IMO^3: Interactive Multi-Objective Off-Policy Optimization

Nan Wang[1] , **Hongning Wang**[1] , **Maryam Karimzadehgan**[2] ,
Branislav Kveton[3*] , **Craig Boutilier**[2]
[1]University of Virginia
[2]Google Research
[3]Amazon
{nw6a, hw5x}@virginia.edu, maryamk@google.com, bkveton@amazon.com, cboutilier@google.com

Abstract

Most real-world optimization problems have multiple objectives. A system designer needs to find a policy that trades off these objectives to reach a desired operating point. This problem has been studied extensively in the setting of known objective functions. However, we consider a more practical but challenging setting of unknown objective functions. In industry, optimization under this setting is mostly approached with online A/B testing, which is often costly and inefficient. As an alternative, we propose *Interactive Multi-Objective Off-policy Optimization* (IMO^3). The key idea of IMO^3 is to interact with a system designer using policies evaluated in an *off-policy fashion* to uncover which policy maximizes her unknown utility function. We theoretically show that IMO^3 identifies a near-optimal policy with high probability, depending on the amount of designer's feedback and training data for off-policy estimation. We demonstrate its effectiveness empirically on several multi-objective optimization problems.

1 Introduction

Most real-world optimization problems involve multiple objectives. *Multi-objective optimization (MOO)* has been studied and applied in various fields of system design, including engineering, economics, and logistics, where optimal policies need to trade off multiple, potentially conflicting objectives [Keeney and Raiffa, 1976]. The system designer aims to find the optimal policy that respects her design principles, preferences and trade-offs. For example, when designing an investment portfolio, one's investment strategy requires trading off maximizing expected gain with minimizing risk [Liang and Qu, 2013].

Two key issues need to be addressed in MOO before policy optimization. First, the objectives in most real world problems do not have explicit functional forms. Given a decision or *policy space*, we need a mapping of policies to the expected values of the objectives in question. These objective values may be obtained by executing new policies on live traffic, which is risky and time-consuming [Deaton and Cartwright, 2018; Kohavi *et al.*, 2009]. A more efficient approach could be learning a model from data, such as for the expected return and risk of an investment portfolio. In practice, acquiring data for learning a model can be costly and the model may be biased due to the data-gathering policy [Strehl *et al.*, 2010]. Correcting for such biases is the target of the literature on *off-policy* evaluation and optimization [Rosenbaum and Rubin, 1983; Strehl *et al.*, 2010; Dudik *et al.*, 2011]. When objective values can be obtained for a new policy, *bandit algorithms* can be used for optimization [Lattimore and Szepesvári, 2020; Wang *et al.*, 2021]. However, these generally require scalar rewards that already dictate a decision maker's desired trade-offs among the objectives.

This leads to the second issue—the specification of a *single objective function* that dictates the desired trade-offs. This can be viewed as the decision maker's *utility function*. In the example above, it might specify how much risk a decision maker can tolerate to attain some expected return. Assessing utility functions almost always requires interaction with the decision maker—requiring human judgements that typically cannot be learned from data in the usual sense [Keeney and Raiffa, 1976]. Moreover, *utility elicitation* is generally challenging and costly due to the cognitive difficulty faced by human decision makers when trying to assess trade-offs among objectives in a quantitatively precise fashion [Tversky and Kahneman, 1974; Camerer, 2004]. While some elicitation techniques attempt to identify the full utility function [Keeney and Raiffa, 1976], others try to minimize this burden in various ways. One common principle is to limit trade-off assessments to only those that are relevant given the *feasible or realizable* combinations of objectives w.r.t. the utility model and policy constraints [Boutilier, 2013].[1] This requires that the model is known.

We propose an interactive off-policy technique which supports a system designer to identify the optimal policy that trades off multiple objectives w.r.t. an *unknown* utility function [Branke *et al.*, 2008]. The utility function is modeled as a linear scalarization of the objectives considered [Keeney and Raiffa, 1976], where the scalarization parameters specify the trade-off among the objectives. We use off-policy estimators to evaluate policies in an unbiased way *without ever executing them*. To learn the desired scalarization, we present

[*]This work started prior to joining Amazon.

[1]Much work in MOO focuses on the identification of *Pareto optimal* solutions [Mas-Colell *et al.*, 1995]. The selection of a solution from this set still requires the decision maker to choose and thus make a trade-off among the objectives, possibly implicitly.

the off-policy estimates of the objective values of candidate policies to the designer for feedback. The candidate policies are chosen judiciously to maximize the information gain from the feedback. Over time, (i) the scalarization converges to the trade-offs embodied in the unknown utility function by learning from the designer's feedback; and (ii) the policy induced by the scalarization converges to the optimal policy. We analyze our approach and prove theoretical guarantees for finding near-optimal policies. Our comprehensive empirical evaluation on four multi-objective optimization problems shows the effectiveness of our method.[2]

