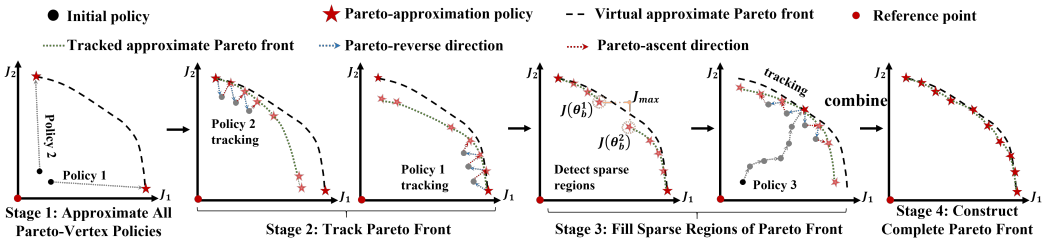

Figure 1: MPFT framework overview ($m = 2$, $K = 1$, $u = 1$, $v = 1$).

Definition 1. (Pareto-vertex policy[1]): For objective $i \in \{1, \ldots, m\}$, by setting $\omega_i = 1$ and $\omega_j = 0$, $\forall j \in \{1, \ldots, m\}$ and $j \neq i$, the optimal solution to problem (P1) is defined as the Pareto-vertex policy of objective i.

Definition 2. (Pareto-interior policy): A policy is a Pareto-interior policy, if it is Pareto-optimal and is not a Pareto-vertex policy.

From Definition 1, the Pareto-vertex policies are obtained by optimizing individual objectives. From Definition 2, the Pareto-interior policy can be obtained by solving the problem (P1) with an extra constraint ensuring that at least two elements in $\boldsymbol{\omega}$ are larger than zero. Since it is usually difficult to obtain the Pareto-vertex and Pareto-interior policies in complex environments, we will find the approximate Pareto-vertex and approximate Pareto-interior policies in the following subsections.

4.2 MPFT Framework Overview

The goal of the proposed MPFT framework is to find the Pareto-approximation policy set, denoted by $\mathcal{F}$, together with its mapping $\mathcal{N}$ to the objective space, which lies tightly close to the Pareto front.

As illustrated in Figure 1, the MPFT framework consists of four stages, and Figure 1 further visualizes the population-free training process in the bi-objective case, where only three policies are kept self-updating independently to track an approximate Pareto front:

1) *Stage 1: Approximate all Pareto-vertex policies.* For each objective i, $\forall i \in \{1, \ldots, m\}$, by continuously training in the direction of the gradient $\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\omega}$, where $\omega_i = 1$ and $\omega_j = 0$, $\forall j \in \{1, \ldots, m\}$ and $j \neq i$, we obtain the approximate Pareto-vertex policy $\pi_{\boldsymbol{\theta}^{i,*}}$.

2) *Stage 2: Track the Pareto front.* Starting from each $\pi_{\boldsymbol{\theta}^{i,*}}$, we design the Pareto-tracking mechanism to track the Pareto front, where the tracked policies form the Pareto-edge policy set[2], denoted by $\mathcal{F}^i_{edge}$, $i \in \{1, \ldots, m\}$. By deleting the dominated policies from the sets $\{\mathcal{F}^i_{edge}\}_{i \in \{1,\ldots,m\}}$, we obtain the initial $\mathcal{F} = \cup_{i=1}^{+m} \{\mathcal{F}^i_{edge}\}$ and its corresponding tracked Pareto front $\mathcal{N}$, where $\cup^+$ is the combined operation of the set union and the deletion of the dominated policies.

3) *Stage 3: Fill sparse regions of the tracked Pareto front $\mathcal{N}$.* We focus on the top-K sparse regions in $\mathcal{N}$, and find the K approximate Pareto-interior policies $\{\pi_{\boldsymbol{\theta}^{k,\star}}\}_{k \in \{1,\ldots,K\}}$. Then starting from each $\pi_{\boldsymbol{\theta}^{k,\star}}$, we continuously updating the policies to track the Pareto front. All the tracked policies form the k-th Pareto-interior policy tracking set, denoted by $\mathcal{F}^k_{inter}$, and are used to fill the k-th sparse region.

4) *Stage 4: Construct the complete $\mathcal{F}$.* The complete $\mathcal{F}$ is obtained as $\mathcal{F} \cup^+ \left\{\cup_{k=1}^{+K} \{\mathcal{F}^k_{inter}\}\right\}$.

Since the tracking mechanism performs local updates, one trajectory cannot move across disconnected regions of the Pareto front. Thus, MPFT launches multiple tracks, each starting from different policies to explore distinct local components. The four stages design inherently accommodates non-connected or discrete Pareto fronts. The Stage 1 and Stage 4 are straightforward, we detail Stages 2 and 3 in the following subsections, respectively.

[1]Here, the term "Pareto-vertex" simply refers to the single-objective extreme policies (not geometric vertices), which serve as the boundary anchors for initialization.

[2]The term "Pareto-edge" denotes the m-dimensional segment induced by any two adjacent preference vectors, not a geometric edge.

4.3 STAGE 2: TRACK PARETO FRONT

Following Li et al. (2025), the Pareto front is continuous under the topology of the policy parameter space for MOMDP with a continuous parameterized policy class, in the sense that sufficiently small perturbations in the policy parameters induce arbitrarily small changes in the objective vector. Therefore, starting from a high-quality approximate Pareto policy, only a small number of updates are typically required to reach a nearby Pareto solution. Enlightened by this, Stage 2 constructs m parallel tracks of the Pareto front, by sequentially generating the Pareto-edge policies $\{\pi_{\boldsymbol{\theta}^{i,(l)}}\}$, where $\pi_{\boldsymbol{\theta}^{i,(l+1)}}$ is updated from $\pi_{\boldsymbol{\theta}^{i,(l)}}$ in each track. For each $\pi_{\boldsymbol{\theta}^{i,*}}$, a total of Ψ_i training episodes are conducted. We also initialize $\pi_{\boldsymbol{\theta}^{i,(0)}} = \pi_{\boldsymbol{\theta}^{i,*}}$ and $\mathcal{F}^i_{edge} = \{\pi_{\boldsymbol{\theta}^{i,*}}\}$.

We propose the Pareto-tracking mechanism to guide the policy updating. Specifically, we newly define the Pareto-reverse direction $\nabla_{\boldsymbol{\theta}}\boldsymbol{J}(\boldsymbol{\theta})^\top\boldsymbol{\omega}$ of objective i as that leads to value increase of all $\{J_j(\boldsymbol{\theta})\}_{j\in\{1,\dots,m\},i\neq j}$ except $J_i(\boldsymbol{\theta})$. The Pareto-reverse direction of objective i, $\forall i \in \{1,\dots,m\}$, can be derived by solving problem (P2) with an extra constraint with $\alpha_i = 0$. By intentionally sacrificing one objective, the Pareto-reverse update induces a controllable policy updating direction that drives the policy away from the Pareto-vertex and along the Pareto front, thereby enabling traversal of the Pareto front. To obtain $\pi_{\boldsymbol{\theta}^{i,(l+1)}}$, $\pi_{\boldsymbol{\theta}^{i,(l)}}$ is first trained along the Pareto-reverse direction for $u \geq 0$ consecutive episodes, and then along the Pareto-ascent direction for $v \geq 0$ consecutive episodes. After obtaining $\pi_{\boldsymbol{\theta}^{i,(l+1)}}$, $\mathcal{F}^i_{edge}$ is updated as $\mathcal{F}^i_{edge} \cup^+ \{\pi_{\boldsymbol{\theta}^{i,(l+1)}}\}$. We also initialize $\mathcal{F}$ as $\cup_{i=1}^{+m}\{\mathcal{F}^i_{edge}\}$ by combing m Pareto front tracks. In Appendix C, we give a proof of exponential convergence of the Pareto-tracking mechanism. As shown in our convergence analysis, the Pareto-tracking mechanism converges exponentially toward a neighborhood of the Pareto-stationary set in any arbitrary objective dimension. Full coverage of the Pareto front is achieved empirically via the multi-track procedure in Stage 2 and Stage 3.

4.4 STAGE 3: FILL SPARSE REGIONS OF $\mathcal{N}$

Sparse regions in the approximated Pareto front naturally arise due to the potentially complex geometric structure of the Pareto front, and are difficult to rack by solely using Stage 2. Stage 3 is designed to address this challenge.

We focus on the top-K sparse regions, and propose the objective weight adjustment method to find K approximate Pareto-interior policies $\{\pi_{\boldsymbol{\theta}^{k,*}}\}_{k\in\{1,\dots,K\}}$, whose mapping to the objective space are located in the top-K sparse regions of $\mathcal{N}$, respectively. Unlike existing heuristic methods such as Hu & Luo (2024) or Liu et al. (2025b), which directly select policies from the archive $\mathcal{F}$, our approach retrains K new approximate Pareto-interior policies to avoid inheriting the coverage gaps produced in Stage 2. This ensures that Stage 3 can effectively fill sparse or disconnected regions of the Pareto front.

Specifically, we represent a sparse region by its m boundary policies, denoted as $\{\pi_{\boldsymbol{\theta}^i_b}\}_{i=1}^m$, whose mappings in the objective space form the bounding simplex that characterizes the location and extent of the sparse region. For the bi-objective case shown in Figure 1, the sparse region is represented by two boundary policies. Please refer to Appendix B.2 for more details on how to find the boundary policies. We denote by $\max^{\mathrm{v}}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_m)$ the element-wise maximum among $\boldsymbol{x}_i$, $i \in \{1,\dots,m\}$, and by $\mathrm{D}[\boldsymbol{x},\boldsymbol{y}]$ the element-wise division of $\boldsymbol{x}$ by $\boldsymbol{y}$. The following introduces the case with $K = 1$ to highlight the key steps of the objective weight adjustment method:

1) Find $\{\pi_{\boldsymbol{\theta}^i_b}\}_{i=1}^m$, and calculate $\boldsymbol{J}_{max} = \max^{\mathrm{v}}\left(\boldsymbol{J}(\boldsymbol{\theta}^1_b),\dots,\boldsymbol{J}(\boldsymbol{\theta}^m_b)\right)$. Initialize a random policy $\pi_{\boldsymbol{\theta}}$.

2) Find $\boldsymbol{\beta} = \mathrm{D}[\mathrm{J}_{\max}, \mathrm{J}(\boldsymbol{\theta})]$. Let weight $\boldsymbol{\omega} = \frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|_1}$[3].

3) Update $\pi_{\boldsymbol{\theta}}$ along $\nabla_{\boldsymbol{\theta}}\boldsymbol{J}(\boldsymbol{\theta})^\top\frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|_1}$ for one episode, resulting in a new policy $\pi_{\boldsymbol{\theta}'}$.

4) Let $\pi_{\boldsymbol{\theta}} = \pi_{\boldsymbol{\theta}'}$, and repeat steps 2) and 3) until $\|\boldsymbol{J}(\boldsymbol{\theta}) - \boldsymbol{J}_{max}\|_2 \leq \epsilon$ or reaching the maximum episode Ξ_k.

[3] $J_i(\boldsymbol{\theta}) \geq 0$ can be easily guaranteed in reward design for each objective $i \in \{1,\dots,m\}$ to assure non-negative weights.

Algorithm 1 MPFT algorithm

Input: episodes $\{\Xi_i\}_{i=1}^m$, $\{\Psi_i\}_{i=1}^m$, $\{\Xi_k\}_{k=1}^K$, $\{\Psi_k\}_{k=1}^K$, u, v, and timestep steps.
Initialize: $\mathcal{F}_{edge}^i = \emptyset$, $\mathcal{F}_{inter}^k = \emptyset$, $\pi_{\boldsymbol{\theta}^i}$, and $\pi_{\boldsymbol{\theta}^k}$, $\forall i \in \{1, \dots, m\}$, $\forall k \in \{1, \dots, K\}$.

1: Stage 1: Follow Section 4.2 to continuously train $\pi_{\boldsymbol{\theta}^i}$ for Ξ_i episodes to get $\pi_{\boldsymbol{\theta}^{i,*}}$. Initialize $\pi_{\boldsymbol{\theta}^{i,(l=0)}} = \pi_{\boldsymbol{\theta}^{i,*}}$ and $\mathcal{F}_{edge}^i = \{\pi_{\boldsymbol{\theta}^{i,*}}\}$;
2: Stage 2: Apply the Pareto-tracking mechanism in Section 4.3 to parallelly train each $\pi_{\boldsymbol{\theta}^{i,(l)}}$ for Ψ_i episodes. Sequentially update the Pareto-edge policy set $\mathcal{F}_{edge}^i = \mathcal{F}_{edge}^i \cup^+ \{\pi_{\boldsymbol{\theta}^{i,(l+1)}}\}$ for $\frac{\Psi_i}{u+v}$ times. Obtain $\mathcal{F} = \mathcal{F} \cup^+ \left\{\cup_{i=1}^{+m}\{\mathcal{F}_{edge}^i\}\right\}$;
3: Stage 3: Follow Section 4.4 to continuously train $\pi_{\boldsymbol{\theta}^k}$ for Ξ_k episodes based on the objective weight adjustment method to get $\pi_{\boldsymbol{\theta}^{k,\star}}$. Initialize $\pi_{\boldsymbol{\theta}^{k,(l=0)}} = \pi_{\boldsymbol{\theta}^{k,\star}}$ and $\mathcal{F}_{inter}^k = \{\pi_{\boldsymbol{\theta}^{k,\star}}\}$. Use the Pareto-tracking mechanism to continuously update $\mathcal{F}_{inter}^k$ for $\frac{\Psi_k}{u+v}$ times;
4: Stage 4: Complete $\mathcal{F} = \mathcal{F} \cup^+ \left\{\cup_{k=1}^{+K}\{\mathcal{F}_{inter}^k\}\right\}$;

Output: $\mathcal{F}$.

Figure 1 shows $\boldsymbol{J}_{max}$ when $m = 2$ and $K = 1$. By the above method, we can find the approximate Pareto-interior policy $\pi_{\boldsymbol{\theta}^{1,\star}}$ for $K = 1$, where the weight $\boldsymbol{\omega} = \frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|_1}$ in step 2) ensures that the update direction of the policy $\pi_{\boldsymbol{\theta}}$ is between the objectives $\{\boldsymbol{J}(\boldsymbol{\theta}^{\boldsymbol{i}})\}_{i=1}^m$. For the more complicated case with $K > 1$ as well as the methods to find the boundary policies and the sparse regions, please refer to Appendix B.2.

Next, as illustrated in Figure 1, by using the Pareto-tracking mechanism in Stage 2 to track toward each of the m objectives, where each $\pi_{\boldsymbol{\theta}^{k,\star}}$ is the start point of the tracking, we can obtain the K approximate Pareto-interior policy tracking sets $\mathcal{F}_{inter}^k$, $k \in \{1, ..., K\}$, parallelly. The top-K sparse regions from $\mathcal{N}$ are filled by $\{\mathcal{F}_{inter}^k\}_{k\in\{1,...,K\}}$.

Finally, by combining all the tracked policies in Stages 2 and 3, we complete $\mathcal{F} = \mathcal{F} \cup^+ \left\{\cup_{k=1}^{+K}\{\mathcal{F}_{inter}^k\}\right\}$ in Stage 4, where the corresponding $\mathcal{N}$ approximates the Pareto front tightly. The MPFT algorithm is presented in Algorithm 1, where each episode includes steps timesteps of policy training. Let Υ represent a fixed upper bound on the cost of training a single policy, independent of m. As analyzed in Appendix D.4, the time complexity of Algorithm 1 is $T_{\text{MPFT}} = \mathcal{O}\left(\Upsilon \times \text{steps} \times \left(\sum_{i=1}^m (\Xi_i + \Psi_i) + \sum_{k=1}^K (\Xi_k + \Psi_k)\right)\right)$, which scales linearly with the number of objectives m.

5 Multi-objective Online and Offline RL

This section takes the offline TD7 Fujimoto et al. (2023) as an example and incorporates it into our proposed MPFT framework. Other typical RL algorithms, such as the online PPO algorithm Schulman et al. (2017) and the offline SAC algorithm Haarnoja et al. (2018), can also be incorporated similarly, as given in Appendices B.3 and B.4, respectively. All these online and offline RL algorithms can be applied for the policy training in each episode of the MPFT algorithm.

The TD7 algorithm, as an improved version of the TD3 algorithm Fujimoto et al. (2018), is one of the SOTA offline RL algorithms. In TD7 algorithm, the high-dimensional state embedding vector z_s and state-action embedding vector z_{sa}, are used to improve the sample efficiency. To the best our knowledge, there is no existing multi-objective TD7 algorithms. In the following, we propose multi-objective TD7 (MOTD7) for the MPFT framework. Specifically, for the MOTD7, $\nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta})$ in (2) is obtained as $\nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}) = -\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathcal{B}\sim\mathcal{D}, s_t \leftarrow \mathcal{B}, a_t' = \pi_{\boldsymbol{\theta}}^{(z_s)}} [\mathbf{Loss}_i(s_t, a_t')] = \mathbb{E}\left[\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|} Q_i^{(z_s, z_{sa})}(s_t, a_t')\right]$, where $\leftarrow$ indicates an extract operation, $\mathcal{D}$ is the replay buffer, $\mathcal{B}$ is a mini-batch, $\pi_{\boldsymbol{\theta}}^{(z_s)}$ is the policy with z_s embedded, $Q_i^{(z_s, z_{sa})}(s_j, a_j)$ is the Q-value of objective i with z_s and z_{sa} embedded. For notation simplicity, we omit z_s and z_{sa}, and use $\pi_{\boldsymbol{\theta}}$ and Q_i to represent $\pi_{\boldsymbol{\theta}}^{(z_s)}$ and $Q_i^{(z_s, z_{sa})}$, respectively, the gradient direction of the policy $\pi_{\boldsymbol{\theta}}$ under a given weight $\boldsymbol{\omega}$ is: $\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\omega} = \mathbb{E}\left[\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|} \nabla_{\boldsymbol{\theta}} \boldsymbol{Q}^{\boldsymbol{\omega}}(s_t, a_t')\right]$, where $\boldsymbol{Q}^{\boldsymbol{\omega}}(s_t, a_t') = \boldsymbol{\omega}^\top \boldsymbol{Q}(s_t, a_t')$ is the weighted Q-value scalar, with $\boldsymbol{Q} = [Q_1, ..., Q_m]^\top$. Details of MOTD7 are provided in Ap-

pendix B.5. It is straightforward to incorporate MOTD7 into the proposed MPFT framework: one can apply Algorithm 5 in Appendix B.5 to policy training of each episode in Algorithm 1.

6 EXPERIMENTS

In our experiments, we adopt three widely-used metrics to evaluate the quality of the Pareto-approximation front: hypervolume (HV), sparsity (SP), and expected utility (EU). Briefly, the HV metric reflects the convergence, distribution, and homogeneity of the Pareto frontier approximation Falcón-Cardona et al. (2022), the SP metric measures its density Hu & Luo (2024), and the EU metric measures expected utility under different sampling preferences Liu et al. (2025b). Let $\mathbf{env\_steps}$ denote the total number of the agent-environment interactions, which is equivalent to the total training number of all the policies. As shown in Table 1, all methods are allocated an $\mathbf{env\_steps}$ budget sufficient to reach convergence, after which their converged performances are compared. Notably, the $\mathbf{env\_steps}$ required by MPFT is substantially lower than that of all benchmark methods across all environments. Please refer to Appendix D for HV, SP, EU, and $\mathbf{env\_steps}$ calculations.

6.1 SIMULATION ENVIRONMENT

The existing offline MORL datasets D4MORL in Zhu et al. (2023) and offline-MOO in Xue et al. (2024) are constructed from the PGMORL algorithm Xu et al. (2020). They are not proper for evaluating the performance of the MOSAC and the MOTD7. Moreover, since MOPPO is also not applicable to the offline dataset, we build our own robot control environment, which is more complicated than that in Xu et al. (2020) and Hu & Luo (2024).

Our main simulation environment is built by Mujuco-3.3.0 Todorov et al. (2012) and the v5 version of the game in Gymnasium-1.1.1. According to the official documentation Farama Foundation (2025), the v5 version fixes the bug in the previous version that the robot can still obtain rewards when it is in an unhealthy state, and increases the difficulty of robot control. Based on the v5 version, we set up seven robot control environments with continuous action spaces, which are HalfCheetah-2, Hopper-2, Swimmer-2, Ant-2, Walker2d-2, Humanoid-2, and Hopper-3, respectively. Except the three-objective environment Hopper-3, the others are all the two-objective environments. Additionally, to validate MPFT performance in discrete and higher-dimensional tasks, we constructed three environments identical to C-MORL Liu et al. (2025b): the discrete 4-objectives environment Lunar-Lander-4, the discrete 6-objectives environment Fruit-Tree-6, and the continuous 9-objectives environment Building-9. For detailed settings, please refer to Appendix E.1.

6.2 BENCHMARKS

To verify that the proposed MPFT framework can support to both online RL and offline RL, we implement the MOPPO, the MOSAC, and the MOTD7 in Section 5 under the MPFT framework, which give MPFT-MOPPO, MPFT-MOSAC, and MPFT-MOTD7, respectively. We also consider three evolutionary framework based MP-MORL algorithms as main benchmarks PGMORL Xu et al. (2020), PA2D-MORL Hu & Luo (2024), and C-MORL Liu et al. (2025b). For a detailed comparison between MPFT and this methods, please refer to Appendix G.

To ensure fairness, all evaluations and comparisons are implemented in the same environment. The PGMORL, the PA2D-MORL, the C-MORL and the MPFT-MOPPO use the online MOPPO algorithm and have the same network structure. We conduct the PGMORL based on the code repository (`https://github.com/mit-gfx/PGMORL`) provided by Xu et al. (2020) and the C-MORL based on the code repository (`https://github.com/RuohLiuq/C-MORL`) provided by Liu et al. (2025b). Since Hu & Luo (2024) did not open the codes, we implement PA2D-MORL exactly as described in Hu & Luo (2024) and put it with our codes online available[4]. Additionally, to show the actual performance of all the algorithms, we do not use the policy interpolation to fill in the sparse regions of the Pareto front. All baselines and MPFT are set with the same policy-buffer size, i.e., 200 for 2-objective environments and 300 for $\geq$3-objective environments. For details on the benchmark and MPFT parameter settings, please refer to Appendix E.2.

