Human-in-the-Loop Policy Optimization for Preference-Based Multi-Objective Reinforcement Learning *

Ke Li[1] and Han Guo[2]

[1]Department of Computer Science, University of Exeter, EX4 4RN, Exeter, UK
[2]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
k.li@exeter.ac.uk, guohan1999@gmail.com

Abstract: Multi-objective reinforcement learning (MORL) aims to find a set of high-performing and diverse policies that address trade-offs between multiple conflicting objectives. However, in practice, decision makers (DMs) often deploy only one or a limited number of trade-off policies. Providing too many diversified trade-off policies to the DM not only significantly increases their workload but also introduces noise in multi-criterion decision-making. With this in mind, we propose a human-in-the-loop policy optimization framework for preference-based MORL that interactively identifies policies of interest. Our method proactively learns the DM's implicit preference information without requiring any a priori knowledge, which is often unavailable in real-world black-box decision scenarios. The learned preference information is used to progressively guide policy optimization towards policies of interest. We evaluate our approach against three conventional MORL algorithms that do not consider preference information and four state-of-the-art preference-based MORL algorithms on two MORL environments for robot control and smart grid management. Experimental results fully demonstrate the effectiveness of our proposed method in comparison to the other peer algorithms.

Keywords: Multi-objective reinforcement learning (MORL), human-in-the-loop, preference learning, decomposition multi-objective optimization.

1 Introduction

Many real-world decision-making tasks involve more than one potentially competing objectives. For example, the trade-offs between makespan and energy consumption in workflow scheduling [1], the balance between microgrid benefits and power grid requirements in microgrid system development [2], and the competing considerations of fuel cost, efficiency, and safety in robot control [3]. Multi-objective reinforcement learning (MORL) algorithms have gained increasing attention as a powerful paradigm for learning control policies that optimize over multiple conflicting criteria simultaneously [4]. The existing MORL literature can be broadly classified into three main categories.

The first and most prevalent category of MORL approaches involves aggregating multiple objective functions into a single scalar reward using a convex combination, a technique commonly referred to as linear scalarization, as exemplified by [5, 6]. Specifically, the weight assigned to each objective represents the decision maker's relative preference for that particular objective. Subsequently, a standard reinforcement learning (RL) algorithm is employed to optimize this scalar reward. While this approach is intuitive, its broader applicability is constrained by the need for *a priori* preferences regarding the different criteria in the underlying MORL task. Obtaining such preferences is often unrealistic in real-world black-box scenarios.

Another widely adopted strategy in MORL involves identifying a set of Pareto-optimal policies that span the entire spectrum of possible preferences in the domain, which are then presented to

*This paper is submitted for potential publication. Reviewers can feel free to use this manuscript in peer review.

the decision maker (DM) to choose the policies that align with their interests, as done in [7–9]. This *a posteriori* decision-making approach, however, not only increases the DM's workload but also introduces potentially irrelevant or even noisy information during the multi-criterion decision-making (MCDM) process. Moreover, it is not guaranteed to yield a diverse set of trade-off solutions that adequately cover the entire Pareto front within a limited budget, thereby potentially hindering the discovery of policies of interest in MCDM.

Distinct from the previous two categories of approaches, the last one is *interactive* MORL paradigm, an emerging area first introduced in [10]. It incorporates human-in-the-loop interaction in MORL, enabling DMs to progressively learn the characteristics of the underlying task. Consequently, the MORL process is 'supervised' to identify policies of interest. As the DM has more frequent opportunities to provide new information, they may feel more in control and engaged in the overall *optimization-cum-decision-making* process. Surprisingly, this line of research has received relatively limited attention in the current MORL community, as highlighted by [11–13].

In this paper, we present a general framework (dubbed `CBOB`) for designing interactive MORL algorithms that involve human-in-the-loop interaction, enabling the progressive learning of the DM's feedback and the adaptation of the search for policies of interest. This framework comprises three key modules: `seeding`, `preference elicitation`, and `optimization`.

- The `seeding` module functions as an initial process to search for a set of trade-off policies that offer a reasonable approximation of the Pareto-optimal front.
- The `preference elicitation` module focuses on gathering the DM's preference information and converting it into a format that can be utilized in the subsequent `optimization` module.
- The `optimization` module encompasses any policy optimization algorithm used in deep RL. It employs the elicited preference information to guide the search for preferred policies in a parallel manner.

In practice, our proposed `CBOB` framework is highly versatile and readily applicable to complex environments featuring high-dimensional and continuous state and action spaces. Moreover, its modular design allows for flexibility, as each module can be adapted to other dedicated configurations. We conducted extensive experiments on two sets of MORL benchmark problems based on MuJoCo [14] and a multi-microgrid system design problem [15], which thoroughly demonstrate the effectiveness and competitiveness of our proposed `CBOB` algorithm instance in comparison to three conventional MORL and four state-of-the-art preference-based MORL peer algorithms.

In the rest of this paper, we will provide some preliminary knowledge pertinent to this paper along with a pragmatic review of existing works on constrained multi-objective optimization in Section 2. In Section 3, we will delineate the algorithmic implementations of our proposed `CBOB` framework. The experimental settings are given in Section 4, and the empirical results are presented and discussed in Section 5. At the end, Section 6 concludes this paper and sheds some lights on future directions.

2 Preliminary

This section starts with some basic definitions pertinent to this paper, followed by a pragmatic overview of some selected up-to-date development of MORL and preference learning.

2.1 Multi-Objective Markov Decision Process

In this paper, a MORL is formulated as a 5-tuple $\langle \mathcal{S}, \mathcal{A}, T, \mathbf{r}, \gamma \rangle$ multi-objective Markov decision process (MOMDP):

- $\mathcal{S}$ is the state space, i.e., the set of all available n-dimensional states $\mathbf{s} = (s_1, \ldots, s_n)^\top \in \mathbb{R}^n$.
- $\mathcal{A}$ is the action space, i.e., the set of all available l-dimensional actions $\mathbf{a} = (a_1, \ldots, a_l)^\top \in \mathbb{R}^l$.

- T is the transition matrix where $T(s_{t+1}|s_t, a_t)$ represents the probability that the state s_t transfers to the other s_{t+1} after taking the action a_t.
- $\mathbf{r} = (r_1(s_t, a_t), \ldots, r_m(s_t, a_t))^\top$ is the reward vector after taking the action a_t.
- $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_m)^\top \in (0, 1]^m$ is a vector of discount factors.

In a MOMDP, a policy $\pi : \mathcal{S} \to \mathcal{A}$ determines how the current state s_t move to the next one s_{t+1} by taking the action $a_t \sim \pi(s_t)$. In particular, π is associated with a vector of expected returns $\mathbf{J}(\pi) = (J_1(\pi), \cdots, J_m(\pi))^\top$ where $J_i(\pi) = \mathbb{E}\left(\sum_{t=0}^{H} \boldsymbol{\gamma}_i^t r_i(s_t, a_t)\right)$, and $H > 0$ is the horizon of the MOMDP.

2.2 Multi-Objective Optimization

The multi-objective optimization (MOP) problem considered in this paper is defined as:

$$\begin{array}{ll} \text{maximize} & \mathbf{F}(\pi) = (f_1(\pi), \cdots, f_m(\pi))^\top \\ \text{subject to} & \pi : \mathcal{S} \to \mathcal{A} \end{array}, \tag{1}$$

where π is a policy and $\mathbf{F}(\pi) = \mathbf{J}(\pi)$ is an objective vector. Note that all objective functions are supposed to be conflicting with each other. That is to say the improvement of one objective can lead to the detriment of the others.

Definition 1. *Given two policies π^1 and π^2, π^1 is said to dominate π^2 (denoted by $\pi^1 \succeq \pi^2$) if and only if $f_i(\pi_1) \geq f_i(\pi_2)$ for all $i \in \{1, \cdots, m\}$ and $\mathbf{F}(\pi^1) \neq \mathbf{F}(\pi^2)$.*

Definition 2. *A policy π^* is said to be Pareto-optimal if and only if $\nexists \pi'$ such that $\pi' \succeq \pi^*$.*

Definition 3. *The set of all Pareto-optimal policies is called the Pareto-optimal set (PS), i.e., $\mathcal{PS} = \{\pi^* | \nexists \pi' \text{ such that } \pi' \succeq \pi^*\}$ and their corresponding objective vectors form the Pareto-optimal front (PF), i.e., $\mathcal{PF} = \{\mathbf{F}(\pi^*) | \pi^* \in PS\}$.*

2.3 Related Works

This section begins with a brief overview of existing research on MORL from both single- and multi-policy perspectives. Subsequently, we highlight selected works on preference learning for MORL and inverse reinforcement learning (IRL). For a more comprehensive literature review, interested readers can refer to excellent survey papers such as [4, 16–18].

2.3.1 Single-policy MORL

Single-policy MORL methods rely on *a priori* preferences for different objectives to first convert a multi-objective problem into a single-objective one through aggregation. Subsequently, any off-the-shelf RL algorithm can be applied to obtain a single optimal policy. Aggregation can take various forms, such as linear scalarization [5, 6] or more complex approaches. For instance, in [19], Rolf proposed an exponential function with the sum of different objectives as the exponential term. In [20], a distributional method was introduced to first calculate a set of policies, each addressing a selected objective and represented as a distribution over actions. A new policy is then synthesized by combining all these policies. Although single-policy MORL algorithms are easy to implement, their practical applicability is limited due to the unrealistic assumption of exact a priori preferences.

2.3.2 Multi-policy MORL

Multi-policy MORL methods decompose the original problem into several subproblems. In principle, any scalarization method can be employed to formulate the subproblems. Subsequently, multiple single-objective RL algorithms are applied to solve these subproblems. Some researchers have proposed

iterative frameworks to address different subproblems sequentially [8, 21, 22]. This approach involves sampling one subproblem at a time, optimizing the corresponding policy, and then switching to another subproblem based on the performance of the previous policy optimization. Alternatively, there have been attempts to solve all subproblems simultaneously within a parallel framework [3, 23, 24]. Multi-policy MORL can discover a set of Pareto-optimal policies without any a priori information or adaptation during the policy optimization process. However, it incurs higher computational and storage costs compared to single-policy MORL, which may be detrimental when dealing with problems involving more than two objectives.

2.3.3 IRL and Preference Learning

An emerging research direction involves incorporating human-in-the-loop reinforcement learning by proactively interacting with decision makers (DMs) and learning their preferences, known as preference learning [13, 25]. This is especially useful and practical when the DM's exact demand is not available beforehand. Manually crafted reward functions notoriously struggle to align with the DM's true aspirations [26]. Preference-based reinforcement learning can be roughly divided into two categories: preference-based IRL and preference-based MORL.

Preference-based IRL does not assume a reward function before the algorithm; instead, it infers one from observed demonstrations according to the DM's preference. In works such as [27–29], the algorithms attempt to update the reward function to fit given preference relationships as closely as possible, using preferred demonstrations as input. While these approaches consider the DM's preferences, it can be challenging to explicitly define all preference relationships a priori [30]. In [31], a direct method is proposed that adds an offset to the policy's output during interaction with the DM. However, the policy optimization process remains unmodified, limiting the influence of preference information. [32, 33] further learn a parameterized representation of the DM's preference interactively, guiding the learning process of reward functions and optimal policies.

Preference-based MORL, the second category, has gained popularity in recent years and is the focus of this paper. Unlike IRL, preference-based MORL addresses a well-defined MOMDP problem and learns a utility function representing the trade-off between different objective functions based on the DM's feedback. For example, [17, 34, 35] formulate the preference learning problem as a multi-armed bandit problem, learning the utility function through the DM's feedback in the form of pairwise comparisons or ranked sequences. These methods provide good examples of modeling and iteratively learning the DM's preferences. However, continuous, high-dimensional RL problems lack the availability of comparisons between a set of fixed arms. In [8], Yang et al. proposes *a posteriori* policy adaptation in continuous environments, approximating the PF and using a Gaussian process-based preference model to learn the DM's preference and guide preferred policy selection. However, many trade-off alternative policies on the PF may not align with the DM's preferences, making the search for the entire PF a waste of computational resources. Furthermore, trade-off alternative policies outside the region of interest can introduce irrelevant or noisy information during the decision-making process.

3 Proposed Method

Fig. 1 presents the modular flow chart of our proposed `CBOB` framework. It consists of three core modules: *i*) the `seeding` module that obtains a set of promising seeding policies; *ii*) the `preference elicitation` module that plays as the interface where the DM interacts with the MORL and it learns the DM's preference from her feedback; and *iii*) the `policy optimization` module that leverages the learned preference to guide the search for the policy of interest. Note that the last two modules iterate between each other until the stopping criterion is met. In the following paragraphs, the implementation of each module is delineated step by step.

