
Multi-Objective Preference Optimization: Improving Human Alignment of Generative Models

Akhil Agnihotri

University of Southern California
agnihotri.akhil@gmail.com

Rahul Jain

Google DeepMind and USC
rahulajain@google.com

Deepak Ramachandran

Google DeepMind
ramachandrand@google.com

Zheng Wen

Google DeepMind
zhengwen@google.com

Abstract

Post-training of LLMs with RLHF, and subsequently preference optimization algorithms such as DPO, IPO, etc. made a big difference in improving human alignment. But all such techniques can only work with a single (human) objective. In practice, however, human users have multiple objectives, such as helpfulness, harmlessness, etc. and there is no natural way to aggregate them into a single objective. In this paper, we address the multi-objective preference alignment problem, where a policy must optimize several, potentially conflicting, objectives. We introduce the Multi-Objective Preference Optimization (MOPO) algorithm which frames alignment as a *constrained* KL-regularized optimization: the primary objective is maximized while secondary objectives are lower-bounded by tunable safety thresholds. Unlike prior work, MOPO operates directly on pairwise preference data, requires no point-wise reward assumption, and avoids heuristic prompt-context engineering. The method recovers policies on the Pareto front whenever the front is attainable; practically, it reduces to simple closed-form iterative updates suitable for large-scale training. On synthetic benchmarks with diverse canonical preference structures, we show that MOPO approximates the Pareto front. When fine-tuning a 1.3B-parameter language model on real-world human-preference datasets, MOPO attains higher rewards and yields policies that Pareto-dominate baselines, and ablation studies confirm optimization stability and robustness to hyperparameters.

1 Introduction

Aligning Large Language Models (LLMs) and other generative models with human preferences [35, 38, 13] has evolved from single-objective to multi-objective [39, 54, 58], aiming to comprehensively capture the inherent heterogeneity of human preferences. Multi-objective alignment jointly considers multiple human preference objectives, such as safety, helpfulness, factuality, and diversity, to optimize the LLM. However, human preferences are neither one dimensional nor fixed: a single conversation may demand an answer that is simultaneously helpful, harmless, concise, and imaginative. Optimizing an LLM for just one of these axes often degrades the others, as seen between helpfulness and safety tasks [5, 39]. Similar trade-offs appear in factuality versus creativity (hallucination control) [54], or in the balance between privacy and transparency required by medical assistants. Consequently, practitioners increasingly seek multi-objective alignment, where the model is judged by reward vector performance on possibly conflicting objectives, rather than a single scalarized score.

From an optimization standpoint this setting is naturally cast as multi-objective optimization (MOO) [30, 22], where desirable solutions form a Pareto front – the set of policies that cannot be improved

on one objective without harming another. Existing alignment pipelines largely collapse this vector into a weighted sum and run reinforcement learning from human feedback (RLHF) on the resulting scalar reward [35, 38], but linear or non-linear scalarization hides important corner cases, and requires retraining for each weight choice. To overcome this, there is some work on decoding at inference time, but it still aims to cast user preferences as vector inputs to the model [42, 49]. However, at deployment time, users seldom articulate explicit weights; they expect models to adapt interactively and at low latency. These observations motivate algorithms that (i) approximate the Pareto front *offline* under reasonable compute budgets, and (ii) provide preference-conditioned inference that selects points on that front without further training. Motivated by these insights, we introduce **MOP**O, an offline constrained-optimization framework that unifies these desiderata and delivers multi-objective alignment with a single preference-conditioned policy.

We begin by formulating the problem as a concave constrained optimization problem where preferences along the ‘primary’ objective are maximized while preferences along the ‘secondary’ objectives are constrained to be above a tunable threshold. We then motivate bounding the *lower bound* of preferences (instead of the naively constraining these secondary objectives), followed by a behaviour cloning approach to extract the optimal policy from the resulting optimal importance sampling ratio. Overall, this procedure results in iterative updates of the underlying optimization variables, which is scalable and robust to the hyperparameters. Our contributions are as follows: (i) We propose **MOP**O, an offline constrained optimization based preference-only learning algorithm that optimizes for multiple objectives and achieves Pareto optimality provably. (ii) We empirically validate the correctness on a variety of canonical preference dataset types that show how **MOP**O approximates the Pareto front when it is known. (iii) We conduct extensive experiments on a tiny-LLM on real world data to validate the effectiveness of **MOP**O, and show optimization stability through ablation studies.

Related works. RLHF [8, 60] has become the de-facto paradigm for aligning LLMs such as GPT-4 [1] and LLaMA-3 [14]. Most RLHF pipelines fit a reward model to pairwise preferences and then fine-tune the policy with PPO [41, 35]. Instability and sample inefficiency have motivated alternatives that still target a scalar reference-regularized objective, including RAFT [57], RRHF [55], DPO [38], Ψ PO [13], and Nash-RLHF [33]. These methods (except Ψ PO) inherit a fundamental limitation: all preferences are collapsed into a single reward signal, obscuring trade-offs between objective(s). Recent works attempt to optimize multiple objectives by learning scalarization functions or prompt contexts [19, 58, 16, 53, 26, 24]. Although effective in specific domains, such approaches seldom achieve Pareto-optimal solutions even when the Pareto front is known [54, 32]. While [39] mitigate tuning via ‘Rewarded Soups’, and MORLHF [25] and MODPO [59] borrow ideas from multi-objective RL, they still learn with respect to a single functional combination of rewards. Rewards-in-Context (RiC) [54], the multi-objective supervised finetuning variant, Hypervolume Maximization (HaM) [32], Multi-Objective Decoding (MOD) [42], and Directional Preference Alignment (DPA) [49] move beyond heuristic scalarization by conditioning on multiple rewards at inference time. While these algorithms improve controllability, they still rely on inference-time user preference input to optimize multiple objectives, which can misrepresent complex preference structures and are hard to quantify practically (for instance, what does “0.6 helpful, 0.4 safe” imply?). Classical multi-objective RL (MORL) focuses on discovering Pareto-efficient policies under vector rewards [40, 45, 17]. Constrained MORL [20, 28, 2, 3] methods maximize a primary objective while enforcing lower bounds on the other objectives, a strategy that inspires our formulation. However, constrained MORL assumes point-wise reward access and thus cannot be applied directly to human preference data.

2 Preliminaries

We first begin with a motivating example that inspires development of **MOP**O as an offline constrained optimization algorithm learning from preference data. Following this, we describe the problem setting with a formal problem statement. Throughout the main text, we keep notation light and refer the reader to Appendix A for a complete discussion, where we prove that current literature as discussed above fails to achieve Pareto-optimality.

A Motivating Example. In this example, we empirically demonstrate the necessity of *principled* multi-objective optimization methods that account for multi-dimensional preferences. We benchmark various approaches for multi-objective alignment and show that existing state-of-the-art techniques consistently fail to reach the Pareto front. To ensure clarity, we conduct experiments on synthetic datasets where true Pareto front is known, allowing for precise evaluation of alignment quality. Due to space constraints we keep discussion concise, and refer the reader to Appendix A.1 for completeness.

Consider this toy preference example with input and action spaces $\mathcal{X} = \mathcal{Y} = [0, 1]$. For any triplet (x, y, y') we draw a preference label $z \in \{y, y'\}$ from the Bradley-Terry model $\text{BT}(r)$ with $\Pr[z = y | x, y, y'] = \exp(r(x, y)) / (\exp(r(x, y)) + \exp(r(x, y')))$, where $r(\cdot)$ is the underlying reward model [6]. We study two bi-objective settings: *Set A* with $r_1 = e^x + \sqrt{y} - y$ and $r_2 = -\sin x - y^2$, and *Set B* with $r_1 = (x+y)^2$ and $r_2 = \log((1+x)/(1+y))$. From i.i.d. samples $(x, y, y') \sim \mathcal{U}([0, 1]^3)$ we construct four datasets: (i) \mathcal{D}_1 , labeled by r_1 only; (ii) \mathcal{D}_2 , by r_2 only; (iii) \mathcal{D}_J , the joint dataset retaining samples where the two labels coincide; and (iv) \mathcal{D}_C , the combined dataset labeled by $\text{BT}(wr_1 + (1-w)r_2)$ for $w \in [0, 1]$. As seen in Figure 1(a), policies trained with DPO on \mathcal{D}_1 or \mathcal{D}_2 alone ignore one objective, those trained on \mathcal{D}_J see only non-conflicting pairs, and those trained on \mathcal{D}_C are biased toward a single scalarization of reward functions, so all three miss large portions of the Pareto front. A constrained optimization baseline $\pi_{\text{COP}}(x) = \operatorname{argmax}_y r_1(x, y) \text{ s.t. } r_2(x, y) \geq b$ for some $b \in \mathbb{R}$ approaches the Pareto front. These limitations motivate an algorithm that optimizes all objectives *jointly*. Hence, we develop **MOPO** as an offline constrained optimization algorithm that recovers policies which lie near the true Pareto front as in Figure 1(b). Given this motivation, we now turn our attention to introducing notations and providing a formal problem statement.

Problem Setting. We define a finite set of contexts \mathcal{X} and a finite action space \mathcal{Y} . A policy $\pi \in \Delta_{\mathcal{Y}}$ defines a probability distribution over actions given a context, where $\Delta_{\mathcal{Y}}$ is the probability simplex over \mathcal{Y} . The policy is learned from human preferences, which are provided in a pairwise manner over actions. For each context $x \in \mathcal{X}$, two actions $y, y' \sim \mu(\cdot | x)$ are sampled from a behavior policy μ , and a human annotator provides a preference signal indicating which action is preferred. We also let the contexts x be sampled from a context distribution ν , denote a vector by \mathbf{v} , let \mathbf{v}_j to be the element at the j^{th} dimension of \mathbf{v} , and let $[N]$ denote the set $\{1, \dots, N\}$ for some $N \in \mathbb{N}$.

Typically, in single objective preference optimization, the preference for one generation over another is denoted as $y_w \succ y_l$, where y_w and y_l denote the preferred and dis-preferred actions amongst $\{y, y'\}$ respectively. This true human preference takes the form $p(y \succ y' | x)$, the probability of y being preferred to y' knowing the context x . In our multi-objective preference setting, we extend this notation to K objectives, wherein $p_k(y \succ y' | x)$ denotes the preferred and dis-preferred actions amongst $\{y, y'\}$ for k^{th} objective with $k \in [K]$. Moreover, we also set the expected preference of a generation y over a distribution μ knowing x for the k^{th} objective as $p_k(y \succ \mu | x) = \mathbb{E}_{y' \sim \mu(\cdot | x)} [p_k(y \succ y' | x)]$. We also let for any two policies $\pi, \mu \in \Delta_{\mathcal{Y}}$ and a context distribution ν the total preference of policy π to μ w.r.t. k^{th} objective as $p_k(\pi \succ \mu) = \mathbb{E}_{x \sim \nu, y \sim \pi(\cdot | x)} [p_k(y \succ \mu | x)]$. Without loss of generality and clarity of notation, we let $p_K(y \succ y' | x) \equiv p(y \succ y' | x)$ denote the preference for the K^{th} (the primary) objective, and $\mathbf{q}(y \succ y' | x) \in [0, 1]^{K-1}$ denotes the vector of preferences for the $K-1$ (the secondary) remaining objectives, wherein the preferences are applied objective-wise i.e. $q_k(y \succ y' | x) = p_k(y \succ y' | x)$ for $k \in [K-1]$. Following this notation, we also have the following definitions.

$$\mathbf{q}(y \succ \mu | x) = \mathbb{E}_{y' \sim \mu(\cdot | x)} [\mathbf{q}(y \succ y' | x)] \quad \text{and} \quad \mathbf{q}^\nu(\pi \succ \mu) = \mathbb{E}_{x \sim \nu, y \sim \pi(\cdot | x)} [\mathbf{q}(y \succ \mu | x)].$$

Pareto optimality. In multi-objective preference optimization (**MOPO**), a policy that simultaneously optimizes all objectives does not exist. Thus, a set of non-dominated solutions is desired. We say policy π is dominated by policy π' when there is no objective under which π' is *worse* than π , i.e., $p_k^\nu(\pi \succ \mu | x) \leq p_k^\nu(\pi' \succ \mu | x)$ for $\forall k \in [K]$. A policy π is Pareto-optimal if and only if it is not dominated by any other policy. The Pareto set is composed of non-dominated solutions, denoted as Π_P . Overall, the goal of **MOPO** is to obtain an optimal policy in Π_P .

Problem statement. The goal is to propose a general solution for RLHF with multiple objectives, based on constrained optimization of a function of preferences. We propose this constrained optimization problem as maximizing a primary objective, and constraining the remaining objective values. To this end, we consider a reference policy $\pi_{\text{ref}} \in \Delta_{\mathcal{Y}}$, a real positive regularization parameter $\tau \in \mathbb{R}_+$, and let $b \in [0, 1]^{K-1}$. The concave constrained optimization problem (**COP**) for **MOP** becomes,

$$\max_{\pi} \underbrace{\mathbb{E}_{\substack{x \sim \nu \\ y \sim \pi(\cdot | x) \\ y' \sim \mu(\cdot | x)}} [p(y \succ y' | x)] - \tau \text{KL}(\pi || \pi_{\text{ref}})}_{\mathcal{F}(\rho)} \quad \text{s.t.} \quad \underbrace{\mathbb{E}_{\substack{x \sim \nu \\ y \sim \pi(\cdot | x) \\ y' \sim \mu(\cdot | x)}} [\mathbf{q}(y \succ y' | x)]}_{\mathcal{G}(\rho)} \geq b. \quad (1)$$

See Definition A.1 for the definition of KL divergence. We now focus our attention on designing a **MOP** algorithm to solve the **COP** problem above.

3 The MOP Algorithm

First, to find an optimal policy that lies in Π_P , it is crucial to set proper constraint values b such that the solution of Problem (1) contributes to the Pareto front. See Appendix A.2 and A.3 for the theoretical discussion, and Figure 2, which illustrates that if b is correctly initialized, then solving the **COP** problem yields solutions on the Pareto-front. In Section 3.1 we will propose a more practical method of specifying constraint values. For now, we focus on solving Problem (1). To deal with the optimization variable in expectation, we let the importance sampling ratio be $\rho(y) = \frac{\pi(y)}{\pi_{\text{ref}}(y)}$. For this we assume that $\text{Supp}(\pi) = \text{Supp}(\pi_{\text{ref}})$. In addition, for clarity, we shall omit the dependency on context x as the all results hold true for all $x \in \text{Supp}(\nu)$. Then the final **MOP** problem takes the form,

$$\max_{\rho} \underbrace{\mathbb{E}_{\substack{y \sim \pi_{\text{ref}} \\ y' \sim \mu}} [\rho(y) p(y \succ y')] - \tau \mathbb{E}_{y \sim \pi_{\text{ref}}} [\rho(y) \ln(\rho(y))]}_{\mathcal{F}(\rho)} \quad \text{s.t.} \quad \underbrace{\mathbb{E}_{\substack{y \sim \pi_{\text{ref}} \\ y' \sim \mu}} [\rho(y) \mathbf{q}(y \succ y')]}_{\mathcal{G}(\rho)} \geq b, \quad (2)$$

which is a strictly concave optimization problem w.r.t. ρ . We then formulate the Lagrangian of the above **MOP** problem. For some $\lambda := \{\lambda_k\}_{k=1}^{K-1} \geq \mathbf{0}$, we have the Lagrangian as $\mathcal{L}(\rho, \lambda) = \mathcal{F}(\rho) - \lambda^T (\mathbf{b} - \mathcal{G}(\rho))$. This leads to the following proposition.