6.3 RESULTS

In experiments, all the MPFT framework based MP-MORL algorithms, we consider the top-1 sparse region with $K = 1$ if not otherwise specified.

[4]In order not to violate the principle of blind review, the link will be provided after the paper is published.

Table 1: Evaluation of HV, SP, EU, and **env_steps** for ≤3-objective environments. All data are based on three independent runs.

Environment	Metrics	MPFT-MOTD7	MPFT-MOSAC	MPFT-MOPPO	PGMORL	PA2D-MORL	C-MORL
Walker2d-2	HV ↑ ($\times 10^7$)	**2.04 ± 0.10**	1.72 ± 0.22	1.37 ± 0.03	1.11 ± 0.25	1.31 ± 0.09	1.21 ± 0.16
	SP ↓ ($\times 10^4$)	1.13 ± 0.53	1.52 ± 0.24	2.75 ± 0.56	2.02 ± 1.05	**0.32 ± 0.13**	0.57 ± 0.29
	EU ↑ ($\times 10^3$)	**4.44 ± 0.06**	4.01 ± 0.33	3.51 ± 0.02	3.25 ± 0.32	3.47 ± 0.15	3.29 ± 0.24
	env_steps ↓	**11,200,000**	16,400,000	17,612,800	41,943,040	41,943,040	41,943,040
Humanoid-2	HV ↑ ($\times 10^7$)	**6.66 ± 0.31**	1.28 ± 0.71	-	-	-	-
	SP ↓ ($\times 10^4$)	1.09 ± 0.33	**0.83 ± 0.16**	-	-	-	-
	EU ↑ ($\times 10^3$)	**7.70 ± 0.24**	3.58 ± 0.14	-	-	-	-
	env_steps ↓	**34,800,000**	42,400,000	43,008,000	137,625,600	137,625,600	137,625,600
HalfCheetah-2	HV ↑ ($\times 10^6$)	**3.51 ± 0.01**	3.25 ± 0.18	3.10 ± 0.26	3.04 ± 0.21	3.09 ± 0.17	2.93 ± 0.34
	SP ↓ ($\times 10^4$)	0.24 ± 0.13	1.48 ± 0.64	0.25 ± 0.17	0.13 ± 0.11	0.03 ± 0.01	**0.02 ± 0.01**
	EU ↑ ($\times 10^3$)	**1.78 ± 0.01**	1.72 ± 0.02	1.68 ± 0.17	1.66 ± 0.01	1.67 ± 0.16	1.61 ± 0.12
	env_steps ↓	**5,000,000**	7,800,000	13,516,800	41,943,040	41,943,040	41,943,040
Hopper-2	HV ↑ ($\times 10^7$)	**3.68 ± 0.27**	3.47 ± 0.44	3.37 ± 0.24	3.16 ± 0.15	3.18 ± 0.05	3.16 ± 0.24
	SP ↓ ($\times 10^4$)	**1.55 ± 0.55**	12.0 ± 5.96	2.66 ± 0.37	2.36 ± 1.68	2.28 ± 0.67	18.9 ± 4.17
	EU ↑ ($\times 10^3$)	**5.85 ± 0.15**	5.66 ± 0.49	5.44 ± 0.23	5.34 ± 0.21	5.39 ± 0.05	5.36 ± 0.37
	env_steps ↓	**19,200,000**	22,800,000	27,033,600	85,196,800	85,196,800	85,196,800
Ant-2	HV ↑ ($\times 10^6$)	**7.144 ± 0.189**	3.75 ± 0.82	0.43 ± 0.11	0.68 ± 0.17	0.87 ± 0.06	0.94 ± 0.35
	SP ↓ ($\times 10^3$)	**1.18 ± 0.11**	5.63 ± 2.21	1.32 ± 0.09	2.32 ± 1.00	1.36 ± 0.35	1.86 ± 0.38
	EU ↑ ($\times 10^3$)	**2.47 ± 0.04**	1.77 ± 0.25	0.54 ± 0.10	0.79 ± 0.10	0.88 ± 0.11	0.92 ± 0.14
	env_steps ↓	**19,200,000**	22,800,000	24,576,000	85,196,800	85,196,800	85,196,800
Swimmer-2	HV ↑ ($\times 10^6$)	**3.54 ± 0.07**	3.52 ± 0.07	3.49 ± 0.01	3.46 ± 0.08	3.51 ± 0.00	3.52 ± 0.04
	SP ↓ ($\times 10^3$)	0.81 ± 0.36	0.81 ± 0.15	0.33 ± 0.13	0.64 ± 0.30	**0.19 ± 0.03**	0.36 ± 0.09
	EU ↑ ($\times 10^3$)	**1.75 ± 0.02**	1.74 ± 0.01	1.74 ± 0.00	1.74 ± 0.01	1.74 ± 0.00	1.74 ± 0.01
	env_steps ↓	**3,800,000**	5,400,000	5,529,600	20,971,520	20,971,520	20,971,520
Hopper-3	HV ↑ ($\times 10^{10}$)	**9.61 ± 0.09**	8.54 ± 0.34	7.58 ± 0.50	7.62 ± 0.80	6.73 ± 0.35	7.19 ± 0.82
	SP ↓ ($\times 10^4$)	**0.32 ± 0.11**	2.03 ± 0.44	1.62 ± 0.40	4.85 ± 0.58	3.00 ± 0.58	3.28 ± 0.60
	EU ↑ ($\times 10^3$)	**4.44 ± 0.05**	4.37 ± 0.08	4.28 ± 0.04	4.30 ± 0.08	4.21 ± 0.06	4.27 ± 0.07
	env_steps ↓	**39,000,000**	41,600,000	55,705,600	135,168,000	135,168,000	135,168,000

Table 1 gives the evaluation results for all ≤3-objective environments[5], which shows that our proposed MPFT-MOTD7 framework can achieve better HV and EU performance with less **env_steps**. Specifically, **env_steps** of the MPFT-MOTD7, the MPFT-MOSAC, and the MPFT-PPO average decrease by 77.72%, 71.64%, and 66.63%, respectively, as compared to the evolutionary framework based algorithms PGMORL, PA2D-MORL and C-MORL.

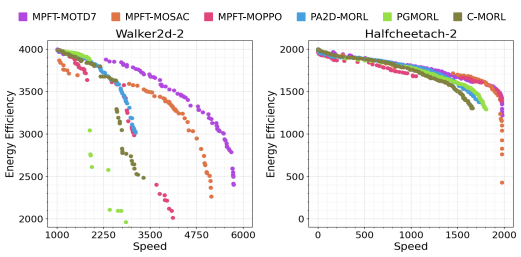

Figure 2: Comparison of Pareto front approximations.

It is also observed that in some environments our proposed MPFT framework does not achieve optimal SP performance, but this is a problem that cannot be avoided due to the reduced **env_steps** or achieving the same level of policy diversity as the benchmarks. This problem can be alleviated, however, by applying more advanced RL algorithms, such as the TD7 algorithm. Notably, MOPPO-based methods fail in the challenging Humanoid-2 (v5) task, whereas MOTD7 and MOSAC based methods succeed. This underscores a key advantage of MPFT framework: it can seamlessly leverage advanced RL algorithms with minimal hardware cost, in stark contrast to evolutionary framework whose integration of such algorithms demands prohibitive resources.

Figure 2 shows the Pareto front of all the algorithms in two different multi-objective environments, Walker2d-2 and HalfCheetah-2. It shows that our proposed MPFT framework can better explore policies whose mappings to the objective space locate almost at the edge of the Pareto front; moreover, our MPFT-MOTD7 obtains the best Pareto-approximating policy set among all the algorithms.

Table 2 further shows the ablation results of the MPFT-MOTD7 in the two-objective environments Walker2d-2 and HalfCheetah-2. Specifically, for different values of K, both the mean and standard deviation of the HV and SP metrics are reported. All data are based on three independent runs.

[5]MPFT-MOPPO, PGMORL, and PA2D-MORL are omitted since they fail to complete the task in Humanoid-2 (v5), where the maximum episode length is fixed at 1000 timesteps (the robot collapses before reaching the maximum timestep). The original PGMORL, PA2D-MORL, and C-MORL papers evaluated on Humanoid-2 (v2/v4) with a shorter horizon of 500 timesteps, making the results not directly comparable.

From Table 2, we generally observe an increased HV value and a reduced SP value as K increases, and the performance is lowest when $K = 0$, i.e., without Stage 3. This validates that the proposed Stage 3 can effectively fill the sparse regions of the tracked Pareto front in Stage 2. However, larger K values require more agent-environment interactions, and therefore trade-offs usually need to be made in practice.

Table 2: MPFT-MOTD7 with different K

K	Walker2d-2		HalfCheetah-2	
	HV ↑ ($\times 10^7$)	SP ↓ ($\times 10^4$)	HV ↑ ($\times 10^5$)	SP ↓ ($\times 10^3$)
0	1.95 ± 0.09	2.09 ± 0.99	34.7 ± 0.04	5.95 ± 5.42
1	2.04 ± 0.10	1.13 ± 0.53	35.1 ± 0.13	2.38 ± 1.30
3	2.15 ± 0.06	1.09 ± 0.28	35.2 ± 0.01	1.10 ± 0.25
6	2.20 ± 0.01	0.89 ± 0.13	35.4 ± 0.002	0.63 ± 0.04

To further evaluate MPFT in environments with more than three objectives and in discrete-action settings, we compare MPFT-MOTD7[6] against **GPI-LS** Alegre et al. (2023), **Envelope** Yang et al. (2019) and **C-MORL** across the Lunar-Lander-4, Fruit-Tree-6, and Building-9 benchmarks. For a fair comparison, Envelope, GPI-LS, and C-MORL are run under the same **env_steps** as reported in Liu et al. (2025b). Moreover, to highlight the efficiency of the proposed Pareto-tracking mechanism in tracking high-dimensional Pareto front, we intentionally disable *multiple policy tracking* and retain only a *single policy tracking*, which is called as Single-policy Pareto Front Tracking (SPFT). The hyperparameter configuration is reported in Appendix E.2. Table 3 gives the evaluation results for all >3-objectives environments, which demonstrate that SPFT achieves best performance on both HV and EU metrics with fewer agent-environment interactions, indicating the efficiency of the Pareto-tracking mechanism in tracing high-dimensional Pareto front. Additional results are reported in Appendix F, and further discussions and limitations of the proposed MPFT framework are presented in Appendix H.

Table 3: Evaluation of HV, SP, EU, and **env_steps** for >3-objective environments. All data are based on three independent runs. T/O indicates that the trainingtime exceeded the maximum limit of 100 hours. N/A indicates not applicable.

Environment	Metrics	Envelope	GPI-LS	C-MORL	Our (SPFT)
Fruit-Tree-6	HV ↑ ($\times 10^4$)	3.66 ± 0.23	3.57 ± 0.05	3.67 ± 0.14	$\mathbf{5.02 \pm 0.09}$
	SP ↓ ($\times 10^{-1}$)	5.46 ± 0.15	5.29 ± 0.21	$\mathbf{0.42 \pm 0.03}$	0.50 ± 0.18
	EU ↑ ($\times 10^0$)	6.15 ± 0.00	6.15 ± 0.00	6.53 ± 0.03	$\mathbf{6.59 \pm 0.08}$
	env_steps ↓	500,000	500,000	500,000	**67,200**
Lunar-Lander-4	HV ↑ ($\times 10^9$)	0.43 ± 0.18	1.06 ± 0.16	1.12 ± 0.03	$\mathbf{1.24 \pm 0.11}$
	SP ↓ ($\times 10^3$)	0.19 ± 0.16	$\mathbf{0.13 \pm 0.01}$	1.04 ± 0.24	1.75 ± 0.56
	EU ↑ ($\times 10^1$)	-2.84 ± 4.06	1.69 ± 0.34	$\mathbf{2.35 \pm 0.18}$	$\mathbf{2.36 \pm 0.29}$
	env_steps ↓	500,000	500,000	500,000	**480,000**
Building-9	HV ↑ ($\times 10^{31}$)	N/A	T/O	7.93 ± 0.07	$\mathbf{8.22 \pm 0.10}$
	SP ↓ ($\times 10^3$)	N/A	T/O	2.79 ± 0.40	$\mathbf{1.92 \pm 0.63}$
	EU ↑ ($\times 10^3$)	N/A	T/O	$\mathbf{3.50 \pm 0.00}$	$\mathbf{3.53 \pm 0.00}$
	env_steps ↓	N/A	2,500,000	2,500,000	**630,000**

7 Conclusion

We propose the MPFT framework to efficiently learn a high-quality Pareto-approximation policy set. It integrates both online and offline RL, leveraging a Pareto-tracking mechanism to reduce agent–environment interactions and an objective weight adjustment to guide policy training toward sparse Pareto regions. Experiments show MPFT achieves SOTA performance with lower computational and interaction costs.

We expect that more advanced RL algorithms can be integrated into our efficient MPFT framework in the future, to tackle sampling-difficult and resource-constrained tasks. Additionally, it is promising to apply the Pareto-tracking mechanism to various multi-objective decision-making problems to improve the training controllability and interpretability.

References

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

[6] Note that the original MOTD7 does not support discrete action settings. We adapted it for discrete versions using one-hot encoding.

Lucas Nunes Alegre, Ana Bazzan, 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*, pp. 2003–2012, 2023.

Xi Chen, Ali Ghadirzadeh, Mårten Björkman, and Patric Jensfelt. Meta-learning for multi-objective reinforcement learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 977–983. IEEE, 2019.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique*, 350:313–318, 2012.

Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pp. 1329–1338. PMLR, 2016.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

Jesús Guillermo Falcón-Cardona, Michael TM Emmerich, and CA Coello Coello. On the construction of Pareto-compliant combined indicators. *Evolutionary Computation*, 30(3):381–408, 2022.

Farama Foundation. Mujoco environments in gymnasium documentation. `https://gymnasium.farama.org/environments/mujoco/`, 2025. Accessed: 2025-04-15.

Shuo Feng, Haowei Sun, Xintao Yan, Haojie Zhu, Zhengxia Zou, Shengyin Shen, and Henry X Liu. Dense reinforcement learning for safety validation of autonomous vehicles. *Nature*, 615(7953): 620–627, 2023.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Scott Fujimoto, David Meger, and Doina Precup. An equivalence between loss functions and non-uniform sampling in experience replay. *Advances in neural information processing systems*, 33: 14219–14230, 2020.

Scott Fujimoto, Wei-Di Chang, Edward Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For sale: State-action representation learning for deep reinforcement learning. *Advances in neural information processing systems*, 36:61573–61624, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

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

Naoto Horie, Tohgoroh Matsui, Koichi Moriyama, Atsuko Mutoh, and Nobuhiro Inuzuka. Multi-objective safe reinforcement learning: the relationship between multi-objective reinforcement learning and safe reinforcement learning. *Artificial Life and Robotics*, 24:352–359, 2019.

Tianmeng Hu and Biao Luo. PA2D-MORL: Pareto ascent directional decomposition based multi-objective reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 12547–12555, 2024.

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518. Springer, 1992.

Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

Yining Li, Peizhong Ju, and Ness B Shroff. How to find the exact pareto front for multi-objective MDPs? In *The Thirteenth International Conference on Learning Representations*, 2025.

Qian Lin, Zongkai Liu, Danying Mo, and Chao Yu. An offline adaptation framework for constrained multi-objective reinforcement learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

Qian Lin, Chao Yu, Zongkai Liu, and Zifan Wu. Policy-regularized offline multi-objective reinforcement learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 1201–1209, 2024b.

Xi Lin, Zhiyuan Yang, Xiaoyuan Zhang, and Qingfu Zhang. Pareto set learning for expensive multi-objective optimization. *Advances in neural information processing systems*, 35:19231–19247, 2022.

Erlong Liu, Yu-Chang Wu, Xiaobin Huang, Chengrui Gao, Ren-Jian Wang, Ke Xue, and Chao Qian. Pareto set learning for multi-objective reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 18789–18797, 2025a.

Ruohong Liu, Yuxin Pan, Linjie Xu, Lei Song, Pengcheng You, Yize Chen, and Jiang Bian. Efficient discovery of Pareto front for multi-objective reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Jaume Llibre, Douglas D Novaes, Marco A Teixeira, et al. Higher order averaging theory for finding periodic solutions via brouwer degree. *Nonlinearity*, 27(3):563–583, 2014.

Minh-Duc Nguyen, Phuong Mai Dinh, Quang-Huy Nguyen, Long P Hoang, and Dung D Le. Improving Pareto set learning for expensive multi-objective optimization via stein variational hypernetworks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19677–19685, 2025.

Jan A Sanders, Ferdinand Verhulst, and James Murdock. *Averaging methods in nonlinear dynamical systems*, volume 59. Springer, 2007.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

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. IEEE, 2012.

Hai-Long Tran, Long Doan, Ngoc Hoang Luong, and Huynh Thi Thanh Binh. A two-stage multi-objective evolutionary reinforcement learning framework for continuous robot control. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 577–585, 2023.

Yue Wang and Shaofeng Zou. Policy gradient method for robust reinforcement learning. In *International conference on machine learning*, pp. 23484–23526. PMLR, 2022.

Lu Wen, Jingliang Duan, Shengbo Eben Li, Shaobing Xu, and Huei Peng. Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–7. IEEE, 2020.

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

Ke Xue, Rongxi Tan, Xiaobin Huang, and Chao Qian. Offline multi-objective optimization. In *International Conference on Machine Learning*, pp. 55595–55624. PMLR, 2024.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *Advances in neural information processing systems*, 32, 2019.

Ze Yu Zhao, Yue Ling Che, Sheng Luo, Gege Luo, Kaishun Wu, and Victor CM Leung. On designing multi-UAV aided wireless powered dynamic communication via hierarchical deep reinforcement learning. *IEEE Transactions on Mobile Computing*, 23(12):13991–14004, 2024.

Tianchen Zhou, Fnu Hairi, Haibo Yang, Jia Liu, Tian Tong, Fan Yang, Michinari Momma, and Yan Gao. Finite-time convergence and sample complexity of actor-critic multi-objective reinforcement learning. In *International Conference on Machine Learning*, pp. 61913–61933. PMLR, 2024.

Baiting Zhu, Meihua Dang, and Aditya Grover. Scaling Pareto-efficient decision making via offline multi-objective RL. In *The Eleventh International Conference on Learning Representations*. PMLR, 2023.

USE OF LLMS

This work made limited use of state-of-the-art large language models (LLMs) for language-related assistance. The LLMs were employed only for grammar correction, style polishing, typographical improvements, figure formatting, and debugging minor code errors (e.g., diagnosing runtime issues). Importantly, no LLM was used to generate or modify the core research content, algorithms, data, results, or conclusions. All experimental findings reported in this paper are genuine results produced by the authors' own implementations and experiments.

A KEY NOTATIONS

Notation	Description
$\pi_{\boldsymbol{\theta}}$	Policy parameterized by $\boldsymbol{\theta}$
$\boldsymbol{J}(\boldsymbol{\theta})$	Multi-objective expected-return vector of $\pi_{\boldsymbol{\theta}}$
$J_i(\boldsymbol{\theta})$	Expected return for the i-th objective
$\nabla_\theta J_i(\boldsymbol{\theta})$	Policy-gradient vector for the i-th objective
$\boldsymbol{\omega}$, $\boldsymbol{\alpha}$	Weight vector over objectives, element of the simplex Δ_m
$\pi_{\boldsymbol{\theta}^{i,*}}$	Pareto-vertex policy of objective i
$\pi_{\boldsymbol{\theta}^{k,\star}}$	Pareto-interior policy in k-th sparse region
$\mathcal{F}$	Pareto-approximating policy set
$\mathcal{F}^i_{edge}$	Pareto-edge policy set tracked from $\pi_{\boldsymbol{\theta}^{i,*}}$
$\mathcal{F}^k_{inter}$	Pareto-interior policy tracking set tracked from $\pi_{\boldsymbol{\theta}^{k,\star}}$
$\mathcal{N}$	The mapping of $\mathcal{F}$ in the objective space (approximated Pareto front)
K	Number of top-K sparse regions
u, v	Training episode number for Pareto-reverse and Pareto-ascent phase
Ξ_i, Ψ_i	Training episodes for $\pi_{\boldsymbol{\theta}^{i,*}}$ and $\mathcal{F}^i_{edge}$
Ξ_k, Ψ_k	Training episodes for $\pi_{\boldsymbol{\theta}^{k,\star}}$ and $\mathcal{F}^k_{inter}$

B ALGORITHM DETAIL

B.1 PARETO-ASCENT DIRECTION

For problem (P2), the stochastic gradient descent can be used to find the $\boldsymbol{\alpha}^*$. Specifically, for the constraints, the projection function $\mathcal{P}_{ro}(\cdot)$ can be used with $\boldsymbol{\alpha}^{t+1} = \mathcal{P}_{ro}(\boldsymbol{\alpha}^t - \eta \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}^t))$, where $f(\boldsymbol{\alpha}) = \left\| \nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\alpha} \right\|_2^2$ and η is the step size. For the case with two objectives, i.e., $m = 2$, we can find the analytical solution of $\boldsymbol{\alpha}^* = [\alpha_1^*, \alpha_2^*]^\top$, which is either orthogonal to the difference of

the two gradient vectors $\nabla_{\boldsymbol{\theta}} J_1(\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}} J_2(\boldsymbol{\theta})$ or coincides with one of the gradient vectors, i.e.,

$$\alpha_1^* = \max\left(\min\left(\frac{\left(\nabla_{\boldsymbol{\theta}} J_2(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} J_1(\boldsymbol{\theta})\right)^\top \nabla_{\boldsymbol{\theta}} J_2(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}} J_2(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} J_1(\boldsymbol{\theta})\|_2^2}, 1\right), 0\right),$$

other is $\alpha_2^* = 1 - \alpha_1^*$. The Pareto-ascent direction is therefore found as $\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}^*$ for the policy $\pi_{\boldsymbol{\theta}}$.