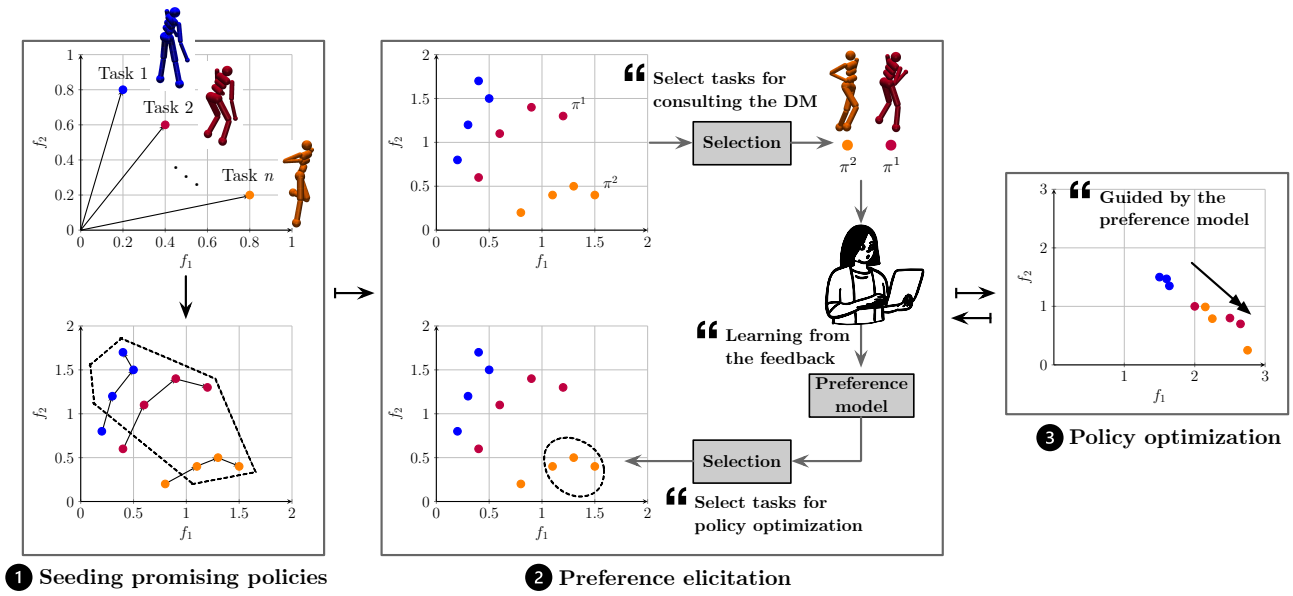

Figure 1: Flowchart of `CBOB`. It iterates between the `preference elicitation` and the `policy optimization` modules until the stopping criterion is met, and outputs the preferred policies.

3.1 Seeding Promising Policies

The `seeding` module operates as a conventional MORL, disregarding the DM's preference information and instead searching for a set of trade-off policies that approximate the PF. In this paper, our fundamental approach for MORL is inspired by the prevalent multi-objective evolutionary algorithm based on decomposition (MOEA/D) [**?**, 36]. We propose decomposing the MORL problem into a collection of subproblems, each representing a linear aggregation of different objective functions:

$$\tilde{J}(\pi, \mathbf{w}) = \sum_{i=1}^{m} w_i J_i(\pi), \tag{2}$$

where $\mathbf{w} = (w_1, \ldots, w_m)^\top$ and $\sum_{i=1}^{m} w_i = 1$ is the weight vector that represents certain preferences for different objectives. Note that any aggregation function can be applied in the `CBOB` framework, while we will investigate this aspect in Section 5.3.6 of our empirical study. To balance convergence and diversity, we employ the Das and Dennis' method [37] to define a set of weight vectors $\mathcal{W} = \{\mathbf{w}^i\}_{i=1}^{N}$ evenly distributed along a unit simplex. Here, $N = \binom{H+m-1}{m-1}$, and $H > 1$ is the number of equally spaced divisions along each coordinate.

In the `CBOB` framework, all subproblems are handled simultaneously in a parallel manner, with each treated as an individual RL task. It is important to note that each RL policy is associated with a dedicated weight vector (i.e., a subproblem) in our `CBOB` framework. Given the continuous state and action spaces of the RL tasks considered in this paper, we employ proximal policy optimization (PPO) [38], a policy-gradient algorithm that optimizes the policy by explicitly updating the policy network π_θ using a gradient-based method. The objective function of the PPO is:

$$\begin{aligned} J^{\text{clip}}(\pi_\theta) = &\mathbb{E}_t\Bigg[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t,\right. \\ &\left.\text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\Bigg], \end{aligned} \tag{3}$$

where $\hat{A}_t$ is an advantage function that estimates the performance of the current state-action tuple $\langle s_t, a_t \rangle$, and $\epsilon \in (0, 1)$ controls the range of policy updates to avoid excessive optimization. The pseudo code of this multi-objective PPO (`MOPPO`) is given in Algorithm 1, the output of which is the set of non-dominated policies, denoted as Π.

Algorithm 1: `MOPPO`

Input: t_{r}: # of environment steps, $\overline{\Pi}$: stored policies

Initialize $\Pi := \emptyset$ for storing optimized policies;
foreach $\langle \pi^i, \mathbf{w}^i \rangle \in \overline{\Pi}$ **do**
  /* every task works parallelly */
  while *the stopping criterion is not met* **do**
    Execute π^i and store the collected trajectory data in a buffer $\mathcal{S}^i := \left(\langle s_1^i, a_1^i, r_1^i \rangle, \ldots, \langle s_{t_{\mathrm{r}}}^i, a_{t_{\mathrm{r}}}^i, r_{t_{\mathrm{r}}}^i \rangle\right)$;
    while *the stopping criterion is not met* **do**
      Sample a trajectory from $\mathcal{S}^i$, and use $\mathbf{w}^i$ to aggregate its rewards;
      Calculate $\hat{\mathcal{A}}_t$ as mentioned in [39];
      Update π^i by optimizing $J^{\mathrm{clip}}(\pi^i)$ in equation (3);
    Insert updated task $\langle \pi^i, \mathbf{w}^i \rangle$ to Π;
return Non-dominated policies in Π

Remark 1. *The motivation of this* `seeding` *module stems from our preliminary experiments, which showed that interacting with the DM and conducting preference learning at the beginning of* `CBOB` *is ineffective. This observation can be attributed to the fact that the initial trade-off policies are far from the PF, which can be misleading for the DM during preference elicitation. As a result, we introduce the* `seeding` *module to first search for a set of reasonably good trade-off policies before proceeding with preference learning. This approach aims to provide a better starting point for the subsequent steps in the* `CBOB` *framework.*

3.2 Preference Elicitation

This module is the crux of proposed `CBOB`, whose pseudo code is given in Algorithm 2. It consists of three steps including `consultation`, `preference learning`, and `preference translation`. They are designed to address the following three questions.

3.2.1 How to collect the DM's feedback

In the `preference elicitation` module, the `consultation` step serves as an interface where the DM is asked to provide feedback on the quality of selected policies based on their preferences. In this work, we consider an indirect preference information in the form of holistic pairwise comparisons of policy pairs $\langle \pi^1, \pi^2 \rangle$. The outcome of this pairwise comparison can be π^1 being *better*, *worse*, or *indifferent* compared to π^2, denoted as $\pi^1 \succ \pi^2$, $\pi^1 \prec \pi^2$, or $\pi^1 \simeq \pi^2$, respectively. Asking the DM to compare all $\binom{|\Pi|}{2}$ policy pairs derived from the seeding policies obtained in the `seeding` module may overwhelm them. To alleviate this issue, we only choose the two most informative policies from Π to consult the DM. Our preliminary experiments showed that this approach not only significantly reduces the DM's workload but is also sufficient for preference learning in `CBOB`. Inspired by [40], we evaluate the information of a policy $\pi \in \Pi$ as:

$$\mathtt{I}(\pi) = \mu(\pi) + \alpha \sqrt{\sigma(\pi) \tilde{n} / \tilde{n}_\pi}, \tag{4}$$

where $\mu(\pi)$ and $\sigma(\pi)$ represent the mean and variance predicted by the preference model, which will be introduced in Section 3.2.2. $\tilde{n}_\pi$ denotes the number of times π has been queried in previous interactions, $\tilde{n}$ represents the total number of queries incurred so far, and $\alpha > 0$ is a parameter that balances exploitation and exploration. Based on this definition, two policies $\pi^1 = \mathrm{argmax}_{\pi \in \Pi} \mathtt{I}(\pi)$ and $\pi^2 = \mathrm{argmax}_{\pi \in \Pi \setminus \pi^1} \mathtt{I}(\pi)$ are chosen for the `consultation` step. The queried policies are stored in a dedicated set $\tilde{\Pi}$.

Remark 2. *In principle, we can employ* `MOPPO`*, as in the* `seeding` *module, to obtain a set of policies that approximate the entire PF. However, these non-dominated policies may be too diverse for DMs to identify their preferred ones that satisfy their aspirations in practice. Thus, a more refined approach is required to accommodate the DM's specific preferences.*

Remark 3. *At the beginning of the* `preference elicitation` *module, no data has been collected from the DM to train the preference model. To address this, during the first $\tau > 1$ iterations, we randomly select two policies in the* `consultation` *step. This approach allows us to gather initial data and subsequently refine the preference model as more information becomes available.*

Remark 4. *In the context of MORL, the DM's preference can be represented by assigning weights to different objective functions based on their importance [7,8,32,41]. However, this presents a paradox in our context, as the preference is unknown a priori. An alternative approach is to assign a numeric score to each candidate [42]. Nevertheless, as discussed in [43], this method can be vague and challenging for non-domain experts to comprehend and apply effectively.*

Remark 5. *Note that the* `preference elicitation` *module in* `CBOB` *is highly efficient where we only need consult the DM once at each round with a pair of candidates.*

3.2.2 How to model the DM's preference information

Based on the holistic pairwise comparison(s) collected in the `consultation` step, the goal of the `preference learning` step is to learn a preference model that evaluates the quality of solutions according to the DM's preference. Here our preference model is represented as a utility function $u(\pi) : \mathbb{R}^m \to \mathbb{R}$ such that the ranking order of a set of testing policies $\hat{\Pi}$ satisfy that $\forall \pi^1, \pi^2 \in \hat{\Pi}$, $u(\pi^1) > u(\pi^2)$ if $\pi^1 \succ \pi^2$ and $u(\pi^1) = u(\pi^2)$ if $\pi^1 \simeq \pi^2$. This paper applies a zero-mean Gaussian process (GP) [44] with the radial basis function kernel to serve as the regression model. In particular, $\tilde{\Pi} = \{\tilde{\pi}^i\}_{i=1}^{\kappa}$ is used as the training data where κ is the number of policies queried at the `consultation` step. Then, we train $u(\pi)$ based on maximum a posterior estimation as:

$$u^\star = \operatorname*{argmax}_u \mathbb{P}(u)\mathbb{P}(\tilde{\Pi}|u), \tag{5}$$

where $\mathbb{P}(u)$ is a zero-mean Gaussian and $\mathbb{P}(\tilde{\Pi}|u) = \prod_{i=1}^{\frac{\kappa}{2}} \Phi(\frac{u(\pi_i^1)-u(\pi_i^2)}{\sqrt{2}})$ is the likelihood of $u(\pi)$, $\Phi(\cdot)$ is the standard normal cumulative distribution. As in [45], for each pair of $\langle u(\pi_i^1), u(\pi_i^2)\rangle$ where $\pi_i^1 \succ \pi_i^2$, solving equation (5) is equivalent to solving the following minimization problem:

$$\text{minimize} - \sum_{i=1}^{\frac{\kappa}{2}} \left[\Phi(\frac{u(\pi_i^1) - u(\pi_i^2)}{\sqrt{2}}) + \frac{1}{2}u^\top(\pi_i^1)K^{-1}u(\pi_i^2)\right], \tag{6}$$

where K is the covariance matrix of $\tilde{\Pi}$. By applying Newton-Raphson formula, we can finally get the numerical approximation of the real $u^\star$. Given a testing policy $\pi^\star$, the mean and variance of $u^\star(\pi^\star)$ are predicted as $\mu(\pi^\star) = \mathbf{k}^{\star\top}K^{-1}\mathbf{U}$ and $\sigma(\pi^\star) = k(\pi^\star, \pi^\star) - \mathbf{k}^{\star\top}(K + \Lambda^{-1})^{-1}\mathbf{k}^\star$, respectively, where $\mathbf{U} = (u^\star(\tilde{\pi}^1), \ldots, u^\star(\tilde{\pi}^\kappa))^\top$. $\mathbf{k}^\star$ is the covariance vector between $\tilde{\Pi}$ and $\pi^\star$. Λ^{-1} is a $\kappa \times \kappa$ matrix for estimating $\mathbf{U}$.

3.2.3 How to leverage the learned preference information

In the `preference translation` step, the learned preference information is converted into a format that can be used to guide the `policy optimization` module. In the context of our proposed `CBOB`, this involves generating a new policy archive Π for the next round of the policy optimization. This process consists of five steps:

Step 1: Assign a preference score $\psi(\pi) = \mu(\pi) + \beta\sigma(\pi)$ to each policy $\pi \in \Pi$.

Algorithm 2: `preference elicitation`

Input: Π: stored non-dominated policies, $\tilde{\Pi}$: queried policies as training data, $u(\pi)$: preference model

/* consultation step */
if *random selection* **then**
  Randomly sample two policies π^1, π^2 from Π.
else
  foreach $\pi \in \Pi$ **do**
    Evaluate its utility value $u(\pi)$ and $\mathtt{I}(\pi)$;
  $\pi^1 := \text{argmax}_{\pi \in \Pi} \mathtt{I}(\pi)$, $\pi^2 := \text{argmax}_{\pi \in \Pi \setminus \{\pi^1\}} \mathtt{I}(\pi)$;
Query the DM with π^1 and π^2, store the result in $\tilde{\Pi}$;
/* preference learning step */
Use the DM's feedback to update $\tilde{\Pi}$ and the preference model $u(\pi)$ as in Section 3.2.2;
/* preference translation step */
Generate the new policy set Π as in Section 3.2.3;
return Π, $\tilde{\Pi}$, $u(\pi)$.