Proposition 3.1. *The dual formulation of Problem (2) is given by,*

$$\text{Dual}^* \triangleq \min_{\lambda \geq 0} \max_{\rho} \mathcal{L}(\rho, \lambda) = \min_{\lambda \geq 0} \mathcal{L}(\rho_{\lambda}^*, \lambda) = \min_{\lambda \geq 0} \mathcal{F}(\rho_{\lambda}^*) - \lambda^T (\mathbf{b} - \mathcal{G}(\rho_{\lambda}^*))$$

where, $\rho_{\lambda}^*(y) = \exp \left(\tau^{-1} \mathbb{E}_{y' \sim \mu} [p(y \succ y') + \lambda^T \mathbf{q}(y \succ y')] - 1 \right) \quad \forall y \in \mathcal{Y}. \quad (3)$

See Appendix A.4 for proof. The inner maximization in Equation (3) corresponds to computing an optimal policy (importance sampling ratio ρ) that maximizes scalarized preferences for the K^{th} objective, while the outer minimization corresponds to balancing the penalty of suboptimal policy w.r.t. the other $(K-1)$ objectives: if the current policy (ρ) is under performing w.r.t. the k^{th} objective, λ_k increases so that the under performance is penalized more, and vice versa.

However, constraining the preference vector $\mathbf{q}(\cdot)$ naively can result in constraint violation when deployed to the real environment. This is due to the fact that empirical importance sampling weighted preferences $\hat{\mathbf{q}}(\cdot)$ collected from a finite dataset inevitably have estimation error (see Figure 3). For the K^{th} objective, preference estimation error may be tolerated as long as those estimates are useful as policy improvement signals, i.e., it is sufficient to maintain the relative order of preferences. For the remaining $(K-1)$ constrained objectives, Equation (3) instead relies on the estimated values directly. Hence, to make a policy robust against these estimation errors, we consider a constrained policy optimization scheme that instead constrains the *lower bound* of the preference estimates $\hat{\mathbf{q}}(\cdot)$, i.e.,

$$\max_{\rho} \mathcal{F}(\rho) \quad \text{s.t.} \quad \text{LowerBound}(\mathcal{G}(\rho)) \geq b.$$

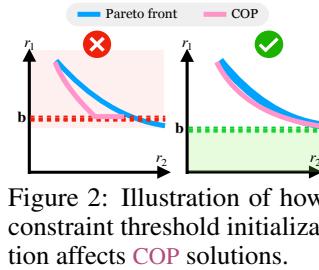


Figure 2: Illustration of how constraint threshold initialization affects **COP** solutions.

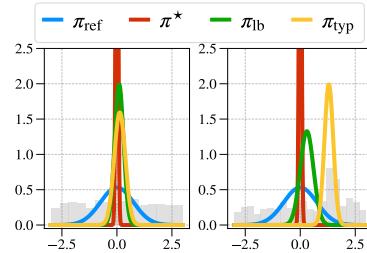


Figure 3: **MOP**: Impact of preference distribution (gray) over output space with lower-magnitude outputs being preferred with probability 1. π_{lb} constrains lower bound of $\mathcal{G}(\rho)$, while π_{typ} constrains $\mathcal{G}(\rho)$ directly (typically).

Then, the key question is how to estimate the lower bound of the ρ -weighted preference vector $\mathbf{q}(\cdot)$. One natural way is to exploit bootstrap confidence interval estimation [12], by sampling bootstrap datasets \mathcal{D}_i from \mathcal{D} and constructing population statistics for confidence interval estimation. However, this procedure is computationally expensive. Instead, we take a different, computationally efficient approach. Specifically, we solve a constrained optimization problem for each objective $k \in [K - 1]$. For a policy $\pi_k \in \Delta_{\mathcal{Y}}$ and some $\epsilon \in \mathbb{R}_+$, the lower bound optimization problem of $\mathcal{G}(\rho)$ becomes:

$$\min_{\pi_k} \mathbb{E}_{\substack{y \sim \pi_k \\ y' \sim \mu}} [\rho(y) \mathbf{q}_k(y \succ y')] \quad \text{s.t.} \quad \text{KL}(\pi_k \parallel \pi_{\text{ref}}) \leq \epsilon \quad \text{and,} \quad \sum_{y \in \mathcal{Y}} \pi_k(y) = 1 \quad (4)$$

In essence, we want to adversarially optimize a distribution π_k so that it underestimates the preference objective k , and simultaneously, we enforce that this π_k should not be perturbed too much from π_{ref} . As ϵ increases, the degree of underestimation of preference probabilities also increases. Now we simplify the constrained optimization problem into a single unconstrained problem as follows.

Proposition 3.2. *The optimal solution to Problem (4) can be obtained by solving the following optimization problem.*

$$\begin{aligned} \chi_k^* &= \underset{\chi_k \geq 0}{\operatorname{argmax}} \mathcal{L}_k(\chi_k; \rho) := -\chi_k \ln (\mathbb{E}_{y \sim \pi_{\text{ref}}, y' \sim \mu} [\exp(\chi_k^{-1} \rho(y) \mathbf{q}_k(y \succ y'))]) - \chi_k \epsilon \\ \text{with, } \pi_k^*(y) &\propto \pi_{\text{ref}}(y) \underbrace{\exp \left((\chi_k^*)^{-1} \mathbb{E}_{y' \sim \mu} [\rho(y) \mathbf{q}_k(y \succ y')] \right)}_{w(y) \text{ (unnormalized weight)}}. \end{aligned} \quad (5)$$

See Appendix A.4 for proof. Note that each term in Equation (5) can be estimated using samples from the offline dataset \mathcal{D} , thus it can be optimized in a fully offline manner. This procedure can be understood as computing the weights for each sample while adopting reweighting, that is, $\text{LowerBound}(\mathcal{G}(\rho)) = \mathbb{E}_{y \sim \pi_{\text{ref}}, y' \sim \mu} [\tilde{w}(y) \rho(y) \mathbf{q}_k(y \succ y')]$, where $\tilde{w}(y)$ is normalized $w(y)$. Solving this unconstrained optimization problem and plugging it in the main Dual* Problem (3) corresponds to the following iterative updates.

$$\chi^* \leftarrow \underset{\chi \geq 0}{\operatorname{argmax}} \sum_{k=1}^{K-1} \mathcal{L}_k(\chi_k; \rho_{\lambda}) \quad \text{and,} \quad \lambda^* \leftarrow \underset{\lambda \geq 0}{\operatorname{argmin}} \mathcal{F}(\rho_{\lambda}) - \lambda^T (\mathbf{b} - \underbrace{\mathcal{L}(\chi^*; \rho_{\lambda})}_{\text{LowerBound}(\mathcal{G}(\rho))}), \quad (6)$$

where $\mathcal{L}(\chi^*; \rho_{\lambda}) = (\mathcal{L}_1(\chi^*, \rho_{\lambda}), \dots, \mathcal{L}_{K-1}(\chi^*, \rho_{\lambda}))^T$. Compared to the original dual Problem (3), the additional maximization for χ is introduced to estimate the lower bound of preference probabilities for the constrained objectives. Once the optimal solution λ^* is computed, $\rho_{\lambda^*}^*(y) \equiv \rho^*(y) = \pi^*(y)/\pi_{\text{ref}}(y)$ is also derived from Equation (3).

Policy Extraction. The current procedure estimates the importance sampling ratio $\rho^*(y)$ of the optimal policy, rather than directly obtaining the policy itself. Since the importance sampling ratio does not provide a direct way to sample an action, we need to extract the optimal policy π^* from ρ^* in order to select actions when deployed. For tabular cases, it is straightforward to obtain $\pi^*(y) = (\pi_{\text{ref}}(y) \rho^*(y)) / (\sum_{y' \sim \mathcal{Y}} \pi_{\text{ref}}(y') \rho^*(y'))$. However, the same method cannot directly be applied to large scale optimization problems due to the intractability of computing the normalization constant. For such cases, we instead extract the policy using importance-weighted behavioral cloning by solving the following problem:

$$\max_{\pi} \mathbb{E}_{y \sim \pi^*} [\log(\pi(y))] = \max_{\pi} \mathbb{E}_{y \sim \pi_{\text{ref}}} [\rho^*(y) \log(\pi(y))], \quad (7)$$

which maximizes the log-likelihood of actions to be selected by the optimal policy π^* . Based on the above discussion, we now turn our attention to the practical implementation of MOP.

3.1 Practical Algorithm with Function Approximation

For this section we consider the practical implementation of MOP, using a given offline dataset of preferences \mathcal{D} . First, we discuss the function approximations used to parameterize the optimization variables, and then we discuss how to implement the procedure discussed above.

Function Approximations. We let the optimization variables $\lambda, \chi \in \mathbb{R}_+^{K-1}$, and the policy π_{ψ} to be parameterized by ψ . The parameter $\chi = \{\chi_k\}_{k=1}^{K-1}$ is trained by minimizing the following loss:

$$\min_{\chi \geq 0} \sum_{k=1}^{K-1} [\chi_k \ln (\mathbb{E}_{y \sim \pi_{\text{ref}}, y' \sim \mu} [\exp(\chi_k^{-1} \rho(y) \mathbf{q}_k(y \succ y'))]) + \chi_k \epsilon].$$

Since this involves a logarithm outside of the expectation, to overcome bias we use mini-batch approximation for computational efficiency. The empirical form is then given by:

$$\min_{\boldsymbol{\chi} \geq \mathbf{0}} J(\boldsymbol{\chi}; \rho) := \mathbb{E}_{\text{batch}(\mathcal{D}) \sim \mathcal{D}} \left[\sum_{k=1}^{K-1} \left[\chi_k \ln \left(\mathbb{E}_{y, y' \sim \text{batch}(\mathcal{D})} [\exp(\chi_k^{-1} \rho(y) \mathbf{q}_k(y \succ y'))] \right) + \chi_k \epsilon \right] \right]. \quad (8)$$

Finally, following the discussion before, $\boldsymbol{\lambda}$ and the policy parameterized by ψ are optimized by:

$$\min_{\boldsymbol{\lambda} \geq \mathbf{0}} J(\boldsymbol{\lambda}; \boldsymbol{\chi}) := \mathcal{F}(\rho) - \boldsymbol{\lambda}^T (\mathbf{b} - J(\boldsymbol{\chi}; \rho)) \quad \text{and}, \quad \min_{\psi} J_\rho(\pi_\psi) := - \mathbb{E}_{y \sim \pi_{\text{ref}}} [\rho(y) \log(\pi_\psi(y))], \quad (9)$$

where all variables are optimized jointly. Now we detail how to empirically solve the problem.

Empirical MOP0. We now formulate the problem given a fixed offline dataset \mathcal{D} of the form $\mathcal{D} := \{(x_i, y_i, y'_i, \mathbb{I}(y_i, y'_i))\}_{i=1}^N$, where context $x_i \in \mathcal{X}$, y_i, y'_i are two generations from π_{ref} and μ respectively, and $\mathbb{I}(\cdot, \cdot) \in \{0, 1\}^K$ is preference indicator vector over K objectives, i.e., $\mathbb{I}_k(y, y') = 1$ if $y \succ_k y'$, and 0 otherwise for $k \in [K]$. With a slight abuse in notation, we let $\mathbb{I}_p(\cdot, \cdot) \triangleq \mathbb{I}_K(y, y')$ and $\mathbb{I}_q(\cdot, \cdot) \triangleq (\mathbb{I}_1(y, y'), \dots, \mathbb{I}_{K-1}(y, y'))^T$. Then, the empirical optimization problem is given by:

$$\max_{\rho} \underbrace{\frac{1}{N} \sum_{i=1}^N \rho(y_i) \mathbb{I}_p(y_i, y'_i) - \tau \rho(y_i) \ln(\rho(y_i))}_{\widehat{\mathcal{F}}(\rho)} \quad \text{s.t.} \quad \underbrace{\frac{1}{N} \sum_{i=1}^N \rho(y_i) \mathbb{I}_q(y_i, y'_i)}_{\widehat{\mathcal{G}}(\rho)} \geq \mathbf{b}. \quad (10)$$

See Appendix A.5 for proof. Note that each data point $(x_i, y_i, y'_i, \mathbb{I}(y_i, y'_i))$ contributes two terms to the empirical problem above: one with $(x, y, y', \mathbb{I}(y, y')) = (x_i, y_i, y'_i, \mathbb{I}(y_i, y'_i))$ and another with $(x, y, y', \mathbb{I}(y, y')) = (x_i, y'_i, y_i, \mathbb{I}(y'_i, y_i))$. This symmetry is important to exploit since it reduces gradient variance and improves stability during optimization. For clarity, we omit the symmetric term in notation as incorporating it is trivial – simply augment the current dataset by swapping y_i and y'_i and bit flipping $\mathbb{I}(\cdot, \cdot)$ element-wise.

Now, as before, for some $\boldsymbol{\lambda} := \{\lambda_k\}_{k=1}^{K-1} \geq \mathbf{0}$, we have the Lagrangian as $\widehat{\mathcal{L}}(\rho, \boldsymbol{\lambda}) = \widehat{\mathcal{F}}(\rho) - \boldsymbol{\lambda}^T (\mathbf{b} - \widehat{\mathcal{G}}(\rho))$, and the dual as,

$$\begin{aligned} \widehat{\text{Dual}}^* &\triangleq \min_{\boldsymbol{\lambda} \geq \mathbf{0}} \max_{\rho} \widehat{\mathcal{L}}(\rho, \boldsymbol{\lambda}) = \min_{\boldsymbol{\lambda} \geq \mathbf{0}} \widehat{\mathcal{L}}(\rho_{\boldsymbol{\lambda}}^*, \boldsymbol{\lambda}) = \min_{\boldsymbol{\lambda} \geq \mathbf{0}} \widehat{\mathcal{F}}(\rho_{\boldsymbol{\lambda}}^*) - \boldsymbol{\lambda}^T (\mathbf{b} - \widehat{\mathcal{G}}(\rho_{\boldsymbol{\lambda}}^*)) \\ \text{where, } \rho_{\boldsymbol{\lambda}}^*(y) &= \exp \left((\tau N)^{-1} \sum_{i=1}^N [\mathbb{I}_p(y, y') + \boldsymbol{\lambda}^T \mathbb{I}_q(y, y')] - 1 \right). \end{aligned} \quad (11)$$

Following our earlier discussion on lower bounding the preference probabilities for $[K-1]$ constraints, we wish to constrain $\text{LowerBound}(\widehat{\mathcal{G}}(\rho))$ to be greater than \mathbf{b} . Then, following Equation (8), the empirical lower bound is obtained by solving below for M batches of \mathcal{D} , where each batch m of size $N_M = \lfloor N/M \rfloor$ is of the form $(x_{m,j}, y_{m,j}, y'_{m,j}, \mathbb{I}(y_{m,j}, y'_{m,j}))_{j=1}^{N_M}$.

$$\min_{\boldsymbol{\chi} \geq \mathbf{0}} \widehat{J}(\boldsymbol{\chi}; \rho) := \frac{1}{M} \sum_{m=1}^M \left[\sum_{k=1}^{K-1} \left[\chi_k \ln \left(\frac{1}{N_M} \sum_{j=1}^{N_M} \exp(\chi_k^{-1} [\rho(y_{m,j}) (\mathbb{I}_q)_k(y_{m,j}, y'_{m,j})]) \right) + \chi_k \epsilon \right] \right]. \quad (12)$$

This transforms the empirical dual Problem (11) into the below optimizations for $\boldsymbol{\lambda}$ and policy π_ψ :

$$\min_{\boldsymbol{\lambda} \geq \mathbf{0}} \widehat{J}(\boldsymbol{\lambda}; \boldsymbol{\chi}) := \widehat{\mathcal{F}}(\rho) - \boldsymbol{\lambda}^T (\mathbf{b} - \widehat{J}(\boldsymbol{\chi}; \rho)) \quad \text{and}, \quad \min_{\psi} \widehat{J}_\rho(\pi_\psi) := - \frac{1}{N} \sum_{i=1}^N \rho_{\boldsymbol{\lambda}}^*(y_i) \log(\pi_\psi(y_i)), \quad (13)$$

where $\rho_{\boldsymbol{\lambda}}^*(\cdot)$ is computed using Equation (11). However, two caveats still remain.