B.2 Finding Boundary Policies for Top-K Sparse Regions

To find the top-K sparse regions from $\mathcal{N}$, we can first view the solutions in $\mathcal{N}$ as points in an m-dimensional (m-D) space. We focus on the case with $m \leq 3$, as shown in Algorithm 2.

When $m = 2$, the algorithm proceeds as follows: 1) Based on the "non-dominated" property of the solutions in $\mathcal{N}$, we can sort each point in ascending order according to the value of objective 1; 2) Calculate the Euclidean distance between each adjacent pair of points; 3) Select the top-K pairs of points with the largest distances; 4) The K point pairs represent the top-K sparse regions, and the boundaries of these K sparse regions are returned as $\{\boldsymbol{J}_{max}^1, \ldots, \boldsymbol{J}_{max}^K\}$.

Algorithm 2 Boundary Detection of Top-K Sparse Regions

Input `arr` = $\mathcal{N} \in \mathbb{R}^{n \times m}$: A set of n solutions in m-dimensional objective space (Pareto Front); K: Number of sparse regions to detect;
Ensure `SparseRegions`: Set of detected sparse regions (line segments for $m = 2$, triangles for $m = 3$);
Initialize `SparseRegions` as empty set;
if $m == 2$ **then**
  Sort `arr` by the first objective value;
  Compute pairwise Euclidean distances between adjacent points: $d_i = \|\texttt{arr}[i+1] - \texttt{arr}[i]\|_2$ for $i = 0, \cdots, n-2$;
  Select the indices of top-K largest distances;
  for each selected index i **do**
    Append segment $(\texttt{arr}[i], \texttt{arr}[i+1])$ to `SparseRegions`;
  end for
else if $m == 3$ **then**
  Project 3D points onto a best-fit 2-D plane using PCA:
  $\texttt{arr\_proj}, \texttt{pca} \leftarrow$ ProjectToPlane$(\texttt{arr})$
  Perform Delaunay triangulation on projected 2-D points;
  For each triangle, compute its area;
  Select top-K triangles with largest areas;
  Map selected triangles back to original 3-D space using `pca`;
  Add them to `SparseRegions`;
end if
Caculate the boundaries $\{\boldsymbol{J}_{max}^1, \ldots, \boldsymbol{J}_{max}^K\}$ of the `SparseRegions` $\in \mathbb{R}^{K \times m \times m}$
Output $\{\boldsymbol{J}_{max}^1, \ldots, \boldsymbol{J}_{max}^K\}$

When $m = 3$, the points in $\mathcal{N}$ form a convex surface in 3-D space Li et al. (2025). To identify sparse regions on the surface, we develop the following algorithm: 1) Use principal component analysis (PCA) to project the 3-D point set $\mathcal{N}$ onto the best-fitting 2-D plane; 2) Use Delaunay triangulation to construct a triangular network on the 2-D plane; 3) Calculate the area of each triangle in the triangular network and select the top-K triangles with the largest areas; 4) Map the selected top-K triangles back to the 3-D space; 5) Use the vertices of these K triangles to represent the sparse regions and return the boundaries of these K sparse regions $\{\boldsymbol{J}_{max}^1, ..., \boldsymbol{J}_{max}^K\}$.

Moreover, although not the focus of this paper, for higher dimensions with $m > 3$, we can use PCA dimension reduction to reduce it to 3-D, and then use the above algorithm to obtain the boundaries of the top K sparse regions. In addition, we can also apply the crowding distance defined in Liu et al. (2025b) to find the top-K solutions that lie in the sparse region of the case with $m > 3$. However, both methods are heuristic, and neither has a formal proof.

B.3 MOPPO ALGORITHM

Most existing MP-MORL research are based on the PPO algorithm Schulman et al. (2017), primarily because PPO supports large-scale parallel training and aligns well with the evolutionary framework used in existing MP-MORL algorithms. Although the evolutionary framework can fully explore more policies on the Pareto front to obtain a denser Pareto policy set, this is feasible only in simulation, while in real applications, each policy needs to interact with the environment and adopt its own trajectory for training, which is obviously impractical. We design an MOPPO algorithm that can be applied within our proposed MPFT framework.

Unlike the traditional single-objective PPO algorithm, the expected return in MOPPO is a weighted sum of multiple objectives. To avoid relearning the state value function when $\boldsymbol{\omega}$ changes, we need to define a vectorized state value function $\boldsymbol{V}(s;\boldsymbol{\phi}) = [V_1(s;\boldsymbol{\phi}), ..., V_m(s;\boldsymbol{\phi})]^\top \in \mathbb{R}^m$ to evaluate each objective $i \in \{1, ..., m\}$'s state value, where $\boldsymbol{\phi}$ is parameter of $\boldsymbol{V}$. $\boldsymbol{V}(s;\boldsymbol{\phi})$ is defined as

$$\boldsymbol{V}(s;\boldsymbol{\phi}) = \mathbb{E}_{a_t\sim\pi_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{T}\gamma^t \boldsymbol{R}(s_t,a_t)\mathbf{Done}_t \middle| s_0 = s\right],$$

which is updated following Bellman equation. $\mathbf{Done}_t \in \{0,1\}$ is the termination condition, with $\mathbf{Done}_t = 0$ if the task terminates at timestep t, or $\mathbf{Done}_t = 1$, otherwise. Note that, although the $\mathbf{Done}_t = 1$ variable already denotes task completion, a finite maximal horizon T is retained in practice to avoid pathological non-termination cases and to align with standard RL environments. The theoretical analysis uses $T = \infty$, whereas the finite T in implementation serves only as a practical safeguard

$$\mathcal{T}^\pi \boldsymbol{V}(s_t;\boldsymbol{\phi}) = \hat{\boldsymbol{V}}(s_t) = \sum_{a_t}\pi_{\boldsymbol{\theta}}(a_t|s_t)\left[\boldsymbol{R}(s_t,a_t)+\gamma \boldsymbol{V}(s_{t+1};\boldsymbol{\phi})\right],$$

where $\mathcal{T}^\pi$ is the Bellman backup operator. For each objective i, the policy gradient with advantage estimation Schulman et al. (2015) is defined as $\nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}) = -\nabla \mathbb{E}_{s\sim e_{\boldsymbol{\theta}}, a\sim\pi_{\boldsymbol{\theta}}}[\mathbf{Loss}_i(s_t,a_t)] = \mathbb{E}\left[\sum_{t=0}^{T} A_i(s_t,a_t)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t|s_t)|s_0 = s\right]$, where $e_{\boldsymbol{\theta}}$ is the state distribution based on policy $\pi_{\boldsymbol{\theta}}$, $A_i(s_t,a_t) \in \boldsymbol{A}(s_t,a_t)$ is the advantage function for objective i, and $\boldsymbol{A}(s_t,a_t) = [A_1(s_t,a_t), ..., A_m(s_t,a_t)]^\top \in \mathbb{R}^m$. The gradient direction of the policy $\pi_{\boldsymbol{\theta}}$ under a given weight $\boldsymbol{\omega}$ is

$$\begin{aligned}\boldsymbol{\omega}^\top \nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta}) &= \sum_{i=1}^{m} w_i \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}) \\ &= \mathbb{E}\left[\sum_{t=0}^{T}\boldsymbol{\omega}^\top \boldsymbol{A}(s_t,a_t)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t|s_t)\right] \\ &= \mathbb{E}\left[\boldsymbol{A}^{\boldsymbol{\omega}}(s_t,a_t)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t|s_t)\right],\end{aligned}$$

where $\boldsymbol{A}^{\boldsymbol{\omega}}(a_t,s_t) = \boldsymbol{\omega}^\top \boldsymbol{A}(a_t,s_t)$ is the weighted advantage scalar. Integrating the above into PPO algorithm, we obtain the MOPPO policy training process:

1) Collect trajectories (i.e., rollout) using the current policy $\pi_{\boldsymbol{\theta}}$.
2) Calculate the advantage by the GAE approach in Schulman et al. (2015), $\boldsymbol{A}^{\boldsymbol{\omega}}(s_t,a_t) = \boldsymbol{\omega}^\top \sum_{l=0}^{\infty}(\gamma\lambda)^l \boldsymbol{\delta}_{t+1}$ with $\boldsymbol{\delta}_t = \boldsymbol{R}(s_t,a_t) + \gamma \boldsymbol{V}(s_{t+1};\boldsymbol{\phi}) - \boldsymbol{V}(s_t;\boldsymbol{\phi})$, and $t \in \{1, ..., T\}$ is the timestep.
3) Update the policy $\pi_{\boldsymbol{\theta}}$ by minimizing the clipped surrogate loss $L^\pi(\boldsymbol{\theta}) = -\mathbb{E}[\min(r_t(\boldsymbol{\theta})\boldsymbol{A}^{\boldsymbol{\omega}}(s_t,a_t), \texttt{clip}(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\boldsymbol{A}^{\boldsymbol{\omega}}(s_t,a_t))]$, where $r_t(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(a_t|s_t)}{\pi_{old}(a_t|s_t)}$, $\epsilon \in [0,1)$, $\texttt{clip}(x, 1-\epsilon, 1+\epsilon)$ is a clip function make the value of x lies in the range of $[1-\epsilon, 1+\epsilon]$, and π_{old} is the old policy.
4) Update the state value function $\boldsymbol{V}(s_t;\boldsymbol{\phi})$ by the MSE Loss $L^V(\boldsymbol{\phi}) = \mathbb{E}\left[\boldsymbol{V}(s_t;\boldsymbol{\phi}) - \hat{\boldsymbol{V}}(s_t)\right]$.
5) Repeat steps 3) and 4) until reaching the maximum epoch.

In each MPFT-MOPPO episode, the policy training process is shown in Algorithm 3.

Algorithm 3 MOPPO Algorithm

1: **Input** $\pi_{\boldsymbol{\theta}}$, and $\boldsymbol{V}(s;\boldsymbol{\phi})$;
2: **Obtain** $\boldsymbol{\omega}$ based on the Pareto-tracking mechanism;
3: **Fixed** initialize ϵ, T, and epoch;
4: `Rollout`: Run policy π_{old} in environment for steps timesteps to collect trajectories. (Agent interacts with the environment steps times);
5: `Compute advantage`: $\{\boldsymbol{A}^{\boldsymbol{\omega}}(s_t,a_t)\}_{t=1}^{T}$;
6: **for** e $\leftarrow 1,\cdots,$ epoch **do**
7: `Update the policy`: Use loss function L^{π} to update $\boldsymbol{\theta}$;
8: `Update the state value network`: Use loss function $L^{V}(\boldsymbol{\phi})$ to update $\boldsymbol{\phi}$;
9: **end for**
10: **Output**: $\pi_{\boldsymbol{\theta}}$ and $\boldsymbol{V}(s;\boldsymbol{\phi})$;

B.4 MOSAC ALGORITHM

The SAC algorithm is an offline RL algorithm based on entropy maximization Haarnoja et al. (2018). Its output, similar to PPO, is a probability distribution over actions.

Unlike MOPPO, MOSAC has a state-action value (Q-value) network $\boldsymbol{Q}(s,a;\boldsymbol{\varphi}) = [Q_1(s,a;\boldsymbol{\varphi}),\ldots,Q_m(s,a;\boldsymbol{\varphi})]^{\top} \in \mathbb{R}^m$ in addition to $\boldsymbol{V}(s;\boldsymbol{\phi}) = [V_1(s;\boldsymbol{\phi}),\ldots,V_m(s;\boldsymbol{\phi})]^{\top} \in \mathbb{R}^m$, where $\boldsymbol{\varphi}$ are the parameters of $\boldsymbol{Q}(s,a;\boldsymbol{\varphi})$. The update of $\boldsymbol{Q}(s,a;\boldsymbol{\varphi})$ is given by the following Bellman equation:

$$\mathcal{T}^{\pi}\boldsymbol{Q}(s_t,a_t;\boldsymbol{\varphi}) = \hat{\boldsymbol{Q}}(s_t,a_t) = \mathbb{E}_{\mathcal{B}\sim\mathcal{D},(\boldsymbol{R}(s_t,a_t),s_{t+1},\mathbf{Done}_t)\leftarrow\mathcal{B}}\left[\boldsymbol{R}(s_t,a_t) + \gamma\boldsymbol{V}(s_{t+1};\boldsymbol{\phi})\mathbf{Done}_t\right],$$

where $\leftarrow$ indicates extract operation, $\mathcal{D}$ is the replay buffer, $\mathcal{B}$ is a mini-batch, and $\boldsymbol{V}(s_t;\boldsymbol{\phi}) = \mathbb{E}_{s_t\leftarrow\mathcal{B},a_t'\sim\pi_{\boldsymbol{\theta}}}\left[\boldsymbol{Q}(s_t,a_t';\boldsymbol{\varphi}) - \log\pi_{\boldsymbol{\theta}}(a_t'|s_t)\right]$ is the soft state value function.

To prevent overestimation of the Q-value, we used two state-action value networks, $\boldsymbol{Q}_1(s,a;\boldsymbol{\varphi}_1)$ and $\boldsymbol{Q}_2(s,a;\boldsymbol{\varphi}_2)$ to estimate the minimum Q-value $\boldsymbol{Q}_{est}(s,a) = \min\left(\{\boldsymbol{Q}_j(s,a;\boldsymbol{\varphi}_j)\}_{j=1}^{2}\right) \in \mathbb{R}^m$. The policy gradient of objective $i \in \{1,...,m\}$ is defined as: $\nabla_{\boldsymbol{\theta}}J_i(\boldsymbol{\theta}) = -\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathcal{B}\sim\mathcal{D},s_t\leftarrow\mathcal{B},a_t'\sim\pi_{\boldsymbol{\theta}}}\left[\mathbf{Loss}_i(s_t,a_t')\right] = \mathbb{E}_{s_t\leftarrow\mathcal{B}}\left[\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|}\kappa_i\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t'|s_t) - \nabla_{\boldsymbol{\theta}}Q_i(s_t,a_t')\right]$, where κ_i is the temperature parameter for the entropy regularizer of objective i, and $Q_i(s_t,a_t')$ is the i-th element of $\boldsymbol{Q}_{est}(s_t,a_t')$. The gradient direction of the policy $\pi_{\boldsymbol{\theta}}$ under a given weight $\boldsymbol{\omega}$ is:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\boldsymbol{J}(\boldsymbol{\theta})^{\top}\boldsymbol{\omega} &= -\nabla_{\boldsymbol{\theta}}\boldsymbol{\omega}^{\top}\mathbb{E}_{s_t\leftarrow\mathcal{B}}\left[\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|}\log\pi_{\boldsymbol{\theta}}(\pi_{\boldsymbol{\theta}}(s_t)|s_t)\boldsymbol{\kappa} - \boldsymbol{Q}(s_t,\pi_{\boldsymbol{\theta}}(s_t))\right] \\
&= -\mathbb{E}\left[\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|}\boldsymbol{\kappa}^{\boldsymbol{\omega}}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\pi_{\boldsymbol{\theta}}(s_t)|s_t) - \nabla_{\boldsymbol{\theta}}\boldsymbol{Q}^{\boldsymbol{\omega}}(s_t,\pi_{\boldsymbol{\theta}}(s_t))\right] \\
&= -\mathbb{E}\left[\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|}\boldsymbol{\kappa}^{\boldsymbol{\omega}}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t'|s_t) - \nabla_{\boldsymbol{\theta}}\boldsymbol{Q}^{\boldsymbol{\omega}}(s_t,a_t')\right]
\end{aligned}$$

where $\boldsymbol{\kappa}^{\boldsymbol{\omega}} = \boldsymbol{\omega}^{\top}\boldsymbol{\kappa}$ and $\boldsymbol{Q}^{\boldsymbol{\omega}}(s_t,a_t') = \boldsymbol{\omega}^{\top}\boldsymbol{Q}_{est}(s_t,a_t')$ are weighted entropy temperature scalar and weighted Q-value scalar, respectively, with $\boldsymbol{\kappa} = [\kappa_1,...,\kappa_m]^{\top}$.

The following is the multi-objective loss function for all networks:

1) State value network loss:

$$L^{V}(\boldsymbol{\phi}) = \mathbb{E}_{s_t\leftarrow\mathcal{B},a_t'\sim\pi_{\boldsymbol{\theta}}}\left[\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|}\left\|\boldsymbol{V}(s_t;\boldsymbol{\phi}) - \left(\boldsymbol{Q}_{est}(s_t,a_t') - \nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t'|s_t)\cdot\boldsymbol{\kappa}\right)\right\|_2^2\right].$$

2) State-action value network loss:

$$L_j^Q(\boldsymbol{\varphi}_j) = \mathbb{E}_{(s_t,a_t,\boldsymbol{R}(s_t,a_t),s_{t+1})\leftarrow\mathcal{B}}\left[\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|}\left\|\boldsymbol{Q}_j(s_t,a_t;\boldsymbol{\varphi}_j)-\hat{\boldsymbol{Q}}(s_t,a_t)\right\|_2^2\right], j\in\{1,2\}.$$

3) Policy loss:

$$L^\pi(\boldsymbol{\theta}) = \mathbb{E}_{s_t\leftarrow\mathcal{B},a_t'\sim\pi_{\boldsymbol{\theta}}}\left[\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|}\boldsymbol{\kappa}^{\boldsymbol{\omega}}\log\pi_{\boldsymbol{\theta}}(a_t'|s_t)-\boldsymbol{\omega}^\top\boldsymbol{Q}_{est}\left(s_t,a_t'\right)\right].$$

4) Entropy temperature loss:

$$L^\kappa(\boldsymbol{\kappa}) = \mathbb{E}_{s_t\leftarrow\mathcal{B},a_t'\sim\pi_{\boldsymbol{\theta}}}\left[-\frac{1}{|\mathcal{B}|\cdot m}\sum_{t=1}^{|\mathcal{B}|}\sum_{i=1}^{m}\left(\log\kappa_i\cdot\left(\log\pi_{\boldsymbol{\theta}}(a_t'|s_t)+\hat{\mathcal{H}}_i\right)\right)\right],$$

where $\hat{\mathcal{H}}_i$ is the target entropy of the objective i, it is usually set to $-|\mathcal{A}|$.

In each MPFT-MOSAC episode, the policy training process is shown in Algorithm 4, where the `Data Storage` operation can be omitted if the offline dataset $\mathcal{D}$ is given and the amount of data is sufficient.

Algorithm 4 MOSAC Algorithm

1: **Input** $\pi_{\boldsymbol{\theta}}$, $\{\boldsymbol{Q}_j(s,a;\boldsymbol{\varphi}_j)\}_{j=1}^2$, $\boldsymbol{V}(s;\boldsymbol{\phi})$, and $\mathcal{D}$;
2: **Obtain** $\boldsymbol{\omega}$ based on the Pareto-tracking mechanism;
3: **Fixed** initialize steps and $\{\hat{\mathcal{H}}_i\}_{i=1}^m$;
4: `Data Storage`: Run policy $\pi_{\boldsymbol{\theta}}$ for steps timesteps in the environment to collect experiences, and store these experiences into $\mathcal{D}$. (Agent interacts with the environment steps times);
5: **for** t $\leftarrow 1,\cdots,$ steps **do**
6: `Update the policy`: Use loss function L^π to update $\boldsymbol{\theta}$;
7: `Update the state value network`: Use loss function $L^V(\boldsymbol{\phi})$ to update $\boldsymbol{\phi}$;
8: `Update the state-action value network`: Use loss function $L_j^Q(\boldsymbol{\varphi}_j)$ to update $\boldsymbol{\varphi}_j$, $j\in\{1,2\}$;
9: `Update the temperature`: Use loss function $L^\kappa(\boldsymbol{\kappa})$ to update $\boldsymbol{\kappa}$;
10: **end for**
11: **Output**: $\pi_{\boldsymbol{\theta}}$, $\{\boldsymbol{Q}_j(s,a;\boldsymbol{\varphi}_j)\}_{j=1}^2$, $\boldsymbol{V}(s;\boldsymbol{\phi})$, and $\mathcal{D}$;

B.5 MOTD7 Algorithm

The TD7 algorithm Fujimoto et al. (2023) introduces four enhancements to the TD3 algorithm Fujimoto et al. (2018), which are the state-action learned embeddings (SALE), the Loss-Adjusted Prioritized (LAP) replay Fujimoto et al. (2020), the checkpoint, and the behavior cloning. The SALE maps the low-dimensional state and state-action into high-dimensional embedding vectors z_s and z_{sa}, which enhance the inputs to the value function and policy to improve sample efficiency and performance. The LAP is a prioritized experience replay technique that prioritizes samples in the replay buffer based on their importance (e.g. TD error). During training, higher-priority samples are selected first, which helps RL algorithms learn important experiences faster and improve learning efficiency. Checkpoint maintains training stability. Sample efficiency, learning efficiency, and training stability are critical for applying MORL algorithm to real-world tasks, thus SALE, LAP, and checkpoint are retained in our MOTD7 setting. The checkpoint is the same as the setting in Fujimoto et al. (2023), and the SALE and LAP settings are as follows.