2 Problem Formulation

We denote the set $\{1, \dots, n\}$ by $[n]$. Consider a policy optimization problem with $d \geq 1$ *(potentially conflicting) objectives*. Let $\mathcal{X}$ be a *context space* and $\mathcal{A}$ be an *action space* with K actions. In each round, $x \in \mathcal{X}$ is sampled from a context distribution P_x. An action $a \in \mathcal{A}$ is taken in response following a *stochastic policy* $\pi(\cdot \mid x)$, which is a distribution over $\mathcal{A}$ for any $x \in \mathcal{X}$. The policy space is $\Pi = \left\{\pi \,\middle|\, \pi(\cdot \mid x) \in \Delta_{K-1}, \forall x \in \mathcal{X}\right\}$, where Δ_K is the K-simplex with $K + 1$ vertices. After taking action a, the agent receives a d-dimensional reward vector $r \in [0, 1]^d$ sampled from a reward distribution $P_r(\cdot \mid x, a)$, corresponding to d objectives. The *expected value* of policy π is $V(\pi) = \mathbb{E}_{x\sim P_x, a\sim\pi(\cdot|x), r\sim P_r(\cdot|x,a)}\left[r\right]$. Note that $V(\pi)$ is a d-dimensional vector whose i-th entry $V_i(\pi)$ is the expected value of objective i under policy π.

We assume that there exists a *utility function* u_θ, parameterized by θ, which is used by the designer to assess the quality $u_\theta(v)$ of any objective-value vector $v \in \mathbb{R}^d$. Without loss of generality, we assume that u_θ is *absolutely monotonic* in each objective; but the correlations and conflicts among the objectives are unknown. We adopt the common assumption that u_θ is linear [Keeney and Raiffa, 1976] and determined by a *scalarization* $u_\theta(v) = \theta^\top v$ of the objective values, where $\theta \in \mathbb{R}^d$ determines the designer's trade-off among the objectives. We treat θ as *a priori* unknown and that it is not easy to specify directly by the designer. Hence, we learn it through the *interactions* with the designer.

The *optimal policy*, for any fixed designer's trade-off preferences $\theta_* \in \mathbb{R}^d$, is defined as

$$\pi_* = \arg\max_{\pi\in\Pi} u_{\theta_*}\big(V(\pi)\big) . \tag{1}$$

Since the interactions can be costly, we consider a fixed budget of T rounds of interactions with the designer. Our goal is to find a near-optimal policy with high probability after the interactions. Specifically, we use *simple regret* [Lattimore and Szepesvári, 2020] to measure the optimality of a policy π identified after T rounds of interactions, which is the difference in the utilities of π_* and π,

$$R_T^{sim} = u_{\theta_*}\big(V(\pi_*)\big) - u_{\theta_*}\big(V(\pi)\big) . \tag{2}$$

Note that we do not consider the optimality of policies presented during the interactions.

[2]An extended version including all appendices can be found at https://arxiv.org/abs/2201.09798.

3 General Algorithm Design

We first describe our approach in general terms, motivating it by the *de facto* standard approach to A/B testing in industry [Kohavi *et al.*, 2009]. In the standard "iterative" approach, a policy designer proposes a candidate policy π and evaluates it on live traffic for some time period (say, two weeks, to average out basic seasonal trends). If π outperforms a *production* policy (e.g., it improves some metrics/objectives and does not degrade others, or it achieves a desired trade-off among all objectives), π is *accepted* and deployed. If it does not, the designer proposes a new candidate policy and the process is repeated. This approach has three major shortcomings. First, each iteration takes a long time and many iterations may be needed to find a good policy. Second, it is difficult to propose good candidate policies, because the policy space is large and it is not a priori clear which objective trade-offs are feasible. Finally, due to the difficulty of managing changes in the control and treatment groups in large-scale platforms, online randomized experiments often lead to unexpected results [Kohavi *et al.*, 2009; Kohavi and Longbotham, 2011], which limit their efficiency and application in the fast-evolving industrial settings.

Consider an idealized scenario where $V(\pi)$ is known for any policy $\pi \in \Pi$. Then we can learn θ_* in (1) by interacting with the designer. A variety of preference elicitation techniques could be used [Keeney and Raiffa, 1976; Boutilier, 2013]. We study the following approach. In round (interaction) t, we (i) propose a policy π_t; (ii) present the value vector $V(\pi_t)$ to the designer; and (iii) obtain a *noisy response* based on the designer's true utility $u_{\theta_*}(V(\pi_t))$. The feedback can take different forms, but ultimately reflects the designer's perceived value for π_t. We assume a binary feedback of the form "Is policy π *acceptable*?", motivated by our industry example.