Lagged reference policy. The KL regularizer in Equation (2) keeps π_ψ close to a fixed reference π_{ref} . Because successive iterates move toward the Pareto front, it is advantageous to regularize against a *stronger* policy than the initial prior. Analogous to target networks in Q-learning [31] and recent self-improvement loops for LLMs [7, 36], we update the reference every t_0 steps: $\pi_{\text{ref}} \leftarrow \pi_\psi^{(t-t_0)}$.

All expectations in Equation (10) are then reweighted by the ratio $\rho_{\text{lag.ref}}(y) = \pi_\psi^{(t-t_0)}(y)/\pi_{\text{ref}}(y)$ for all $y \in \mathcal{Y}$, requiring no additional data collection. We find that this leads to more stable optimization and consistent progression to the Pareto front.

Adaptive constraint schedule. In practice, exact values of constraint thresholds \mathbf{b} are unknown. We therefore, after every t_0 steps, set the constraint vector *only* from the policy of the previously optimized iterates: $\mathbf{b} = \boldsymbol{\beta}^\top \widehat{\mathcal{G}}(\rho^{(t-t_0)})$ for some hyperparameter $\boldsymbol{\beta} \in (0, 1)^{K-1}$ and

Algorithm 1 Multi-objective Preference Optimization (**MOP**O)

```

1: Input: Dataset  $\mathcal{D} = \{x_i, y_i, y'_i, \mathbb{I}(y_i, y'_i)\}_{i=1}^N$ , batch size  $M$ , learning rate  $\eta$ , epochs  $T, t_0, \beta$ .
2: Initialize parameter vectors  $\lambda^{(0)}, \chi^{(0)}, \psi^{(0)}, b$  and  $\rho_{\lambda^{(0)}}(\cdot)$ .
3: for  $t = 1, 2, \dots, T$  do
4:   Sample  $M$  mini-batches from  $\mathcal{D}$ .
5:    $\chi^{(t)} = [\chi^{(t-1)} - \eta \nabla_{\chi} \hat{J}(\chi; \rho_{\lambda^{(t-1)}})]_+$  using Equation (12). // Lower bound estimation
6:    $\lambda^{(t)} = [\lambda^{(t-1)} - \eta \nabla_{\lambda} \hat{J}(\lambda; \chi^{(t)})]_+$  using Equation (13). // Solve original dual
7:   Compute  $\rho_{\lambda^{(t)}}(\cdot)$  using Equation (11). // Retrieve primal solution
8:   Update policy  $\psi^{(t)} = \psi^{(t-1)} - \eta \nabla_{\psi} \hat{J}_{\rho_{\lambda^{(t)}}}(\pi_{\psi})$  using Equation (13). // Extract policy
9:   if  $t \bmod t_0 = 0$  then
10:     $b = \beta^\top \hat{\mathcal{G}}(\rho^{(t-t_0)})$  // Update constraint thresholds
11:     $\pi_{\text{ref}} \leftarrow \pi_{\psi}^{(t-t_0)}$  // Update reference policy
12:   end if
13: end for
14: Output: Optimal policy  $\pi_{\psi}^{(T)}$ .

```

$\rho^{(t-t_0)}(\cdot) = \pi_{\psi}^{(t-t_0)}(\cdot)/\pi_{\text{ref}}(\cdot)$ is the importance sampling ratio. This retains the theoretical lower bound interpretation of the constraints while avoiding a global search across constraint thresholds.

Summarizing the above discussion gives the final *Multi-Objective Preference Optimization* (**MOP**O) algorithm, shown in Algorithm 1. At each step we maximize the primary preference objective subject to the time-varying lower bounds b , while penalizing divergence from the current reference policy. The result is a scalable, offline algorithm that steadily advances toward Pareto-optimal solutions.

4 Empirical Results

We conducted extensive experimental evaluation on the relative empirical performance of the **MOP**O algorithm to arrive at the following conclusions: (i) **MOP**O exactly recovers – or provably approximates – the optimal policy under total, partial and antichain orderings of preferences, (ii) in contrast to several DPO-style baselines [13], **MOP**O does not overfit to the preference dataset and preserves performance on held-out comparisons, (iii) it performs better or nearly as well as all baseline algorithms when evaluated on a tiny-LLM on real-world data, and (iv) it is more robust to stochastic and complex environments as compared to previous methods as we will see in the ablations.

4.1 Synthetic Sanity Check

This experiment verifies that **MOP**O solves the constrained optimization problem exactly when the ground-truth optimum is known. We consider a two-objective, context-free bandit setting with discrete action set $\mathcal{Y} = \{y_1, y_2, y_3\}$ and a uniform reference policy π_{ref} . Training data $\mathcal{D} = \{(y_i, y_j, \mathbb{I}(y_i, y_j))\}$ are sampled *with replacement* from one of five canonical preference structures:

$$\begin{aligned}
\mathcal{D}_1 &= \{(y_1, y_2, (1, 1)), (y_2, y_3, (1, 1)), (y_1, y_3, (1, 1))\} && \text{(total order)} \\
\mathcal{D}_2 &= \{(y_1, y_2, (1, 1)), (y_1, y_3, (1, 0)), (y_2, y_3, (0, 0))\} && \text{(partial order)} \\
\mathcal{D}_3 &= \{(y_1, y_2, (1, 1)), (y_1, y_3, (1, 0)), (y_2, y_3, (1, 0))\} && \text{(antichain)} \\
\mathcal{D}_4 &= \{(y_1, y_2, (1, 0)), (y_1, y_3, (0, 1)), (y_2, y_3, (0, 0))\} && \text{(pairwise incomparable)} \\
\mathcal{D}_5 &= \{(y_1, y_2, (1, 1)), (y_2, y_1, (0, 1))\} && \text{(unobserved preferences)}
\end{aligned}$$

Learning Protocol. Mini-batches are drawn uniformly with replacement from each of \mathcal{D}_j for $j \in \{1, \dots, 5\}$ and optimized with Algorithm 1 for 20k steps. Policy is encoded simply as $\pi_{\psi}(y_i) = \text{softmax}(\psi)_i$ using a vector $\psi \in \mathbb{R}^3$, and is optimized for 20k steps using Adam [23] with a learning rate of 0.015 and mini-batch size 12. We also sweep the KL-penalty coefficient $\tau \in \{0.1, 0.5, 1\}$.

Results. The learned action probabilities for each \mathcal{D}_j (column-wise) are seen in Figure 4. For \mathcal{D}_1 the policy converges to the Condorcet winner y_1 . For \mathcal{D}_2 , in which y_1 and y_3 are undominated, **MOP**O assigns them equal probability. Dataset \mathcal{D}_3 yields an antichain; the optimal mixture places roughly two-thirds mass on y_1 and one-third on y_3 , which **MOP**O reproduces. In the pairwise-incomparable set \mathcal{D}_4 the learned distribution stays close to π_{ref} , as expected. Finally, with the inconsistent set \mathcal{D}_5 , **MOP**O successively down-weights the unobserved action y_3 as τ decreases. Across all cases, increasing τ smoothly interpolates between π_{ref} and the optimal policy, confirming controlled regularization.

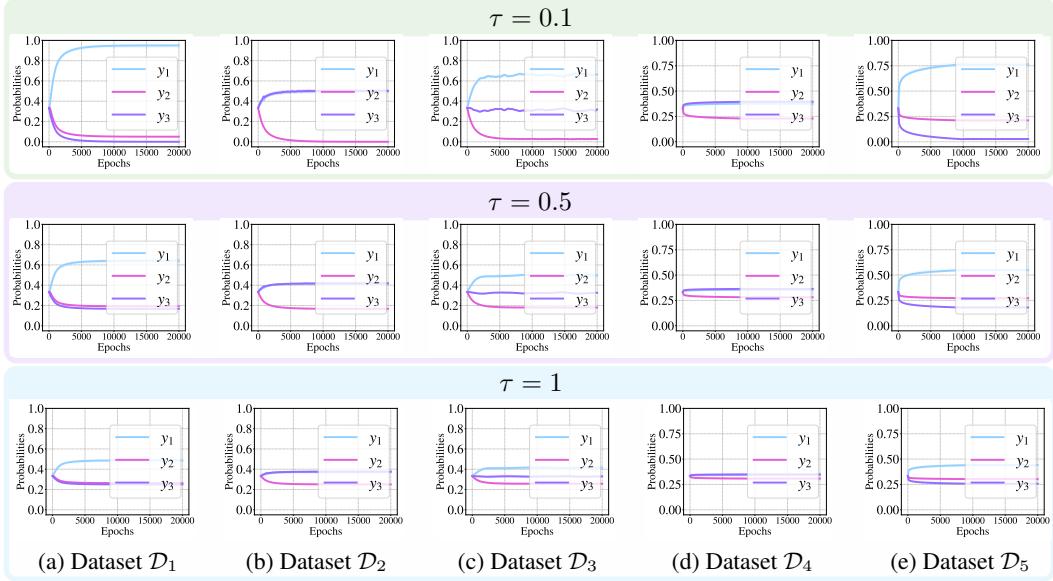


Figure 4: Learning Curves of Action Probabilities of **MOPPO** on various dataset types (read column-wise). Shaded region around mean line represents 1 standard deviation over 5 independent runs.

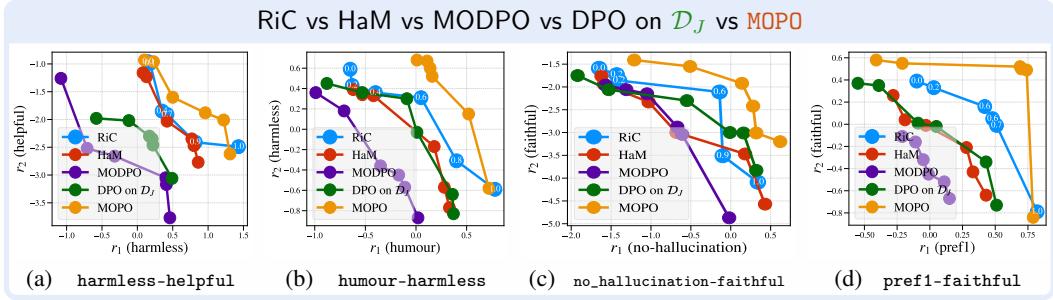


Figure 5: Results of the Helpful Assistant task and the Reddit Summary task with normalized rewards.

4.2 Experiments on Text Generation Tasks

Having established the validity of finetuning using **MOPPO**, in this section, we aim to evaluate the performance of our **MOPPO** algorithm on two text generation tasks that involve diverse rewards. Furthermore, we will conduct ablation studies to analyze the individual contributions of the components within **MOPPO** to show that it is robust and scalable.

Baselines. We consider (i) Rewards-in-Context (RiC) [54] with in-context rewards and human preferences (this is a state-of-the-art baseline that outperforms Rewarded Soups [39] and MORLHF [25]), (ii) Hypervolume maximization Method (HaM) [32] which learns diverse LLM policies that maximize their hypervolume, (iii) Multi-objective DPO (MODPO) [59], which is a more nuanced version of scalarizing multiple reward models into one (this corresponds to “DPO on $w^T r$ ” in Figure 1), and (iv) DPO on \mathcal{D}_J , which trains DPO on context-output pairs which are preferred under both reward models similar to our discussion in Section 2.

Training protocol. We finetune a pretrained phi-1.5, a 1.3B parameter model with performance on natural language tasks comparable to models $\sim 5\times$ larger [27]. Note that the implementation of **MOPPO** can be extended to larger models, such as Llama 3 [14] and Gemma 3 [44], as well, but due to resource constraints we focus on tiny-LLMs to showcase practical *relative* performance (w.r.t. baselines) on real world datasets. All likelihood maximization problems use parameter efficient fine-tuning with LoRA [18] for 5k steps with a batch size of 4. LoRA is applied to the shared transformer backbone, and is optimized together with the policy parameters.

Evaluation protocol. All methods are evaluated based on the quality of their empirical Pareto fronts on the Helpful Assistant task [5] and the Reddit Summariza-

tion task [43]. The Helpful Assistant task uses the HH-RLHF dataset containing 160k prompts with human preference annotations. Evaluation is performed using three HuggingFace reward models – ‘helpful’, ‘harmless’, and ‘humor’ – which score responses from different perspectives [51]. The Reddit Summary task consists of 14.9k posts and corresponding summaries. We consider three reward models: ‘pref1’ and ‘no-hallucination’, which evaluate human preference for summaries, and a ‘faithful’ reward that measures the faithfulness of the summary to the original post. For both datasets, we uniformly sample 3k prompts from the test sets and cluster them into $j = 6$ clusters, and compute the average reward for each objective for each cluster group [32, 54]. Performance is measured by comparing the resulting multi-objective reward values; methods with *outer* Pareto fronts are better.

See Figure 5 for the empirical Pareto fronts. Each point represents the average rewards evaluated on sampled cluster groups from the test set. For **MOP0**, we let the primary objective be r_1 and constrain preferences w.r.t. r_2 . For RiC, the numbers at the centers of the markers indicate the (normalized) preference for r_1 in each cluster that achieves the highest reward within that cluster. See Appendix A.6 for more details.

To assess the scalability of **MOP0**, we aim to optimize three objectives in the Helpful Assistant task, i.e., ‘harmless’, ‘helpful’, and ‘humor’. For easy interpretation, we sample 2k prompts from the test set and plot the average rewards. The results in Table 1 reveal that RLHF [54], when optimized for a single reward (see Appendix A.6 for the problem formulation), achieves high performance on the targeted reward but degrades substantially on the remaining objectives. In contrast, multi-objective algorithms yield more uniform performance across all rewards, with **MOP0** achieving the most balanced trade-offs. The results demonstrate the effectiveness of **MOP0** in scaling to more than two objectives as well.