SALE uses two independent encoders $f(s)$ and $g(f(s),a)$ to encode the state s and the state-action (s,a) into z_s and z_{sa}, respectively. Since $f(\cdot)$ and $g(\cdot)$ are decoupled, the encoders $f(\cdot)$ and $g(\cdot)$ can be jointly trained using the mean squared error (MSE) between the state-action embedding $z_{sa}^t = g(f(s_t),a_t)$ and the next state embedding $z_s^{t+1} = f(s_{t+1})$ with $(s_t,a_t,s_{t+1})\leftarrow\mathcal{B}$ and $\mathcal{B}\sim\mathcal{D}$. The encoder loss function is defined as:

$$L^{enc}(f,g) = \frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|}\|g(f(s_t),a_t)-|f(s_{t+1})|_\times\|_2^2,$$

where $|\cdot|_\times$ is the stop-gradient operation. Therefore, the original state-action value function $\boldsymbol{Q}(s,a;\boldsymbol{\varphi})$ becomes $\boldsymbol{Q}^{(z_s,z_{sa})}(s,a;\boldsymbol{\varphi}) \triangleq \boldsymbol{Q}(z_s,z_{sa},s,a;\boldsymbol{\varphi})$ with z_s and z_{sa} embedded. The original policy $\pi_{\boldsymbol{\theta}}(s)$ becomes $\pi_{\boldsymbol{\theta}}^{(z_s)}(s) \triangleq \pi_{\boldsymbol{\theta}}(z_s,s)$ with z_s and z_{sa} embedded. For notation simplicity, we omit the symbols z_s and z_{sa}, i.e., $\pi_{\boldsymbol{\theta}} = \pi_{\boldsymbol{\theta}}^{(z_s)}$, $\boldsymbol{Q}(s,a;\boldsymbol{\varphi}) = \boldsymbol{Q}^{(z_s,z_{sa})}(s,a;\boldsymbol{\varphi})$. In implementation, MOTD7 also uses two state-action value networks $\boldsymbol{Q}_1(s,a;\boldsymbol{\varphi}_1)$ and $\boldsymbol{Q}_2(s,a;\boldsymbol{\varphi}_2)$ to alleviate the impact of overestimating the Q-value. The Q-value is represented by the minimum value estimated by the two networks, i.e., $\boldsymbol{Q}_{est}(s,a) = \min\left(\{\boldsymbol{Q}_j(s,a;\boldsymbol{\varphi}_j)\}_{j=1}^2\right) \in \mathbb{R}^m$.

For the LAP, the probability of sampling a transition tuple $t := (s_t, a_t, \boldsymbol{R}(s_t,a_t), s_{t+1}, \mathbf{Done}_t)$ from the replay buffer $\mathcal{D}$ is defined as:

$$p(t) = \frac{\max\left(\texttt{mean}(\boldsymbol{\delta}(t))^\xi, 1\right)}{\sum_{t'=1}^{|\mathcal{D}|}\max\left(\texttt{mean}(\boldsymbol{\delta}(t'))^\xi, 1\right)},$$

where $\texttt{mean}(\boldsymbol{x})$ is an operation to obtain the mean of all elements in $\boldsymbol{x}$, $\boldsymbol{\delta}(t) = \frac{\boldsymbol{\delta}_1(t)+\boldsymbol{\delta}_2(t)}{2}$, $\boldsymbol{\delta}_j(t) = \texttt{abs}\left(\boldsymbol{Q}_j(s_t,a_t;\boldsymbol{\varphi}_j) - \hat{\boldsymbol{Q}}(s_t,a_t)\right) = [\delta_{j,1}(t),...,\delta_{j,m}(t)]^\top \in \mathbb{R}_+^m$ is TD error of $\boldsymbol{Q}_j(\cdot)$ with $j \in \{1,2\}$. $\texttt{abs}(\boldsymbol{x})$ is an operation to take the element-wise absolute value in $\boldsymbol{x}$, and $\hat{\boldsymbol{Q}}(s_t,a_t) = \boldsymbol{R}(s_t,a_t) + \gamma\texttt{clip}\left(\boldsymbol{Q}_{est}(s_{t+1},\pi_{\boldsymbol{\theta}}(s_{t+1})), \boldsymbol{Q}_{min}, \boldsymbol{Q}_{max}\right)$ is target Q-value. $\boldsymbol{Q}_{min} = \min_{(s,a)\in\mathcal{D}}\boldsymbol{Q}_{est}(s,a) \in \mathbb{R}^m$ and $\boldsymbol{Q}_{max} = \max_{(s,a)\in\mathcal{D}}\boldsymbol{Q}_{est}(s,a) \in \mathbb{R}^m$ are the minimum and maximum Q-values that have been recorded during training. In order to reduce the effect of overestimation or underestimation of $\boldsymbol{Q}_{est}$ by a small number of samples in replay buffer $\mathcal{D}$, the value of $\boldsymbol{Q}_{est}$ will be restricted between $\boldsymbol{Q}_{min}$ and $\boldsymbol{Q}_{max}$.

The policy gradient of objective $i \in \{1,\ldots,m\}$ is defined as: $\nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}) = -\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathcal{B}:\sim\mathcal{D},s_t\leftarrow\mathcal{B},a_t'=\pi_{\boldsymbol{\theta}}(s_t)}\left[\mathbf{Loss}_i(s_t,a_t')\right] = -\mathbb{E}\left[\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|}Q_i(s_t,a_t')\right]$, where $:\sim$ is the LAP sample operation, and $Q_i(s_t,a_t')$ is the i-th element of the $\boldsymbol{Q}_{est}(s_t,a_t')$. The gradient direction of the policy $\pi_{\boldsymbol{\theta}}$ under a given weight $\boldsymbol{\omega}$ is:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\boldsymbol{J}(\boldsymbol{\theta})^\top\boldsymbol{\omega} &= -\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathcal{B}:\sim\mathcal{D},s_t\leftarrow\mathcal{B}}\left[-\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|}\boldsymbol{\omega}^\top\boldsymbol{Q}_{est}\left(s_t,\pi_{\boldsymbol{\theta}}(s_t)\right)\right] \\ &= \mathbb{E}\left[\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|}\nabla_{\boldsymbol{\theta}}\boldsymbol{Q}^{\boldsymbol{\omega}}\left(s_t,a_t'\right)\right]\end{aligned}$$

where $\boldsymbol{Q}^{\boldsymbol{\omega}}\left(s_t,a_t'\right) = \boldsymbol{\omega}^\top\boldsymbol{Q}_{est}(s_t,a_t')$.

The following are the multi-objective loss functions for all networks:

1) State-action value network loss: Instead of the MSE loss, we use the LAP Huber loss Huber (1992), which is defined as

$$L_j^Q(\boldsymbol{\varphi}_j) = \mathbb{E}_{\mathcal{B}:\sim\mathcal{D}}\left[\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|}\texttt{mean}\left(\boldsymbol{\delta}_j^{LAP}(t)\right)\right], j\in\{1,2\},$$

where $\boldsymbol{\delta}_j^{LAP}(t) = \left[\delta_{j,1}^{LAP}(t),...,\delta_{j,m}^{LAP}(t)\right]^\top$, and $\delta_{j,i}^{LAP}(t)$ equals to $\frac{1}{2}\left(\delta_{j,i}(t)\right)^2$ if $\delta_{j,i}(t) < 1$ else $\delta_{j,i}(t)$, for all $i \in \{1,\ldots,m\}$.

2) Policy loss:

$$L^\pi(\boldsymbol{\theta}) = -\mathbb{E}_{\mathcal{B}:\sim\mathcal{D},s_t\leftarrow\mathcal{B}}\left[\frac{1}{|\mathcal{B}|}\sum_{t=1}^{|\mathcal{B}|}\boldsymbol{\omega}^\top\boldsymbol{Q}_{est}\left(s_t,\pi_{\boldsymbol{\theta}}(s_t)\right)\right].$$

In each MPFT-MOTD7 episode, the policy training process is shown in Algorithm 5, where the `Data Storage` operation can be omitted if the offline dataset $\mathcal{D}$ is given and the amount of data is sufficient.

Algorithm 5 MOTD7 Algorithm

1: **Input** $\pi_{\boldsymbol{\theta}}$, $\{\boldsymbol{Q}_j(s,a;\boldsymbol{\varphi}_j)\}_{j=1}^2$, $(f(s), g(z_s,a))$, and $\mathcal{D}$;
2: **Obtain** $\boldsymbol{\omega}$ based on the Pareto-tracking mechanism;
3: **Fixed** initialize steps;
4: `Data Storage`: Run policy $\pi_{\boldsymbol{\theta}}$ for steps timesteps in the environment to collect the experiences, and store these experiences into the LAP $\mathcal{D}$. (Agent interacts with the environment steps times);
5: **for** t $\leftarrow 1, \cdots,$ steps **do**
6: `Update the policy`: Use loss function L^{π} to updates $\boldsymbol{\theta}$;
7: `Update the state-action value network`: Use loss function $L_j^{Q}(\boldsymbol{\varphi}_j)$ to updates $\boldsymbol{\phi}_j$, $j \in \{1,2\}$;
8: `Update the encoders`: Use loss function $L^{enc}(f,g)$ to updates $(f(s), g(z_s,a))$;
9: **end for**
10: **Output**: $\pi_{\boldsymbol{\theta}}$, $\{\boldsymbol{Q}_j(s,a;\boldsymbol{\varphi}_j)\}_{j=1}^2$, $(f(s), g(z_s,a))$, and $\mathcal{D}$;

C Convergence Proof of the Pareto-Tracking Mechanism

To isolate the Pareto-tracking mechanism from specific algorithmic details, we adopt a continuous-time ordinary differential equation (ODE) formulation and interpret discrete updates as Euler discretization. The alternating reverse/ascent dynamics are modeled as a switched system, which is approximated by the averaged vector field with duty cycle λ Sanders et al. (2007). To guarantee smoothness and uniqueness of the inner minimizer, we introduce a μ-regularization based on the Moreau–Yosida envelope, ensuring differentiability and bounding the perturbation error by $\mathcal{O}(\mu)$. This enables us to define the Lyapunov function $V_\mu(\boldsymbol{\theta}) = \frac{1}{2}\|\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}_\mu^*\|^2$, whose derivative can be analyzed via Danskin-type results. By combining projection inequalities, the directional curvature assumption, and Young's inequality, we derive a linear decay bound of the form $\dot{V}_\mu(\boldsymbol{\theta}) \leq -cV_\mu(\boldsymbol{\theta}) + C\mu$. Grönwall's inequality and Lyapunov theory Khalil & Grizzle (2002) then establish exponential convergence to an $\mathcal{O}(\mu)$ neighborhood. A first-order Euler discretization with bounded step size preserves this stability, while robustness results from averaging theory Llibre et al. (2014) ensure that the switched system tracks the averaged dynamics up to an $\mathcal{O}(\tau)$ error. Collectively, these results yield exponential convergence to an $\mathcal{O}(\mu+\tau)$ neighborhood of the Pareto-stationary set.

Note that, to avoid symbolic ambiguity, all symbol definitions introduced in this section are local and valid only within its context.

C.1 Notations and Basic Assumptions

For the multi-objective optimization problem with m objectives, the objective function vector $\boldsymbol{J}(\boldsymbol{\theta}) = [J_1(\boldsymbol{\theta}), \ldots, J_m(\boldsymbol{\theta})]^\top$ is determined by the parameters $\boldsymbol{\theta} \in \mathbb{R}^d$, where d is the dimension of $\boldsymbol{\theta}$. We assume the parameter space $\boldsymbol{\Theta}$ is convex and compact, and all objective functions $\{J_i(\boldsymbol{\theta})\}$ are second-order continuous differentiable, $i \in \{1, \ldots, m\}$. Since the gradient descent algorithm (GD) is used to update $\boldsymbol{\theta}$, our goal is to minimize the weighted sum $\boldsymbol{J}^\top \boldsymbol{\omega}$, where $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_m]^\top \in \Delta_m = \{\boldsymbol{\omega} \in \mathbb{R}^m$, and $\Delta_m := \{\omega \in \mathbb{R}^m : \omega_i \geq 0, \|\omega\|_1 = 1\}$ denotes the simplex. Note that $J_i(\boldsymbol{\theta})$ denotes any second-order continuous differentiable function, not solely the cumulative discounted reward. For maximization problems, it suffices to prefix it with a negative sign, which does not alter our conclusions.

Denote the gradient of each objective by $\nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}) \in \mathbb{R}^d$, and the gradient matrix as

$$\boldsymbol{g}(\boldsymbol{\theta}) := \nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta}) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}} J_1(\boldsymbol{\theta})^\top \\ \vdots \\ \nabla_{\boldsymbol{\theta}} J_m(\boldsymbol{\theta})^\top \end{bmatrix} \in \mathbb{R}^{m \times d}, \quad g_i(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}).$$

By compactness of $\boldsymbol{\Theta}$ we obtain boundedness and smoothness constants:

$$G_{\max} := \sup_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sup_{i \in \{1,\ldots,m\}} \|\nabla J_i(\boldsymbol{\theta})\| < \infty,$$

and

$$H_{\max} := \sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sup_{i\in\{1,\dots,m\}} \left\|\nabla^2 J_i(\boldsymbol{\theta})\right\|_{\mathrm{op}} < \infty,$$

where $\|\cdot\|_{\mathrm{op}}$ is the operator norm of a matrix.

We aim to show that the Pareto-tracking mechanism (alternating Pareto-reverse and Pareto-ascent phases) converges exponentially to a neighborhood of the Pareto-stationary set, under appropriate duty-cycle and step-size conditions.

C.2 Regularization and Differentiability

Since Pareto-ascent and Pareto-reverse directions rely on the solution of a minimum-norm combination

$$\boldsymbol{\alpha}_0^*(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\alpha}\in\Delta_m} \|\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}\|^2,$$

the minimizer may not be unique everywhere, leading to non-differentiability. To obtain a smooth Lyapunov function, we introduce a quadratic regularization (Moreau-type):

$$\boldsymbol{\alpha}_\mu^*(\boldsymbol{\theta}) := \arg\min_{\boldsymbol{\alpha}\in\Delta_m} \|\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}\|^2 + \frac{\mu}{2}\|\boldsymbol{\alpha}\|^2, \tag{3}$$

where $\mu > 0$ is a constant. This ensures a unique solution $\alpha_\mu(\theta)$, and by Danskin's theorem the smoothed energy function

$$V_\mu(\boldsymbol{\theta}) = \tfrac{1}{2}\|\boldsymbol{v}_\mu(\boldsymbol{\theta})\|^2,$$

is differentiable, where $\boldsymbol{v}_\mu(\boldsymbol{\theta}) := \boldsymbol{g}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}_\mu^*(\boldsymbol{\theta}) \in \mathbb{R}^d$. The gradient of $V_\mu(\boldsymbol{\theta})$ is defined as

$$\nabla_{\boldsymbol{\theta}} V_\mu(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \left(\tfrac{1}{2}\|\boldsymbol{g}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}_\mu^*(\boldsymbol{\theta})\|^2\right) = \boldsymbol{H}_\mu(\boldsymbol{\theta})\boldsymbol{v}_\mu(\boldsymbol{\theta}),$$

where $\boldsymbol{H}_\mu(\boldsymbol{\theta}) := \sum_{i=1}^m \boldsymbol{\alpha}_{\mu,i}^*(\boldsymbol{\theta})\nabla^2 J_i(\boldsymbol{\theta})$ is the weighted Hessian matrix of $J_i(\boldsymbol{\theta})$.

Remark. When $\mu \to 0$, the minimizer $\alpha_\mu(\theta)$ converges to the minimizer of the original problem. If the original problem is unique over all $\boldsymbol{\theta}$, no regularization is required.

In Appendix D.C.4, we quantify the projection error when $\mu > 0$, and show that when using μ-regularization, the regularization introduces only a small perturbation, and the projection property is approximately preserved.

C.3 System Model (Averaged System)

For each Pareto-vertex policy $\pi_{\boldsymbol{\theta}^{i,*}}$, the Pareto-tracking mechanism alternates between two phases:

Phase 1 (Pareto-reverse): move along the Pareto-reverse direction of the objective i, the objective weight $\boldsymbol{\omega}$ equals $\boldsymbol{\alpha}_\mu^{*,(i)}(\boldsymbol{\theta}) := \arg\min_{\boldsymbol{\alpha}\in\Delta_m, \boldsymbol{\alpha}_i=0} \|\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}\|^2 + \frac{\mu}{2}\|\boldsymbol{\alpha}\|$. The Pareto-reverse direction of objective i is defined as

$$d_{\mathrm{rev}}^{(i)}(\boldsymbol{\theta}) := -\boldsymbol{u}_\mu^{(i)}(\boldsymbol{\theta}) := -\boldsymbol{g}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}_\mu^{*,(i)}(\boldsymbol{\theta}).$$

Phase 2 (Pareto-ascent): move along the Pareto-ascent direction, which is defined as

$$d_{\mathrm{asc}}(\boldsymbol{\theta}) := -\boldsymbol{v}_\mu(\boldsymbol{\theta}) := -\boldsymbol{g}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}_\mu^*(\boldsymbol{\theta}).$$

In practice these two phases alternate periodically. For analysis we first study the averaged system with duty-cycle $\lambda \in [0, 1]$:

$$\dot{\boldsymbol{\theta}} = F_i(\boldsymbol{\theta}) := \lambda d_{\mathrm{rev}}^{(i)}(\boldsymbol{\theta}) + (1-\lambda) d_{\mathrm{asc}}(\boldsymbol{\theta}).$$

For any $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and $\mu > 0$, The derivative of $V_\mu(\boldsymbol{\theta})$ along the vector field $F_i(\boldsymbol{\theta})$ is

$$\dot{\boldsymbol{V}}_\mu(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \boldsymbol{V}_\mu(\boldsymbol{\theta})^\top \cdot F_i(\boldsymbol{\theta}) = -\boldsymbol{v}_\mu(\boldsymbol{\theta})^\top \boldsymbol{H}_\mu(\boldsymbol{\theta}) \left(\lambda \boldsymbol{u}_\mu^{(i)}(\boldsymbol{\theta}) + (1-\lambda)\boldsymbol{v}_\mu(\boldsymbol{\theta})\right).$$

Let $\boldsymbol{v} := \boldsymbol{v}_\mu(\boldsymbol{\theta})$, $\boldsymbol{u} := \boldsymbol{u}_\mu^{(i)}(\boldsymbol{\theta})$, and $\boldsymbol{H} := \boldsymbol{H}_\mu(\boldsymbol{\theta})$. Then we have

$$\dot{V}_\mu(\boldsymbol{\theta}) = -\lambda \boldsymbol{v}^\top \boldsymbol{H}\boldsymbol{u} - (1-\lambda)\boldsymbol{v}^\top \boldsymbol{H}\boldsymbol{v} \tag{4}$$

C.4 μ-PERTURBED PROJECTION APPROXIMATION

Define the convex set for a fixed $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ as

$$\mathcal{U}(\boldsymbol{\theta}) := \{\boldsymbol{g}(\boldsymbol{\theta})^\top \boldsymbol{\omega} : \boldsymbol{\omega} \in \Delta_m\} \subset \mathbb{R}^d.$$

For the original problem where $\mu = 0$, let $\boldsymbol{v}_0(\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}_0^*$ denote the minimally normed point from the set $\mathcal{U}(\boldsymbol{\theta})$ to the origin. Then the projection inequality (projection theorem) holds:

$$\langle \boldsymbol{v}_0(\boldsymbol{\theta}), \boldsymbol{u} \rangle \geq \|\boldsymbol{v}_0(\boldsymbol{\theta})\|^2, \forall \boldsymbol{u} \in \mathcal{U}(\boldsymbol{\theta}), \quad \text{(Proj)}$$

Where $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^\top \boldsymbol{b}$ denotes the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. When employing μ-regularisation, the set undergoes a slight perturbation while its projection properties are approximately preserved. The following quantifies the projection error to demonstrate that for $\mu > 0$ and sufficiently small values, μ-regularisation does not significantly impact the original problem, i.e., $\left\|\boldsymbol{\alpha}_\mu^*(\boldsymbol{\theta}) - \boldsymbol{\alpha}_0^*(\boldsymbol{\theta})\right\| = \mathcal{O}(\mu)$.

Lemma 1 (μ-Perturbed Projection Approximation)**.** *There exists a constant $C_1 = \mathcal{O}(G_{\max}\sqrt{m})$ such that for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and $\mu > 0$, we have*

$$\|\boldsymbol{v}_\mu(\boldsymbol{\theta}) - \boldsymbol{v}_0(\boldsymbol{\theta})\| \leq C_1 \mu. \tag{5}$$

Proof. The proof process follows the following two facts:

1) From the compactness of $\boldsymbol{\Theta}$ and the boundedness of $\|\boldsymbol{g}(\boldsymbol{\theta})\|$, it follows that the set $\mathcal{U}(\boldsymbol{\theta})$ is uniformly up-bounded on $\boldsymbol{\theta}$.

2) The regularization term $\frac{1}{2}\|\boldsymbol{\alpha}\|$ unambiguously defines $\boldsymbol{\alpha}_\mu^*(\boldsymbol{\theta})$ from the original $\boldsymbol{\alpha}_0^*(\boldsymbol{\theta})$. Since the equation in (3) has a lower bound on the minimum eigenvalue of the Hessian matrix with respect to $\boldsymbol{\alpha}$, i.e., $\boldsymbol{g}(\boldsymbol{\theta})\boldsymbol{g}(\boldsymbol{\theta})^\top + \mu \boldsymbol{I}$, which is greater than or equal to μ, the sensitivity of the inner-layer solution to $\boldsymbol{\theta}$ is controlled.