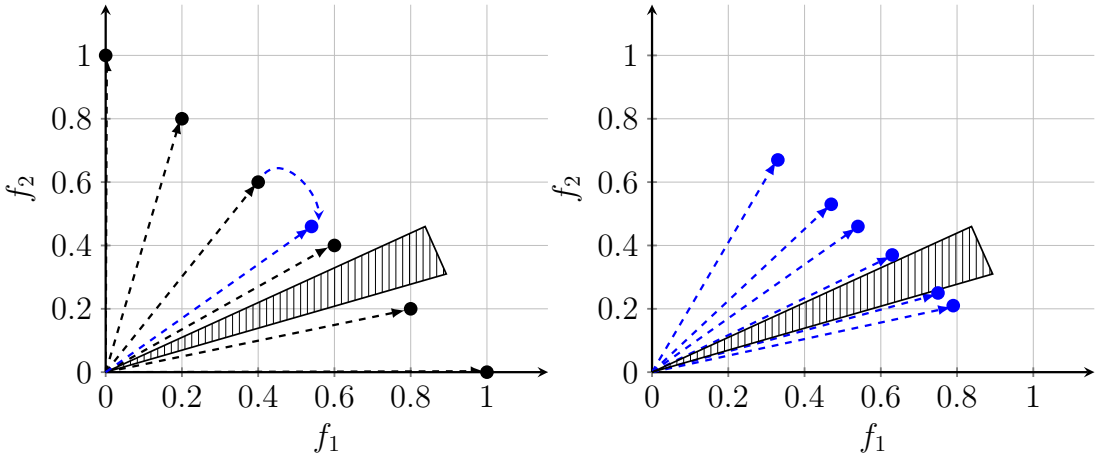

Figure 2: Illustration of the weight vector adjustment in Step 4. (a) The ROI is highlighted as the shaded cone region versus the evenly distributed weight vectors (denoted as ●). (b) Adjusted weight vectors towards the ROI (denoted as ●).

Step 2: Rank the policies in Π based on the preference scores from Step 1. Identify the top κ_1 policies to construct a temporary policy archive $\dot{\Pi}$. Store the weight vectors associated with these policies, considered as promising, in a temporary archive $\dot{\mathcal{W}} = \{\dot{\mathbf{w}}^i\}_{i=1}^{\kappa_1}$.

Step 3: Generate a set of evenly distributed weight vectors $\tilde{\mathcal{W}} = \{\tilde{\mathbf{w}}^i\}_{i=1}^{\tilde{N}}$, as in the `seeding` module, where $\tilde{N} > N$. Set another temporary archive $\ddot{\mathcal{W}} = \emptyset$.

Step 4: Generate a set of weight vectors biased towards the region of interest (ROI), as illustrated in Fig. 2. Then, for $i = 1$ to $\tilde{N}$, perform the following:

  Step 4.1: Determine the reference point of $\tilde{\mathbf{w}}^i$ as $\mathbf{w}^{\text{r}} = \text{argmin}_{\dot{\mathbf{w}} \in \dot{\mathcal{W}}} |\dot{\mathbf{w}} - \tilde{\mathbf{w}}^i|$.

  Step 4.2: Move $\tilde{\mathbf{w}}^i$ towards $\mathbf{w}^{\text{r}}$ as $\ddot{w}_j^i = \tilde{w}_j^i + \eta(w_j^{\text{r}} - \tilde{w}_j^i)$, where $j \in \{1, \cdots, m\}$.

  Step 4.3: Update $\ddot{\mathcal{W}}$ with the new weight vector: $\ddot{\mathcal{W}} = \ddot{\mathcal{W}} \bigcup \{\ddot{\mathbf{w}}\}$ and return to Step 4.

Step 5: Randomly select κ_2 weight vectors from $\ddot{\mathcal{W}}$ and remove the others to form a reduced $\ddot{\mathcal{W}}$. Randomly choose κ_2 policies from $\dot{\Pi}$ and associate each selected policy with a weight vector in $\ddot{\mathcal{W}}$

to form a temporary archive $\ddot{\Pi}$. Construct the new policy archive Π for the next round of the policy optimization as $\Pi = \dot{\Pi} \bigcup \ddot{\Pi}$.

Remark 6. *In Step 1, $\psi(\pi)$ serves as an approximate value function that evaluates the satisfaction of policy π concerning the DM's preference information. Notably, the parameter $\beta > 0$ helps strike a balance between exploitation and exploration in the* `PBMORL` *process.*

Remark 7. *Note that the preference score of a policy is distinct from the information of a policy, as defined in equation (4). This difference arises because the objective of Step 2 is to identify policies that best represent the DM's implicit preferences. By making this distinction, our approach effectively captures the nuances of preference elicitation.*

Remark 8. *In our preliminary experiments, we observed that MORL can become overly exploitative and prone to being trapped in local optima when the* `policy optimization` *module is solely driven by $\dot{\mathcal{W}}$. To address this issue, Steps 3 and 4 are introduced to incorporate additional exploration outside of the ROI. This approach effectively balances exploitation and exploration, ensuring a more robust optimization process*

Remark 9. *In Step 5, the composition of $\mathcal{W}$ is designed to strike a balance between exploitation and exploration, with a focus on enhancing exploitation to accelerate convergence. In practice, we set $\kappa_1 = 80\% \times |\Pi|$ and $\kappa_2 = 20\% \times |\Pi|$. We will conduct a parameter sensitivity study on the settings of κ_1 and κ_2 in Section 5.3.4 of our empirical study.*

3.3 Policy Optimization

In the `policy optimization` module, we employ the `MOPPO` algorithm, as done in the `seeding` module introduced in Section 3.1, which utilizes the biased weight vectors generated in the `preference elicitation` module to guide the MORL process. It is important to note that the outputs of this `policy optimization` module are fed back into the `preference elicitation` module, further enhancing the navigation towards the DM's preferred non-dominated policies until the stopping criterion are met.

3.4 Time Complexity Analysis

Since the `seeding` and the `policy optimization` modules of `CBOB` are MORL based on PPO, the corresponding time complexity is bounded by the PPO itself. Therefore, here we focus on analyzing the time complexity of the `preference elicitation` module, which consists of three steps. Specifically, during the `consultation` step, the preference model first evaluate the quality of policies in Π which incurs $\mathcal{O}(|\Pi|)$ evaluations. Then, it chooses two elite policies to query the DM. After the DM's feedback is collected, the complexity of the `preference learning` step is bounded by $\mathcal{O}(\kappa^3)$, where κ is the number of data instances in the training set. During the `preference translation` step, the computational complexity is mainly dominated by the ranking of policies in Π, i.e., $\mathcal{O}(|\Pi| \log |\Pi|)$. All in all, the time complexity of the `preference elicitation` module is $\max\left\{\mathcal{O}(\kappa^3), \mathcal{O}(|\Pi| \log |\Pi|)\right\}$.

4 Experimental Settings

This section introduces the settings of our empirical study including the benchmark problems, the peer algorithms, and the performance metrics.

4.1 Benchmark Problems

In empirical study, we consider two types of benchmark problems: one is from the popular MuJoCo environment [14] and the other is a multi-microgrid system design (MMSD) problem [15].

Table 1: Lookup table of the mathematical notations used in the MMSD problem [15].

Symbol	Description
n	The number of microgrids
n_{s}	The number of microgrids with power storage
t	The current time step
T	The length of one episode
$P_g(t)$	The sum of power given to all of microgirds at the t-th time step
$p_i^g(t)$	The power given to the i-th microgrid at t-th time step
$p_i^d(t)$	The power demand of the i-th microgrid at the t-th time step
$\lambda(t)$	The power price at the t-th time step
$s_i(t)$	The power storage of the i-th microgrid at the t-th time step
$v_i(t)$	The energy that the distributed power gives to the i-th microgrid at the t-th time step
$b_i(t)$	The base load of the i-th microgrid at the t-th time step
c_i	The maximum rate of storage charging and discharging
l_i	The lower bound of the energy storage of the i-th microgrid
u_i	The upper bound of the energy storage of the i-th microgrid

4.1.1 MuJoCo environment

Following [3], we examine seven benchmark problems developed from MuJoCo. Their objective functions and search spaces are outlined below:

- `Ant-v2`: This problem uses the speeds at the x and y axes as the two objectives. The state space is $\mathcal{S} \in \mathbb{R}^{27}$, and the action space is $\mathcal{A} \in \mathbb{R}^8$.
- `HalfCheetah-v2`: Two objectives are considered, including forward speed and energy consumption. The state space is $\mathcal{S} \in \mathbb{R}^{17}$, and the action space is $\mathcal{A} \in \mathbb{R}^6$.
- `Hopper-v2`: This problem considers forward speed and jumping height as the objectives. The state space is $\mathcal{S} \in \mathbb{R}^{11}$, and the action space is $\mathcal{A} \in \mathbb{R}^3$.
- `Humanoid-v2`: Two objectives are evaluated, including forward speed and energy consumption. The state space is $\mathcal{S} \in \mathbb{R}^{376}$, and the action space is $\mathcal{A} \in \mathbb{R}^{17}$.
- `Swimmer-v2`: This problem focuses on forward speed and energy consumption objectives. The state space is $\mathcal{S} \in \mathbb{R}^8$, and the action space is $\mathcal{A} \in \mathbb{R}^2$.
- `Walker2d-v2`: Two objectives are considered, including forward speed and energy consumption. The state space is $\mathcal{S} \in \mathbb{R}^{17}$, and the action space is $\mathcal{A} \in \mathbb{R}^6$.
- `Hopper-v3`: This problem incorporates three objectives, including forward speed, jumping height, and energy consumption. The state space is $\mathcal{S} \in \mathbb{R}^{11}$, and the action space is $\mathcal{A} \in \mathbb{R}^3$.

4.1.2 MMSD environment

It considers the following three-objective optimization problem:

$$\begin{array}{ll} \text{maximize} & \mathbf{r} = \left(\mathrm{U}_{\mathrm{pg}}(T), \mathrm{U}_{\mathrm{mg}}(T), \sum_{i=1}^{n_s} s_i(T)\right)^\top \\ \text{subject to} & \|s_i(t) - s_i(t-1)\| < c_i, \forall i \in \{1, \ldots, n_{\mathrm{s}}\} \end{array}, \tag{7}$$

where $\mathbf{r} \subseteq \mathbb{R}^3$ is the reward, $l_i < s_i(t) < u_i$. The mathematical definitions of these three objectives are delineated as follows, while the related notations are listed in Table 1.

- $\mathrm{U}_{\mathrm{pg}}(T)$ evaluates the utility value achieved by the power grid after one episode:

$$\mathrm{U}_{\mathrm{pg}}(T) = \sum_{t=1}^{T} \sum_{i=1}^{n} \left(\mathrm{U}(p_i^d(t), w_i) - \lambda(t) p_i^d(t) \right), \tag{8}$$

where

$$\mathrm{U}(p_i^d(t), w_i) = \begin{cases} \dfrac{w_i}{\alpha}, & \text{if } p_i^d(t) \geq \dfrac{w_i}{\alpha} \\ w_i p_i^d(t) - \dfrac{\alpha p_i^d(t)^2}{2}, & \text{if } 0 \leq p_i^d(t) \leq \dfrac{w_i}{\alpha} \end{cases}, \tag{9}$$

where w_i and α are pre-defined parameters.

- $\mathrm{U}_{\mathrm{mg}}(T)$ is the total utility value achieved by all microgrids after one episode:

$$\mathrm{U}_{\mathrm{mg}}(T) = \sum_{t=1}^{T} \lambda(t) P_g(t) - \beta P_g(t)^2, \tag{10}$$

where β is a pre-defined parameter.

- To measure the stability of a multi-microgrid system, we use $\sum_{i=1}^{n_s} s_i(T)$ to evaluate its total energy storage. In particular, we have:

$$s_i(t) = s_i(t-1) + p_i^g(t) - p_i^d(t) + v_i(t), \tag{11}$$

where $p_i^d(t)$ is decided by both the base load of the i-th microgrid [15] and a scaling factor:

$$p_i^d(t) = (1 + h_i(t)) b_i(t), \tag{12}$$

where $h_i(t)$ is defined as:

$$h_i(t) = \begin{cases} 0.01\lambda^2(t) - 0.12\lambda(t) + 0.26, & \text{if } i = 1 \\ -0.01\lambda^2(t) + 0.13, & \text{if } i = 2 \\ -0.01\lambda^2(t) + 0.02\lambda(t) + 0.08, & \text{if } i = 3 \end{cases}. \tag{13}$$

The state space is $\mathcal{S} \subseteq \mathbb{R}^{n+2}$ where $\forall \mathbf{s} \in \mathcal{S}$, we have $\mathbf{s} = (t, p_1^g(t), \ldots, p_n^g(t), \lambda(t))^\top$. The action space is $\mathcal{A} \subseteq \mathbb{R}^{n_s+1}$ where $\forall a \in \mathcal{A}$, we have $a = (\Delta\lambda(t), \Delta p_1^g(t), \ldots, \Delta p_{n_s}^g(t))^\top$.

4.2 Peer Algorithms

In our experiments, we consider two types of peer algorithms. The first category comprises conventional MORL algorithms that do not take the DM's preference information into account.

- `RA` [23]: `RA` transforms a MORL problem into several single-objective RL tasks by weighted aggregations. Different from our proposed `CBOB`, `RA` solves these tasks independently.
- `PGMORL` [3]: Designed to optimize multiple policies in parallel, `PGMORL` iteratively adjusts the directions based on a predictive method.
- `MOIA` [15]: `MOIA` is a dedicated multi-objective evolutionary algorithm tailored for the multi-microgrid system design problem.

The second category consists of preference-based MORL algorithms from the literature:

- `MORL-Adaptation` [8]: By generalizing the Bellman operator to MORL problems, `MORL-Adaptation` learns a universal parametric representation for all latent preferences. After the training phase, the learned policy can adapt to a given preference without further adaptation.
- `META-MORL` [46]: `META-MORL` formulates MORL as a meta-learning problem conditioned on a task distribution over preferences. After the training phase, the learned policy can be adapted to a DM-specified preference through a fine-tuning phase.

Table 2: List of the hyperparameter settings used in our experiments.