Table 1: Three objective alignment for Helpful Assistant task with normalized rewards.

	helpful	humour	harmless
RLHF-r1	0.76	-0.42	-0.23
RLHF-r2	-0.81	0.53	-0.40
RLHF-r3	-0.79	-0.92	0.42
MOP0	0.39	0.22	0.17
RiC	0.18	0.07	0.20
HaM	0.01	0.03	0.11
MODPO	0.04	-0.09	0.08
DPO on \mathcal{D}_J	0.12	-0.01	0.05

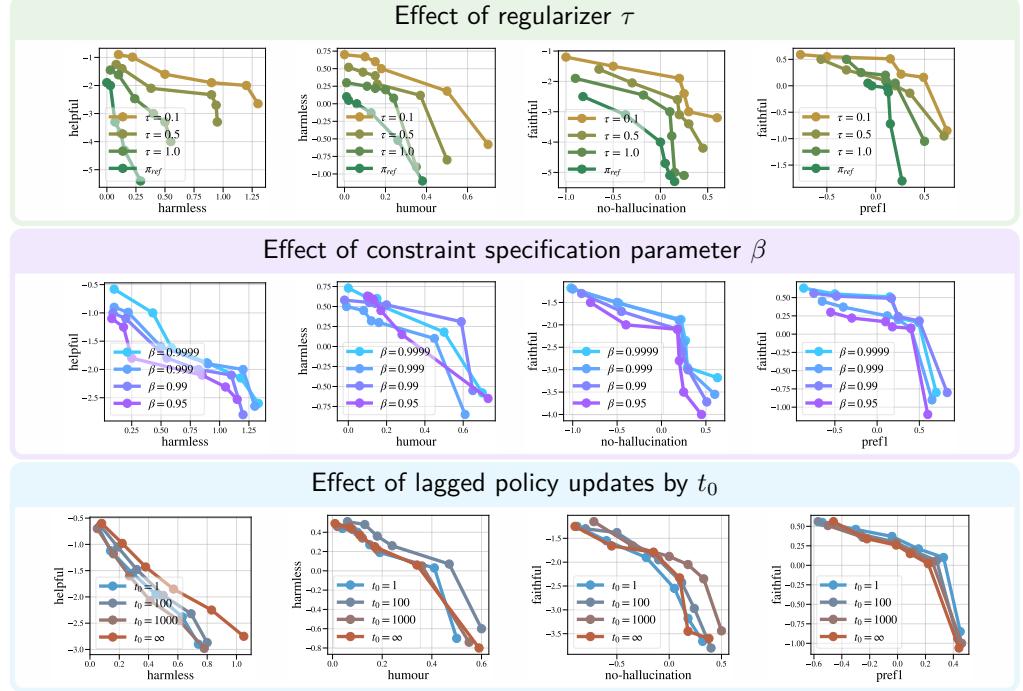


Figure 6: Empirical Pareto fronts of **MOP0** under different regularization, constraint threshold updates, and lagged reference policy updates in the Helpful Assistant and Reddit Summary tasks.

4.3 Secondary Evaluation

In this section, we take a deeper dive into the empirical performance of **MOP0**. We discuss its dependence on various factors, and try to verify its merits.

Effect of regularization. **MOP**O employs a *lagged* reference policy to impose a dynamic KL-regularization. This moving baseline stabilizes gradient estimates and prevents overfitting, allowing maximization of multiple reward functions and extension towards the Pareto frontier. In contrast, DPO variants use a static reference and tend to overfit irrespective of regularization [13, 54]. **MOP**O’s adaptive reference updates yield more robust multi-objective alignment as seen in Figure 6.

Sensitivity analysis. As discussed before, the updates in **MOP**O are governed by optimization hyperparameters such as the constraint relaxation factor β and the lag interval t_0 . In practice, robustness to these hyperparameters is desirable, as their tuning can significantly affect performance. Figure 6 shows that **MOP**O remains stable and effective even under suboptimal choices of β and t_0 .

5 Conclusion

In this paper, we introduced **MOP**O, an offline, multi-objective constrained optimization algorithm that learns from preference data and maximizes a primary objective while enforcing tunable lower bound constraints on secondary objectives. On synthetic benchmarks **MOP**O accurately recovers the true Pareto front. Experiments on real-world datasets show that **MOP**O matches or surpasses baselines, and ablation studies prove robustness to hyperparameters. An important future direction is to develop a rigorous theoretical analysis of **MOP**O to provably advance the field of RLHF.

References

- [1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [2] A. Agnihotri, R. Jain, and H. Luo. ACPO: A policy optimization algorithm for average MDPs with constraints. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 397–415. PMLR, 21–27 Jul 2024.
- [3] A. Agnihotri, R. Jain, D. Ramachandran, and S. Singla. e-cop : Episodic constrained optimization of policies. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 110909–110934. Curran Associates, Inc., 2024.
- [4] S. M. Ali and S. D. Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society. Series B (Methodological)*, 28(1):131–142, 1966.
- [5] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- [6] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.
- [7] Z. Chen, Y. Deng, H. Yuan, K. Ji, and Q. Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.
- [8] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.
- [9] I. Csiszár. Eine informationstheoretische ungleichung und ihre anwendung auf beweis der ergodizitaet von markoffschen ketten. *Magyer Tud. Akad. Mat. Kutato Int. Koezl.*, 8:85–108, 1964.
- [10] I. Csiszár. On information-type measure of difference of probability distributions and indirect observations. *Studia Sci. Math. Hungar.*, 2:299–318, 1967.
- [11] R. Durrett. *Probability: Theory and Examples, 4th Edition*. Cambridge University Press, 2010.
- [12] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. Chapman and Hall/CRC, 1994.
- [13] M. Gheshlaghi Azar, Z. Daniel Guo, B. Piot, R. Munos, M. Rowland, M. Valko, and D. Calandriello. A general theoretical paradigm to understand learning from human preferences. In S. Dasgupta, S. Mandt, and Y. Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 4447–4455. PMLR, 02–04 May 2024.
- [14] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- [15] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.
- [16] Y. Guo, G. Cui, L. Yuan, N. Ding, Z. Sun, B. Sun, H. Chen, R. Xie, J. Zhou, Y. Lin, et al. Controllable preference optimization: Toward controllable multi-objective alignment. *arXiv preprint arXiv:2402.19085*, 2024.
- [17] C. F. Hayes, R. Rădulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, 2022.
- [18] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

- [19] J. Hu, L. Tao, J. Yang, and C. Zhou. Aligning language models with offline learning from human feedback. *arXiv preprint arXiv:2308.12050*, 2023.
- [20] S. Huang, A. Abdolmaleki, G. Vezzani, P. Brakel, D. J. Mankowitz, M. Neunert, S. Bohez, Y. Tassa, N. Heess, M. Riedmiller, and R. Hadsell. A constrained multi-objective reinforcement learning framework. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 883–893. PMLR, 08–11 Nov 2022.
- [21] J. Jang, S. Kim, B. Y. Lin, Y. Wang, J. Hessel, L. Zettlemoyer, H. Hajishirzi, Y. Choi, and P. Ammanabrolu. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *CoRR*, abs/2310.11564, 2023.
- [22] R. L. Keeney and H. Raiffa. *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge university press, 1993.
- [23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [24] K. Lee, X. Li, Q. Wang, J. He, J. Ke, M.-H. Yang, I. Essa, J. Shin, F. Yang, and Y. Li. Calibrated multi-preference optimization for aligning diffusion models. *arXiv preprint arXiv:2502.02588*, 2025.
- [25] K. Li, T. Zhang, and R. Wang. Deep reinforcement learning for multiobjective optimization. *IEEE transactions on cybernetics*, 51(6):3103–3114, 2020.
- [26] M. Li, Y. Zhang, W. Wang, W. Shi, Z. Liu, F. Feng, and T.-S. Chua. Self-improvement towards pareto optimality: Mitigating preference conflicts in multi-objective alignment. *arXiv preprint arXiv:2502.14354*, 2025.
- [27] Y. Li, S. Bubeck, R. Eldan, A. Del Giorno, S. Gunasekar, and Y. T. Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.
- [28] R. Liu, Y. Pan, L. Xu, L. Song, P. You, Y. Chen, and J. Bian. C-morl: Multi-objective reinforcement learning through efficient discovery of pareto front. *arXiv preprint arXiv:2410.02236*, 2024.
- [29] C. Mavromatis, P. Karypis, and G. Karypis. Pack of llms: Model fusion at test-time via perplexity optimization. *arXiv preprint arXiv:2404.11531*, 2024.
- [30] K. Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 1999.
- [31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- [32] S. Mukherjee, A. Lalitha, S. Sengupta, A. Deshmukh, and B. Kveton. Multi-objective alignment of large language models through hypervolume maximization. *arXiv preprint arXiv:2412.05469*, 2024.
- [33] R. Munos, M. Valko, D. Calandriello, M. G. Azar, M. Rowland, Z. D. Guo, Y. Tang, M. Geist, T. Mesnard, A. Michi, et al. Nash learning from human feedback. *arXiv preprint arXiv:2312.00886*, 18, 2023.
- [34] Y. Nesterov. *Lectures on Convex Optimization*. Springer Publishing Company, Incorporated, 2nd edition, 2018.
- [35] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- [36] R. Y. Pang, W. Yuan, H. He, K. Cho, S. Sukhbaatar, and J. Weston. Iterative reasoning preference optimization. *Advances in Neural Information Processing Systems*, 37:116617–116637, 2024.
- [37] B. Peng, C. Li, P. He, M. Galley, and J. Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.
- [38] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

- [39] A. Rame, G. Couairon, C. Dancette, J.-B. Gaya, M. Shukor, L. Soulier, and M. Cord. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. *Advances in Neural Information Processing Systems*, 36:71095–71134, 2023.
- [40] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.
- [41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [42] R. Shi, Y. Chen, Y. Hu, A. Liu, H. Hajishirzi, N. A. Smith, and S. S. Du. Decoding-time language model alignment with multiple objectives. *Advances in Neural Information Processing Systems*, 37:48875–48920, 2024.
- [43] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.
- [44] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.
- [45] K. Van Moffaert and A. Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.
- [46] L. von Werra, Y. Belkada, L. Tunstall, E. Beeching, T. Thrush, N. Lambert, and S. Huang. Trl: Transformer reinforcement learning. <https://github.com/huggingface/trl>, 2020.
- [47] L. von Werra, Y. Belkada, L. Tunstall, E. Beeching, T. Thrush, N. Lambert, S. Huang, K. Rasul, and Q. Gallouédec. Trl: Transformer reinforcement learning. <https://github.com/huggingface/trl>, 2020.
- [48] C. Wang, Y. Jiang, C. Yang, H. Liu, and Y. Chen. Beyond reverse KL: Generalizing direct preference optimization with diverse divergence constraints. In *The Twelfth International Conference on Learning Representations*, 2024.
- [49] H. Wang, Y. Lin, W. Xiong, R. Yang, S. Diao, S. Qiu, H. Zhao, and T. Zhang. Arithmetic control of llms for diverse user preferences: Directional preference alignment with multi-objective rewards. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8642–8655, 2024.
- [50] H. Wang, Y. Lin, W. Xiong, R. Yang, S. Diao, S. Qiu, H. Zhao, and T. Zhang. Arithmetic control of llms for diverse user preferences: Directional preference alignment with multi-objective rewards. *CoRR*, abs/2402.18571, 2024.
- [51] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.
- [52] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. G. Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, and L. Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 2022.
- [53] Z. Wu, Y. Hu, W. Shi, N. Dziri, A. Suhr, P. Ammanabrolu, N. A. Smith, M. Ostendorf, and H. Hajishirzi. Fine-grained human feedback gives better rewards for language model training. *Advances in Neural Information Processing Systems*, 36:59008–59033, 2023.
- [54] R. Yang, X. Pan, F. Luo, S. Qiu, H. Zhong, D. Yu, and J. Chen. Rewards-in-context: Multi-objective alignment of foundation models with dynamic preference adjustment, 2024.
- [55] Z. Yuan, H. Yuan, C. Tan, W. Wang, S. Huang, and F. Huang. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023.
- [56] R. Zhang, J. Han, C. Liu, P. Gao, A. Zhou, X. Hu, S. Yan, P. Lu, H. Li, and Y. Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023.
- [57] T. Zhang, S. G. Patil, N. Jain, S. Shen, M. Zaharia, I. Stoica, and J. E. Gonzalez. Raft: Adapting language model to domain specific rag. In *First Conference on Language Modeling*, 2024.

- [58] Y. Zhong, C. Ma, X. Zhang, Z. Yang, H. Chen, Q. Zhang, S. Qi, and Y. Yang. Panacea: Pareto alignment via preference adaptation for llms. *Advances in Neural Information Processing Systems*, 37:75522–75558, 2024.
- [59] Z. Zhou, J. Liu, J. Shao, X. Yue, C. Yang, W. Ouyang, and Y. Qiao. Beyond one-preference-fits-all alignment: Multi-objective direct preference optimization. *arXiv preprint arXiv:2310.03708*, 2023.
- [60] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Appendix

Definition A.1. For any two policies $P, Q \in \Delta_{\mathcal{Y}}^{\mathcal{X}}$ such that $\text{Supp}(P) = \text{Supp}(Q)$, their KL divergence is defined as:

$$\text{KL}(P || Q) = \mathbb{E}_{\substack{x \sim \nu \\ y \sim P(\cdot|x)}} \left[\log \left(\frac{P(y|x)}{Q(y|x)} \right) \right].$$

A.1 A Detailed Motivating Example

In this example, we empirically demonstrate the necessity of *principled* multi-objective optimization methods that account for multi-dimensional preferences. We benchmark various approaches for multi-objective alignment and show that existing state-of-the-art techniques consistently fail to reach the Pareto front. To ensure clarity, we conduct experiments on synthetic datasets where the true Pareto front is known, allowing for precise evaluation of alignment quality.

Now define the input space $\mathcal{X} := \mathcal{U}[0, 1]$ and output space $\mathcal{Y} := \mathcal{U}[0, 1]$. For input $x \in \mathcal{X}$ and outputs $y, y' \in \mathcal{Y}$, the Bradley-Terry preference model $\text{BT}(r(\cdot))$ [6] w.r.t. to a reward model $r(\cdot)$ provides preference z as,

$$\Pr(z = y | x, y, y') = r(x, y) / (r(x, y) + r(x, y')) \quad (14)$$

Reward models. For $(x, y) \in \mathcal{X} \times \mathcal{Y}$, we consider two pairs of reward functions $r : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$:

$$\begin{aligned} r_1^A(x, y) &= e^x + \sqrt{y} - y \quad \text{and} \quad r_2^A(x, y) = -\sin(x) - y^2. \\ r_1^B(x, y) &= (x + y)^2 \quad \text{and} \quad r_2^B(x, y) = \log\left(\frac{1+x}{1+y}\right). \end{aligned} \quad (15)$$

Dataset construction. For arbitrary $N \in \mathbb{N}$, generate $x_i, y_i, y'_i \sim \mathcal{U}[0, 1]$ for $i \in [N]$. Let $z_i^{(1)} \sim \text{BT}(r_1)$ and $z_i^{(2)} \sim \text{BT}(r_2)$ with $z_i^{(1)}, z_i^{(2)} \in \{y_i, y'_i\}$. We now construct four datasets: (i) $\mathcal{D}_1 = \{(x_i, y_i, y'_i, z_i^{(1)})\}_{i=1}^N$ incorporating preferences w.r.t. reward model $r_1(\cdot)$ *only*, (ii) $\mathcal{D}_2 = \{(x_i, y_i, y'_i, z_i^{(2)})\}_{i=1}^N$ incorporating preferences w.r.t. reward model $r_2(\cdot)$ *only*, (iii) $\mathcal{D}_J = \{(x_i, y_i, y'_i, z) : z = z_i^{(1)} = z_i^{(2)}\}_{i=1}^N$ incorporating preferences *only if they are consistent with reward models $r_1(\cdot)$ and $r_2(\cdot)$* , and (iv) $\mathcal{D}_C = \{(x_i, y_i, y'_i, z_i^{(C)}) : z_i^{(C)} \sim \text{BT}(wr_1 + (1-w)r_2)\}_{i=1}^N$ for some $w \in [0, 1]$, which incorporates preferences based on some convex weighting of both reward models.