According to Moreau smoothing theory, we have $\left\|\boldsymbol{\alpha}_\mu^*(\boldsymbol{\theta}) - \boldsymbol{\alpha}_0^*(\boldsymbol{\theta})\right\| = \mathcal{O}(\mu)$. Recall $\boldsymbol{v}_\mu(\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}_\mu^*(\boldsymbol{\theta})$ and $\boldsymbol{v}_0(\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}_0^*(\boldsymbol{\theta})$, we have

$$\|\boldsymbol{v}_\mu(\boldsymbol{\theta}) - \boldsymbol{v}_0(\boldsymbol{\theta})\| = \|\boldsymbol{g}(\boldsymbol{\theta})^\top (\boldsymbol{\alpha}_\mu^*(\boldsymbol{\theta}) - \boldsymbol{\alpha}_0^*(\boldsymbol{\theta}))\| \leq \|\boldsymbol{g}(\boldsymbol{\theta})^\top\|_{\text{op}} \cdot \|\boldsymbol{\alpha}_\mu^*(\boldsymbol{\theta}) - \boldsymbol{\alpha}_0^*(\boldsymbol{\theta})\|,$$

where $\|\boldsymbol{g}(\boldsymbol{\theta})^\top\|_{\text{op}} = \sup_{\|\mathbf{x}\|=1} \|\boldsymbol{g}(\boldsymbol{\theta})^\top \mathbf{x}\|$. For any $\mathbf{x} \in \mathbb{R}^m$, we have

$$\|\boldsymbol{g}(\boldsymbol{\theta})^\top \mathbf{x}\| = \left\| \sum_{i=1}^m x_i \nabla J_i(\boldsymbol{\theta}) \right\| \leq \sum_{i=1}^m |x_i| \cdot \|\nabla J_i(\boldsymbol{\theta})\| \leq G_{\max} \sum_{i=1}^m |x_i| = G_{\max} \|\mathbf{x}\|_1 \leq G_{\max}\sqrt{m}\|\mathbf{x}\|.$$

This means that

$$\|\boldsymbol{g}(\boldsymbol{\theta})^\top\|_{\text{op}} = \sup_{\|\mathbf{x}\|=1} \|\boldsymbol{g}(\boldsymbol{\theta})^\top \mathbf{x} \leq G_{\max}\sqrt{m}$$

and

$$\|\boldsymbol{v}_\mu(\boldsymbol{\theta}) - \boldsymbol{v}_0(\boldsymbol{\theta})\| \leq G_{\max}\sqrt{m} \cdot \|\boldsymbol{\alpha}_\mu^*(\boldsymbol{\theta}) - \boldsymbol{\alpha}_0^*(\boldsymbol{\theta})\| = G_{\max}\sqrt{m} \cdot \mathcal{O}(\mu) = \mathcal{O}(G_{\max}\sqrt{m})\mu = C_1 \mu,$$

and we complete our proof. □

Remark. Lemma 1 serves to relate quantities involving μ to the original ($\mu = 0$) projection properties, and provides an upper bound on the error introduced by perturbation; if the original problem can be guaranteed to be globally unique and continuous, then we can directly set $\mu = 0$.

C.5 EXPONENTIAL CONVERGENCE

To obtain a uniform negative convergence rate, we make a key assumption: there exists a constant $h_{\min} \in (0, 1)$ such that for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}_s$ and $\mu > 0$, the following holds:

$$\boldsymbol{v}^\top \boldsymbol{H} \boldsymbol{v} \geq h_{\min} \|\boldsymbol{v}\|^2, \quad \text{(A1)}$$

where $\boldsymbol{\Theta}_s := \left\{\boldsymbol{\theta} | \boldsymbol{\theta} \in \boldsymbol{\Theta}, \min_{\boldsymbol{\alpha} \in \Delta_m} \|\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})^\top \boldsymbol{\alpha}\|^2 \leq \epsilon\right\}$ is defined as the set consisting of policies located near the Pareto-stationary point, and $\epsilon > 0$ is a small constant. An intuitive interpretation of this assumption: if the weighted Hessian $\boldsymbol{H}$ is a positive definite matrix then the assumption (A1) holds, i.e., the objective function $\boldsymbol{J}(\boldsymbol{\theta})$ is locally convex (i.e. directional positive curvature), which is common in the vicinity of Pareto-stationary set.

Lemma 2. *There exist constants $C_2 = 2C_1$ and $C_3 = C_1 G_{max}$ such that for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}_s$, we have*

$$\langle \boldsymbol{v}, \boldsymbol{u} \rangle \geq \|\boldsymbol{v}\|^2 - C_2\mu\|\boldsymbol{v}\| - C_3\mu.$$

Proof.

$$\langle \boldsymbol{v}, \boldsymbol{u} \rangle = \langle \boldsymbol{v}_0 + (\boldsymbol{v} - \boldsymbol{v}_0), \boldsymbol{u} \rangle = \langle \boldsymbol{v}_0, \boldsymbol{u} \rangle + \langle \boldsymbol{v} - \boldsymbol{v}_0, \boldsymbol{u} \rangle$$

According to the Cauchy-Schwarz inequality, we have

$$\langle \boldsymbol{v} - \boldsymbol{v}_0, \boldsymbol{u} \rangle \geq -|\langle \boldsymbol{v} - \boldsymbol{v}_0, \boldsymbol{u} \rangle| \geq -\|\boldsymbol{v} - \boldsymbol{v}_0\| \cdot \|\boldsymbol{u}\|,$$

and according to the projection inequality (Proj) $\langle \boldsymbol{v}_0, \boldsymbol{u} \rangle \geq \|\boldsymbol{v}_0\|^2$, we have

$$\langle \boldsymbol{v}, \boldsymbol{u} \rangle \geq \|\boldsymbol{v}_0\|^2 - \|\boldsymbol{v} - \boldsymbol{v}_0\| \cdot \|\boldsymbol{u}\|.$$

According to the triangle inequality $\|\boldsymbol{v}_0\| \geq \|\boldsymbol{v}\| - \|\boldsymbol{v} - \boldsymbol{v}_0\|$, we have

$$\|\boldsymbol{v}_0\|^2 \geq (\|\boldsymbol{v}\| - \|\boldsymbol{v} - \boldsymbol{v}_0\|)^2 = \|\boldsymbol{v}\|^2 - 2\|\boldsymbol{v}\|\|\boldsymbol{v} - \boldsymbol{v}_0\| + \|\boldsymbol{v} - \boldsymbol{v}_0\|^2 \geq \|\boldsymbol{v}\|^2 - 2\|\boldsymbol{v}\|\|\boldsymbol{v} - \boldsymbol{v}_0\|$$

$$\Rightarrow \langle \boldsymbol{v}, \boldsymbol{u} \rangle \geq \|\boldsymbol{v}\|^2 - 2\|\boldsymbol{v}\|\|\boldsymbol{v} - \boldsymbol{v}_0\| - \|\boldsymbol{v} - \boldsymbol{v}_0\| \cdot \|\boldsymbol{u}\|$$

Finally, substituting $\|\boldsymbol{v}_\mu(\boldsymbol{\theta}) - \boldsymbol{v}_0(\boldsymbol{\theta})\| \leq C_1\mu$ (Lemma 1) and $\|\boldsymbol{u}\| \leq G_{\max}$ into above yields

$$\langle \boldsymbol{v}, \boldsymbol{u} \rangle \geq \|\boldsymbol{v}\|^2 - 2C_1\mu\|\boldsymbol{v}\| - C_1 G_{\max}\mu,$$

and we complete our proof. □

Lemma 3 (The Upper Bound of $\dot{V}_\mu(\boldsymbol{\theta})$)**.** *There exist constants $C = \frac{(\lambda C_4)^2}{2c\mu} + \lambda C_3$, $C_4 = C_2\mu + (H_{\max} + 1)G_{\max}$, $c = \lambda + (1-\lambda)h_{\min}$, $\lambda \in [0, \frac{h_{\min}}{1+h_{\min}})$, and $\mu > 0$ such that for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}_s$, we have*

$$\dot{V}_\mu(\boldsymbol{\theta}) \leq -cV_\mu(\boldsymbol{\theta}) + C\mu. \tag{6}$$

Proof.

$$\boldsymbol{v}^\top \boldsymbol{H}\boldsymbol{u} = \langle \boldsymbol{v}, \boldsymbol{u} \rangle + \boldsymbol{v}^\top(\boldsymbol{H} - \boldsymbol{I})\boldsymbol{u}$$

According to the Cauchy-Schwarz inequality, we have

$$\boldsymbol{v}^\top \boldsymbol{H}\boldsymbol{u} \geq \langle \boldsymbol{v}, \boldsymbol{u} \rangle - \|\boldsymbol{H} - \boldsymbol{I}\|_{\text{op}}\|\boldsymbol{v}\|\|\boldsymbol{u}\|.$$

Since $\|\boldsymbol{H}\|_{\text{op}} \leq H_{\max}$ and $\|\boldsymbol{I}\|_{\text{op}} = 1$, it follows that $\|\boldsymbol{H} - \boldsymbol{I}\|_{\text{op}} \leq H_{\max} + 1$. Further, with $\|\boldsymbol{u}\| \leq G_{\max}$, we obtain $\|\boldsymbol{H} - \boldsymbol{I}\|_{\text{op}}\|\boldsymbol{v}\|\|\boldsymbol{u}\| \leq (H_{\max} + 1)G_{\max}\|\boldsymbol{v}\|$. Combining $\langle \boldsymbol{v}, \boldsymbol{u} \rangle \geq \|\boldsymbol{v}\|^2 - C_2\mu\|\boldsymbol{v}\| - C_3\mu$ (Lemma 2) yields

$$\boldsymbol{v}^\top \boldsymbol{H}\boldsymbol{u} \geq \|\boldsymbol{v}\|^2 - C_2\mu\|\boldsymbol{v}\| - C_3\mu - (H_{\max} + 1)G_{\max}\|\boldsymbol{v}\| = \|\boldsymbol{v}\|^2 - C_4\|\boldsymbol{v}\| - C_3\mu$$

$$\Rightarrow -\dot{V}_\mu(\boldsymbol{\theta}) = \lambda \boldsymbol{v}^\top \boldsymbol{H}\boldsymbol{u} + (1-\lambda)\boldsymbol{v}^\top \boldsymbol{H}\boldsymbol{v} \geq \lambda(\|\boldsymbol{v}\|^2 - C_4\|\boldsymbol{v}\| - C_3\mu) + (1-\lambda)h_{\min}\|\boldsymbol{v}\|^2$$

$$\Rightarrow \dot{V}_\mu(\boldsymbol{\theta}) \leq -(\lambda + (1-\lambda)h_{\min})\|\boldsymbol{v}\|^2 + \lambda C_4\|\boldsymbol{v}\| + \lambda C_3\mu. \tag{7}$$

Observing the above inequality, we find that for $\dot{\boldsymbol{V}}_\mu(\boldsymbol{\theta})$ to be negative, we must have $\lambda + (1-\lambda)h_{\min} > 0$, thus $\lambda > -\frac{h_{\min}}{1-h_{\min}}$. Moreover, since $-\frac{h_{\min}}{1-h_{\min}} \leq 0 \leq \lambda$ always holds ($h_{\min} \in [0, 1)$), we let $c = \lambda + (1-\lambda)h_{\min}$ and $0 \leq \lambda \leq 1$ yielding $h_{min} \leq c \leq 1$. At this point, we obtain

$$\dot{V}_\mu(\boldsymbol{\theta}) \leq -c\|\boldsymbol{v}\|^2 + \lambda C_4\|\boldsymbol{v}\| + \lambda C_3\mu.$$

By defining $V_\mu(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{v}\|^2$ and $\lambda C_4\|\boldsymbol{v}\| \leq \frac{c}{2}\|\boldsymbol{v}\|^2 + \frac{(\lambda C_4)^2}{2c}$ (Young's inequality), we obtain $\dot{V}_\mu(\boldsymbol{\theta}) \leq -cV_\mu(\boldsymbol{\theta}) + C\mu$. We complete our proof. □

Remark. Note that by relaxing inequality (7) to $\dot{V}_\mu(\boldsymbol{\theta}) \leq -(-\lambda + (1-\lambda)h_{\min})\|\boldsymbol{v}\|^2 + \lambda C_4\|\boldsymbol{v}\| + \lambda C_3\mu$, and let $-\lambda + (1-\lambda)h_{\min} > 0$, we can deduce that $\lambda < \frac{h_{\min}}{1+h_{\min}} < 0.5$. This suggests that the duty-cycle λ should not be excessively large, meaning we require a greater number of Pareto-ascent phases to enable the policy to converge to the Pareto-stationary.

By Grönwall's inequality, we solve the inequality (6) as:

$$\dot{V}_\mu(\boldsymbol{\theta}(t)) + cV_\mu(\boldsymbol{\theta}(t)) \leq C\mu.$$

$$\Rightarrow e^{ct}\dot{V}_\mu(\boldsymbol{\theta}(t)) + ce^{ct}V_\mu(\boldsymbol{\theta}(t)) \leq C\mu e^{ct} \Rightarrow \frac{d}{dt}[e^{ct}V_\mu(\boldsymbol{\theta}(t))] \leq C\mu e^{ct}$$

$$\Rightarrow \int_0^t \frac{d}{ds}[e^{ct}V_\mu(\boldsymbol{\theta}(s))]ds \leq \int_0^t C\mu e^{cs}ds$$

$$\Rightarrow e^{ct}V_\mu(\boldsymbol{\theta}(t)) - V_\mu(\boldsymbol{\theta}(0)) \leq \frac{C\mu}{c}(e^{ct}-1) \leq \frac{C\mu}{c}e^{ct}$$

$$\Rightarrow V_\mu(\boldsymbol{\theta}(t)) \leq e^{-ct}V_\mu(\boldsymbol{\theta}(0)) + \frac{C}{c}\mu. \tag{8}$$

Thus the value of $V_\mu(\boldsymbol{\theta})$ converges exponentially to an $O(\mu)$ neighborhood, i.e., $\boldsymbol{\theta}(0) \in \boldsymbol{\Theta}_s$ converges exponentially to a certain Pareto-stationary neighborhood by Pareto-tracking mechanism, denoted as $\lim_{t\to\infty} \sup \operatorname{dist}(\boldsymbol{\theta}(t), \boldsymbol{\Theta}_s) = \mathcal{O}(\mu)$.

C.6 Euler Discretization and Step-Size Condition

For an arbitrary track $i \in \{1, ..., m\}$ we approximate the continuous dynamics by an explicit Eulerian discretization. The continuous dynamics is $\dot{\boldsymbol{\theta}} = F_i(\boldsymbol{\theta})$, the discrete iteration is

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta\dot{\boldsymbol{\theta}}(t),$$

where $\eta > 0$ is step-size (i.e. learning rate).

Lemma 4. *If the step size satisfies* $0 \leq \eta \leq \eta_{\max}$, $\eta_{\max} = \min\{\frac{c}{2L_V M_F^2}, \frac{1}{c}\}$, *we have*

$$\lim_{t\to\infty} V_\mu(\boldsymbol{\theta}(t)) = \mathcal{O}(\mu). \tag{9}$$

Proof. Using the gradient boundedness and Lipschitz continuity of $V_\mu(\boldsymbol{\theta})$, a first-order Taylor expansion is obtained:

$$V_\mu(\boldsymbol{\theta}(t+1)) = V_\mu(\boldsymbol{\theta}(t)) + \eta\nabla_{\boldsymbol{\theta}}V_\mu(\boldsymbol{\theta}(t))^\top F_i(\boldsymbol{\theta}(t)) + \tfrac{\eta^2}{2}L_V\|F_i(\boldsymbol{\theta}(t))\|^2,$$

where $L_V = \mathcal{O}(H_{\max}G_{\max})$ denotes the Lipschitz constant of $\nabla_{\boldsymbol{\theta}}V_\mu(\boldsymbol{\theta})$. According to lemma 3, we have

$$V_\mu(\boldsymbol{\theta}(t+1)) \leq V_\mu(\boldsymbol{\theta}(t)) + \eta(-cV_\mu(\boldsymbol{\theta}) + C\mu) + \tfrac{\eta^2}{2}L_V M_F^2$$

$$\Rightarrow V_\mu(\boldsymbol{\theta}(t+1)) \leq (1-\eta c)V_\mu(\boldsymbol{\theta}(t)) + \eta C\mu + \tfrac{\eta^2}{2}L_V M_F^2,$$

where $M_F := \sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} V\|F_i(\boldsymbol{\theta})\| \leq G_{\max}$ is a constant. In order to control the second-order term $\frac{\eta^2}{2}L_V M_F^2$, the step size should satisfy $0 < \eta \leq \min\{\frac{c}{2L_V M_F^2}, \frac{1}{c}\}$, then it can be guaranteed that

$$\Rightarrow V_\mu(\boldsymbol{\theta}(t+1)) \leq (1-\eta c)V_\mu(\boldsymbol{\theta}(t)) + \eta C\mu,$$

$$\Rightarrow V_\mu(\boldsymbol{\theta}(t)) \leq (1-\eta c)^t\left(V_\mu(\boldsymbol{\theta}(0)) - \frac{C}{c}\mu\right) + \frac{C}{c}\mu$$

$$\Rightarrow \lim_{t\to\infty} \sup V_\mu(\boldsymbol{\theta}(t)) = \frac{C}{c}\mu = \mathcal{O}(\mu),$$

and we complete our proof. □

Remark. Lemma 4 indicates that when using the GD algorithm to update the parameters $\boldsymbol{\theta}$ discretely, if the step-size satisfies $0 < \eta \leq \eta_{\max}$, then the value of $V_\mu(\boldsymbol{\theta}(0))$ will ultimately converge to the $\mathcal{O}(\mu)$-neighborhood.

C.7 From Averaged System to Switching System

The real Pareto-tracking mechanism alternates periodically with period $\tau = \tau_{rev} + \tau_{asc}$, $\tau > 0$ (i.e. $\lambda = \frac{\tau_{rev}}{\tau}$). By standard averaging theory we have:

Proposition 1 (Switching vs Averaged System)**.** *Suppose that the vector field for each phase satisfies Lipschitz continuity. Then for any period $\tau \geq 1$:*

- *On the timescale $t = \mathcal{O}(1)$, the distance between the switching system trajectory $\bar{\boldsymbol{\theta}}(t)$ and the average system trajectory $\boldsymbol{\theta}(t)$ has an upper bound:*

$$\|\bar{\boldsymbol{\theta}}(t) - \boldsymbol{\theta}(t)\| \leq \mathcal{O}(\tau), \forall t \in [0, \tau].$$

- *When τ is small, the exponential convergence property of the average system is approximately preserved for the switching system: there exist $C' > 0$ ($C' = \mathcal{O}(C)$) such that the Lyapunov function of the switching system satisfies the same form as (8), but with an additional periodic switching error term $\mathcal{O}(\tau)$:*

$$\dot{V}_\mu(\boldsymbol{\theta}(t)) \leq -cV_\mu(\boldsymbol{\theta}(t))C'(\mu + \tau).$$

$$\Rightarrow V_\mu(\boldsymbol{\theta}(t)) \leq e^{-ct}V_\mu(\boldsymbol{\theta}(0)) + \frac{C'}{c}(\mu + \tau). \tag{10}$$

Proof. According to the standard averaging theory we know that the period average of the vector field converges approximately to the average vector field, and the error is $\mathcal{O}(\tau)$ when the period τ is small enough Llibre et al. (2014); the exponential stability is robust to small perturbations, therefore, combining the average system error $\mathcal{O}(\mu)$ with the periodic switching system error $\mathcal{O}(\tau)$ yields (10), and the proof is complete. □

Remark. Proposition 1 demonstrates that when the period τ is sufficiently small, the exponential convergence property of the switching system is approximately preserved, with a convergence rate of c. Eulerian discretisation also yields $\lim_{t\to\infty} \sup V_\mu(\boldsymbol{\theta}(t)) = \mathcal{O}(\mu + \tau)$.

C.8 Main Theorem

According to the Lemmas 1, 2, 3, 4 and Proposition 1, Theorem 1 holds as follows:

Theorem 1 (Convergence of Pareto-Tracking Mechanism)**.** *Suppose:*

- *Parameter space $\boldsymbol{\Theta}$ is convex and compact, and for all objective functions $\{J_i(\boldsymbol{\theta})\}$ are second-order continuous differentiable, $i \in \{1, \ldots, m\}$;*
- *Regularization parameter $\mu > 0$ is small enough (Lemma 1 holds);*
- *There exist a constant $h_{\min} \in (0, 1)$ such that directional positive curvature condition holds near Pareto-stationary set (Assumption (A1) holds);*
- *Duty-cycle λ satisfies $0 \leq \lambda < \frac{h_{\min}}{1+h_{\min}} < 0.5$;*
- *Switching period τ is small;*
- *Step-size η satisfies $0 < \eta \leq \eta_{\max} < 1$.*

Then there exist constants $h_{min} \leq c \leq 1$ and $C > 0$ such that for the average system the Lyapunov function satisfies

$$V_\mu(\boldsymbol{\theta}(t)) \leq (1 - \eta c)^t \left(V_\mu(\boldsymbol{\theta}(0)) - \frac{C}{c}\mu \right) + \frac{C}{c}\mu,$$

and for the switching system the Lyapunov function satisfies

$$V_\mu(\boldsymbol{\theta}(t)) \leq (1 - \eta c)^t \left(V_\mu(\boldsymbol{\theta}(0)) - \frac{C'}{c}(\mu + \tau) \right) + \frac{C'}{c}(\mu + \tau), \quad C' = \mathcal{O}(C),$$

i.e., for any $\boldsymbol{\theta} \in \boldsymbol{\Theta}_s$ converges exponentially to certain neighborhood of the Pareto-stationary set in any arbitrary objective dimension by Pareto-tracking mechanism.

Remark. From Theorem 1, the policies tracked by the MPFT framework remain within an $\mathcal{O}(\mu+\tau)$-neighborhood of the Pareto-stationary set in any arbitrary objective dimension, as corroborated by Figures 6 and 8 in Appendix F. The period τ depends on the number of training episodes per phase, denoted by u and v. The duty cycle $0 \leq \lambda < \frac{h_{\min}}{(1+h_{\min})} < 0.5$ implies that $u < v$, and λ is proportional to the convergence rate c, such that larger λ leads to faster convergence. Hence, parameter settings should be adapted to the specific environment. Since the initial policy of the Pareto-tracking mechanism is the Pareto-vertex policy $\boldsymbol{\theta}^{i,*}$, and $\boldsymbol{\theta}^{i,*} \in \boldsymbol{\Theta}_s$ by definition, the mechanism ensures that policies remain close to the Pareto front. Consequently, the parameters stay within the vicinity of the Pareto-stationary set, guaranteeing that assumption (A1) always holds.