Hyperparameter	MuJoCo ($m = 2$)	MuJoCo ($m = 3$)	Microgrid
# of environment steps	8×10^6	8×10^6	4×10^6
# of environment steps for the `seeding` stage	4×10^5	4×10^5	4×10^4
# of interactions with the DM	40	40	40
# of subtasks built before starting `CBOB`	6	21	21
# of microgrids	\	\	3
# of microgrids with energy storage	\	\	2

- `MOMPO` [20]: `MOMPO` learns an action distribution for each objective in each round of policy training and updates the policy by fitting it to a combination of action distributions. Before the training phase, a set of parameters representing the DM's preference for each objective is provided, controlling the learning rate of the corresponding action distribution.
- `MORAL` [13]: `MORAL` first learns a multi-objective reward function from demonstrations. Then, similar to our proposed `CBOB`, it learns the DM's preference as a weight vector based on the Bradley-Terry model [47] to guide the policy optimization.

All peer algorithms used the same number of environment steps, and Table 2 lists the key hyperparameters used in our experiments.

4.3 Performance Metrics

As discussed in [48], quality assessment of non-dominated trade-off policies is far from trivial when considering DM's preference information regarding multiple conflicting criteria. In our experiments, we consider the following two metrics to serve this purpose.

- The first one is the *approximation accuracy* that evaluates the closeness of the best non-dominated policy to the DM preferred one:

$$\epsilon^\star(\Pi) = \min_{\pi \in \Pi} \|\pi - \pi^\star\|_2, \tag{14}$$

  where $\|\cdot\|_2$ is the Euclidean norm, $\pi^\star$ is the golden policy specified by the DM but is unknown to the algorithm, and Π is the non-dominated policy set obtained by MORL.

- The second one is the *average accuracy* that evaluates the average distance of all non-dominated policies in Π to the DM preferred one:

$$\bar{\epsilon}(\Pi) = \frac{\sum_{\pi \in \Pi} \|\pi - \pi^\star\|_2}{|\Pi|}, \tag{15}$$

  where $|\cdot|$ is the cardinality of a set.

Note that calculating both $\epsilon^\star(\Pi)$ and $\bar{\epsilon}(\Pi)$ needs to specify a golden policy $\pi^\star$, which is represented as a m-dimensional hyperplane. Specifically, if we consider that the DM always prefers one objective over the others, $\pi^\star$ is defined as $f_i(\pi^\star) = 10,000$, where i is the objective index preferred by the DM. As for the MMSD problem, we set $f_1(\pi^\star) = 0$ or $f_2(\pi^\star) = 0$ if the DM prefers the first or the second objective, while we set $f_3(\pi^\star) = 450$ if the DM prefers the third objective.

5 Experimental Results

In this section, we present and analyze the results obtained in our experiments.

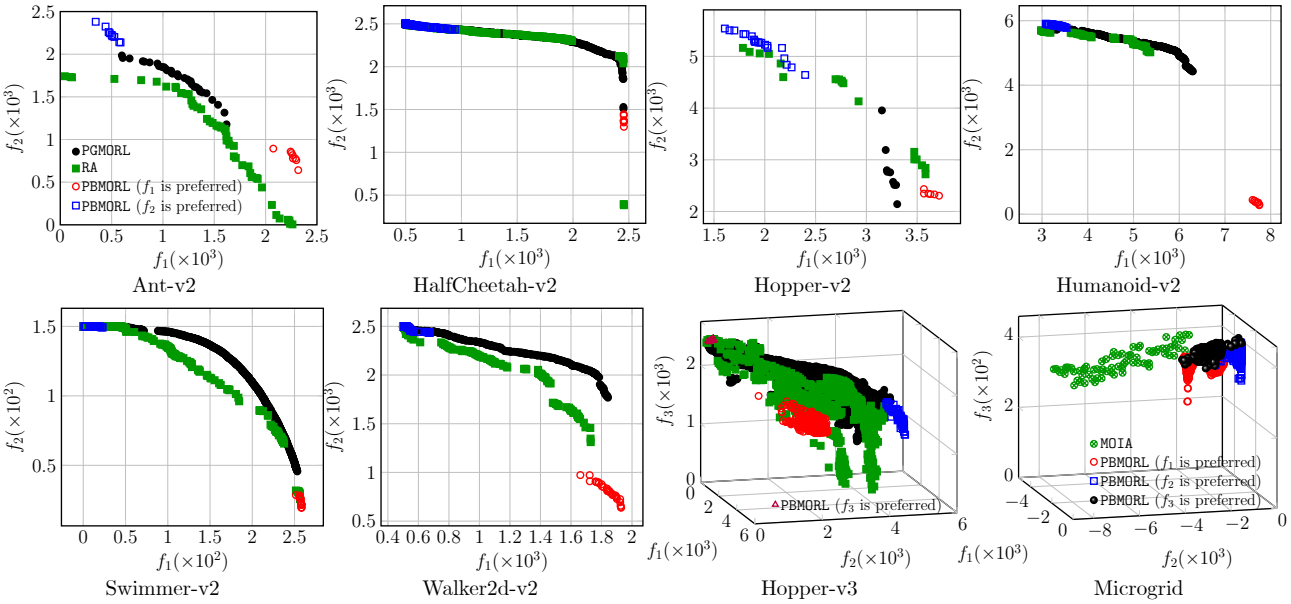

Figure 3: Plots of non-dominated policies obtained by CBOB versus PGMORL, RA, and MOIA with different preferences.

Table 3: Comparison results of $\epsilon^\star(\Pi)$ and $\bar{\epsilon}(\Pi)$ of CBOB versus PGMORL, RA and MOIA over 10 runs with mean and standard deviation.

		CBOB		PGMORL		RA		MOIA	
		$\epsilon^\star(\Pi)$	$\bar{\epsilon}(\Pi)$	$\epsilon^\star(\Pi)$	$\bar{\epsilon}(\Pi)$	$\epsilon^\star(\Pi)$	$\bar{\epsilon}(\Pi)$	$\epsilon^\star(\Pi)$	$\bar{\epsilon}(\Pi)$
Ant-v2	f_1	**7.709(1.68E − 3)**	**7.880(3.32E − 2)**	8.383(5.07E−3)	8.898(7.39E−2)	7.795(2.40E−3)	8.506(6.13E−3)		
	f_2	**7.637(6.98E − 4)**	**7.751(1.65E − 4)**	8.042(9.23E−3)	8.322(1.57E−2)	8.287(6.99E−3)	8.965(1.66E−3)		
HalfCheetah-v2	f_1	**7.541(3.21E − 6)**	**7.542(7.53E − 6)**	7.554(1.84E−5)	8.584(7.46E−4)	7.542(1.14E−5)	8.583(4.29E−4)		
	f_2	**7.500(3.70E − 9)**	**7.508(1.17E − 4)**	7.506(2.31E−5)	7.729(2.61E−4)	7.504(2.44E−5)	7.670(4.61E−5)		
Hopper-v2	f_1	**6.221(1.14E − 2)**	**6.299(1.57E − 2)**	6.744(9.76E−4)	6.891(1.25E−1)	6.467(3.78E−2)	7.270(9.95E−2)		
	f_2	**4.587(2.25E − 2)**	**4.972(3.11E − 2)**	6.160(8.05E−2)	7.305(6.48E−2)	4.903(3.99E−2)	5.910(1.48E−3)		
Humanoid-v2	f_1	**2.362(2.51E − 2)**	**2.583(1.49E − 1)**	3.796(1.19E−2)	5.066(2.95E−2)	4.719(9.07E−2)	5.783(1.80E−2)		
	f_2	**4.099(6.96E − 7)**	**4.143(9.79E − 5)**	4.393(2.76E−2)	5.06(6.93E−3)	4.451(2.06E−2)	4.67(1.54E−3)		
Swimmer-v2	f_1	**9.747(1.12E − 4)**	**9.749(1.49E − 4)**	9.753(1.25E−5)	9.842(9.00E−5)	9.756(4.32E−4)	9.913(4.84E−4)		
	f_2	**9.850(1.00E − 12)**	**9.850(3.67E − 12)**	**9.850(1.08E − 7)**	9.894(1.80E−6)	9.852(1.11E−6)	9.898(2.63E−5)		
Walker2d-v2	f_1	**8.048(9.77E − 4)**	**8.116(3.95E − 3)**	8.189(1.10E−2)	8.870(2.90E−2)	8.297(5.48E−3)	9.110(1.23E−2)		
	f_2	**7.500(3.21E − 8)**	**7.506(8.84E − 5)**	**7.500(1.23E − 7)**	7.786(3.84E−4)	7.502(8.30E−6)	7.835(2.54E−3)		
Hopper-v3	f_1	**6.020(8.91E − 8)**	**6.203(1.44E − 3)**	6.169(1.74E−3)	7.56(4.22E−3)	6.283(2.37E−3)	8.331(1.55E−2)		
	f_2	**4.346(2.02E − 3)**	**4.539(1.97E − 2)**	4.825(1.89E−3)	7.131(3.34E−3)	5.017(4.73E−3)	8.428(2.40E−2)		
	f_3	**7.500(1.90E − 8)**	**7.502(1.23E − 5)**	**7.500(7.84E − 8)**	8.201(4.15E−3)	7.501(6.27E−7)	7.984(3.80E−4)		
Microgrid	f_1	**1.149(8.85E − 4)**	**1.846(3.88E − 5)**					1.845(1.81E−3)	3.902(2.56E−3)
	f_2	**0.000(4.87E − 7)**	**0.511(3.41E − 6)**					1.001(7.35E−7)	5.051(4.09E−7)
	f_3	**0.030(9.78E − 8)**	**0.050(5.74E − 8)**					0.061(2.14E−7)	0.090(5.22E−8)

5.1 Comparison with Conventional MORL

We first compare our proposed CBOB against three conventional MORL algorithms, which do not consider preferences. Note that since RA and PGMORL were merely designed for the MuJoCo environment while MOIA was deliberately designed for the MMSD environment, we only compare with them on their dedicated environment, respectively. To have a better visual interpretation, we plot the non-dominated policies found by different algorithms in Fig. 3. Overall, the comparison results can be classified into two categorizes. The first category is on Ant-v2, Swimmer-v2, Walker2d-v2, and Hopper-v3. The non-dominated policies obtained by RA and PGMORL approximate a wide range of objective space, while the policies found by our proposed CBOB are notably biased towards the DM's specified objectives, i.e., the ROI. This explains the comparable results of PBMORL regarding RA and PGMORL on $\epsilon^\star(\Pi)$ shown in Table 3 where CBOB and PGMORL even achieve the same mean $\epsilon^\star(\Pi)$ values.

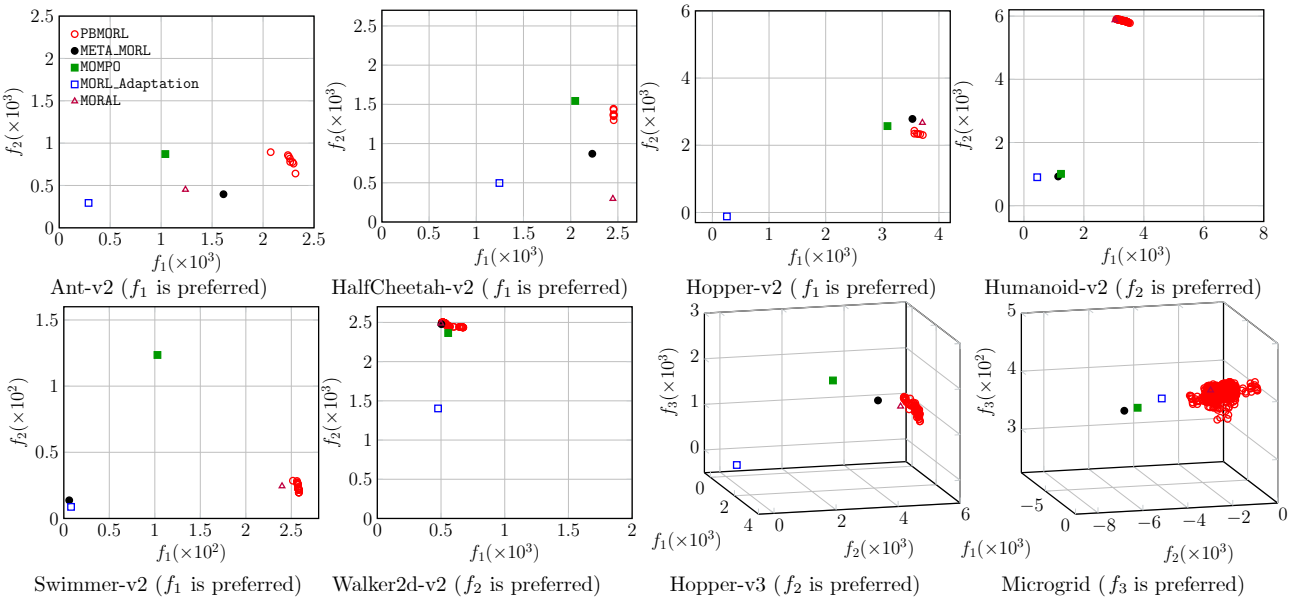

Figure 4: Selected plots of non-dominated policies obtained by `CBOB` vs `MORL-Adaptation`, `META-MORL`, `MOMPO` and `MORAL`.

However, if we refer to the $\bar{\epsilon}(\Pi)$, we can see that `CBOB` is significantly better, indicating that the average performance of `CBOB` in approximating the policies of interest outperforms `RA` and `PGMORL`. This is expected, as both `RA` and `PGMORL` identify too many policies outside the ROI. As for the other cases, i.e., `HalfCheetah-v2`, `Hopper-v2`, and `Humanoid-v2`, neither `RA` nor `PGMORL` can find the complete PF. In contrast, `CBOB` can discover reasonable non-dominated policies that meet the DM's preferred objectives. Consequently, as the comparison results of $\epsilon^\star(\Pi)$ and $\bar{\epsilon}(\Pi)$ shown in Table 3, we can see that `CBOB` are consistently the best. As for the MMSD environment shown in Fig. 3, we can see that the policies obtained by `MOIA` are completely dominated by those found by `CBOB`. This observation is also supported by the superior performance of $\epsilon^\star(\Pi)$ and $\bar{\epsilon}(\Pi)$ shown in Table 3.