Another approach of solving the multi-objective preference problem is a constrained optimization approach **COP**, where we can solve for the optimal policy as $\pi_{\text{COP}}(x) = \operatorname{argmax}_y r_1(x, y) \text{ s.t. } r_2(x, y) \geq b$ for some $b \in \mathbb{R}$. Now, given the four datasets and the constrained optimization approach, we wish to compare learning the optimal policy as described by a Pareto frontier in the (r_1, r_2) space. We train a neural network policy with DPO [38] for each of the four datasets, and solve a constrained optimization problem for the **COP** approach. See Figure 7(a) for empirical results under reward model sets *A* (left) and *B* (right).

While it is somewhat trivial to see why learning from \mathcal{D}_1 and \mathcal{D}_2 alone yields suboptimal rewards, the case for the jointly preferred dataset \mathcal{D}_J and a convex weighted reward model dataset \mathcal{D}_C is not obvious. The issue with \mathcal{D}_J is that it only contains samples where r_1 and r_2 agree, effectively discarding all points that exhibit a meaningful trade-off between the two objectives. This results in a sparse and biased preference signal that does not span the entire Pareto front. In contrast, \mathcal{D}_C encodes preferences with respect to a *single* scalarized reward model, which inherently biases learning toward one specific convex combination of the objectives. While more sophisticated approaches have been proposed for learning from multi-dimensional preferences - such as RiC [54], MODPO [59], Rewarded Soups [39], and SIPO [26] - they remain fundamentally limited in their expressivity. Ultimately, each method relies on learning with respect to a *single* scalarized reward signal of the form $\tilde{r} = f(r_1, r_2)$, where the function f varies across methods. As a result, these approaches do not recover the full structure of the underlying preference landscape and cannot characterize the Pareto front in the multi-objective setting.

In contrast, constrained optimization (**COP**) over r_1 and r_2 yields solutions that lie close to the true Pareto front. This highlights the need for optimization methods that explicitly account for trade-offs

across objectives, rather than collapsing them into a single reward signal, in order to fully leverage multi-dimensional preference data. **MOPPO** follows this principle by directly optimizing within the multi-objective space, and empirically achieves solutions that approach the Pareto front as in Figure 7(b).

We further empirically validate the correctness of **MOPPO** and consider whether it is able to generalize and regularize effectively w.r.t. the reference policy. See Figure 8 for comparison of the policy learned through **MOPPO** under various regularization values. We observe that even with an uninformed π_{ref} , **MOPPO** is able to push toward the Pareto frontier and is limited only by the strength of regularization.

Remark A.1.1. Note that reward models are only used for evaluation, and are not assumptions or requirements to finetune policies using **MOPPO**. **MOPPO** learns strictly from preference data, without: (i) assuming the existence of a mapping from preferences to pointwise rewards, and (ii) learning this mapping (reward model) from preference data.

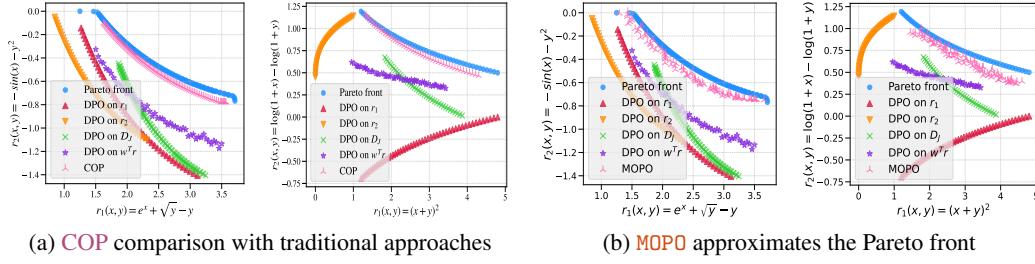


Figure 7: Illustration of how a COP approach, and hence **MOPPO**, achieves Pareto-optimal alignment in comparison with DPO on \mathcal{D}_1 , \mathcal{D}_2 , \mathcal{D}_3 , and \mathcal{D}_C under two sets of reward models.

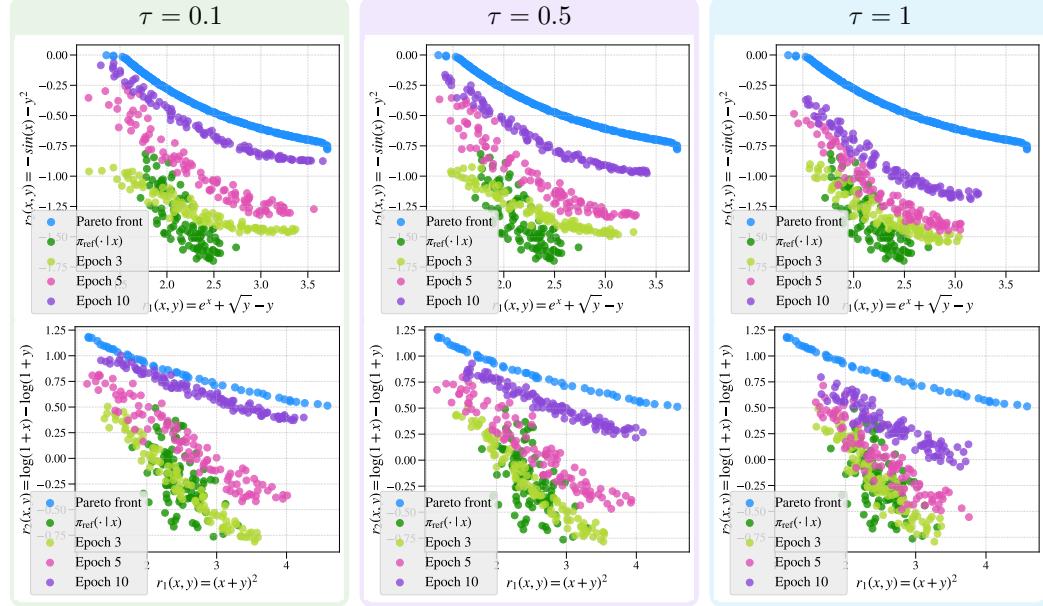


Figure 8: Comparison of the KL-regularized policy learned using **MOPPO** with the reference policy π_{ref} and the Pareto frontier, visualized in the reward space for two reward model pairs, A (top) and B (bottom).

A.2 Sub-optimality of baselines.

To begin, we list some common divergence metrics that have been used in literature to characterize two probability distributions. Then, we follow with some definitions before listing the main proofs for the sub-optimality of baselines.

A.2.1 Divergence measures and closed-form policies

We acknowledge that commonly used f -divergence measures have been introduced in [48, 42] and show them here for completeness:

Divergence measure	$f(x)$	$\nabla f(x)$	barrier function
Reverse KL-divergence	$x \log x$	$\log x + 1$	✓
Forward KL-divergence	$-\log x$	$-1/x$	✓
JSD	$x \log x - (x+1) \log \frac{x+1}{2}$	$\log \frac{2x}{1+x}$	✓
α -divergence	$\frac{x^{1-\alpha} - (1-\alpha)x^{-\alpha}}{\alpha(1-\alpha)}$	$(1-x^{-\alpha})/\alpha$	✓
Jeffery divergence	$x \log x - \log x$	$\log x - \frac{1}{x} + 1$	✓
Total Variation	$ x-1 /2$	$\text{sgn}(x-1)/2$	✗
Chi-squared	$(x-1)^2$	$2(x-1)$	✗

Here we show the optimal sampling policies for multi-objective w.r.t. these divergence measures:

Divergence measure	Optimal policy
Reverse KL-divergence	$\left(\prod_{i=1}^K \pi_i(y x)^{w_i}\right) \cdot \exp(-Z(x))$
Forward KL-divergence	$\pi_{\text{ref}}(y x) \cdot \left(Z(x) + \sum_{i=1}^K \frac{w_i \pi_{\text{ref}}(y x)}{\pi_i(y x)}\right)^{-1}$
JSD	$\pi_{\text{ref}}(y x) \cdot \left(-1 + \exp(Z(x)) \prod_{i=1}^K \left(\frac{\pi_{\text{ref}}(y x)}{\pi_i(y x)} + 1\right)^{w_i}\right)^{-1}$
α -divergence	$\pi_{\text{ref}}(y x) \cdot \left(\alpha Z(x) + \sum_{i=1}^K w_i \left(\frac{\pi_{\text{ref}}(y x)}{\pi_i(y x)}\right)^{\alpha}\right)^{-\frac{1}{\alpha}}$

A.2.2 Definitions

We first begin with some definitions.

Definition A.2.1 (f -divergence [4, 9, 10]). *For probability measures P and Q , let μ be a dominating measure of P and Q (i.e. $P, Q \ll \mu$), and let p, q be the Radon-Nikodym derivative [11] $\frac{dP}{d\mu}, \frac{dQ}{d\mu}$ respectively. For simplicity, here we assume $q > 0$ almost surely. Then f -divergence from P to Q is defined as*

$$I_f(p\|q) := \int qf\left(\frac{p}{q}\right) d\mu,$$

where f is convex on \mathbb{R}_+ , satisfying $f(1) = 0$. Most useful divergence measures are included in f -divergences, and the commonly used ones and corresponding f are introduced in Appendix A.2.1.

Definition A.2.2 (Barrier function [34]). *Given conditions satisfied in Definition A.2.1, if additionally $0 \notin \text{dom}(\nabla f)$, then f is a barrier function. If a barrier function f is continuously differentiable and strongly convex on \mathbb{R}_+ , then f is a strongly convex and smooth barrier function (abbreviated as strong-barrier function).*

Definition A.2.3 (Expected calibration error [15, 48]). *Denote the ground truth distribution as \mathbb{P} , context as X and response as Y . The expected calibration error of a stochastic policy π is defined as*

$$\text{ECE}(\pi) := \mathbb{E}_{\substack{x \sim \mathcal{X} \\ y \sim \pi(\cdot|x)}} |\mathbb{P}(Y=y|X=x) - \pi(y|x)|.$$

Hypothesis 1 (Reducible reward misspecification [52, 39, 21]). *Let θ_k be the parameter of the optimal policy for objective value J_k , $\forall k \in [K]$, and θ_w^* be the parameter of the optimal policy for the interpolated objective $\sum_{k=1}^K w_k \cdot J_k$, then this hypothesis claims that*

$$\theta_w^* \in \left\{ \sum_{k=1}^K \lambda_k \cdot \theta_k, \lambda \in \Delta^{K-1} \right\}, \quad \forall w \in \Delta^{K-1}.$$

Extending the results of [48] to the multi-objective setting, we prove the necessity of f being barrier functions to find an optimal policy π^* for multi-objective alignment. We refer the reader to [42] for a complete discussion.

Theorem A.2.4. *If f is not a barrier function, then for $\forall C \in \mathbb{R}_+$, $N \in \mathbb{Z}_{>4}$, $K \in \mathbb{Z}_{\geq 2}$, $\mathcal{Y} = \{y_i\}_{i=1}^N$, any multi-objective decoding or merging algorithm $\mathcal{A} : \mathcal{S}^{K+1} \times \Delta^{K-1} \rightarrow \mathcal{S}$, there exists a reference policy π_{ref} , policies $\{\pi_i\}_{i=1}^K$ and π' , reward functions $\{\mathcal{R}_i\}_{i=1}^K$, preference weightings $w \in \Delta^{K-1}$ and $\tau \in \mathbb{R}_+$, s.t. π_i is the optimal policy for \mathcal{R}_i w.r.t. $\tau \cdot I_f(\cdot \| \pi_{\text{ref}})$, $\forall i \in [K]$, but*

$$\begin{aligned} \mathbb{E}_{y \sim \pi_{\mathcal{A},w}} \left[\sum_{i=1}^K w_i \mathcal{R}_i(y) \right] &\leq \mathbb{E}_{y \sim \pi'} \left[\sum_{i=1}^K w_i \mathcal{R}_i(y) \right] - C, \text{ and} \\ \mathbb{E}_{y \sim \pi_{\mathcal{A},w}} \left[\sum_{i=1}^K w_i \mathcal{R}_i(y) \right] - \tau I_f(\pi_{\mathcal{A},w} \| \pi_{\text{ref}}) &\leq \mathbb{E}_{y \sim \pi'} \left[\sum_{i=1}^K w_i \mathcal{R}_i(y) \right] - \tau I_f(\pi' \| \pi_{\text{ref}}) - C, \end{aligned}$$

where $\pi_{\mathcal{A},w}(y) := \mathcal{A}(\pi_{\text{ref}}, \pi_1, \pi_2, \dots, \pi_K, w)(y)$.

Remark A.2.5 (Motivating example). *Here we provide a motivating example where $f \equiv 0$: let $K = 4$, $\mathcal{R}_1(y_1) = \mathcal{R}_2(y_2) = 1$, $\mathcal{R}_1(y_2) = \mathcal{R}_2(y_1) = -1$, $\mathcal{R}_1(y_{3+k}) = \mathcal{R}_2(y_{3+k}) = 0$, $\mathcal{R}_1(y_{4-k}) = \mathcal{R}_2(y_{4-k}) = 1/2$, where $k \in \{0, 1\}$. Then the optimal policy for \mathcal{R}_1 is $\pi_1(y_i) := \delta_{1i}$, for \mathcal{R}_2 is $\pi_2(y_i) := \delta_{2i}$, and for $\mathcal{R}_1/2 + \mathcal{R}_2/2$ is $\pi^*(y_i) := \delta_{4-k,i}$. Thus $\pi_{\mathcal{A},w}$ cannot fit π^* both for $k = 0, 1$.*

Proof. Since f is not a barrier function, $0 \in \text{dom}(\nabla f)$. Now we can define $p := \max_{x \in [0,N]} \nabla f(x)$, $q := \min_{x \in [0,N]} \nabla f(x)$, $r := \max_{x \in [0,N]} f(x) - \min_{x \in [0,N]} f(x)$, $s := \frac{N-2}{N-3} \cdot C$. Let $w = (0.5, 0.5, \underbrace{0, \dots, 0}_{N-2})$, and we pick $k = \operatorname{argmin}_{j \in \{3,4,\dots,N\}} \pi_{\mathcal{A},w}(y_j)$. Let $\pi_{\text{ref}}(y_i) = \frac{1}{N}$, $\pi_1(y_i) = \delta_{1i}$, $\pi_2(y_i) = \delta_{2i}$, $\pi_j(y_i) = \frac{1}{N}$ and $\pi'(y_i) = \delta_{ik}$, $\forall i \in [N]$, $j \in \{3, 4, \dots, K\}$. And set $\mathcal{R}_1(y_i) = \begin{cases} 2p + 2r + 2s & i = 1 \\ 4q - 2p - 2r - 2s & i = 2 \\ p + q + r + s & i = k \\ 2q & \text{o/w} \end{cases}$, and $\mathcal{R}_2(y_i) = \begin{cases} 4q - 2p - 2r - 2s & i = 1 \\ 2p + 2r + 2s & i = 2 \\ p + q + r + s & i = k \\ 2q & \text{o/w} \end{cases}$, and $\mathcal{R}_j \equiv 0$, $\forall j \in \{3, 4, \dots, K\}$.