D Evaluation Metrics, and Time and Space Complexity

D.1 HV, SP, and EU

The HV is given by $\mathbf{HV} = \Omega_m\left(\cup_{j=1}^{|\mathcal{N}|}\mathbb{V}_j\right)$, where Ω_m represents the volume measurement in m-dimensional Euclidean space (in two-dimensional space, Ω_2 represents the area), and $\mathbb{V}_j$ is the space enclosed by the j-th solution in the set $\mathcal{N}$ and the reference point. A larger $\mathbf{HV}$ indicates a better approximation of Pareto front. The SP is given by $\mathbf{SP} = \frac{1}{|\mathcal{N}|-1}\sum_{i=1}^{m-1}\sum_{j=1}^{|\mathcal{N}|}\left(\tilde{\mathcal{N}}_j(i+1) - \tilde{\mathcal{N}}_j(i)\right)^2$, where $\mathcal{N}_j \in \mathbb{R}^m$ represents the mapping vector of the j-th policy in $\mathcal{F}$, $\mathcal{N}_j(i)$ is the value of objective i in $\mathcal{N}_j$, and $\tilde{\mathcal{N}}_j(i)$ is the value after sorting the i-th objective in ascending order. A smaller $\mathbf{SP}$ indicates that the solutions in $\mathcal{N}$ are more densely distributed. The EU is given by $\mathbf{EU} = \mathbb{E}_{\boldsymbol{\omega}\sim\Omega}\left[\max\{\boldsymbol{J}(\boldsymbol{\theta}_j)^\top\boldsymbol{\omega}\}_{j=1}^{|\mathcal{N}|}\right]$, where Ω is a specified preferences set. A higher expected utility is better.

D.2 Calculation of env_steps for MPFT framework

$\mathbf{env\_steps} = \text{steps} \times (\sum_{i=1}^{m}(\Xi_i + \Psi_i) + \sum_{k=1}^{K}(\Xi_k + \Psi_k))$, where Ξ_i is the number of episodes in finding the approximate Pareto-vertex policies $\pi_{\boldsymbol{\theta}^{i,*}}$, Ξ_k is the number of episodes in finding the approximate Pareto-interior policies $\pi_{\boldsymbol{\theta}^{k,\star}}$, Ψ_i is the number of episodes required to construct the Pareto-edge policy set $\mathcal{F}_{edge}^{i,*}$, and Ψ_k is the number of episodes required to construct the Pareto-interior policy tracking set $\mathcal{F}_{interior}^{k,\star}$ for all $i \in \{1,\ldots,m\}$ and all $k \in \{1,\ldots,K\}$. The steps is the number of timesteps in an episode of Algorithms 1.

D.3 Calculation of env_steps for Evolutionary-based MORL Framework

$\mathbf{env\_steps} = n \times D \times (m_w + (G \times m_t))$, where n is the number of policies selected for each iteration, D is the number of interactions with the environment during one policy iteration (i.e., timesteps per actorbatch), m_w is the number of warm-up iterations, m_t is the number of iterations required for the population to complete one generation of evolution, and G is the total number of generations of evolution of the population.

D.4 Time and Space Complexity

Let Υ denotes an upper bound on the cost of training a single policy, and set the number of times a policy is trained equal to the number of times corresponding agent interacts with the environment. If parallel training is not considered, the time complexity of the Stage 1 is $\mathcal{O}(\text{steps} \times \sum_{i=1}^{m}\Xi_i)$. The time complexity of the Stage 2 is $\mathcal{O}(\text{steps} \times \sum_{i=1}^{m}\Psi_i)$. The time complexity of the Stage 3 is $\mathcal{O}\left(\text{steps} \times \sum_{k=1}^{K}(\Xi_k + \Psi_k)\right)$. The Stage 4 is negligible. Therefore, the time complexity of the MPFT is

$$T_{\text{MPFT}} = \mathcal{O}\left(\Upsilon \times \text{steps} \times \left(\sum_{i=1}^{m}(\Xi_i + \Psi_i) + \sum_{k=1}^{K}(\Xi_k + \Psi_k)\right)\right), \tag{11}$$

and the time complexity of the evolutionary framework based MP-MORL is

$$T_{\text{EVOL}} = \mathcal{O}\left(\Upsilon \times n \times D \times (m_w + (G \times m_t)) + P_{sel}\right) = \mathcal{O}\left(\Upsilon \times D \times (m_w + (G \times m_t)) + \mathcal{O}(P_{sel})\right), \tag{12}$$

where P_{sel} denotes the time spent on policy selection, which varies from algorithm to algorithm, e.g., PGMORL is $P_{sel} = Gn^{m-2}$ (needs to solve a knapsack problem), PA2D-MORL is $P_{sel} = G$ (randomly select n policies from the candidate policies), and C-MORL is $P_{sel} = m|\mathcal{F}|\log|\mathcal{F}|$ (select n policies from $\mathcal{F}$ based on crowding distance). Note that, unlike evolutionary frameworks, MPFT does not require policy selection, and thus $P_{\text{sel}} = 0$. Consequently, when the same policy gradient algorithm (e.g., PPO) is used, we typically have $T_{\text{MPFT}} < T_{\text{EVOL}}$.

For the space complexity, the MPFT is $S_{\text{MPFT}} = \mathcal{O}\left(|\mathcal{F}| \times \Gamma\right)$, and the evolutionary framework based MP-MORL is $S_{\text{EVOL}} = \mathcal{O}\left((|\mathcal{F}| + |\mathcal{P}_{op}|) \times \Gamma\right)$, where $\mathcal{F}$ is the Pareto-approximation policy set, $\mathcal{P}_{op}$ is the policy population, and Γ is the memory space occupied for saving a policy $\pi_{\boldsymbol{\theta}}$. Note that the values of Υ and Γ will be different using different algorithms, but whatever algorithm is used S_{MPFT} is significantly lower than S_{EVOL}. This is due to $|\mathcal{P}_{op}| \gg |\mathcal{F}|$.

E Experiment Detail

E.1 Multi-Objective Robot Control Environment

Our robotics control environment is built using MuJoCo 3.3.0 Todorov et al. (2012) and the v5 version of the tasks in Gymnasium 1.1.1. According to the official documentation Farama Foundation (2025), the v5 version fixes a previous bug where the robot could receive rewards while in an unhealthy state, and it also increases the difficulty of robot control, posing a greater challenge for RL algorithms. The reward function for each environment is defined as R_i for the i-th objective, with all rewards scaled to similar ranges. Each robotics control task has a fixed duration of 1000 timesteps (i.e., maximum timestep).

1) HalfCheetah-2:

Action and observation space dimensionality: $\mathcal{A} \in \mathbb{A}^6$ and $\mathcal{S} \in \mathbb{R}^{17}$. The first objective is forward speed:

$$R_1 = \min(0.5 \times v_x, 2).$$

The second objective is energy efficiency:

$$R_2 = 2 - \sum_{i=1}^{6} a_i^2,$$

where v_x is the speed in x direction, and a_i is the action of each actuator.

2) Hopper-2:

Action and observation space dimensionality: $\mathcal{A} \in \mathbb{R}^3$ and $\mathcal{S} \in \mathbb{R}^{11}$. The first objective is speed:

$$R_1 = 2 \times v + R_{\text{alive}} - C_{\text{cost}}.$$

The second objective is jumping height:

$$R_2 = 20 \times \max(0, h - h_{\text{init}}) + R_{\text{alive}} - C_{\text{cost}},$$

where v is the speed, $R_{\text{alive}} = 1$ is the alive bonus, $C_{\text{cost}} = 0.0002 \times \sum_{i=1}^{3} a_i^2$ is the control cost, h is the current height, h_{init} is the initial height, and a_i is the action of each actuator.

3) Swimmer-2:

Action and observation space dimensionality: $\mathcal{A} \in \mathbb{R}^2$ and $\mathcal{S} \in \mathbb{R}^8$. The first objective is speed:

$$R_1 = v.$$

The second objective is energy efficiency:

$$R_2 = 2 - \sum_{i=1}^{2} a_i^2,$$

where v is the speed, and a_i is the action of each actuator.

4) Ant-2:

Action and observation space dimensionality: $\mathcal{A} \in \mathbb{R}^8$ and $\mathcal{S} \in \mathbb{R}^{105}$. The first objective is x-axis speed:

$$R_1 = 0.35 \times v_x + R_{\text{alive}} - C_{\text{cost}}.$$

The second objective is y-axis speed:

$$R_2 = 0.35 \times v_y + R_{\text{alive}} - C_{\text{cost}},$$

where v_x is the x-axis speed, v_y is the y-axis speed, $R_{\text{alive}} = 1$ is the alive bonus, $C_{\text{cost}} = \sum_{i=1}^{8} a_i^2$ is the control cost, and a_i is the action of each actuator.

5) Walker2d-2:

Action and observation space dimensionality: $\mathcal{A} \in \mathbb{R}^6$ and $\mathcal{S} \in \mathbb{R}^{17}$. The first objective is speed:

$$R_1 = v + R_{\text{alive}}.$$

The second objective is energy efficiency:

$$R_2 = 3 - \sum_{i=1}^{6} a_i^2 + R_{\text{alive}},$$

where v is the speed, $R_{\text{alive}} = 1$ is the alive bonus, and a_i is the action of each actuator.

6) Humanoid-2:

Action and observation space dimensionality: $\mathcal{A} \in \mathbb{R}^{17}$ and $\mathcal{S} \in \mathbb{R}^{348}$. The first objective is speed:

$$R_1 = 1.25 \times v + R_{\text{alive}}.$$

The second objective is energy efficiency:

$$R_2 = 4 - 3 \times \sum_{i=1}^{17} a_i^2 + R_{\text{alive}},$$

where v is the speed, $R_{\text{alive}} = 4$ is the alive bonus, and a_i is the action of each actuator.

7) Hopper-3:

Action and observation space dimensionality: $\mathcal{A} \in \mathbb{R}^3$ and $\mathcal{S} \in \mathbb{R}^{11}$. The first objective is speed:

$$R_1 = 2 \times v + R_{\text{alive}}.$$

The second objective is jumping height:

$$R_2 = 20 \times \max(0, h - h_{\text{init}}) + R_{\text{alive}}.$$

The third objective is energy efficiency:

$$R_3 = \max(0, 3 - 20 \times \sum_{i=1}^{3} a_i^2) + R_{\text{alive}},$$

where v is the speed, $R_{\text{alive}} = 1$ is the alive bonus, h is the current height, h_{init} is the initial height, and a_i is the action of each actuator.

Additionally, the discrete 4-D environment Lunar-Lander-4, the discrete 6-D environment Fruit-Tree-6, and the continuous 9-D environment Building-9 are built same as C-MORL Liu et al. (2025b). For EU metric evaluation, we evenly generate an preference set Ω in a systematic manner-with specified intervals Δ. For all environment configuration details, see Table 4.

E.2 Training Details

We run all our experiments on a workstation with an Intel i9-13900K CPU, NVIDIA GeForce RTX-4090, and 128G memory. For MOPPO, MOSAC, and MOTD7, their neural network structures are shown in the Pseudocodes 1–3, respectively. Each hidden layer comprises 256 units, activated by both `Tanh(·)` and `ReLU(·)`. For PGMORL and PA2D-MORL, we use the same neural network structure as MOPPO.

Hyperparameter Settings:

- **env_steps**: The total number of environment steps, with details provided in Appendix D.
- **MOPPO:** All hyperparameters are consistent across all benchmarks and multi-objective robotic control environments, as listed in Table 5.
- **MOSAC:** All hyperparameters are consistent across all multi-objective environments, as listed in Table 6.
- **MOTD7:** All hyperparameters are consistent across all multi-objective environments, as listed in Table 7.
- **Benchmarks:** Following Xu et al. (2020) and Hu & Luo (2024), hyperparameters for PGMORL, PA2D-MORL and C-MORL are set to ensure convergence of all algorithms, as shown in Table 8 across all $\leq$ 3-objective environments. The meaning of each parameter is described in Appendix D.
- **MPFT framework:** Episode settings for training (i.e., $\{\Xi_i, \Psi_i\}_{i=1}^{m}$, $\{\Xi_k, \Psi_k\}_{k=1}^{K}$, u, and v in Algorithm 1) are summarized in Table 10 across all $\leq$ 3-objective environments, and steps settings are listed in Table 9. For all $>$ 3-objective environments the hyperparameter configuration is as: for Fruit-Tree-6, $\Xi_{i=1} = 60$, $\{\Xi_i\}_{i=2}^{6} = 0$, $\Psi_{i=1} = 60$, $\{\Psi_i\}_{i=2}^{6} = 0$, $u = 1$, $v = 2$, steps $= 280$, and $K = 0$; for Lunar-Lander-4, $\Xi_{i=2} = 240$, $\{\Xi_i\}_{i\neq 2} = 0$, $\Psi_{i=1} = 80$, $\{\Psi_i\}_{i\neq 2} = 0$, $u = 1$, $v = 2$, steps $= 1000$, and $K = 0$; and for Building-9, $\Xi_{i=1} = 30$, $\{\Xi_i\}_{i=2}^{9} = 0$, $\Psi_{i=1} = 200$, $\{\Psi_i\}_{i=2}^{9} = 0$, $u = 1$, $v = 2$, steps $= 1000$, $K = 0$.

Table 4: Environment details of continuous control benchmarks

Environments	m	Continuous	State Space	Action Space	Policy buffer sizes	Reference point	Δ
HalfCheetah-2	2	Yes	$\mathcal{S} \subset \mathbb{R}^{17}$	$\mathcal{A} \subset \mathbb{R}^{6}$	200	$[0, 0]$	0.01
Hopper-2	2	Yes	$\mathcal{S} \subset \mathbb{R}^{11}$	$\mathcal{A} \subset \mathbb{R}^{3}$	200	$[0, 0]$	0.01
Swimmer-2	2	Yes	$\mathcal{S} \subset \mathbb{R}^{8}$	$\mathcal{A} \subset \mathbb{R}^{2}$	200	$[0, 0]$	0.01
Ant-2	2	Yes	$\mathcal{S} \subset \mathbb{R}^{105}$	$\mathcal{A} \subset \mathbb{R}^{8}$	200	$[0, 0]$	0.01
Walker2d-2	2	Yes	$\mathcal{S} \subset \mathbb{R}^{17}$	$\mathcal{A} \subset \mathbb{R}^{6}$	200	$[0, 0]$	0.01
Humanoid-2	2	Yes	$\mathcal{S} \subset \mathbb{R}^{348}$	$\mathcal{A} \subset \mathbb{R}^{17}$	200	$[0, 0]$	0.01
Hopper-3	3	Yes	$\mathcal{S} \subset \mathbb{R}^{11}$	$\mathcal{A} \subset \mathbb{R}^{3}$	300	$[0, 0]$	0.1
Lunar-Lander-4	4	No	$\mathcal{S} \subset \mathbb{R}^{8}$	$\mathcal{A} \subset \mathbb{R}^{4}$	300	$[-101, -1001, -101, -101]$	0.1
Fruit-Tree-6	6	No	$\mathcal{S} \subset \mathbb{R}^{2}$	$\mathcal{A} \subset \mathbb{A}^{2}$	300	$[-1, -1, -1, -1, -1, -1]$	0.5
Building-9	9	Yes	$\mathcal{S} \subset \mathbb{R}^{29}$	$\mathcal{A} \subset \mathbb{A}^{23}$	300	$[0, 0, 0, 0, 0, 0, 0, 0, 0]$	0.5

Table 5: MOPPO hyperparameters.

parameter name	**value**
learning rate	3×10^{-4}
discount (γ)	0.99
gae lambda	0.95
mini-batch size	32
ppo epoch	10
entropy coef	0.01
value loss coef	0.5
optimizer	Adam

Table 6: MOSAC hyperparameters.

parameter name	value
learning rate	3×10^{-4}
discount (γ)	0.99
replay buffer size	10^6
mini-batch size	256
target smoothing coef (τ)	0.001
target entropy ($\hat{\mathcal{H}}_i$)	-$\lvert\mathcal{A}\rvert$
optimizer	Adam

Table 7: MOTD7 hyperparameters.

parameter name	value
learning rate	3×10^{-4}
discount (γ)	0.99
replay buffer size	10^6
mini-batch size	256
target policy noise	$\mathcal{N}(0, 0.2^2)$
target policy noise clipping ϵ	(-0.5, 0.5)
policy update frequency	2
Target update rate	0.004
optimizer	Adam

Table 9: steps setting.

Algorithm	Walker2d-2	Humanoid-2	Halfcheetah-2	Hopper-2	Ant-2	Swimmer-2	Hopper-3
MPFT-MOPPO	2048	4096	2048	8192	4096	2048	8192
MPFT-MOSAC	2000	2000	2000	4000	4000	2000	4000
MPFT-MOTD7	2000	2000	2000	2000	2000	2000	2000

F Additional Results

In this section, we provide additional experimental results across all evaluated environments, including visualizations of the Pareto-approximation fronts for all algorithms, tracking analyses of the MPFT framework, convergence analyses of both MPFT and the benchmark approaches, and hyperparameter sensitivity analyses of the MPFT framework. For all MPFT framework based MP-MORL algorithms, we focus on the top-1 sparse region (i.e., $K = 1$).

F.1 Pareto-Approximation Fronts

We plot the Pareto fronts discovered by each algorithm across all robotic control environments in Figures 3 and 5. Additionally, we present a visualization of the Pareto front for Building-9 in Figure 4, demonstrating the Pareto-tracking mechanism's capability to track the Pareto front under high-dimensional objectives. The results demonstrate that the proposed MPFT framework can integrate both online and offline RL algorithms to obtain a Pareto-approximation policy set with state-of-the-art (SOTA) performance. In particular, the MPFT/SPFT, which incorporates the advanced offline RL algorithm TD7, achieves the highest HV values across all environments and produces a densely tracked Pareto front. Moreover, MPFT-based MP-MORL algorithms can better explore policies that map to the edges of the Pareto front in the objective space.

Given that MPFT-MOTD7 achieves the best overall performance, the subsequent subsections focus on analyzing the MPFT framework based on this algorithm.

Table 8: PA2D-MORL/PGMORL/C-MORL hyperparameters setting.

Environment	n	m_w	m_t	D	G	**env_steps**
Walker2d-2	8	80	20	4096	60	8×4096×(80+(60×20))
Humanoid-2	8	200	40	4096	100	8×4096×(200+(100×40))
Halfcheetah-2	8	80	20	4096	60	8×4096×(80+(60×20))
Hopper-2	8	200	40	4096	60	8×4096×(200+(60×40))
Ant-2	8	200	40	4096	60	8×4096×(200+(60×40))
Swimmer-2	8	40	10	4096	60	8×4096×(40+(60×10))
Hopper-3	15	200	40	4096	50	15×4096×(200+(50×40))

Table 10: Hyperparameters of MPFT framework based algorithms in $\leq$ 3-D environment ("$(1+2)$" indicates the numbers of consecutive Pareto-reverse and Pareto-ascent updates, i.e., u=1 and v=2).

Environment	**Algorithm (MPFT-)**	$\Xi_{i=1}$ / $\Psi_{i=1}$	$\Xi_{i=2}$ / $\Psi_{i=2}$	$\Xi_{i=3}$ / $\Psi_{i=3}$	$\{\Xi_k/\Psi_k\}_{k=1}^K$
Walker2d-2	MOPPO	1500 / 300×(1+2)	600 / 800×(1+2)	- / -	800 / 800×(1+2)
	MOSAC	1000 / 500×(1+2)	400 / 1000×(1+2)	- / -	500 / 600×(1+2)
	MOTD7	500 / 500×(1+2)	100 / 500×(1+2)	- / -	500 / 500×(1+2)
Humanoid-2	MOPPO	1600 / 500×(1+2)	1000 / 900×(1+2)	- / -	1000 / 900×(1+2)
	MOSAC	3000 / 1200×(1+2)	2000 / 2000×(1+2)	- / -	1500 / 1700×(1+2)
	MOTD7	2500 / 1200×(1+2)	600 / 1700×(1+2)	- / -	1400 / 1400×(1+2)
Halfcheetah-2	MOPPO	600 / 800×(0+2)	600 / 800×(0+2)	- / -	600 / 800×(0+2)
	MOSAC	500 / 400×(0+2)	500 / 400×(0+2)	- / -	500 / 400×(0+2)
	MOTD7	100 / 300×(0+2)	100 / 300×(0+2)	- / -	100 / 500×(0+2)
Hopper-2	MOPPO	200 / 300×(1+2)	200 / 300×(1+2)	- / -	200 / 300×(1+2)
	MOSAC	700 / 400×(1+2)	700 / 400×(1+2)	- / -	700 / 400×(1+2)
	MOTD7	800 / 800×(1+2)	800 / 800×(1+2)	- / -	800 / 800×(1+2)
Ant-2	MOPPO	1200 / 400×(0+2)	1200 / 400×(0+2)	- / -	1200 / 400×(0+2)
	MOSAC	700 / 400×(1+2)	700 / 400×(1+2)	- / -	700 / 400×(1+2)
	MOTD7	800 / 800×(1+2)	800 / 800×(1+2)	- / -	800 / 800×(1+2)
Swimmer-2	MOPPO	400 / 150×(1+2)	400 / 150×(1+2)	- / -	400 / 200×(1+2)
	MOSAC	300 / 200×(1+2)	300 / 200×(1+2)	- / -	300 / 200×(1+2)
	MOTD7	100 / 200×(0+2)	100 / 200×(0+2)	- / -	100 / 400×(0+2)
Hopper-3	MOPPO	500 / 400×(1+2)	500 / 400×(1+2)	500 / 400×(1+2)	500 / 400×(1+2)
	MOSAC	800 / 600×(1+2)	800 / 600×(1+2)	800 / 600×(1+2)	800 / 600×(1+2)
	MOTD7	800 / 1200×(1+2)	500 / 1500×(1+2)	400 / 1400×(1+2)	1000 / 1500×(1+2)

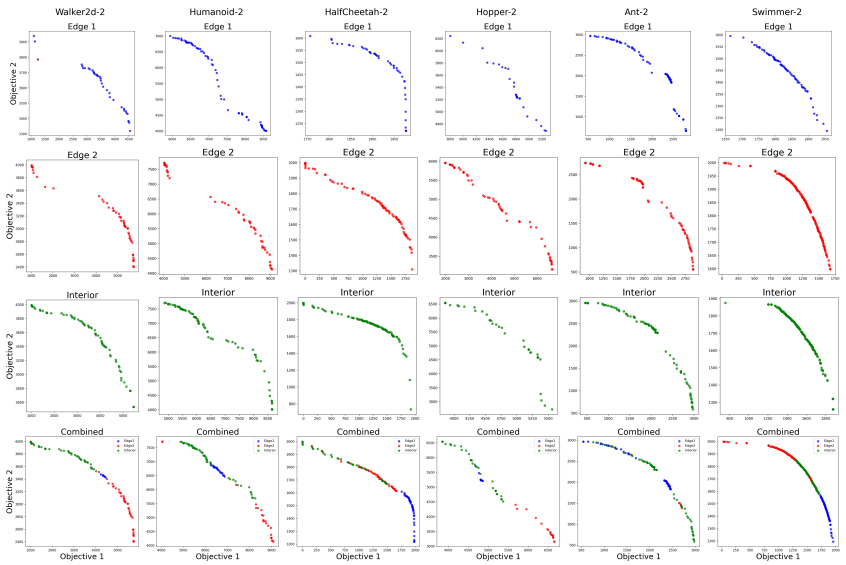

Figure 6: Pareto front tracking process for the MPFT-MOTD7 algorithm in all two-objective environments.