5.2 Comparison with Preference-based MORL

Next, we compare `CBOB` with four other preference-based MORL algorithms. Note that since these peer algorithms are designed to find only one preferred policy, we only evaluate the performance on $\epsilon^\star(\Pi)$ and the corresponding comparison results are shown in Table 4. From these results, we can see that `CBOB` is the most competitive algorithm in all test cases, though it achieves the same $\epsilon^\star(\Pi)$ as some peer algorithms on selected cases. To have a better visual interpretation, like Section 5.3.1, we plot the non-dominated policies obtained by different algorithms. Due to the page limit, we only show one type of preference while the full results can be found in Fig. 9 of the Appendix. From these plots, we again divide the performance into two categorizes. For the first category, including `Ant-v2`, `Hopper-v3`, `Humanoid-v2`, and `Microgrid`, it is evident that `CBOB` consistently outperforms its peers. Notably, all four preference-based MORL algorithms can only find one policy, which is dominated by nearly all trade-off alternative policies identified by `CBOB`. For the other problems, the performance of `CBOB` is comparable to some of the other peers. This can be explained by the fact that the PFs of such problems are relatively easier to approximate, thus allowing for better exploration towards the region of interest therein.

5.3 Further Analysis

The previous experiments demonstrated the effectiveness of `CBOB` for finding the policies of interest according to the DM's preferences. In this subsection, we plan to further analyze several core algorithmic components by addressing the following five research questions (RQs).

Table 4: Comparison results of $\epsilon^\star(\Pi)$ of CBOB versus MOMPO, META-MORL, MORL-Adaptation and MORAL over 10 runs with mean and standard deviation.

		PBMORL	MOMPO	META-MORL	MORL-Adaptation	MORAL
Ant-v2	f_1	**7.709(1.68E − 3)**	8.910(1.81E−2)	8.355(1.36E−2)	9.686(8.53E−3)	8.926(1.18E−2)
	f_2	**7.637(6.98E − 4)**	8.914(2.05E−3)	8.446(4.52E−3)	9.633(3.37E−3)	8.377(2.49E−3)
HalfCheetah-v2	f_1	**7.541(3.21E − 6)**	7.953(1.05E−4)	7.752(1.77E−4)	8.743(1.52E−4)	7.545(9.73E−5)
	f_2	**7.500(3.70E − 9)**	**7.500(7.72E − 8)**	7.980(4.57E−8)	8.561(1.55E−7)	**7.500(5.66E − 8)**
Hopper-v2	f_1	**6.221(1.14E − 2)**	6.828(3.60E−3)	6.404(3.54E−3)	9.662(2.32E−3)	6.236(2.42E−3)
	f_2	**4.587(2.250E − 2)**	8.583(2.31E−2)	4.930(4.68E−1)	9.942(6.12E−2)	9.761(7.94E−1)
Humanoid-v2	f_1	**2.362(2.51E − 2)**	9.292(4.17E−1)	8.679(5.18E−2)	9.554(8.33E−2)	5.599(6.40E−2)
	f_2	**4.099(6.96E − 7)**	8.993(1.84E−2)	9.083(2.84E−3)	9.912(8.34E−2)	9.776(1.34E−2)
Swimmer-v2	f_1	**9.747(1.12E − 4)**	9.913(6.88E−4)	9.933(5.58E−3)	9.941(1.87E−3)	9.759(2.99E−5)
	f_2	**9.850(1.00E − 12)**	**9.850(2.87E − 9)**	9.860(3.34E−7)	**9.850(1.09E − 8)**	**9.850(7.22E − 9)**
Walker2d-v2	f_1	**8.048(9.77E − 4)**	8.905(5.78E−3)	8.267(1.26E−2)	9.271(4.22E−3)	8.850(6.31E-3)
	f_2	**7.500(3.21E − 8)**	7.643(6.78E−3)	7.528(2.39E−6)	8.602(6.61E−3)	**7.500(1.92E − 7)**
Hopper-v3	f_1	**6.020(8.91E − 8)**	6.821(2.91E−7)	6.773(3.51E−5)	9.070(8.77E−4)	6.391(6.98E−6)
	f_2	**4.346(2.02E − 3)**	7.295(1.72E−3)	6.032(3.55E−3)	9.411(3.87E−2)	5.263(9.21E−3)
	f_3	**7.500(1.90E − 8)**	**7.500(2.63E − 6)**	7.522(3.96E−6)	**7.500(5.47E − 7)**	**7.500(4.54E − 7)**
Microgrid	f_1	**1.149(8.85E − 4)**	2.418(3.39E−3)	2.737(9.05E−3)	4.69(7.22E−3)	2.48(6.72E−4)
	f_2	**0.000(4.87E − 7)**	1.366(9.02E−4)	1.317(8.62E−4)	7.020(2.31E−4)	0.672(5.56E−5)
	f_3	**0.030(9.78E − 8)**	0.102(4.57E−6)	0.071(8.76E−5)	0.244(9.94E−4)	0.124(1.29E−5)

RQ1: When do we consult the DM?

RQ2: What is impact of the interaction frequency?

RQ3: What is the benefit of our preference learning?

RQ4: What is the impact of the parameters κ_1 and κ_2 in the `preference translation` step?

RQ5: What if the DM's preference is not determinstic?

RQ6: What if we use other the aggregation function other than the linear aggregation in equation (2)?

5.3.1 Investigation on RQ1

In addition to the *interactive* preference elicitation considered in CBOB, DM's preferences can also be incorporated in *a priori* or *a posteriori* manner. From the comparison results discussed in Section 5.1, we can see that the a posteriori method may fail to identify the policy of interest in some of the challenging problems. The better performance of CBOB against the other four peer preference-based MORL, as shown in Section 5.2, demonstrate that the interactive MORL outperforms the a priori method. To have a better understanding of the discrepancy between the a priori and interactive preference elicitation, we plot the corresponding weight vector a priori specified by the DM versus those identified by CBOB, while the full results can be found in Fig. 10 of the Appendix. From these plots, we can see that the weight vector specified by the DM is always outside the region of interest (ROI). From the DM's perspective, it is not intuitive for the DM to elicit an appropriate weight vector a priori given the black-box nature of the problem itself. Our experiments in Section 5.3.1 demonstrate that there is even no guarantee to find good non-dominated policies without considering the DM's preference information before a posteriori decision-making.

5.3.2 Investigation on RQ2

In the `consultation` stage, more frequent interactions with the DM would generate more oracles for preference learning. However, this also significantly increases the DM's workload. To address RQ2, we set the number of interactions to 20 and compared this with 40 and 10 consultations. From the selected results shown in Fig. 6, while full results can be found in Fig. 11 of the Appendix, we observe that

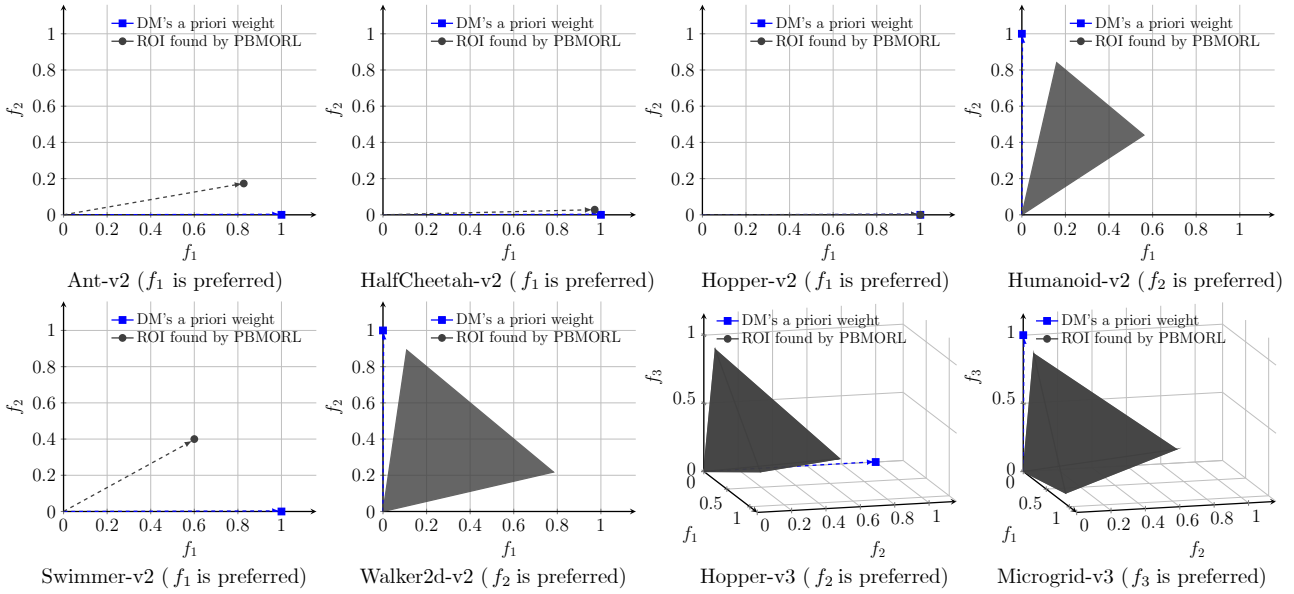

Figure 5: Comparison of the weight vector specified by the DM *a priori* versus the ROI identified by CBOB (shaded in the gray region).

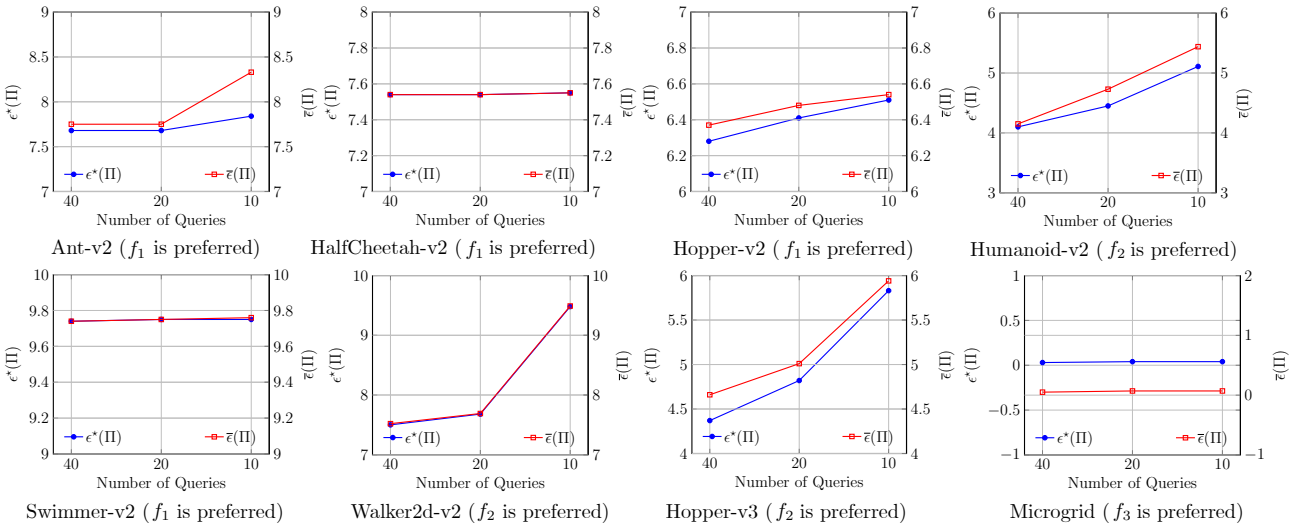

Figure 6: Comparison results of $\epsilon^\star(\Pi)$ and $\bar{\epsilon}(\Pi)$ obtained by CBOB with 10, 20, and 40 interactions, respectively.

reducing the number of interactions does not negatively impact CBOB's performance on some problems like HalfCheetah-v2 with a preference for f_2. However, for other problems like Walker2d-v2 with a preference for f_2, decreasing the number of queries to 10 can significantly hurt CBOB's performance. On the other hand, increasing the number of interactions does not bring significantly better performance but can increase the DM's cognitive load.

5.3.3 Investigation on RQ3

In principle, any off-the-shelf preference learning methods can be used for modeling the DM's preference information. To validate this assertion, we construct two variants of CBOB, dubbed RSMORL and WLMORL. They replace the preference learning method introduced in Section 3.2.2 with a classic preference learning approach ranking-SVM [49] and a preference learning method in RL [32], respectively. Note that all variants apply one query at each round of policy optimization, as done in CBOB. The comparison results shown in Table 5 demonstrate the better performance of CBOB on most problems.

Table 5: Comparison results of $\epsilon^\star(\Pi)$ and $\bar{\epsilon}(\Pi)$ of CBOB versus RSMORL and WLMORL over 10 runs with mean and standard deviation.