Let $\tau = 1$, then the optimization objective for \mathcal{R}_1 w.r.t. I_f is $J_1(\pi) := \mathbb{E}_{y \sim \pi} [\mathcal{R}_1(y)] - I_f(\pi \| \pi_{\text{ref}})$, and the Lagrangian dual is

$$\mathcal{L}_1(\pi) := \sum_{i=1}^N \left(-\mathcal{R}_1(y_i) \cdot \pi(y_i) + \frac{1}{N} f(N \cdot \pi(y_i)) \right) + \lambda \left(\sum_{i=1}^N \pi(y_i) - 1 \right) - \sum_{i=1}^N \mu_i \pi(y_i).$$

As the objective is convex and the constraints are affine, we can directly apply the *Karush-Kuhn-Tucker conditions* [34]:

$$\nabla \mathcal{L}_1(\pi_1^*) = 0, \tag{16}$$

$$\begin{aligned} \sum_{i=1}^N \pi_1^*(y_i) &= 1, \\ \pi_1^*(y_i) &\geq 0, \\ \mu_i^* &\geq 0, \\ \mu_i^* \pi_1^*(y_i) &= 0. \end{aligned} \tag{17}$$

Equation (16) implies

$$-\mathcal{R}_1(y_1) + \nabla f(N \cdot \pi_1^*(y_1)) + \lambda^* - \mu_1^* = 0.$$

If $\pi_1^*(y_1) > 0$, we have

$$\begin{aligned} \lambda^* &= \mathcal{R}_1(y_1) - \nabla f(N \cdot \pi_1^*(y_1)) \\ &\geq p + 2r + 2s, \end{aligned}$$

and then for $\forall j \neq 1$,

$$\begin{aligned}\mu_j^* &= -\mathcal{R}_1(y_j) + \nabla f(N \cdot \pi_1^*(y_j)) + \lambda^* \\ &\geq -p - q - r - s + q + p + 2r + 2s \\ &= r + s \\ &> 0.\end{aligned}$$

Combining it with Equation (17) yields $\pi_1^*(y_j) = 0$ for $\forall j \neq 1$, which is exactly π_1 . Note that we have

$$J(\pi_1) \geq 2p + 2r + 2s - \max_{x \in [0, N]} f(x).$$

For any π' with $\pi'(y_1) = 0$, we have

$$\begin{aligned}J(\pi') &\leq p + q + r + s - \min_{x \in [0, N]} f(x) \\ &= p + q + 2r + s - \max_{x \in [0, N]} f(x) \\ &< J(\pi_1).\end{aligned}$$

Thus π_1 is the optimal policy for \mathcal{R}_1 w.r.t. $I_f(\cdot \|\pi_{\text{ref}})$. Similarly, π_2 is the optimal policy for \mathcal{R}_2 w.r.t. $I_f(\cdot \|\pi_{\text{ref}})$. By convexity of f , the minimum of $I_f(\pi \|\pi_{\text{ref}})$ is obtained when $\pi = \pi_{\text{ref}}$, and thus π_j is the optimal policy for \mathcal{R}_j w.r.t. $I_f(\cdot \|\pi_{\text{ref}})$, for $\forall j \in \{3, 4, \dots, K\}$. Therefore, all conditions are well satisfied by this construction. Note that

$$\mathbb{E}_{y \sim \pi'} \left[\sum_{i=1}^K w_i \mathcal{R}_i(y) \right] = p + q + r + s. \quad (18)$$

While by the selection of k , we have

$$\mathbb{E}_{y \sim \pi_{\mathcal{A}, w}} \left[\sum_{i=1}^K w_i \mathcal{R}_i(y) \right] \leq \frac{(N-3) \cdot 2q + p + q + r + s}{N-2}. \quad (19)$$

Comparing Equation (18) with Equation (19), we have

$$\begin{aligned}\mathbb{E}_{y \sim \pi_{\mathcal{A}, w}} \left[\sum_{i=1}^K w_i \mathcal{R}_i(y) \right] &\leq \mathbb{E}_{y \sim \pi'} \left[\sum_{i=1}^K w_i \mathcal{R}_i(y) \right] - \frac{N-3}{N-2}s \\ &= \mathbb{E}_{y \sim \pi'} \left[\sum_{i=1}^K w_i \mathcal{R}_i(y) \right] - C.\end{aligned}$$

Note that π_{ref} is a uniform distribution and both $\pi_{\mathcal{A}, w}, \pi'$ are one-point distributions, thus $I_f(\pi_{\mathcal{A}, w} \|\pi_{\text{ref}}) = I_f(\pi' \|\pi_{\text{ref}})$. We have

$$\mathbb{E}_{y \sim \pi_{\mathcal{A}, w}} \left[\sum_{i=1}^K w_i \mathcal{R}_i(y) \right] - I_f(\pi_{\mathcal{A}, w} \|\pi_{\text{ref}}) \leq \mathbb{E}_{y \sim \pi'} \left[\sum_{i=1}^K w_i \mathcal{R}_i(y) \right] - I_f(\pi' \|\pi_{\text{ref}}) - C. \quad \square$$

A.2.3 Baselines are not Pareto-optimal

Given the necessity of f being a barrier-function, we now show how parameter-merging paradigm algorithms ([16, 54, 21, 59]) fail to achieve Pareto-optimality. The optimality of parameter-merging paradigm primarily relies on reduced reward mis-specification hypothesis (see Hypothesis 1). The following theorem demonstrates that this hypothesis does not hold for almost all f -divergence regularized policies.

Theorem A.2.6. *For any f -divergence satisfying one of the following conditions: (i) f is not a barrier function; (ii) I_f is Reverse KL-divergence; (iii) f is a strong-barrier function, with finite roots of*

$$2\nabla f \left(\frac{3\sqrt{1-2x}}{2\sqrt{1-2x} + \sqrt{x}} \right) - 2\nabla f \left(\frac{3\sqrt{x}}{2\sqrt{1-2x} + \sqrt{x}} \right) - \nabla f(3-6x) + \nabla f(3x),$$

$\exists N, K \in \mathbb{N}, \mathcal{Y} = \{y_i\}_{i=1}^N, \tau \in \mathbb{R}_+$, a neural network $nn = \text{softmax}(h_\theta(z_0))$ where $z_0 \in \mathbb{R}^n$ and $h_\theta : \mathbb{R}^n \rightarrow \mathbb{R}^N$ is a continuous mapping, preference weightings $w \in \Delta^{K-1}$, reference policy π_{ref} , and the objectives J_1, J_2, \dots, J_K representing reward functions $\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_K$ w.r.t. $\tau \cdot I_f(\cdot \| \pi_{\text{ref}})$, s.t. Hypothesis 1 does not hold.

Proof. (i) If f is not a barrier function, Hypothesis 1 does not hold immediately from Theorem A.2.4.

(ii) If I_f is Reverse KL-divergence, we let $N = 3, K = 3$, and $h_\theta(z_0) = W_\theta^{(2)} \sigma(W_\theta^{(1)} z_0)$, where σ is ReLU(\cdot). We set $\mathcal{R}_i(y_j) = \delta_{ij}$, $\pi_{\text{ref}}(y_i) = 1/3$ for $\forall i, j \in [3]$, $z_0 = 1$ and $\tau = 1$. Then the optimal policies are $W_{\theta_1}^{(1)} = e_1, W_{\theta_1}^{(2)} = \begin{pmatrix} 100 \\ 000 \\ 000 \end{pmatrix}$ for \mathcal{R}_1 w.r.t. $\text{KL} \cdot \pi_{\text{ref}}$, $W_{\theta_2}^{(1)} = e_2, W_{\theta_2}^{(2)} = \begin{pmatrix} 000 \\ 010 \\ 000 \end{pmatrix}$ for \mathcal{R}_2 w.r.t. $\text{KL} \cdot \pi_{\text{ref}}$, and $W_{\theta_3}^{(1)} = e_3, W_{\theta_3}^{(2)} = \begin{pmatrix} 000 \\ 000 \\ 001 \end{pmatrix}$ for \mathcal{R}_3 w.r.t. $\text{KL} \cdot \pi_{\text{ref}}$. Thus we have $h_{\sum_{j=1}^3 \lambda_j \theta_j}(z_0) = (\lambda_1^2, \lambda_2^2, \lambda_3^2)^\top$. Given $w = (0, 1/3, 2/3)$, the optimal policy π^* should output $\pi^*(y_1) = \frac{1}{1+\exp(1/3)+\exp(2/3)}$, $\pi^*(y_2) = \frac{\exp(1/3)}{1+\exp(1/3)+\exp(2/3)}$ and $\pi^*(y_3) = \frac{\exp(2/3)}{1+\exp(1/3)+\exp(2/3)}$. Note that

$$\sqrt{t} + \sqrt{t+1/3} + \sqrt{t+2/3} > 1, \quad \forall t \in \mathbb{R}_+,$$

thus there is no solution $\lambda \in \Delta^2, t \in \mathbb{R}_+$ for $(\lambda_1^2, \lambda_2^2, \lambda_3^2)^\top = (t, t + \frac{1}{3}, t + \frac{2}{3})^\top$, i.e. there is no λ s.t. $\text{softmax}\left(h_{\sum_{j=1}^3 \lambda_j \theta_j}(z_0)\right) = (\pi^*(y_1), \pi^*(y_2), \pi^*(y_3))$, i.e. Hypothesis 1 does not hold.

(iii) If f is a strong-barrier function, with finite roots of

$$2\nabla f\left(\frac{3\sqrt{1-2x}}{2\sqrt{1-2x}+\sqrt{x}}\right) - 2\nabla f\left(\frac{3\sqrt{x}}{2\sqrt{1-2x}+\sqrt{x}}\right) - \nabla f(3-6x) + \nabla f(3x),$$

we let $N = 3, K = 2, h_\theta(z_0) = W_\theta(z_0), z_0 = 1, \mathcal{R}_1(y_i) = \delta_{1i}, \mathcal{R}_2(y_i) = \delta_{2i}$ and $\pi_{\text{ref}}(y_i) = 1/3$, for $\forall i \in [3]$. The optimal policy for J_1 is $\pi_{\theta_1}(y_i) = \frac{1}{3}(\nabla f)^{(-1)}\left(\frac{1}{\tau}\delta_{1i} - Z\right)$, and the optimal policy for J_2 is $\pi_{\theta_2}(y_i) = \frac{1}{3}(\nabla f)^{(-1)}\left(\frac{1}{\tau}\delta_{2i} - Z\right)$, where Z is the normalization factor. And these policies can be learned by setting $W_{\theta_i} = (\log \pi_{\theta_i}(y_1), \log \pi_{\theta_i}(y_2), \log \pi_{\theta_i}(y_3))^\top$.

We set $a := \pi_{\theta_1}(y_1) = \frac{1}{3}(\nabla f)^{(-1)}\left(\frac{1}{\tau} - Z\right)$, $b := \pi_{\theta_1}(y_2) = \pi_{\theta_1}(y_3) = \frac{1}{3}(\nabla f)^{(-1)}(-Z)$. Thus we have

$$\nabla f(3a) - \nabla f(3b) = \frac{1}{\tau}, \tag{20}$$

$$a + 2b = 1. \tag{21}$$

The optimal policy for $w_1 \cdot J_1 + w_2 \cdot J_2$ (see [42] for proof) is

$$\pi_w^*(y_i) = \frac{1}{3}(\nabla f)^{(-1)}\left(-Z_w^* + \frac{w_1}{\tau}\delta_{1i} + \frac{w_2}{\tau}\delta_{2i}\right), \tag{22}$$

where Z_w^* is the normalization factor. By linearly merging the weights of π_{θ_1} and π_{θ_2} , we have

$$\begin{aligned} \pi_{\lambda_1 \theta_1 + \lambda_2 \theta_2}(y_i) &= \text{softmax}(\lambda_1 W_{\theta_1}(z_0) + \lambda_2 W_{\theta_2}(z_0))(y_i) \\ &= \frac{1}{Z_\lambda} \left((\nabla f)^{(-1)}\left(\frac{1}{\tau}\delta_{1i} - Z\right) \right)^{\lambda_1} \left((\nabla f)^{(-1)}\left(\frac{1}{\tau}\delta_{2i} - Z\right) \right)^{\lambda_2}, \end{aligned} \tag{23}$$

where Z_λ is the normalization factor.

With symmetry, Equation (22), (23) and Hypothesis 1 indicate that $\pi_{\frac{1}{2}\theta_1 + \frac{1}{2}\theta_2} = \pi_{(\frac{1}{2}, \frac{1}{2})}^*$, thus

$$\frac{1}{3}(\nabla f)^{(-1)}\left(-Z_{(0.5, 0.5)}^* + \frac{1}{2\tau}\right) = \frac{\sqrt{a}}{2\sqrt{a} + \sqrt{b}},$$

$$\frac{1}{3}(\nabla f)^{(-1)}\left(-Z_{(0.5, 0.5)}^*\right) = \frac{\sqrt{b}}{2\sqrt{a} + \sqrt{b}},$$

and combining them with Equation (20) yields

$$2\nabla f\left(\frac{3\sqrt{a}}{2\sqrt{a}+\sqrt{b}}\right)-2\nabla f\left(\frac{3\sqrt{b}}{2\sqrt{a}+\sqrt{b}}\right)=\nabla f(3a)-\nabla f(3b). \quad (24)$$

Given the condition, the solution set (a, b) to Equation (21), (24) is finite, thus there exists $\tau \in \mathbb{R}_+$ s.t. Equation (20) does not hold, implying that Hypothesis 1 does not hold. \square

A.3 Constraint Threshold Initialization

We begin this section by describing an equivalence between the preference-learning based optimization problem and a reward-learning based optimization problem. Note that **MOP**O does not assume this equivalence, and directly works with preference data, and this equivalence is established for analysis only. Following this equivalence discussion, we specify a provable method for setting constraint thresholds b such that the optimal solution learned by **MOP**O is a Pareto-optimal solution.

For all preference instances in the preference dataset, there exists an underlying, unknown reward model based on which preferences are provided. For all contexts-output pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, let the reward model for the k^{th} objective be $r_k(x, y) \in \mathbb{R}$ for $k \in [K]$.

Preference–reward link. For every objective $k \in [K]$ there exists a *strictly increasing* function $\phi_k : \mathbb{R} \rightarrow \mathbb{R}$ and a strictly increasing transfer function $\sigma : \mathbb{R} \rightarrow (0, 1)$ such that for all contexts $x \in \mathcal{X}$ and actions $y, y' \in \mathcal{Y}$ we have,

$$p_k(y \succ y' | x) = \sigma(\phi_k(r_k(x, y) - r_k(x, y'))). \quad (25)$$

Now, with respect to the k^{th} objective, let the expected reward $R_k(\pi)$ and preference-based objective value $F_k(\pi)$ for a policy be given by,

$$\begin{aligned} R_k(\pi) &:= \mathbb{E}_{x \sim \nu, y \sim \pi(\cdot|x)}[r_k(x, y)] \text{ for } k \in [K], \\ F_k(\pi) &:= \mathbb{E}_{x \sim \nu, y \sim \pi(\cdot|x), y' \sim \mu(\cdot|x)}[p_k(y \succ y' | x)], \quad k \in [K-1], \\ F_K(\pi) &:= \mathbb{E}_{x \sim \nu, y \sim \pi(\cdot|x), y' \sim \mu(\cdot|x)}[p(y \succ y' | x)] - \tau \text{KL}(\pi \| \pi^{\text{ref}}). \end{aligned}$$

Lemma A.3.1 (Order preservation). *Under Equation (25) and some $u \in \mathbb{R}$, the mapping $H_k(u) := \mathbb{E}_{z \sim \text{Unif}[-u, u]}[\sigma \circ \phi_k(z)]$ is strictly increasing. Moreover, for every policy π*

$$F_k(\pi) = H_k(R_k(\pi) - R_k(\mu)), \quad k \in [K-1],$$

so that for any π, π' , we have $R_k(\pi) \geq R_k(\pi') \iff F_k(\pi) \geq F_k(\pi')$.