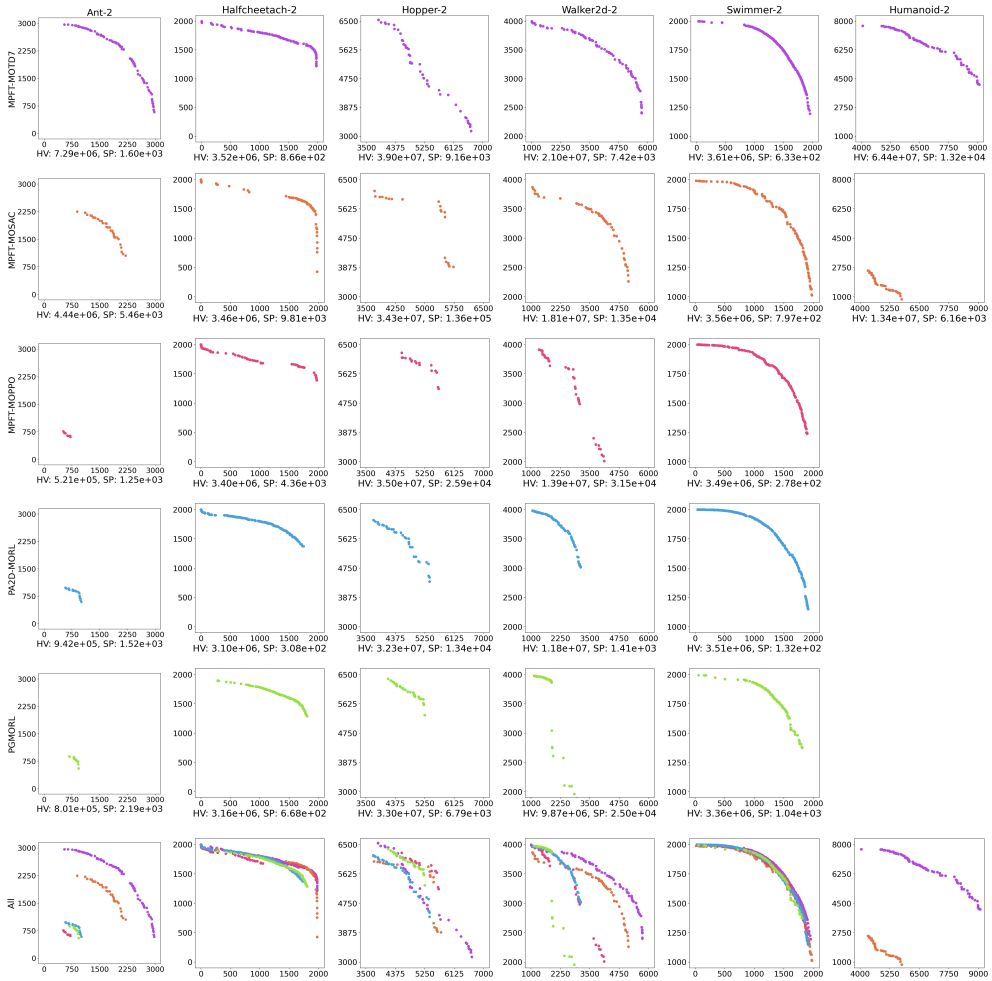

Figure 3: The Pareto front approximation comparison for all two-objective environments. The Pareto fronts of MPFT-MOPPO, PGMORL, and PA2D-MORL on the Humanoid-2 environment are omitted.

F.2 MPFT Framework Tracking Analysis

In this subsection, we analyzed and visualized the tracked Pareto front for both Stage 2 and Stage 3. Figures 7 and 6 show the MPFT-MOTD7 algorithm tracking the Pareto-edge and Pareto-interior fronts in three-objective robotic control environment Hopper-3 and in all two-objective robotic control environments, respectively. The results indicate that the Pareto front tracked starting from the Pareto-vertex policies in Stage 2 is more densely distributed along the edge of the Pareto front, whereas the exploration of some Pareto-interior policies remains insufficient. Consequently, Stage 2 alone is insufficient to obtain a complete Pareto front, especially in challenging environments such as Hopper-2, Hopper-3, and Humanoid.

In Stage 3, the proposed objective weight adjustment method guides policy training toward the sparse regions of the tracked Pareto front, enabling the discovery of the Pareto-interior policies. These policies are then used as anchor points to track the Pareto front in the direction of each objective using the proposed Pareto-tracking mechanism. From the combined results in Figures7 and 6, the sparse regions of the tracked Pareto front are visibly reduced after Stage 3.

Moreover, our experimental results show that the proposed MPFT framework can independently trace a Pareto front starting from any Pareto-vertex policy. This is particularly advantageous in edge

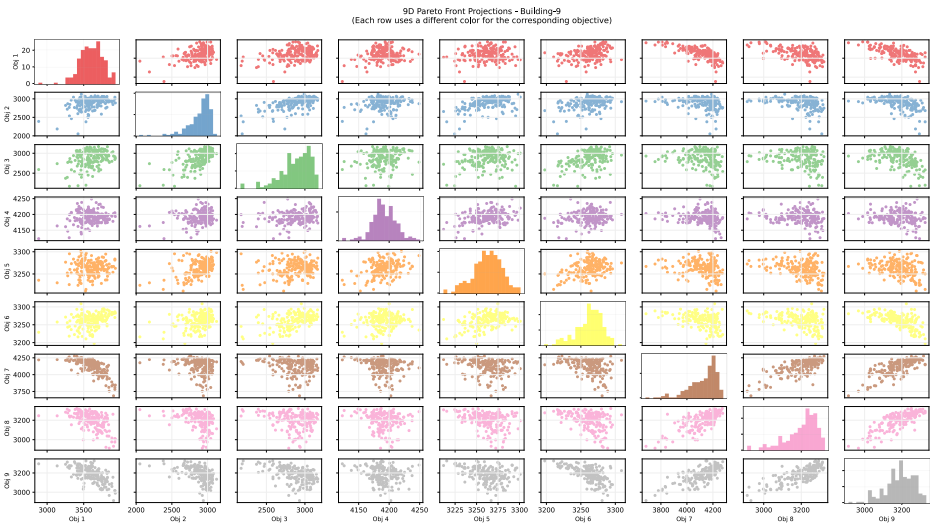

Figure 4: The Pareto front visualization of SPFT for Building-9. (HV: 8.33×10^{31}, SP: 1.74×10^{3}, EU: 3.53×10^{31})

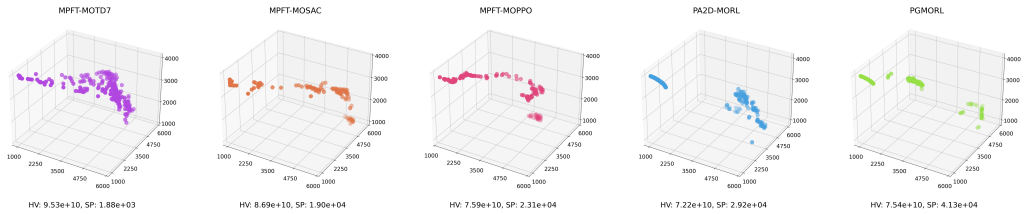

Figure 5: The Pareto front approximation comparison for the three-objective Hopper-3 environment.

scenarios with limited hardware resources, as only a single Pareto front tracing is required rather than maintaining m parallel tracks—an ability that evolutionary frameworks cannot readily achieve.

F.3 Convergence Analysis

We do not compare the HV and SP learning curves of the MPFT framework based algorithms and the benchmarks in the same figure. This is because both HV and SP metrics can only be evaluated in Stage 4. But to verify the convergence of the MPFT framework, we present the HV and SP learning curves for $\mathcal{F}^i_{edge}$ and $\mathcal{F}^k_{inter}$, which are traced in Stage2 and Stage3, respectively.

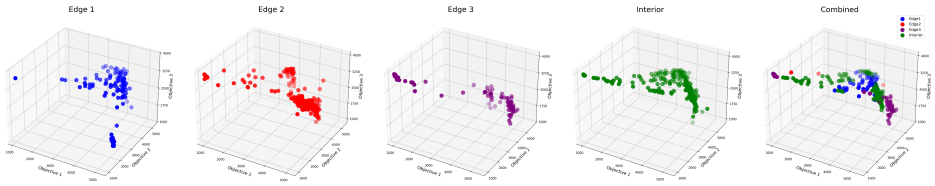

Figure 7: Pareto front tracking process of the MPFT-MOTD7 algorithm in the three-objective Hopper-3 environment.

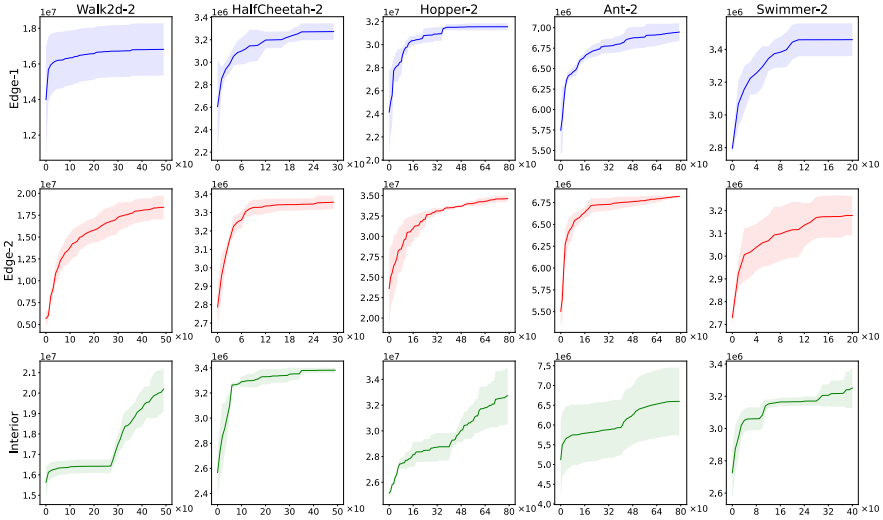

Figure 8: HV metric learning curves of the MPFT-MOTD7 algorithm on five robotic control environments.

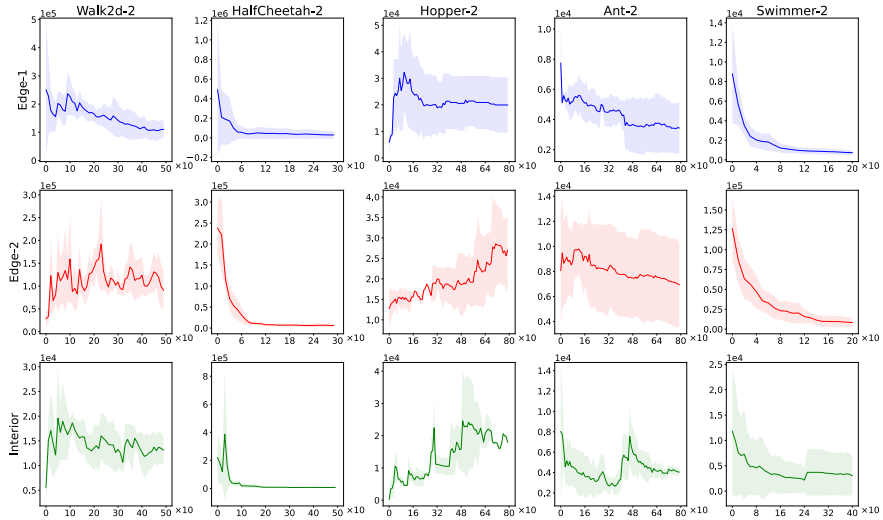

Figure 9: SP metric learning curves of the MPFT-MOTD7 algorithm on five robotic control environments.

Figures 8 and 9 show the HV and SP metrics learning curves of the MPFT-MOTD7 algorithm in five robotic control environments. The x-axis denotes the Ψ_i or Ψ_k episodes, the y-axis denotes the HV or SP metrics, and the shaded area represents the standard deviation calculated from three independent runs. The Humanoid-2 curves are omitted because no benchmark data are available for this environment, and the Hopper-3 curves are omitted because it is a three-objective environment and the Edge-3 figure cannot be properly aligned. From Figures 8 and 9, we can observe that each track, $\mathcal{F}^i_{edge}$ or $\mathcal{F}^k_{inter}$, eventually converges, indicating the eventual convergence of the overall track set $\mathcal{F}$ traced by the MPFT framework. In addition, to ensure a fair comparison, the results of all evolutionary framework based benchmarks are reported after their convergence, and their HV and SP learning curves are plotted in Figure 10. In this figure, the x-axis denotes the generations, the y-axis denotes the HV and SP metrics, and the shaded areas represent the standard deviations derived from three independent runs.

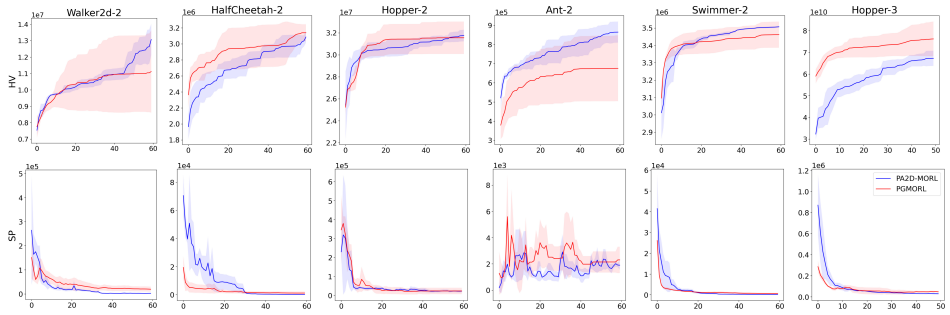

Figure 10: HV and SP metric learning curves of all benchmarks on six robotic control environments.

F.4 Sensitivity Analysis of Pareto Tracking Mechanism

We first analyze the sensitivity of the Pareto-tracking mechanism to the hyperparameters u and v, in order to validate the conclusion $u < v$ derived in Appendix C. We fix $K = 1$ and keep all parameters consistent with Table 10, except for u and v. Figure 11 reports the mean and standard deviation over three independent runs of MPFT-MOTD7 algorithm. We observe that the MPFT framework performs well when $u < v$, yielding stable HV and SP values, whereas the SP performance deteriorates when $u > v$. Specifically, the setting $u = 0, v = 2$ achieves the best performance in simpler environments (e.g., Swimmer-2 and HalfCheetah-2), while $u = 1, v = 2$ performs best in more complex environments (e.g., Ant-2). These results suggest that in simpler environments, a higher proportion of Pareto-ascent phase helps mitigate the excessive exploration induced by the Pareto-reverse phase, whereas in more complex environments, increasing the proportion of Pareto-reverse phase is beneficial for covering a broader Pareto front though its proportion should remain lower than that of the Pareto-ascent phase.

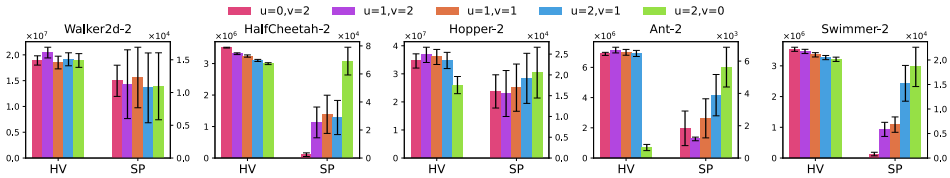

Figure 11: Performance of MPFT-MOTD7 under different (u, v) settings on five robotic control environments. Error bars denote ± 1 standard deviation.

We further investigate the impact of the hyperparameter steps on the performance of the MPFT framework. Specifically, we fix $K = 1$ and keep all other settings identical to Table 10, except for steps. Figure 12 presents the mean and standard deviation over three independent runs of MPFT-MOTD7. The results show that increasing in steps has little effect on HV in relatively simple environments (e.g., Swimmer-2 and HalfCheetah-2), whereas more complex environments benefit from larger steps, particularly in terms of SP. However, once steps is sufficiently large for Pareto-tracking mechanism to converge, further increases steps provide diminishing returns on both HV

and SP. Thus, selecting an appropriate steps is crucial for balancing sample efficiency and performance.

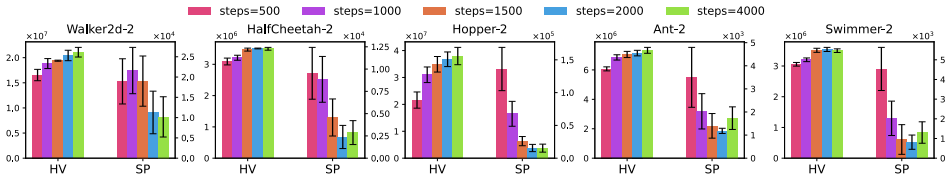

Figure 12: Performance of MPFT-MOTD7 under different steps settings on five robotic control environments. Error bars denote ± 1 standard deviation.

F.5 Robustness Analysis of MPFT

In this subsection, we evaluate the robustness of MPFT-MOTD7 with respect to four key factors: (i) the initialize parameters $\{\Xi_i, \Xi_k\}$, (ii) the tracking parameters $\{\Psi_i, \Psi_k\}$, (iii) the total interaction budget **env_steps**, and (iv) the policy-buffer size. We additionally compare the performance drop of SPFT relative to MPFT.

Table 11: Evaluation of HV, SP, and EU for 2-objective environments. 10% indicates that $\{\Xi_i\}$ and $\{\Xi_k\}$ is 10% of the corresponding $\{\Xi_i\}$ and $\{\Xi_k\}$ in Table 10.

Environment	Metrics	100%	50%	20%	10%	Best Baseline
Walker2d-2	HV ↑ ($\times 10^7$)	**2.044 ± 0.104**	**2.043 ± 0.528**	1.853 ± 0.529	1.689 ± 0.243	1.308 ± 0.094
	SP ↓ ($\times 10^4$)	1.127 ± 0.525	1.012 ± 0.125	0.943 ± 0.066	0.901 ± 0.052	**0.323 ± 0.132**
	EU ↑ ($\times 10^3$)	**4.443 ± 0.063**	**4.439 ± 0.077**	3.966 ± 0.081	3.772 ± 0.048	3.474 ± 0.148
HalfCheetah-2	HV ↑ ($\times 10^6$)	**3.506 ± 0.013**	**3.510 ± 0.170**	3.352 ± 0.030	3.339 ± 0.056	3.345 ± 0.037
	SP ↓ ($\times 10^4$)	0.238 ± 0.130	0.253 ± 0.111	0.240 ± 0.094	0.232 ± 0.078	**0.024 ± 0.012**
	EU ↑ ($\times 10^3$)	**1.784 ± 0.010**	**1.786 ± 0.012**	**1.781 ± 0.015**	1.773 ± 0.019	1.673 ± 0.155
Hopper-2	HV ↑ ($\times 10^7$)	**3.676 ± 0.272**	3.636 ± 0.219	3.564 ± 0.215	3.396 ± 0.168	3.177 ± 0.052
	SP ↓ ($\times 10^4$)	1.552 ± 0.553	1.503 ± 0.965	**1.498 ± 0.590**	1.516 ± 0.518	2.281 ± 0.668
	EU ↑ ($\times 10^3$)	**5.851 ± 0.148**	5.740 ± 0.147	5.682 ± 0.145	5.550 ± 0.123	5.389 ± 0.045
Ant-2	HV ↑ ($\times 10^6$)	**7.144 ± 0.189**	7.093 ± 0.183	6.816 ± 0.185	6.664 ± 0.232	0.865 ± 0.055
	SP ↓ ($\times 10^3$)	1.182 ± 0.113	1.363 ± 0.468	1.281 ± 0.227	**1.141 ± 0.137**	1.359 ± 0.347
	EU ↑ ($\times 10^3$)	**2.467 ± 0.040**	**2.448 ± 0.033**	2.427 ± 0.047	2.394 ± 0.039	0.883 ± 0.105
Swimmer-2	HV ↑ ($\times 10^6$)	**3.539 ± 0.067**	**3.505 ± 0.097**	3.435 ± 0.109	3.406 ± 0.093	3.507 ± 0.004
	SP ↓ ($\times 10^3$)	0.814 ± 0.362	1.221 ± 0.471	1.054 ± 0.351	1.388 ± 0.537	**0.188 ± 0.033**
	EU ↑ ($\times 10^3$)	**1.748 ± 0.017**	**1.746 ± 0.020**	1.734 ± 0.012	1.711 ± 0.017	1.740 ± 0.001

Impact of $\{\Xi_i, \Xi_k\}$. Table 11 reports the effect of $\{\Xi_i, \Xi_k\}$ while keeping all other hyperparameters fixed. When the values are reduced to 50% of the default configuration in Table 10, performance degradation is negligible. When reduced to 10%, HV and EU decrease by at most 20%, while SP remains nearly unchanged. Despite this reduction, MPFT remains competitive relative to the best baselines. This behavior arises because $\{\Xi_i, \Xi_k\}$ directly influence the quality of Pareto-vertex and Pareto-interior policies; larger values allow MPFT to discover solutions further from the reference point, enabling more favorable initialization for Pareto tracking.