		PBMORL		RSMORL		WLMORL	
		$\epsilon^\star(\Pi)$	$\bar{\epsilon}(\Pi)$	$\epsilon^\star(\Pi)$	$\bar{\epsilon}(\Pi)$	$\epsilon^\star(\Pi)$	$\bar{\epsilon}(\Pi)$
Ant-v2	f_1	**7.709(1.68E − 3)**	**7.880(3.32E − 2)**	8.333(1.44E−5)	8.796(6.22E−3)	7.746(1.35E−5)	7.949(5.02E−3)
	f_2	**7.637(6.98E − 4)**	**7.751(1.65E − 4)**	7.722(1.24E−5)	7.998(5.72E−3)	7.695(8.74E−7)	7.956(2.23E−4)
HalfCheetah-v2	f_1	**7.541(3.21E − 6)**	**7.542(7.53E − 6)**	7.566(3.71E−4)	7.721(5.29E−4)	8.397(1.31E−4)	9.233(8.53E−2)
	f_2	**7.500(3.70E − 9)**	**7.508(1.17E − 4)**	**7.500(2.47E − 7)**	7.541(2.14E−6)	**7.508(7.98E − 3)**	7.544(8.98E−2)
Hopper-v2	f_1	**6.221(1.14E − 2)**	**6.299(1.57E − 2)**	9.566(8.17E−5)	9.642(5.58E−2)	9.421(6.43E−7)	9.631(5.16E−3)
	f_2	**4.587(2.25E − 2)**	**4.972(3.11E − 2)**	8.351(5.29E−5)	8.635(3.68E−2)	9.922(3.11E−5)	9.730(6.21E−3)
Humanoid-v2	f_1	**2.362(2.51E − 2)**	**2.583(8.31E − 1)**	6.324(5.21E−5)	6.543(5.19E−3)	9.236(5.66E−5)	9.353(6.74E−2)
	f_2	**4.099(6.96E − 7)**	**4.143(9.79E − 5)**	4.818(5.89E−5)	5.076(2.29E−2)	8.718(1.39E−6)	8.728(5.21E−2)
Swimmer-v2	f_1	**9.747(1.12E − 4)**	**9.749(1.49E − 4)**	9.716(6.23E−5)	9.762(7.22E−3)	9.884(7.22E−3)	9.894(2.39E−3)
	f_2	**9.850(1.00E − 12)**	**9.850(3.67E − 12)**	**9.850(5.39E − 8)**	**9.850(1.12E − 7)**	**9.850(8.84E − 7)**	**9.850(6.23E − 6)**
Walker2d-v2	f_1	**8.048(9.77E − 4)**	**8.116(3.95E − 3)**	8.636(5.13E−5)	8.928(7.11E−2)	8.609(4.26E−5)	8.897(9.21E−3)
	f_2	**7.500(3.21E − 8)**	**7.506(8.84E − 5)**	**7.500(6.64E − 6)**	7.534(8.34E−4)	7.600(1.14E−5)	7.673(9.96E−3)
Hopper-v3	f_1	6.020(8.91E−8)	**6.203(1.44E − 3)**	**5.981(5.12E − 5)**	6.250(9.75E−3)	6.858(6.75E−6)	7.736(7.34E−3)
	f_2	**4.346(2.02E − 3)**	**4.539(1.97E − 2)**	5.000(2.66E−5)	5.272(9.74E−4)	8.611(1.74E−6)	9.657(8.72E−4)
	f_3	**7.500(1.90E − 8)**	**7.502(1.23E − 5)**	**7.500(2.74E − 4)**	7.510(8.82E−3)	**7.500(1.08E − 6)**	**7.500(4.34E − 3)**
Microgrid	f_1	**1.149(8.85E − 4)**	**1.846(3.88E − 5)**	**1.149(1.34E − 3)**	2.382(8.87E−2)	**1.149(5.88E − 5)**	2.397(1.74E−3)
	f_2	**0.000(4.87E − 7)**	**0.511(3.41E − 6)**	**0.000(2.62E − 5)**	1.558(6.62E−3)	**0.000(2.34E − 5)**	1.045(8.78E−4)
	f_3	**0.030(9.78E − 8)**	**0.050(5.74E − 8)**	**0.030(1.22E − 5)**	0.081(8.87E−7)	**0.030(3.54E − 6)**	0.074(3.79E−4)

Table 6: Comparison results of $\epsilon^\star(\Pi)$ and $\bar{\epsilon}(\Pi)$ on different settings of κ_1 and κ_2 over 10 runs with mean and standard deviation.

		$\kappa_1 = 100\% \times \lvert\Pi\rvert, \kappa_2 = 0$		$\kappa_1 = 80\% \times \lvert\Pi\rvert, \kappa_2 = 20\% \times \lvert\Pi\rvert$		$\kappa_1 = 50\% \times \lvert\Pi\rvert, \kappa_2 = 50\% \times \lvert\Pi\rvert$		$\kappa_1 = 20\% \times \lvert\Pi\rvert, \kappa_2 = 80\% \times \lvert\Pi\rvert$	
		$\epsilon^\star(\Pi)$	$\bar{\epsilon}(\Pi)$	$\epsilon^\star(\Pi)$	$\bar{\epsilon}(\Pi)$	$\epsilon^\star(\Pi)$	$\bar{\epsilon}(\Pi)$	$\epsilon^\star(\Pi)$	$\bar{\epsilon}(\Pi)$
HalfCheetah-v2	f_1	**7.541(7.65E − 8)**	**7.542(6.34E − 7)**	**7.541(3.21E − 6)**	**7.542(7.53E − 6)**	7.550(6.24E-8)	7.554(7.78E−7)	7.551(8.75E−8)	7.562(2.56E−7)
	f_2	**7.500(6.76E − 10)**	**7.508(3.19E − 9)**	**7.500(3.70E − 9)**	**7.508(1.17E − 8)**	**7.500(4.42E − 9)**	7.522(8.12E−4)	**7.500(6.24E − 6)**	7.534(3.28E−6)
Swimmer-v2	f_1	9.761(6.79E−5)	9.762(3.78E−4)	**9.747(1.12E − 4)**	**9.749(1.49E − 4)**	9.770(5.63E−4)	9.773(3.29E−5)	9.763(6.79E−5)	9.768(1.12E−5)
	f_2	**9.850(4.55E − 10)**	**9.850(7.87E − 11)**	**9.850(1.00E − 12)**	**9.850(3.67E − 12)**	**9.850(2.56E − 10)**	**9.850(7.73E − 10)**	**9.850(6.66E − 9)**	**9.850(8.34E − 9)**
Walker2d-v2	f_1	8.612(2.37E−4)	8.652(6.17E−4)	**8.048(9.77E − 4)**	**8.116(3.95E − 3)**	8.623(8.13E−3)	8.738(6.32E−2)	8.513(7.93E−4)	8.671(5.35E−3)
	f_2	**7.500(6.61E − 6)**	7.593(6.25E−5)	**7.500(3.21E − 8)**	**7.506(8.84E − 5)**	**7.500(7.27E − 6)**	7.510(3.90E−4)	**7.500(1.22E − 5)**	7.516(8.28E−4)

Specifically, comparing to RSMORL, our better performance not only highlights the superiority of GP for preference learning, but also showcases the benefits of uncertainty provided by GP prediction for better exploration. Comparing to WLMORL as well as MORAL, our better performance demonstrates that learning preferred weight vector(s) is less reliable than learning the DM's preference in the objective space.

5.3.4 Investigation on RQ4

In the `preference translation` step, there are two hyperparameters κ_1 and κ_2 that implicitly control the balance between exploration versus exploitation. Specifically, a larger κ_1 indicates a greater reliance on learned preference information to guide policy optimization, while a larger κ_2 emphasizes random exploration. We set $\kappa_1 = 80\% \times |\Pi|$ and $\kappa_2 = 20\% \times |\Pi|$ as the default. To address RQ4, here we empirically compare the the default setting with three other $\kappa_1 \in \{100\% \times |\Pi|, 50\% \times |\Pi|, 20\% \times |\Pi|\}$ on three example problems, including `Walker2d-v2`, `HalfCheetah-v2`, and `Swimmer-v2`. From the comparison results in Table 6, we find that the setting with $\kappa_1 = 80\% \times |\Pi|$ and $\kappa_2 = 20\% \times |\Pi|$ consistently achieves the best performance when considering $\epsilon^\star(\Pi)$. On the other hand, when considering $\bar{\epsilon}(\Pi)$, the outcomes depend more on the characteristics of the individual problems. In general, we find that a large κ_1 and a nonzero κ_2 are efficient for most situations.

5.3.5 Investigation on RQ5

The DM's preferences discussed so far in our experiments are all *deterministic*. That is to say DMs are assumed to prefer only one objective function over the other(s). However, it is not uncommon that DMs can be blurry about their preference. For instance, the DM can be more into one objective (say 70%), but she also gives certain level of priority (say the remaining 30%) to the other objective(s).

Here, we conduct an experiment that considers a *fuzzy* type of preference. From the plots of the non-dominated policies found by different types of preference settings in Fig. 7, we find that our proposed CBOB can also be used to find trade-off policies with a controllable bias towards one of the objectives, instead of a polarized preference. However, we also find that the trade-off policies found by CBOB in Ant-v2 are further polarized in the less preferred objective. This can be attributed to the intriguing interaction of two objectives in Ant-v2 or the difficulty of CBOB when tackling this environment. All in all, it is an interesting and important future direction to investigate more diversified types of preference elicitation methods under the CBOB framework.

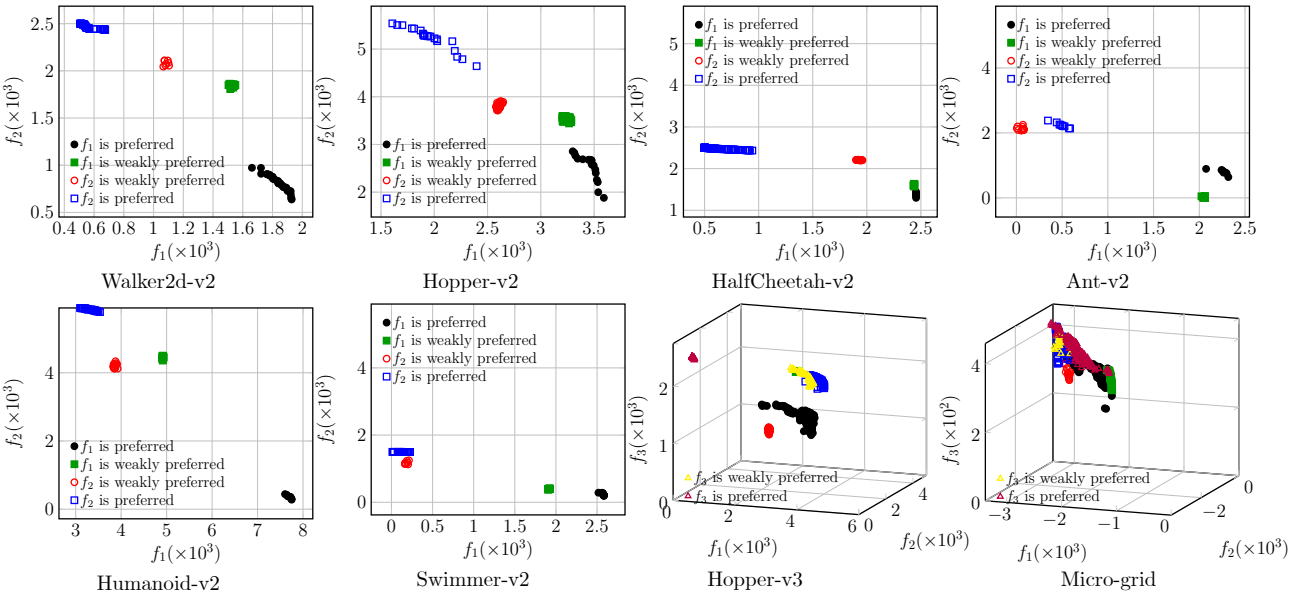

Figure 7: Selected plots of the non-dominated policies obtained by CBOB with different types of preference settings. One is *deterministic* preference on one objective while the other is a *fuzzy* type of preference.

5.3.6 Investigation on RQ6

As one of the key components of CBOB, the `seeding` module works as a conventional MORL to search for a set of promising trade-off policies that approximate the PF. While we applied a linear aggregation in CBOB for a proof-of-concept purpose, it has been notorious in the multi-objective optimization domain (e.g., [24, 36, 50]) that equation (2) can be ineffective to search for solutions located in the non-convex region(s) of the PF. In contrast, the following weighted Tchebycheff aggregation function has been widely recognized to be applicable to problems with both convex and non-convex regions in the PF:

$$\tilde{J}(\pi, \mathbf{w}, \mathbf{z}) = \max_{i \in \{1,\dots,m\}} \{w_i | J_i(\pi) - z_i|\}, \tag{16}$$

where $\mathbf{z} = (z_1, \cdots, z_m)^\top$ is the utopia point. Note that all objective functions in this paper are considered being maximized, whereas there is no *a priori* knowledge about the maximum of each objective function. In this case, we set $\mathbf{z}$ as the nadir point instead, i.e., $z_i = 0$, $i \in \{1, \dots, m\}$. To investigate RQ6, we replace equation (2) as equation (16) in the `seeding` module of CBOB to constitute a variant dubbed PBMORL-TCH. Note that our proposed CBOB is a general framework that each component can be adapted to any other techniques in a plug-in manner. From the comparison results of $\epsilon^\star(\Pi)$ and $\bar{\epsilon}(\Pi)$ shown in Table 7 along with the non-dominated policies shown in Fig. 8, we can see that the performance of CBOB and PBMORL-TCH is close to each other. This can be explained as the PF of the MORL problems considered here are all with convex PFs. The robust performance PBMORL-TCH also provides us confidence to extend our proposed CBOB framework for handling problems in more complex environments.

Table 7: Comparison results of $\epsilon^\star(\Pi)$ and $\bar{\epsilon}(\Pi)$ of CBOB against PBMORL-TCH over 10 runs with mean and standard deviation.