Proof. Fix k . By Equation (25), $\sigma \circ \phi_k$ is strictly increasing, hence so is its odd extension $z \mapsto \sigma \circ \phi_k(z)$. For $U := R_k(\pi) - R_k(\mu)$ let $z := r_k(x, y) - r_k(x, y')$. Because $(y, y') \sim (\pi, \mu)$ are independent, z is symmetrically distributed around U and $z \sim \text{Unif}[U - \delta, U + \delta]$ for some $\delta > 0$ that does not depend on U . Taking expectation yields $F_k(\pi) = H_k(U)$, and strict monotonicity of H_k follows from strict monotonicity of $\sigma \circ \phi_k$. \square

Note there that we introduced $H_k(u) = \mathbb{E}_{z \sim \text{Unif}[-u, u]}[\sigma \circ \phi_k(z)]$. The uniform law is chosen purely for notational brevity; the proof requires only that the base distribution be symmetric and shifted by the reward gap $u = R_k(\pi) - R_k(\mu)$. Consequently, one may replace $\text{Unif}[-u, u]$ by any symmetric density $\rho_u(z) = \rho(z - u)$, and define $H_k(u) = \mathbb{E}_{z \sim \rho_u}[\sigma \circ \phi_k(z)]$. Strict monotonicity of $\sigma \circ \phi_k$ then guarantees that the Lemma A.3.1 holds. Now, let $\mathbf{b} \in [0, 1]^{K-1}$ be the probability thresholds in **COP**, let $\mathbf{F}_{1:K-1}(\pi) = (F_1(\pi), \dots, F_{K-1}(\pi))$ and $\mathbf{R}_{1:K-1}(\pi) = (R_1(\pi), \dots, R_{K-1}(\pi))$, and define component-wise

$$c_k := H_k^{-1}(b_k) + R_k(\mu), \quad k \in [K-1], \quad \mathbf{c} := (c_1, \dots, c_{K-1}).$$

By Lemma A.3.1, $F_k(\pi) \geq b_k \iff R_k(\pi) \geq c_k$. Hence the preference-space constrained problem

$$\max_{\pi} F_K(\pi) \text{ s.t. } \mathbf{F}_{1:K-1}(\pi) \geq \mathbf{b} \quad (\text{COP})$$

is equivalent to the reward-space problem

$$\max_{\pi} \{R_K(\pi) - \tau \text{KL}(\pi \parallel \pi^{\text{ref}})\} \text{ s.t. } \mathbf{R}_{1:K-1}(\pi) \geq \mathbf{c}. \quad (\text{COP-R})$$

Given this equivalence, we now describe the procedure of setting appropriate constraint thresholds \mathbf{c} for the COP-R problem, following which constraint thresholds \mathbf{b} for the original COP can be obtained element-wise via $c_k := H_k^{-1}(b_k) + R_k(\mu)$. This setting of constraint thresholds ensures that the optimal solution of the COP problem (Problem (1)) is also a Pareto-optimal solution. We begin with a definition, following which we state the main result for c_k , which holds for all $k \in [K-1]$.

Definition A.3.2 (Insertion Index). Let $\mathbf{P}_k = (P_k(0), P_k(1), \dots, P_k(M-1))$ be an ascending (sorted) list of the k -th objective values from the Pareto front consisting of M points. For any new value $\alpha \in \mathbb{R}$, the insertion index j is the smallest integer $0 \leq j < M$ satisfying $P_k(j) \geq \alpha$ (if such a j exists), and set $j = M$ if no such index exists.

Proposition A.3.3. For the initial point π_0 of Problem (1), let the insertion index of $\alpha_k := R_k(\pi_0)$ in \mathbf{P}_k be j_k . If $c_k \geq P_k(\max(0, j_k - 1)) \forall k \in [K-1]$, then the optimal solution of Problem COP-R, if it exists, is a Pareto-optimal solution.

Proof. We prove by contradiction. First, define a solution element by the tuple (π, \mathbf{P}^π) , which refers to a policy π along with its corresponding reward vector $\mathbf{P}^\pi = (R_1(\pi), \dots, R_K(\pi))$. Now suppose that the optimal solution $P' = (\pi', \mathbf{P}^{\pi'})$ of Problem COP-R is not a Pareto-optimal solution. By the definition of Pareto-optimal solution, there exists a solution $\widehat{P} = (\widehat{\pi}, \mathbf{P}^{\widehat{\pi}})$ in Π_P that dominates P' , i.e., $R_k(\pi') \leq R_k(\widehat{\pi}) \forall k \in [K]$. Given $P_0 = (\pi_0, \mathbf{P}^{\pi_0})$, we have $R_K(\widehat{\pi}) \geq R_K(\pi') \geq R_K(\pi_0)$ by definition. Since both P_0 and \widehat{P} do not dominate each other, since $R_K(\widehat{\pi}) \geq R_K(\pi_0)$, there exists $k \in [K-1]$ such that $R_k(\pi_0) \geq R_k(\widehat{\pi})$.

Now consider the values of c_k and $R_k(\widehat{\pi})$ for some objective k . Note that $c_k \geq P_k(\max(0, j_k - 1))$. If $R_k(\pi_0) \geq R_k(\widehat{\pi}) > c_k$, then $R_k(\pi_0) \geq R_k(\widehat{\pi}) > P_k(\max(0, j_k - 1))$, which is conflicting with the condition that $P_k(\max(0, j_k - 1))$ is the $(\max(0, j_k - 1))^{th}$ objective value in \mathbf{P}_k . If $R_k(\widehat{\pi}) \leq c_k$, it conflicts with the condition that \widehat{P} dominates P' . Therefore, such a \widehat{P} does not exist, and hence, P' is a Pareto-optimal solution. \square

Proposition A.3.3 formalizes the criteria for specifying appropriate constraint values and provides the condition for which the optimal solution of Problem (1) is a Pareto optimal solution. See Figure 9 for the visualization. Proposition A.3.3 gives a sufficient condition under which the solution of Problem (2) is Pareto optimal. However, in practice this condition is (i) overly conservative and may exclude many feasible Pareto points, and (ii) computationally expensive as it requires re-evaluating all policies for non-dominated sorting at every optimization step. Please see Section 3.1 for an empirically validated practical constraint specification procedure.

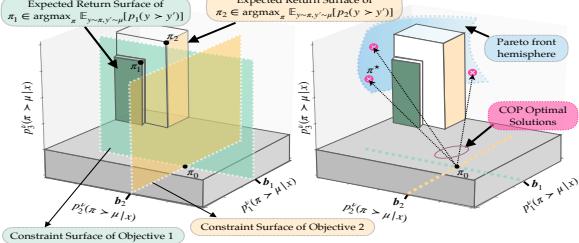


Figure 9: Visualization of criteria for specifying constraint values. The expected return surface of π_1 (π_2), call it S_1 (S_2), in objective 1(2) is the $\max(0, j_k - 1)^{th}$ value in list P_1 (P_2) respectively. Therefore, specifying constraints values $b_1 \geq S_1$ and $b_2 \geq S_2$ is sufficient for the optimal solution of Equation (1) to be a Pareto-optimal solution.

A.4 Main Text Proofs

Proposition A.4.1. The dual formulation of Problem (2) is given by,

$$\text{Dual}^* \triangleq \min_{\lambda \geq 0} \max_{\rho} \mathcal{L}(\rho, \lambda) = \min_{\lambda \geq 0} \mathcal{L}(\rho_\lambda^*, \lambda) = \min_{\lambda \geq 0} \mathcal{F}(\rho_\lambda^*) - \lambda^T(\mathbf{b} - \mathbf{G}(\rho_\lambda^*))$$

$$\text{where, } \rho_\lambda^*(y) = \exp \left(\tau^{-1} \mathbb{E}_{y' \sim \mu} [p(y \succ y') + \lambda^T \mathbf{q}(y \succ y')] - 1 \right) \quad \forall y \in \mathcal{Y}. \quad (3)$$

Proof. Given the Lagrangian $\mathcal{L}(\rho, \boldsymbol{\lambda})$, the dual formulation is given by

$$\max_{\rho} \min_{\boldsymbol{\lambda} \geq 0} \mathcal{L}(\rho, \boldsymbol{\lambda}) \equiv \min_{\boldsymbol{\lambda} \geq 0} \max_{\rho} \mathcal{L}(\rho, \boldsymbol{\lambda}).$$

By the strong duality, it is sufficient to consider KKT conditions for $(\rho^*, \boldsymbol{\lambda}^*)$.

- (i) Primal feasibility i.e. $\mathbb{E}_{y \sim \pi_{\text{ref}}, y' \sim \mu} [\rho^*(y) \mathbf{q}(y \succ y')] \geq \mathbf{b}$.
- (ii) Dual feasibility i.e. $\boldsymbol{\lambda}^* \geq \mathbf{0}$.
- (iii) Complementary slackness i.e. $(\boldsymbol{\lambda}^*)^T (\mathbf{b} - \mathbb{E}_{y \sim \pi_{\text{ref}}, y' \sim \mu} [\rho^*(y) \mathbf{q}(y \succ y')]) = \mathbf{0}$.
- (iv) Stationarity i.e. $\nabla_{\rho} \mathcal{L}(\rho, \boldsymbol{\lambda}) = 0$ i.e.

$$\begin{aligned} & \mathbb{E}_{y' \sim \mu} [p(y \succ y')] - \tau (\ln(\rho(y)) + 1) + \boldsymbol{\lambda}^T \mathbb{E}_{y' \sim \mu} [\mathbf{q}(y \succ y')] = 0. \\ & \Rightarrow \rho^*(y) = \exp \left(\frac{1}{\tau} \mathbb{E}_{y' \sim \mu} [p(y \succ y') + \boldsymbol{\lambda}^T \mathbf{q}(y \succ y')] - 1 \right) \end{aligned}$$

Now we show that conditions (i)-(iii) hold for the above $\rho^*(y)$. For condition (i), by initialization of \mathbf{b} using Proposition A.3.3, we have

$$\mathbb{E}_{\substack{y \sim \pi_{\text{ref}} \\ y' \sim \mu}} [\rho^*(y) \mathbf{q}(y \succ y')] \geq \mathbb{E}_{\substack{y \sim \pi_{\text{ref}} \\ y' \sim \mu}} [\rho_0(y) \mathbf{q}(y \succ y')] \geq \mathbf{b},$$

where $\rho_0(y) = \pi_0(y)/\pi_{\text{ref}}(y)$, π_0 is the initialization point of solving Problem (2), and the second inequality follows by construction. Condition (ii) also holds by construction of $\boldsymbol{\lambda}$. Now, condition (iii) holds by definition if the constraint is active i.e. $\mathbb{E}_{y \sim \pi_{\text{ref}}, y' \sim \mu} [\rho^*(y) \mathbf{q}(y \succ y')] = \mathbf{b}$, and if it is inactive, then dual feasibility also ensures that complementary slackness holds. As a consequence, all KKT conditions are always satisfied with the above $\rho^*(y)$, which concludes the proof. \square

Proposition A.4.2. *The optimal solution to Problem (4) can be obtained by solving the following optimization problem.*

$$\begin{aligned} \chi_k^* &= \operatorname{argmax}_{\chi_k \geq 0} \mathcal{L}_k(\chi_k; \rho) := -\chi_k \ln \left(\mathbb{E}_{y \sim \pi_{\text{ref}}, y' \sim \mu} [\exp(\chi_k^{-1} \rho(y) \mathbf{q}_k(y \succ y'))] \right) - \chi_k \epsilon \\ &\text{with, } \pi_k^*(y) \propto \pi_{\text{ref}}(y) \underbrace{\exp \left((\chi_k^*)^{-1} \mathbb{E}_{y' \sim \mu} [\rho(y) \mathbf{q}_k(y \succ y')] \right)}_{w(y) \text{ (unnormalized weight)}}. \end{aligned} \quad (5)$$

Proof. For the given constrained optimization problem:

$$\min_{\pi_k} \mathbb{E}_{\substack{y \sim \pi_k \\ y' \sim \mu}} [\rho(y) \mathbf{q}_k(y \succ y')] \quad \text{s.t.} \quad \text{KL}(\pi_k || \pi_{\text{ref}}) \leq \epsilon \quad \text{and,} \quad \sum_{y \in \mathcal{Y}} \pi_k(y) = 1,$$

we consider its Lagrangian to find its dual problem. By noticing that, $\pi_k(y)\rho(y) = \pi_k(y) \frac{\pi(y)}{\pi_{\text{ref}}(y)} \approx \pi(y)$, the Lagrangian for some arbitrary multipliers $\chi_k \geq 0$ and $\zeta_k \in \mathbb{R}$ is given by,

$$\begin{aligned} \mathcal{L}(\pi_k, \chi_k, \zeta_k) &= \mathbb{E}_{\substack{y \sim \pi_k \\ y' \sim \mu}} [\rho(y) \mathbf{q}_k(y \succ y')] + \chi_k \left(\mathbb{E}_{y \sim \pi_k} \left[\ln \left(\frac{\pi_k(y)}{\pi_{\text{ref}}(y)} \right) \right] - \epsilon \right) + \zeta_k \left(\sum_{y \in \mathcal{Y}} \pi_k(y) - 1 \right) \\ &= \mathbb{E}_{\substack{y \sim \pi_k \\ y' \sim \mu}} [\rho(y) \mathbf{q}_k(y \succ y')] + \chi_k \left(\sum_{y \in \mathcal{Y}} \pi_k(y) \ln \left(\frac{\pi_k(y)}{\pi_{\text{ref}}(y)} \right) - \epsilon \right) + \zeta_k \left(\sum_{y \in \mathcal{Y}} \pi_k(y) - 1 \right), \end{aligned}$$

where $\chi_k \in \mathbb{R}_+$ is the Lagrange multiplier for KL constraint, and $\zeta_k \in \mathbb{R}$ is the Lagrange multiplier for the normalization constraint that ensures $\sum_{y \in \mathcal{Y}} \pi_k(y) = 1$. Hence, the corresponding optimization problem due to strong duality is: $\min_{\pi_k} \max_{\chi_k \geq 0, \zeta_k} \mathcal{L}(\pi_k, \chi_k, \zeta_k) \equiv \max_{\chi_k \geq 0, \zeta_k} \min_{\pi_k} \mathcal{L}(\pi_k, \chi_k, \zeta_k)$. Now, we can compute the non-parametric closed form solution

for each sample y for the inner minimization problem. Due to the convexity of KL-divergence, it is sufficient to consider $\nabla_{\pi_k} \mathcal{L}(\pi_k, \chi_k, \zeta_k) = 0$. For each y we then have,

$$\begin{aligned}\nabla_{\pi_k} \mathcal{L}(\pi_k, \chi_k, \zeta_k) &= \mathbb{E}_{y' \sim \mu} [\rho(y) \mathbf{q}_k(y \succ y')] + \chi_k^* \left(\ln \left(\frac{\pi_k^*(y)}{\pi_{\text{ref}}(y)} \right) + 1 \right) + \zeta_k = 0 \\ &\Rightarrow \pi_k^*(y) \propto \pi_{\text{ref}}(y) \exp \left((\chi_k^*)^{-1} \mathbb{E}_{y' \sim \mu} [\rho(y) \mathbf{q}_k(y \succ y')] \right)\end{aligned}$$

with some normalization constant Z_k that ensures that $\sum_{y \in \mathcal{Y}} \pi_k^*(y) = 1$, which is described with respect to ζ_k . Then, by plugging the above stationarity condition into the Lagrangian, we have the dual unconstrained optimization problem as,

$$\begin{aligned}\max_{\chi_k \geq 0, \zeta_k} \mathcal{L}(\pi_k^*, \chi_k, \zeta_k) &= \mathbb{E}_{y \sim \pi_k^*, y' \sim \mu} [\rho(y) \mathbf{q}_k(y \succ y')] + \chi_k \left(\mathbb{E}_{y \sim \pi_k^*} \left[\ln \left(\frac{\pi_k^*(y)}{\pi_{\text{ref}}(y)} \right) \right] - \epsilon \right) + \zeta_k \left(\sum_{y \in \mathcal{Y}} \pi_k^*(y) - 1 \right) \\ &= \mathbb{E}_{y \sim \pi_k^*} \left[\mathbb{E}_{y' \sim \mu} [\rho(y) \mathbf{q}_k(y \succ y')] + \chi_k \ln \left(\frac{\pi_k^*(y)}{\pi_{\text{ref}}(y)} \right) \right] - \chi_k \epsilon \\ &= -\chi_k \ln \left(\mathbb{E}_{y \sim \pi_{\text{ref}}, y' \sim \mu} [\exp(\chi_k^{-1} \rho(y) \mathbf{q}_k(y \succ y'))] \right) - \chi_k \epsilon.\end{aligned}$$