Impact of $\{\Psi_i, \Psi_k\}$. Table 12 evaluates sensitivity to $\{\Psi_i, \Psi_k\}$. Reducing these values to 50% results in only mild performance drops; reducing them to 10% yields at most a 23% decrease in HV and EU, but noticeably affects SP. This is expected, as $\{\Psi_i, \Psi_k\}$ determine the density of discovered Pareto-approximate policies. Larger values allow MPFT to populate the Pareto set more densely, ensuring better front coverage. Even when reduced to 10%, MPFT maintains competitive HV and EU compared with the best baselines.

Impact of env_steps. To study interaction efficiency, we adjust $\{\Xi_i, \Xi_k\}$, $\{\Psi_i, \Psi_k\}$, and steps to reduce **env_steps** while keeping all other settings fixed. Table 13 shows that using only 50% of the default interaction budget results in slight performance degradation; using 10% leads to at most a 23% drop in HV and EU, while SP decreases more significantly. Since **env_steps** affects (i) the convergence of Pareto tracking (via steps), (ii) the quality of discovered Pareto-vertex/interior policies (via $\{\Xi_i, \Xi_k\}$), and (iii) the number of discovered Pareto-approximate policies (via $\{\Psi_i, \Psi_k\}$),

Table 12: Evaluation of HV, SP, and EU for 2-objective environments. 10% indicates that $\{\Psi_i\}$ and $\{\Psi_k\}$ is 10% of the corresponding $\{\Psi_i\}$ and $\{\Psi_k\}$ in Table 10.

Environment	Metrics	100%	50%	20%	10%	Best Baseline
Walker2d-2	HV ↑ ($\times 10^7$)	**2.043 ± 0.104**	2.032 ± 0.092	1.880 ± 0.110	1.709 ± 0.129	1.308 ± 0.094
	SP ↓ ($\times 10^4$)	1.127 ± 0.525	1.652 ± 0.695	7.538 ± 2.017	19.27 ± 9.268	**0.323 ± 0.132**
	EU ↑ ($\times 10^3$)	**4.443 ± 0.063**	4.382 ± 0.011	4.207 ± 0.202	4.006 ± 0.191	3.474 ± 0.148
HalfCheetah-2	HV ↑ ($\times 10^6$)	**3.506 ± 0.013**	3.462 ± 0.039	3.390 ± 0.087	3.293 ± 0.119	3.345 ± 0.037
	SP ↓ ($\times 10^4$)	0.238 ± 0.130	0.547 ± 0.366	3.296 ± 0.868	4.822 ± 2.591	**0.024 ± 0.012**
	EU ↑ ($\times 10^3$)	**1.784 ± 0.010**	**1.784 ± 0.012**	**1.780 ± 0.006**	1.772 ± 0.016	1.673 ± 0.155
Hopper-2	HV ↑ ($\times 10^7$)	**3.676 ± 0.272**	3.539 ± 0.103	3.456 ± 0.333	3.375 ± 0.291	3.177 ± 0.052
	SP ↓ ($\times 10^4$)	**1.552 ± 0.553**	6.911 ± 2.934	12.862 ± 3.443	18.51 ± 6.317	2.281 ± 0.668
	EU ↑ ($\times 10^3$)	**5.851 ± 0.148**	5.691 ± 0.166	5.650 ± 0.119	5.501 ± 0.209	5.389 ± 0.045
Ant-2	HV ↑ ($\times 10^6$)	**7.144 ± 0.189**	6.714 ± 0.283	5.681 ± 0.295	5.553 ± 0.467	0.865 ± 0.055
	SP ↓ ($\times 10^3$)	**1.182 ± 0.113**	5.492 ± 0.257	11.53 ± 3.210	13.08 ± 2.786	1.359 ± 0.347
	EU ↑ ($\times 10^3$)	**2.467 ± 0.040**	2.416 ± 0.034	2.283 ± 0.059	2.274 ± 0.070	0.883 ± 0.105
Swimmer-2	HV ↑ ($\times 10^6$)	**3.539 ± 0.067**	3.498 ± 0.077	3.473 ± 0.075	3.348 ± 0.083	3.507 ± 0.004
	SP ↓ ($\times 10^3$)	0.814 ± 0.362	2.873 ± 1.658	8.102 ± 2.371	16.29 ± 4.741	**0.188 ± 0.033**
	EU ↑ ($\times 10^3$)	**1.748 ± 0.017**	**1.746 ± 0.014**	**1.745 ± 0.014**	1.739 ± 0.011	1.740 ± 0.001

Table 13: Evaluation of HV, SP, and EU for 2-objective environments. 10% indicates that **env_steps** is 10% of the corresponding **env_steps** in Table 1.

Environment	Metrics	100%	50%	20%	10%	Best Baseline
Walker2d-2	HV ↑ ($\times 10^7$)	**2.044 ± 0.104**	2.031 ± 0.069	1.875 ± 0.039	1.701 ± 0.572	1.308 ± 0.094
	SP ↓ ($\times 10^4$)	1.127 ± 0.525	1.827 ± 0.042	8.332 ± 3.942	24.11 ± 8.091	**0.323 ± 0.132**
	EU ↑ ($\times 10^3$)	**4.443 ± 0.063**	4.382 ± 0.013	4.119 ± 0.018	4.002 ± 0.074	3.474 ± 0.148
HalfCheetah-2	HV ↑ ($\times 10^6$)	**3.506 ± 0.013**	3.456 ± 0.030	3.339 ± 0.033	3.284 ± 0.062	3.305 ± 0.037
	SP ↓ ($\times 10^4$)	0.238 ± 0.130	0.552 ± 0.351	3.953 ± 1.865	5.422 ± 2.778	**0.024 ± 0.012**
	EU ↑ ($\times 10^3$)	**1.784 ± 0.010**	1.779 ± 0.027	1.774 ± 0.008	1.768 ± 0.012	1.673 ± 0.155
Hopper-2	HV ↑ ($\times 10^7$)	**3.676 ± 0.272**	3.528 ± 0.249	3.404 ± 0.105	3.342 ± 0.268	3.177 ± 0.052
	SP ↓ ($\times 10^4$)	**1.552 ± 0.553**	4.399 ± 2.926	15.967 ± 3.853	19.97 ± 6.833	2.281 ± 0.668
	EU ↑ ($\times 10^3$)	**5.851 ± 0.148**	5.640 ± 0.163	5.625 ± 0.146	5.484 ± 0.196	5.389 ± 0.045
Ant-2	HV ↑ ($\times 10^6$)	**7.144 ± 0.189**	6.677 ± 0.258	5.595 ± 0.248	5.520 ± 0.113	0.865 ± 0.055
	SP ↓ ($\times 10^3$)	**1.182 ± 0.113**	5.536 ± 0.768	12.43 ± 4.992	17.41 ± 4.340	1.359 ± 0.347
	EU ↑ ($\times 10^3$)	**2.467 ± 0.040**	2.410 ± 0.121	2.261 ± 0.038	2.249 ± 0.034	0.883 ± 0.105
Swimmer-2	HV ↑ ($\times 10^6$)	**3.539 ± 0.067**	3.480 ± 0.103	3.340 ± 0.212	3.250 ± 0.236	3.507 ± 0.004
	SP ↓ ($\times 10^3$)	0.814 ± 0.362	3.119 ± 0.658	10.62 ± 4.582	16.42 ± 2.384	**0.188 ± 0.033**
	EU ↑ ($\times 10^3$)	**1.748 ± 0.017**	**1.743 ± 0.018**	1.728 ± 0.031	1.717 ± 0.039	1.740 ± 0.001

a larger budget naturally leads to denser and higher-quality Pareto fronts. Importantly, in practical applications, evaluation typically focuses on EU rather than HV and SP; under this criterion, even at 10% of the default interaction budget, MPFT remains competitive relative to the best baselines.

Table 14: Evaluation of HV, SP, and EU for 2-objective environments under varying policy buffer sizes, alongside a comparison with the best-performing baseline. 100 indicates the policy buffer sizes (i.e., 100 policies model are stored after trianing).

Environment	Metrics	200	100	50	20	Best Baseline
Walker2d-2	HV ↑ ($\times 10^7$)	**2.044 ± 0.104**	**2.044 ± 0.104**	2.029 ± 0.071	2.016 ± 0.377	1.308 ± 0.094
	SP ↓ ($\times 10^4$)	1.127 ± 0.525	1.127 ± 0.525	3.048 ± 0.960	14.88 ± 5.143	**0.323 ± 0.132**
	EU ↑ ($\times 10^3$)	**4.443 ± 0.063**	**4.443 ± 0.063**	**4.407 ± 0.101**	4.392 ± 0.074	3.474 ± 0.148
HalfCheetah-2	HV ↑ ($\times 10^6$)	**3.506 ± 0.013**	**3.488 ± 0.029**	3.446 ± 0.074	3.424 ± 0.062	3.087 ± 0.173
	SP ↓ ($\times 10^4$)	0.238 ± 0.130	0.314 ± 0.139	0.772 ± 0.145	2.126 ± 0.105	**0.024 ± 0.012**
	EU ↑ ($\times 10^3$)	**1.784 ± 0.010**	**1.783 ± 0.016**	**1.780 ± 0.009**	1.777 ± 0.007	1.673 ± 0.155
Hopper-2	HV ↑ ($\times 10^7$)	**3.676 ± 0.272**	**3.651 ± 0.220**	**3.604 ± 0.105**	3.559 ± 0.230	3.177 ± 0.052
	SP ↓ ($\times 10^4$)	**1.552 ± 0.553**	6.128 ± 2.977	10.26 ± 4.998	20.28 ± 6.761	2.281 ± 0.668
	EU ↑ ($\times 10^3$)	**5.851 ± 0.148**	5.752 ± 0.147	5.731 ± 0.134	5.644 ± 0.208	5.389 ± 0.045
Ant-2	HV ↑ ($\times 10^6$)	**7.144 ± 0.189**	7.074 ± 0.261	7.036 ± 0.274	6.833 ± 0.242	0.865 ± 0.055
	SP ↓ ($\times 10^3$)	**1.182 ± 0.113**	5.835 ± 0.571	12.21 ± 1.884	18.04 ± 1.781	1.359 ± 0.347
	EU ↑ ($\times 10^3$)	**2.467 ± 0.040**	**2.467 ± 0.039**	**2.464 ± 0.040**	2.458 ± 0.035	0.883 ± 0.105
Swimmer-2	HV ↑ ($\times 10^6$)	**3.539 ± 0.067**	3.498 ± 0.096	3.422 ± 0.177	3.308 ± 0.192	3.507 ± 0.004
	SP ↓ ($\times 10^3$)	0.814 ± 0.362	5.596 ± 1.569	9.665 ± 3.338	15.61 ± 3.253	**0.188 ± 0.033**
	EU ↑ ($\times 10^3$)	**1.748 ± 0.017**	1.738 ± 0.029	1.736 ± 0.001	1.725 ± 0.032	1.740 ± 0.001

Impact of Policy Buffer Size. Table 14 compares MPFT with varying policy-buffer sizes against the best baseline. Even when the buffer is substantially reduced, MPFT maintains consistently strong performance while using significantly fewer stored policies. In Walker2d-2, the results remain unchanged across buffer sizes, indicating that the Pareto-optimal set in this environment does not exceed the tested buffer capacities.

Table 15: Evaluation of HV, SP, and EU for 2-objective environments, comparison with the SPFT and best-performing baseline.

Environment	Metrics	MPFT	SPFT	Best Baseline
Walker2d-2	HV ↑ ($\times 10^7$)	**2.044 ± 0.104**	**1.682 ± 0.180**	1.308 ± 0.094
	SP ↓ ($\times 10^4$)	1.127 ± 0.525	6.156 ± 3.868	**0.323 ± 0.132**
	EU ↑ ($\times 10^3$)	**4.443 ± 0.063**	**4.054 ± 0.028**	3.474 ± 0.148
HalfCheetah-2	HV ↑ ($\times 10^6$)	**3.506 ± 0.013**	**3.310 ± 0.133**	3.087 ± 0.173
	SP ↓ ($\times 10^4$)	0.238 ± 0.130	0.673 ± 0.338	**0.024 ± 0.012**
	EU ↑ ($\times 10^3$)	**1.784 ± 0.010**	**1.755 ± 0.005**	1.673 ± 0.155
Hopper-2	HV ↑ ($\times 10^7$)	**3.676 ± 0.272**	**3.252 ± 0.170**	3.177 ± 0.052
	SP ↓ ($\times 10^4$)	**1.552 ± 0.553**	2.707 ± 1.610	**2.281 ± 0.668**
	EU ↑ ($\times 10^3$)	**5.851 ± 0.148**	**5.420 ± 0.158**	5.389 ± 0.045
Ant-2	HV ↑ ($\times 10^6$)	**7.144 ± 0.189**	**6.345 ± 0.946**	0.865 ± 0.055
	SP ↓ ($\times 10^3$)	**1.182 ± 0.113**	5.413 ± 1.460	**1.359 ± 0.347**
	EU ↑ ($\times 10^3$)	**2.467 ± 0.040**	**2.339 ± 0.185**	0.883 ± 0.105
Swimmer-2	HV ↑ ($\times 10^6$)	**3.539 ± 0.067**	3.460 ± 0.123	**3.507 ± 0.004**
	SP ↓ ($\times 10^3$)	0.814 ± 0.362	0.690 ± 0.305	**0.188 ± 0.033**
	EU ↑ ($\times 10^3$)	**1.748 ± 0.017**	1.739 ± 0.021	**1.740 ± 0.001**

Comparison Between SPFT and MPFT. Table 15 reports the performance difference between SPFT and MPFT. Despite using only a *single* policy for Pareto tracking, SPFT still achieves HV and EU performance that surpasses the best baseline in most environments, confirming the strong effectiveness of MPFT's Pareto-tracking mechanism.

Overall Summary. When $u < v$, steps is chosen to ensure convergence of the Pareto-tracking mechanism, and $\{\Xi_i, \Psi_i\}_{i=1}^m$ and $\{\Xi_k, \Psi_k\}_{k=1}^K$ are sufficiently large, MPFT yields stable and competitive results. Moreover, under constrained computational or memory resources, MPFT can still produce competitive Pareto fronts by reducing $\mathbf{env\_steps}$, shrinking policy-buffer sizes, or using SPFT.

G Comparision with Evolutionary Based MPMORL

To clarify the conceptual differences between MPFT and existing evolutionary MP-MORL frameworks, we provide a concise comparison as shown in Table 16. PGMORL, PA2D-MORL, and C-MORL all rely on population-based or non-dominated policy set-based policy selection within an evolutionary loop, whereas MPFT eliminates population-based evolution entirely and instead employs a deterministic single-policy *Pareto-tracking mechanism* for each track. This fundamental shift leads to differences in initialization, evolution, sparse-region handling, computational behavior, and offline-RL compatibility.

Table 16: Comparison between evolutionary MP-MORL frameworks and MPFT.

Component	PGMORL	PA2D-MORL	C-MORL	MPFT (ours)
Framework type	Evolutionary	Evolutionary	Evolutionary	**Deterministic tracking**
Initialization	$\mathcal{P}_{op} = \{\pi^{(1)}, \ldots, \pi^{(n)}\}$	$\mathcal{P}_{op}$	$\mathcal{P}_{op}$	$\{\pi^{(1)}, \ldots, \pi^{(m)}\}$
Policies selection	from $\mathcal{P}_{op}$	from $\mathcal{P}_{op}$ and $\mathcal{F}$	from $\mathcal{P}_{op}$	$\{\pi^{(1)}, \ldots, \pi^{(m)}\}$ **only**
Selection rule	Predictive model	Random	Crowding-distance	**None**
Policy updating method	Predictive model	Pareto-ascent direction	Interior point method	**Pareto-tracking mechanism**
Sparse-region handling	Interpolation	Sparse regions sampling	None	**New policy + tracking**
Complexity driver	T_{EVOL} 12	T_{EVOL}	T_{EVOL}	T_{MPFT} 11
Offline RL	×	×	×	✓

Summary. MPFT differs fundamentally from PGMORL, PA2D-MORL, and C-MORL by removing the evolutionary population/archive selection loop and replacing it with independent single-

policy Pareto-tracking trajectories. This yields distinct algorithmic behavior and enables features such as offline-RL compatibility. Thus, MPFT represents a non-evolutionary paradigm rather than an incremental modification of PGMORL, PA2D-MORL, or C-MORL.

H Discussion and Limitations of MPFT

With the novel design of Pareto-tracking mechanism, and sparse regions filling method, the proposed MPFT framework offers a systematic and efficient way to approximate the Pareto front without maintaining a large policy population. Despite its effectiveness in reducing agent-environment interactions, we observe that in some environments the tracked Pareto front still shows sparse regions, indicating underexplored areas. To address this limitation, we plan to develop more advanced sparse regions filling strategies and provide stronger global convergence guarantees. Additionally, while MPFT integrates both online and offline RL, its performance depends on the underlying algorithms: PPO benefits from a simpler network architecture, enabling faster training, but suffers from low sample efficiency and limited performance, whereas SAC and TD7 achieve higher efficiency and performance at the expense of greater training time. In the future, we plan to explore more lightweight and sample-efficient algorithms within MPFT to further enhance its practicality, such as the meta-policy, mixture-of-experts architectures, and multi-head networks can be adopted in the network architecture.

Pseudocode 3. MOTD7 Network Structure

Variables: `zs_dim = 256, hidden_dim = 256`
Sate-Action Value Network $Q^{(z_s,z_{sa})}(s,a;\varphi)$:

```
▷ MOTD7 uses two state value networks each with the same network
and forward pass.
L0 = Linear(state_dim + action_dim, hidden_dim)
L1 = Linear(zs_dim * 2 + hidden_dim, hidden_dim)
L2 = Linear(hidden_dim, hidden_dim)
L3 = Linear(hidden_dim, m)
```

$Q^{(z_s,z_{sa})}(s,a;\varphi)$ Forward Pass:

```
input = concatenate([state, action]); x = AvgL1Norm(L0(input))
x = concatenate([zsa, zs, x]); x = ReLU(L1(x))
x = ReLU(L2(x)); value = L3(x)
```

Policy Network π_θ:

```
L0 = Linear(state_dim, hidden_dim)
L1 = Linear(zs_dim + hidden_dim, hidden_dim)
L2 = Linear(hidden_dim, hidden_dim)
L3 = Linear(hidden_dim, action_dim)
```

π_θ Forward Pass:

```
input = state; x = AvgL1Norm(L0(input)); x = concatenate([zs, x])
x = ReLU(L1(x)),x = ReLU(L2(x)); action = tanh(L3(x))
```

State Encoder Network f:

```
L1 = Linear(state_dim, hidden_dim)
L2 = Linear(hidden_dim, hidden_dim); L3 = Linear(hidden_dim, zs_dim)
```

f Forward Pass:

```
input = state; x = ReLU(L1(input))
x = ReLU(L2(x))
zs = AvgL1Norm(L3(x))
```

State-Action Encoder Network g:

```
L1 = Linear(action_dim + zs_dim, hidden_dim)
L2 = Linear(hidden_dim, hidden_dim)
L3 = Linear(hidden_dim, zs_dim)
```

g Forward Pass:

```
input = concatenate([action, zs])
x = ReLU(L1(input))
x = ReLU(L2(x))
zsa = L3(x)
```

Pseudocode 1. MOPPO Network Structure

Variables:
`hidden_dim = 256`
Sate Value Network $V(s;\phi)$:
```
L0 = Linear(state_dim, hidden_dim)
L1 = Linear(hidden_dim, hidden_dim)
L2 = Linear(hidden_dim, m)
```
$V(s;\phi)$ Forward Pass:
```
input = state
x = Tanh(L0(input))
x = Tanh(L1(x))
value = L2(x)
```
Policy Network π_θ:
```
L0 = Linear(state_dim, hidden_dim)
L1 = Linear(hidden_dim, hidden_dim)
L2 = Linear(hidden_dim, action_dim)
Log_Std = Parameter(zeros(action_dim))
```
π_θ Forward Pass: `input = state`
```
x = Tanh(L0(input))
x = Tanh(L1(x))
mean = Tanh(L2(x))
std = exp(Log_Std).clamp(1e-6,20)
action ~ Normal(mean, std)
```

Pseudocode 2. MOSAC Network Structure

Variables: `hidden_dim = 256`
Sate Value Network $V(s;\phi)$:
```
▷ MOSAC uses two state value networks each with the same network
and forward pass.
L0 = Linear(state_dim, hidden_dim)
L1 = Linear(hidden_dim, hidden_dim); L2 = Linear(hidden_dim, m)
```
$V(s;\phi)$ Forward Pass:
```
input = state
x = ReLU(L0(input))
x = ReLU(L1(x)); value = L2(x)
```
Sate-Action Value Network $Q(s,a;\varphi)$:
```
▷ MOSAC uses two sate-action value networks each with the same
network and forward pass.
L0 = Linear(state_dim + action_dim, hidden_dim)
L1 = Linear(hidden_dim, hidden_dim); L2 = Linear(hidden_dim, m)
```
$Q(s,a;\varphi)$ Forward Pass:
```
input = concatenate([state, action])
x = ReLU(L0(input))
x = ReLU(L1(x)); value = L2(x)
```
Policy Network π_θ:
```
L0 = Linear(state_dim, hidden_dim)
L1 = Linear(hidden_dim, hidden_dim)
Mean = Linear(hidden_dim, action_dim)
Log_Std = Linear(hidden_dim, action_dim)
```
π_θ Forward Pass:
```
input = state; x = ReLU(L0(input)); x = ReLU(L1(x)); mean = Mean(x)
std = exp(Log_Std(x)).clamp(1e-6,20), action ~ Normal(mean, std)
```