		PBMORL-TCH		PBMORL	
		$\epsilon^\star(\Pi)$	$\bar{\epsilon}(\Pi)$	$\epsilon^\star(\Pi)$	$\bar{\epsilon}(\Pi)$
Ant-v2	f_1	**7.738(6.41E − 3)**	**7.866(5.09E − 2)**	7.709(1.68E−3)	7.880(3.32E−2)
	f_2	7.700(3.70E−3)	7.87(2.06E−2)	**7.637(6.98E − 4)**	**7.751(1.65E − 4)**
HalfCheetah-v2	f_1	7.543(2.00E−6)	7.591(1.06E−2)	**7.541(3.21E − 6)**	**7.542(7.53E − 6)**
	f_2	7.501(3.23E−7)	7.510(2.63E−5)	**7.500(3.70E − 9)**	**7.508(1.17E − 4)**
Hopper-v2	f_1	**6.203(3.71E − 3)**	6.301(1.13E−2)	6.221(1.14E−2)	**6.299(1.57E − 2)**
	f_2	4.700(4.34E−2)	**4.904(1.04E − 1)**	**4.587(2.25E − 2)**	4.972(3.11E−2)
Humanoid-v2	f_1	**2.230(3.64E − 4)**	**2.289(3.39E − 5)**	2.362(2.51E−2)	2.583(1.49E−1)
	f_2	4.224(3.55E−3)	4.284(2.72E−3)	**4.099(6.96E − 7)**	**4.143(9.79E − 5)**
Swimmer-v2	f_1	9.751(7.69E−5)	9.752(1.04E−4)	**9.747(1.12E − 4)**	**9.749(1.49E − 4)**
	f_2	**9.850(3.13E − 12)**	**9.850(1.04E − 7)**	**9.850(1.00E − 12)**	**9.850(3.67E − 12)**
Walker2d-v2	f_1	**7.922(1.76E − 1)**	**8.067(1.86E − 1)**	8.048(9.77E−4)	8.116(3.95E−3)
	f_2	**7.500(4.74E − 6)**	7.525(3.65E−4)	**7.500(3.21E − 8)**	**7.506(8.84E − 5)**
Hopper-v3	f_1	6.360(5.29E−2)	6.582(6.21E−2)	**6.020(8.91E − 8)**	**6.203(1.44E − 3)**
	f_2	**4.333(1.58E − 1)**	**4.434(4.21E − 4)**	4.346(2.02E−3)	4.539(1.97E−2)
	f_3	**7.500(5.60E − 6)**	7.506(2.05E−5)	**7.500(1.90E − 8)**	**7.502(1.23E − 5)**
Microgrid	f_1	1.169(3.96E−1)	**1.206(5.20E − 4)**	**1.149(8.85E − 4)**	1.846(3.88E−5)
	f_2	0.016(4.94E−4)	**0.086(1.13E − 2)**	**0.000(4.87E − 7)**	0.511(3.41E−6)
	f_3	0.039(5.00E−7)	**0.047(2.45E − 5)**	**0.030(9.78E − 8)**	0.050(5.74E−8)

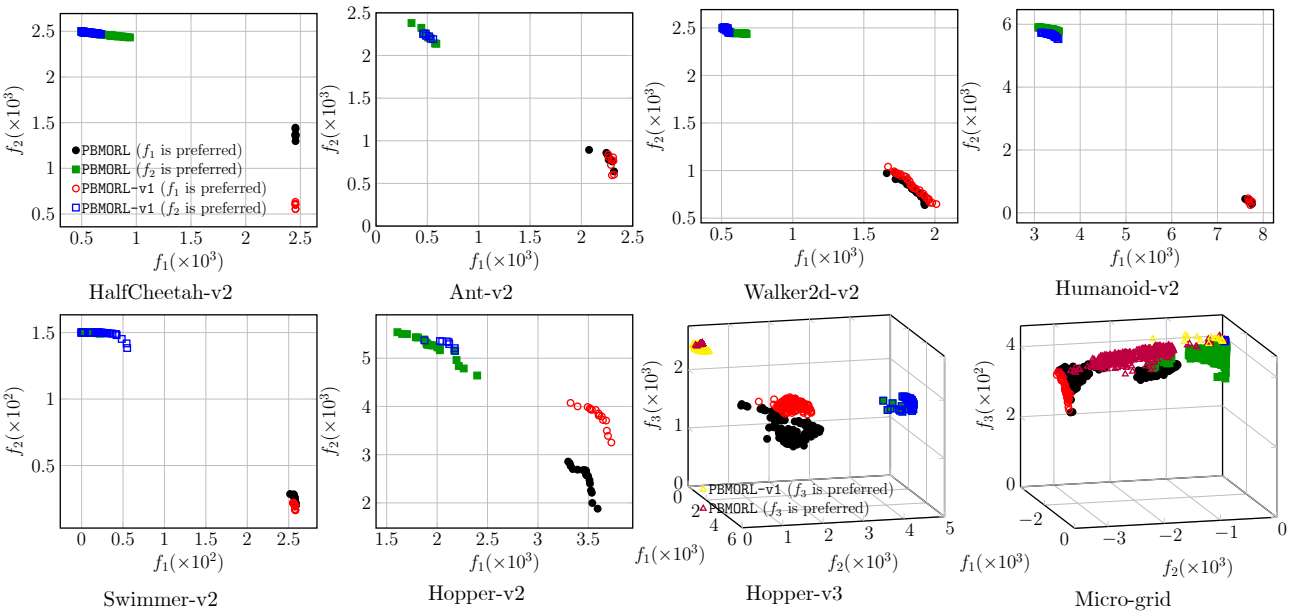

Figure 8: Plots of the non-dominated policies obtained by CBOB versus PBMORL-TCH with different preferences on each objective.

6 Conclusion and Future Directions

This paper proposed a human-in-the-loop framework for preference-based MORL that searches for policies of interest preferred by the DM. This framework proactively learns the DM's preferences in an interactive manner, using the learned preference information to guide policy optimization in MORL. It is worth noting that our proposed CBOB is highly versatile, as all its algorithmic components can be replaced by other related techniques in a plug-in manner. Extensive experiments on the MuJoCo and MMSD environments fully demonstrate the effectiveness of our proposed CBOB for finding the policies of interest.

Human-in-the-loop interactive MORL presents a promising paradigm for realizing human-AI collaboration. However, the field is far from mature, and numerous issues warrant exploration in the future. For instance, this paper assumes that the information provided by MORL is fully understand-

able by the DM, which may not be realistic, particularly when dealing with more than three objective functions. Developing a human-computer interaction platform and mechanism is essential for enhancing the effectiveness of interactive MORL. Furthermore, we assume that the DM's preferences remain consistent throughout the MORL process. Proactively detecting and adapting to changes in the DM's preferences in dynamic and uncertain environments pose a significant challenge. Finally, the explainability of the policies of interest and their implications has rarely been discussed in the literature, representing another area for future investigation.

Acknowledgment

This work was supported in part by the UKRI Future Leaders Fellowship under Grant MR/S017062/1 and MR/X011135/1; in part by NSFC under Grant 62376056 and 62076056; in part by the Royal Society under Grant IES/R2/212077; in part by the EPSRC under Grant 2404317; in part by the Kan Tong Po Fellowship (KTP\R1\231017); and in part by the Amazon Research Award and Alan Turing Fellowship.

References

[1] Y. Qin, H. Wang, S. Yi, X. Li, and L. Zhai, "An energy-aware scheduling algorithm for budget-constrained scientific workflows based on multi-objective reinforcement learning," *J. Supercomput.*, vol. 76, no. 1, pp. 455–480, 2020.

[2] J. Xu, K. Li, and M. Abusara, "Multi-objective reinforcement learning based multi-microgrid system optimisation problem," in *EMO'21: Proc. of 11th International Conference on Evolutionary Multi-Criterion Optimization*, vol. 12654. Springer, 2021, pp. 684–696.

[3] J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik, "Prediction-guided multi-objective reinforcement learning for continuous robot control," in *ICML'20: Proc. of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 10 607–10 616.

[4] C. F. Hayes, R. Radulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz, E. Howley, A. A. Irissappane, P. Mannion, A. Nowé, G. de Oliveira Ramos, M. Restelli, P. Vamplew, and D. M. Roijers, "A practical guide to multi-objective reinforcement learning and planning," *Auton. Agents Multi Agent Syst.*, vol. 36, no. 1, p. 26, 2022.

[5] Z. Gábor, Z. Kalmár, and C. Szepesvári, "Multi-criteria reinforcement learning," in *ICML'98: Proc. of the 15th International Conference on Machine Learning*. Morgan Kaufmann, 1998, pp. 197–205.

[6] S. Mannor and N. Shimkin, "The steering approach for multi-criteria reinforcement learning," in *NIPS'01: Proc. of 14th Annual Conference on Neural Information Processing Systems*. MIT Press, 2001, pp. 1563–1570.

[7] S. Natarajan and P. Tadepalli, "Dynamic preferences in multi-criteria reinforcement learning," in *ICML'05: Proc. of the 22nd International Conference on Machine Learning*, ser. ACM International Conference Proceeding Series, vol. 119. ACM, 2005, pp. 601–608.

[8] R. Yang, X. Sun, and K. Narasimhan, "A generalized algorithm for multi-objective reinforcement learning and policy adaptation," in *NeurIPS'19: Proc. of the 2019 Annual Conference on Neural Information Processing Systems 2019*, 2019, pp. 14 610–14 621.

[9] A. Ikenaga and S. Arai, "Inverse reinforcement learning approach for elicitation of preferences in multi-objective sequential optimization," in *ICA'18: Proc. of the 2018 IEEE International Conference on Agents*. IEEE, 2018, pp. 117–118.

[10] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker, "Empirical evaluation methods for multiobjective reinforcement learning algorithms," *Mach. Learn.*, vol. 84, no. 1-2, pp. 51–80, 2011.

[11] D. M. Roijers, L. M. Zintgraf, and A. Nowé, "Interactive thompson sampling for multi-objective multi-armed bandits," in *ADT'17: Proc. of 5th International Conference on Algorithmic Decision Theory*, J. Rothe, Ed., vol. 10576. Springer, 2017, pp. 18–34.

[12] D. M. Roijers, L. M. Zintgraf, P. Libin, and A. Nowé, "Interactive multi-objective reinforcement learning in multi-armed bandits for any utility function," in *ALA workshop at FAIM*, vol. 8, 2018.

[13] M. Peschl, A. Zgonnikov, F. A. Oliehoek, and L. C. Siebert, "MORAL: aligning AI with human norms through multi-objective reinforced active learning," in *AAMAS'22: Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems*, 2022, pp. 1038–1046.

[14] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IROS'12: Proc. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[15] W. Chiu, H. Sun, and H. V. Poor, "A multiobjective approach to multimicrogrid system design," *IEEE Trans. Smart Grid*, vol. 6, no. 5, pp. 2263–2272, 2015.

[16] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *J. Artif. Intell. Res.*, vol. 48, pp. 67–113, 2013.

[17] D. M. Roijers and S. Whiteson, *Multi-Objective Decision Making*, ser. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2017.

[18] R. Radulescu, P. Mannion, D. M. Roijers, and A. Nowé, "Multi-objective multi-agent decision making: A utility-based analysis and survey," in *AAMAS'20: Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems*, 2020, pp. 2158–2160.

[19] M. Rolf, "The need for MORE: need systems as non-linear multi-objective reinforcement learning," in *ICDL-EpiRob'20: Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics*. IEEE, 2020, pp. 1–8.

[20] A. Abdolmaleki, S. H. Huang, L. Hasenclever, M. Neunert, H. F. Song, M. Zambelli, M. F. Martins, N. Heess, R. Hadsell, and M. A. Riedmiller, "A distributional view on multi-objective policy optimization," in *ICML'20: Proc. of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 11–22.

[21] H. Mossalam, Y. M. Assael, D. M. Roijers, and S. Whiteson, "Multi-objective deep reinforcement learning," *CoRR*, vol. abs/1610.02707, 2016. [Online]. Available: http://arxiv.org/abs/1610.02707

[22] M. Reymond and A. Nowé, "Pareto-DQN: Approximating the Pareto front in complex multi-objective decision problems," in *ALA'19: Proc. of the Adaptive and Learning Agents Workshop at AAMAS*, 2019.

[23] S. Parisi, M. Pirotta, N. Smacchia, L. Bascetta, and M. Restelli, "Policy gradient approaches for multi-objective sequential decision making: A comparison," in *ADPRL'14: Proc. of 2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*. IEEE, 2014, pp. 1–8.

[24] K. Li, T. Zhang, and R. Wang, "Deep reinforcement learning for multiobjective optimization," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3103–3114, 2021.

[25] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz, "A survey of preference-based reinforcement learning methods," *J. Mach. Learn. Res.*, vol. 18, pp. 136:1–136:46, 2017.

[26] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *CoRR*, vol. abs/1606.06565, 2016. [Online]. Available: http://arxiv.org/abs/1606.06565

[27] H. Sugiyama, T. Meguro, and Y. Minami, "Preference-learning based inverse reinforcement learning for dialog control," in *INTERSPEECH'12: Proc. of the 13th Annual Conference of the International Speech Communication Association*. ISCA, 2012, pp. 222–225.

[28] X. Pan and Y. Shen, "Human-interactive subgoal supervision for efficient inverse reinforcement learning," in *AAMAS'18: Proc. of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018, pp. 1380–1387.

[29] D. S. Brown, W. Goo, P. Nagarajan, and S. Niekum, "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations," in *ICML'19: Proc. of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 783–792.

[30] D. S. Brown, W. Goo, and S. Niekum, "Better-than-demonstrator imitation learning via automatically-ranked demonstrations," in *CoRL'19: Proc. of the 3rd Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 100. PMLR, 2019, pp. 330–359.

[31] M. Kollmitz, T. Koller, J. Boedecker, and W. Burgard, "Learning human-aware robot navigation from physical interaction via inverse reinforcement learning," in *IROS'20: Proc. of 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2020, pp. 11 025–11 031.

[32] C. Wirth, J. Fürnkranz, and G. Neumann, "Model-free preference-based reinforcement learning," in *AAAI'16: Proc. of the 30th AAAI Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 2222–2228.

[33] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *NIPS'17: Proc. of 2017 Annual Conference on Neural Information Processing Systems*, 2017, pp. 4299–4307.

[34] D. M. Roijers, L. M. Zintgraf, P. Libin, and A. Nowé, "Interactive multi-objective reinforcement learning in multi-armed bandits for any utility function," in *ALA workshop at FAIM*, vol. 8, 2018.

[35] N. Wanigasekara, Y. Liang, S. T. Goh, Y. Liu, J. J. Williams, and D. S. Rosenblum, "Learning multi-objective rewards and user utility function in contextual bandits for personalized ranking," in *IJCAI'19: Proc. of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 3835–3841.