This concludes the proof. \square

A.5 Empirical Optimization Problem

In this section, in order to obtain the empirical version of **MOPPO**, we need to show that we can build an unbiased estimate of the optimization function from empirical observations. To this end, consider the sampled **COP** as:

$$\max_{\rho} \mathbb{E}_{y \sim \pi_{\text{ref}}, y' \sim \mu} [\rho(y) I_p(y, y')] - \tau \mathbb{E}_{y \sim \pi_{\text{ref}}} [\rho(y) \ln(\rho(y))] \quad \text{s.t.} \quad \mathbb{E}_{y \sim \pi_{\text{ref}}, y' \sim \mu} [\rho(y) \mathbf{I}_q(y, y')] \geq \mathbf{b} \quad (26)$$

, where $\mathbf{I}(\cdot, \cdot)$ is a Bernoulli random preference vector over K objectives i.e. $\mathbf{I}_k(y, y')$ is a random variable sampled from a Bernoulli distribution with mean $p_k(y \succ y')$ such that it is 1 if $y \succ_k y'$, and 0 otherwise, where \succ_k denotes preference with respect to the k^{th} objective for $k \in [K]$. Following the notation discussed before, we let $I_p(\cdot, \cdot)$ to be the preference with respect to the K^{th} objective, and let $\mathbf{I}_q(\cdot, \cdot)$ to be the preference vector for the remaining $(K-1)$ objectives i.e. $I_p(y, y') = \mathbf{I}_K(y, y')$ and $(\mathbf{I}_q)_k = I_k(y, y')$ for $k \in [K-1]$. Note that Problem (2) and Problem (26) are equivalent since $\mathbb{E}_{y \sim \pi_{\text{ref}}, y' \sim \mu} [\rho(y) I_p(y, y')] = \mathbb{E}_{y \sim \pi_{\text{ref}}, y' \sim \mu} [\rho(y) I_p(y, y') | y, y'] = \mathbb{E}_{y \sim \pi_{\text{ref}}, y' \sim \mu} [\rho(y) p(y \succ y')]$. Similar argument follows for \mathbf{I}_q .

A.6 Implementation Details

A.6.1 Background

SFT. Supervised fine-tuning (SFT) with labeled demonstrations is widely adopted to fine-tune LLMs [56, 37]. Given prompt-response pairs $\{(x, y)\}$ sampled from the dataset \mathcal{D} , the SFT loss function is defined as:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(x, y) \sim \mathcal{D}} \left[\sum_i \log \pi_{\text{sft}}(y_i | x, y_{<i}) \right], \quad (27)$$

where π_{sft} refers to the LLM policy and $y_{<i}$ indicates all tokens before the i -th token in response y .

RLHF. RLHF typically involves two steps [35, 53]: reward modeling, and RL training. In reward modeling, a reward model r_ϕ is trained to minimize the loss function $\mathcal{L}_{\text{RM}}(\phi) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} [\log(\sigma(r_\phi(x, y_w)) - r_\phi(x, y_l))]$, where $\sigma(z)$ is the sigmoid function, y_w and y_l refer to preferred and dispreferred responses, respectively. Generally, RL training uses the PPO algorithm [41] with an additional KL penalty relative to the SFT policy:

$$\arg \max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(y|x)} \left[r_\phi(x, y) - \tau \log \frac{\pi_\theta(y|x)}{\pi_{\text{sft}}(y|x)} \right],$$

where $\tau > 0$ is the KL penalty coefficient.

A.6.2 Training details.

We summarize the key implementation details of text generation tasks in Table 2. This table also provides links to the open-sourced datasets and reward models utilized in our study. Implementation is primarily based on trl [47] and the phi-1.5 base model [27]. Especially, SFT fine-tunes the base model, while MORLHF and Rewarded Soups fine-tune the SFT model using the PPO algorithm. In contrast, RiC directly fine-tunes the base model. See [54] for more details. We apply the same 4-bit quantization and LoRA configuration for training all models. During evaluation, we maintain a consistent configuration across different models, generating 64 tokens for the Helpful Assistant task and 32 for the Reddit Summary task.

In **MOP0**, we begin by normalizing the rewards using the mean and standard of the offline dataset before incorporating them into the prompts. During online generation (updating the reference policy) and evaluation, we sample a group of 25,000 random samples from a normal distribution and use the maximum and minimum values (generally around ± 3) of these samples to replace the maximum and minimum values of the dataset. This method can prevent the extreme values in the dataset to impact reward selection.

Incorporating preference vectors. Pareto fronts are generated as in [54]. One point to consider while evaluating empirical Pareto fronts is to incorporate user preferences for a particular objective. For instance, in the case of 2 objectives, in RiC [54], scalarization tuples are passed as in-context human preferences. Preference tuple $w = (w_1, w_2)$ for the two reward dimensions is passed to the model at inference time to adjust the LLM policy according to the user preferences. It is necessary to map these scalarization tuples w to the desired rewards that will be used as conditioning in prompts. Similarly, for the two objective case, HaM [32] uses prompts at inference time and clusters them. Then, they take two diverse policies, apply each policy to each cluster, and compute a multi-objective reward vector for all policy-cluster pairs. A single point in the plots is a reward vector for the policy-cluster pair. For a fair comparison, **MOP0** should also incorporate the preference vectors at inference time. For K objectives, one idea would be to solve K separate optimization problems as in Problem (2), one for each objective being the "main" objective. However, this is not efficient since if the LLM had m parameters, the space complexity of representing K policies would be Km .

We propose a multi-policy architecture in which all policies share a common LLM backbone. Each policy k is parameterized by a matrix $\theta_k \in \mathbb{R}^{L \times d}$, where L is the number of tokens and d is the transformer embedding dimension. The logit for the next token under policy k is computed as $\theta_k \phi$, where $\phi \in \mathbb{R}^d$ is the final-layer embedding summarizing the input sequence. This multi-headed model is illustrated in Figure 10. The total parameter count is reduced to $m - dL + dKL$, where $m - dL$ parameters are in the shared backbone, and dKL correspond to the K policy heads. The shared backbone enables efficient language modeling, while the separate heads provide sufficient flexibility for each policy to optimize distinct objectives and language styles. During training, each policy matrix is jointly trained with the transformer backbone.

Inference code. Here we provide the inference pseudo-code. Notably, to prevent potential precision explosion, we approximate the solution for JSD same as Reverse KL-divergence, as they are inherently similar.

```

def f_divergence(logp, weights, f_type):
    if f_type in ("reverse_kld", "jsd"):
        return torch.stack([w * lp for w, lp in zip(weights,
                                                      logp)]).sum(dim=0)

    if f_type == "forward_kld":
        alpha = 1.0

```

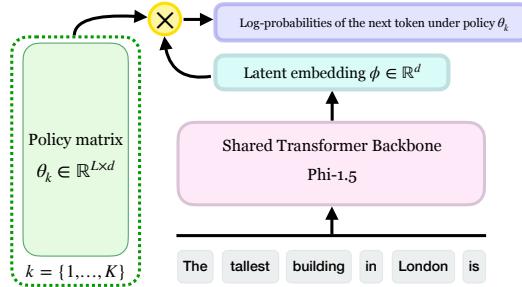


Figure 10: Multi-headed policy architecture for incorporating preferences at inference time in **MOP0**.

Table 2: Key implementations of the text generation experiments.

Basic information	
Architecture	Transformer
Pre-training	phi-1.5 [27]
Hardware	NVIDIA GeForce RTX 5080, GPU 16GB, Memory DDR5 64 GB
Quantization for training	4bit
Fine-tuning strategy	LoRA [18]
LoRA r	16
LoRA alpha	32
LoRA dropout	0.05
Optimizer	Adam
Batch size	8
Inference tokens for evaluation	64 for Helpful Assistant and 32 for Reddit Summary
SFT	
Finetuning steps	10000
Initial learning rate	1.41e-4
Learning rate scheduler	Linear
MOPPO	
Finetuning steps	10000
Initial learning rate	1.87e-4
Learning rate scheduler	Linear
Batch size	8
Regularization τ	0.08
Constraint lower bound ball ϵ	0.15
Constraint relaxation β	0.9995
Reference policy lag t_0	500
RiC	
Offline finetuning steps	10000
Initial learning rate	1.41e-4 for offline finetuning, 1e-5 for online finetuning
Learning rate scheduler	Linear for offline finetuning, constant for online finetuning
Threshold for MORS	0.7-quantile for each reward dimension
Online generation sample size per iteration	10000
Online finetuning steps per iteration	4000
RL step for MORLHF and Rewarded Soups [39]	
RL algorithm	PPO [41]
Implementation	trl [47]
KL regularization	0.2
Epochs	1
learning rate	1e-5
lambda for GAE	0.95
gamma	1
cliprange	0.2
Number of optimisation epochs per batch	4
Target KL	3
Datasets and Reward Models	
Helpful Assistant	
Task name	Provide helpful and harmless answers to potentially complex and sensitive questions.
Description	No prompt, only users' questions.
Prompt	Anthropic/hh-rlhf [5]
Dataset	gpt2-large-harmless-reward_model gpt2-large-helpful-reward_model humor-no-humor
Reddit Summary	
Task name	Provide a summary to a post from Reddit.
Description	Generate a one-sentence summary of this post.
Prompt	openai/summarize_from_feedback [43]
Dataset	gpt2_reward_summarization bart-summary-detector bart-faithful-summary-detector
pref1 reward	
less-hallucination reward	
faithful reward	

```

    elif "-divergence" in f_type:
        alpha = float(f_type.split("_", 1)[0])
    else:
        raise ValueError(f"Unknown f_type: {f_type}")

    terms = [
        -alpha * lp + np.log(w)
        for w, lp in zip(weights, logp)
        if w != 0
    ]
    return -torch.logsumexp(torch.stack(terms), dim=0)

```

Compute resources. For training RLHF, MORLHF models, the number of workers are set as 3, each taking up 7,000M of memory, running for 5 hours; for training DPO, MODPO models, the

number of workers are set as 2, each taking up 11,000M of memory, running for 3 hours. For **MOPO**, see Table 2, which takes 4 hours.

Training hyper-parameters. For PPO, we follow the settings of [54] and train for 25 batches; for DPO, we follow [59], with PERDEVICE_BATCH_SIZE= 1 and MAX_LENGTH= 64.

Inference hyper-parameters. For PPO, we follow the settings of [54] with NUM_BEAMS= 1; for DPO, we follow [59] with BATCH_SIZE= 4, MAX_LENGTH= 50 and NUM_BEAMS= 1.

Codebase. Our codebase is mainly based on trl [46] (<https://github.com/huggingface/trl>), MODPO (<https://github.com/ZHZisZZ/modpo>), RiC (<https://github.com/YangRui2015/RiC>) and Finegrained RLHF (<https://github.com/allenai/FineGrainedRLHF>), and has referred to f-divergence DPO [48] (<https://github.com/alecwangcq/f-divergence-dpo>), PackLLM [29] (<https://github.com/cmavro/PackLLM>), and DPA [50] (<https://github.com/Haoxiang-Wang/directional-preference-alignment>).

A.7 Example Outputs

Example generations for each dataset are shown in Tables 3 and 4. For each dataset, we show a representative prompt in the down-sampled dataset, and one generated response for each model/algorithm, with preference weightings set as $w = (0.5, 0.5)$ for MODPO, HaM, and RiC.

Table 3: Examples of **Reddit Summary**.

Prompt:	Instruction: Rewrite the email below in a friendly, concise tone in a single sentence. Input: Dear team, due to updated budget constraints, we must reduce remote work days from three to one per week effective next month. Please ensure on-site attendance Monday–Thursday. Thank you for your cooperation. Response:
LM response	
HaM:	Starting next month, please work onsite Monday–Thursday; budget changes limit us to one remote day weekly.
MODPO:	Budget limits mean we'll shift to one WFH day—please join us on-site Mon–Thu from next month; thanks for flexing with us!
RiC:	Heads-up: from next month we'll be in the office Mon–Thu (just one WFH day) because of budget constraints—appreciate everyone's flexibility!
MOPO:	Hey team! Budget tweaks mean we'll have one remote day going forward, so plan to be on-site Mon–Thu starting next month—thanks for rolling with the change!

Table 4: Examples of **Helpful Assistants**. Helpfulness & Humor.

Prompt: Assistant:	Human: What is two factor authentication, and how can it help me stay safe online?
LM response	
RLHF (Helpful):	Two factor authentication is a security system that requires you to provide two pieces of information to verify your identity. This can help protect you from fraud and identity theft.
RLHF (Humor):	What do you mean by two factor authentication?
HaM:	Two-factor authentication (2FA) adds a second proof—usually a code from your phone—on top of your password, so a thief who steals the password still can't get in.
MODPO:	Think of 2FA as a double lock: you enter your password, then confirm with a phone code or authenticator app. That quick extra step blocks most account-takeovers—using it on email first is a great start!
RiC:	It's a “prove-it-twice” check—password *plus* something you have (SMS code, hardware key). Turn it on for banking, email, socials and you slam the door on 99% of drive-by hacks.
MOPO:	Picture ordering pizza: the driver needs your address *and* a secret knock. 2FA works the same—password first, then a six-digit code or hardware tap. Even if crooks guess the password, the knock stays secret, so enable it everywhere (especially email and banking) for huge peace-of-mind gains.