[36] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, 2007.

[37] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM J. Optim.*, vol. 8, no. 3, pp. 631–657, 1998.

[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[39] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *ICLR'16: Proc. of the 4th International Conference on Learning Representations*, 2016.

[40] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs," *J. Mach. Learn. Res.*, vol. 3, pp. 397–422, 2002.

[41] K. H. Wray, S. Zilberstein, and A. Mouaddib, "Multi-objective mdps with conditional lexicographic reward preferences," in *AAAI'15: Proc. of the 29th AAAI Conference on Artificial Intelligence*. AAAI Press, 2015, pp. 3418–3424.

[42] K. Li, R. Chen, D. A. Savic, and X. Yao, "Interactive decomposition multiobjective optimization via progressively learned value functions," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 5, pp. 849–860, 2019.

[43] L. M. Zintgraf, D. M. Roijers, S. Linders, C. M. Jonker, and A. Now, "Ordered preference elicitation strategies for supporting multi-objective decision making," in *AAMAS'18: Proc. of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 1477–1485.

[44] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005.

[45] W. Chu and Z. Ghahramani, "Preference learning with Gaussian processes," in *ICML'05: Proc. of Proceedings of the Twenty-Second International Conference on Machine Learning*, ser. ACM International Conference Proceeding Series, vol. 119. ACM, 2005, pp. 137–144.

[46] X. Chen, A. Ghadirzadeh, M. Björkman, and P. Jensfelt, "Meta-learning for multi-objective reinforcement learning," in *IROS'19: Proc. of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2019, pp. 977–983.

[47] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[48] K. Li, K. Deb, and X. Yao, "R-metric: Evaluating the performance of preference-based evolutionary multiobjective optimization using reference points," *IEEE Trans. Evol. Comput.*, vol. 22, no. 6, pp. 821–835, 2018.

[49] T. Joachims, "Optimizing search engines using clickthrough data," in *SIGKDD'02: Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2002, pp. 133–142.

[50] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

Appendix

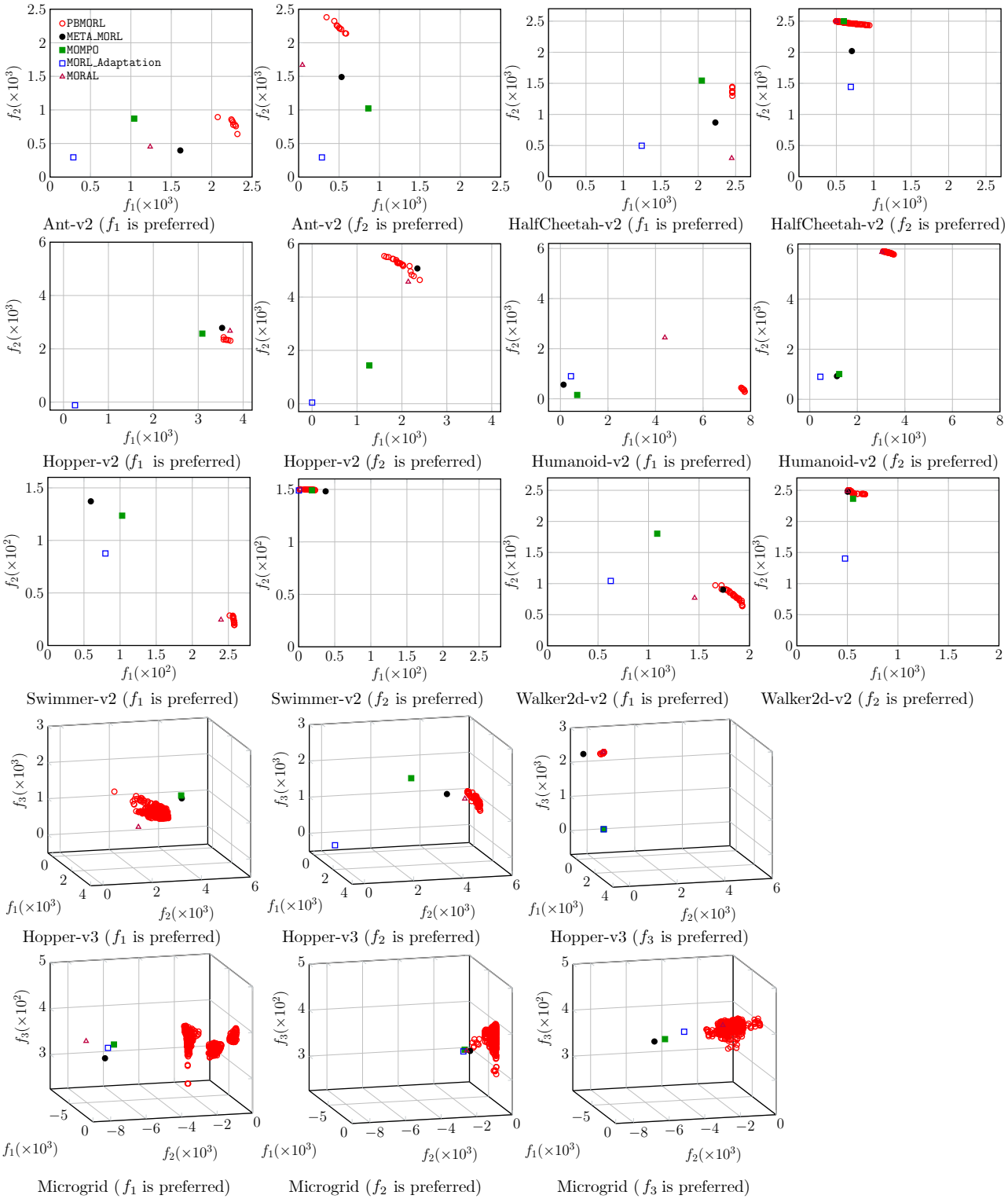

Figure 9: Plots of the non-dominated policies obtained by `CBOB` versus `MOMPO`, `META-MORL`, `MORL-Adaptation` and `MORAL` with different preferences on each objective.

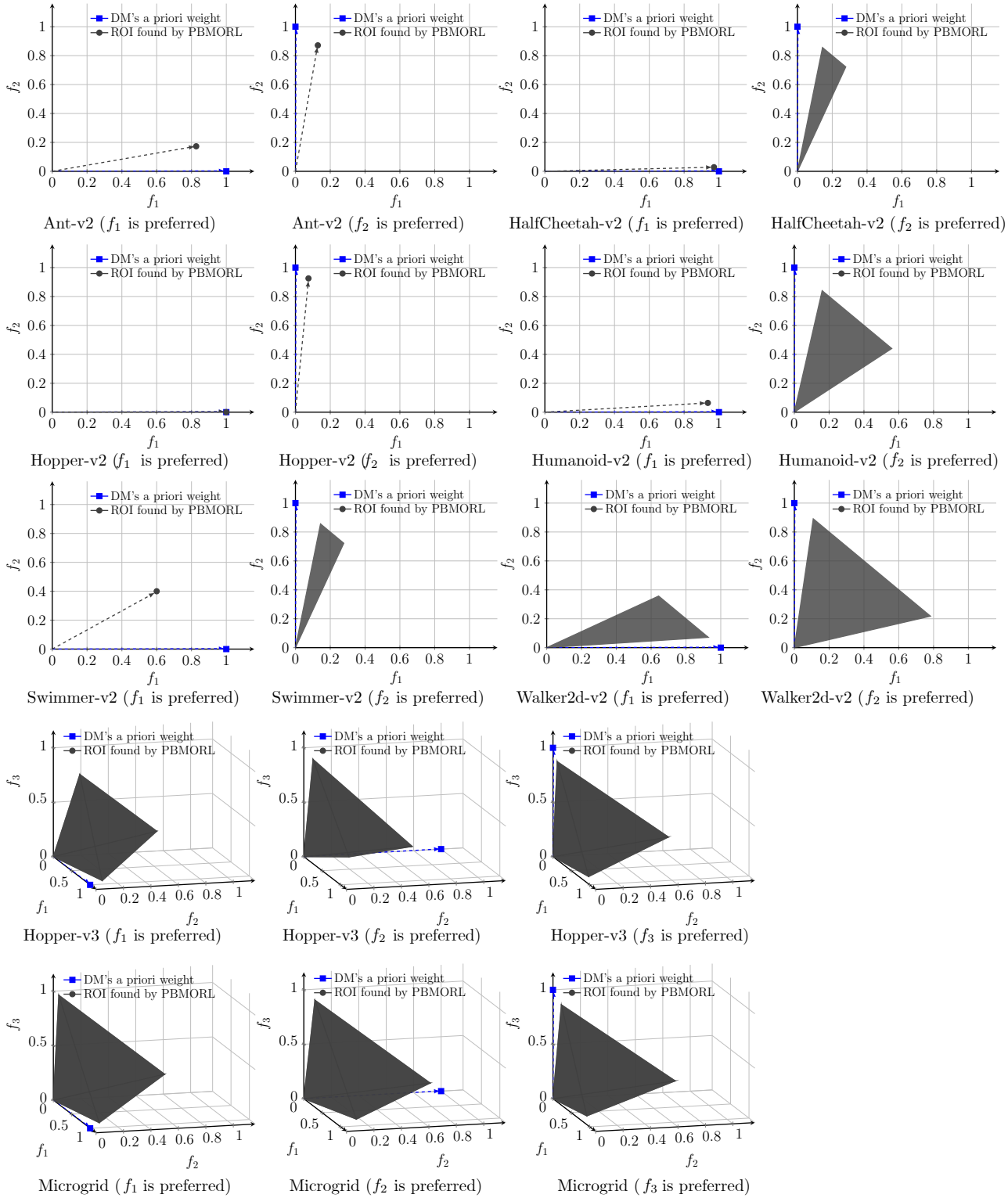

Figure 10: Comparison of the weight vector specified by the DM *a priori* versus the ROI identified by `CBOB` (shaded in the gray region).

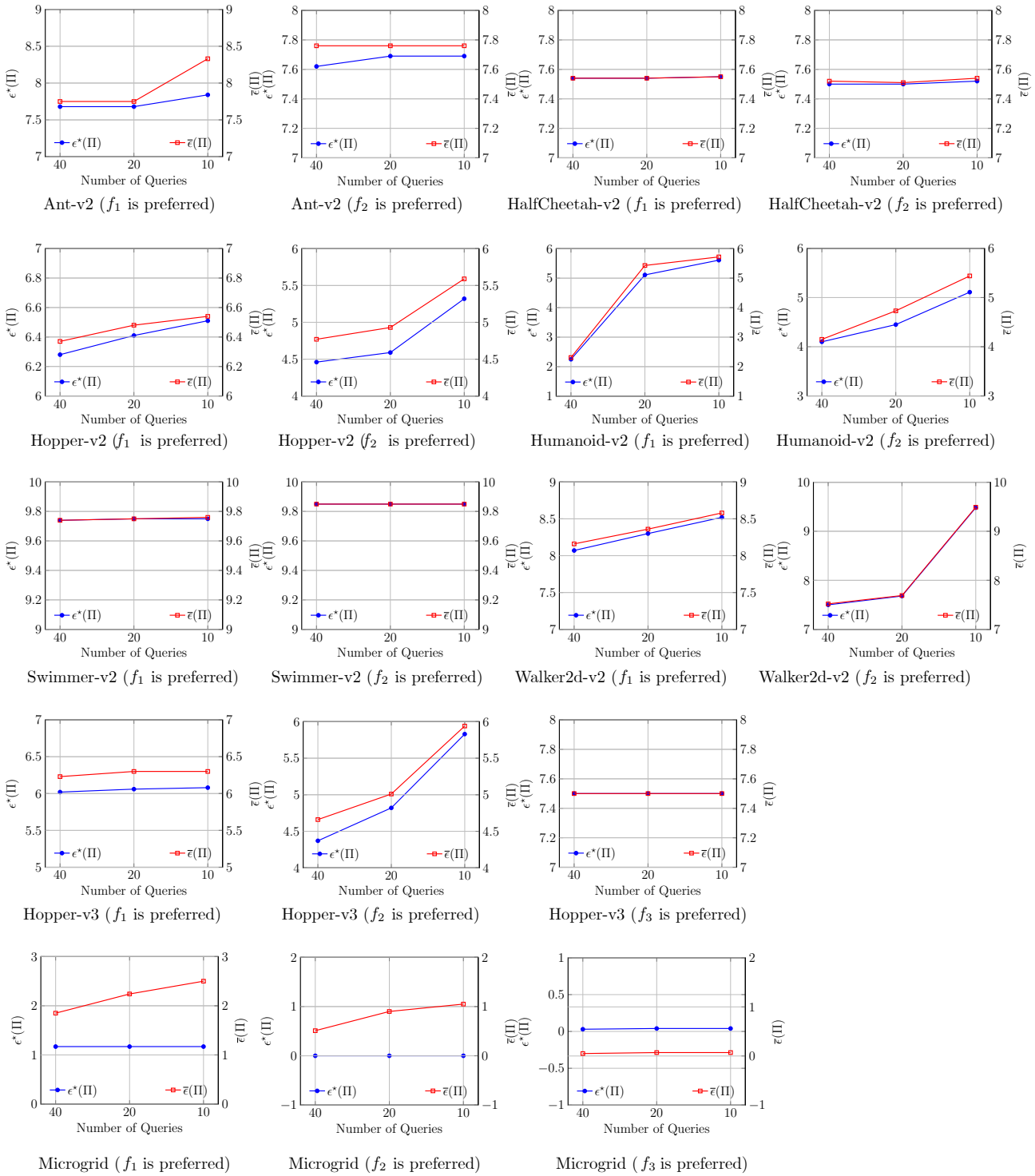

Figure 11: Comparison results of $\epsilon^\star(\Pi)$ and $\bar{\epsilon}(\Pi)$ obtained by `CBOB` with 10, 20, and 40 interactions, respectively.