In this work, we consider a more realistic but also more challenging setting where $V(\pi)$ is unknown. In principle, any policy π can be evaluated on live traffic. However, online evaluation can be costly, inefficient, and time consuming; leading to unacceptable delays in finding π_* [Deaton and Cartwright, 2018; Kohavi *et al.*, 2009]. To address this issue, we evaluate π *offline* using logged data generated by some prior policy, such as the production policy [Swaminathan and Joachims, 2015]. In Section 4, we introduce three most common off-policy estimators for this purpose. The off-policy estimated value vector $\hat{V}(\pi)$ is then used in the elicitation process with the designer. Finally, we learn θ_* and π_* based on the estimated values and *noisy feedback* from interactions with the designer. We present our algorithm and analyze it in Section 5.

4 Multi-Objective Off-Policy Evaluation and Optimization

In this section, we discuss how to evaluate a policy π using logged data generated by another (say, production) policy, and optimize π w.r.t. any (fixed and known) scalarization parameters θ. We have a set of logged records $\mathcal{D} = \left\{(x_j, a_j, r_j)\right\}_{j=1}^N$ collected by a *logging policy* π_0. For the j-th record, x_j is the context, a_j is the action from π_0, and $r_j \in \mathbb{R}^d$ is the realized reward vector corresponding to d objectives. We also assume that the *propensity scores* $\pi_0(a_j \mid x_j)$ (i.e., the probability that

π_0 takes action a_j given context x_j) are logged. If not, they can be estimated from logged data [Strehl *et al.*, 2010].

4.1 Evaluation

Off-policy evaluation has been studied extensively in the single-objective setting [Strehl *et al.*, 2010; Dudik *et al.*, 2011]. Generally, better evaluation leads to better optimization [Strehl *et al.*, 2010]. By treating the reward as a d-dimensional vector rather than a scalar, we can adapt existing off-policy estimators to MOO. We adapt three popular estimators below.

The first estimator, the *direct method (DM)* [Lambert and Pregibon, 2007], estimates the expected reward vector $\mathbb{E}\left[r \mid x, a\right]$ by $\hat{r}(a, x) \in \mathbb{R}^d$, where $\hat{r}$ is some offline-learned reward model. The policy value is estimated by

$$\hat{V}^{\text{DM}}(\pi) = \frac{1}{N} \sum_{j=1}^{N} \sum_{a \in \mathcal{A}} \pi(a \mid x_j) \hat{r}(a, x_j) \,. \tag{3}$$

Since the model is learned without knowledge of π, it may focus on areas irrelevant for estimating $V(\pi)$, resulting in a biased estimate of $V(\pi)$.

The second estimator, *inverse propensity scoring (IPS)* [Rosenbaum and Rubin, 1983], is less prone to bias. Instead of estimating rewards, IPS uses the propensities of logged records to correct the shift between the logging and new policies,

$$\hat{V}^{\text{IPS}}(\pi) = \frac{1}{N} \sum_{j=1}^{N} \min\left\{M, \frac{\pi(a_j \mid x_j)}{\pi_0(a_j \mid x_j)}\right\} r_j \,, \tag{4}$$

where $M > 0$ is a hyper-parameter that trades off the bias and variance in the estimate. The IPS estimator is unbiased for $M = \infty$, but can have a high variance if π takes actions that are unlikely under π_0. When M is small, the variance is small but the bias can be high, since the IPS scores are clipped.

To alleviate the high variance of IPS, we can take advantage of both $\hat{r}$ and IPS to construct the *doubly robust (DR)* estimator [Dudik *et al.*, 2011]

$$\hat{V}^{\text{DR}}(\pi) = \frac{1}{N} \sum_{j=1}^{N} \frac{\pi(a_j \mid x_j)}{\pi_0(a_j \mid x_j)} (r_j - \hat{r}(a_j, x_j)) + \hat{V}^{\text{DM}}(\pi). \tag{5}$$

Intuitively, $\hat{r}$ is used as a baseline for the IPS estimator. If the model for reward estimation is unbiased or the propensities are correctly specified, DR can provide an unbiased estimate of the value. It has been shown that DR achieves lower variance than IPS [Dudik *et al.*, 2011].

4.2 Optimization

A key component in our approach is *policy optimization*, i.e., finding the optimal policy given a scalarization vector θ,

$$\hat{\pi} = \underset{\pi \in \Pi}{\arg\max}\; u_\theta\big(\hat{V}(\pi)\big) = \underset{\pi \in \Pi}{\arg\max}\; \theta^\top \hat{V}(\pi) \,, \tag{6}$$

where $\hat{V}(\pi)$ is some off-policy estimator. The optimized variables are the entries of $\pi \in \Pi$ that represent the probabilities of taking actions. In Appendix A, we prove that (6) can be formulated as a *linear program (LP)* for all off-policy estimators in Section 4.1 in the tabular case, where the policy is parameterized separately for each context. For non-tabular policies, we suggest using gradient-based policy optimization methods [Swaminathan and Joachims, 2015], though we provide no theoretical guarantees for this case.

Since (6) is an LP for all estimators, at least one solution to (6) is a vertex of the feasible set, corresponding to *non-dominated policies*, which cannot be written as a convex combination of other policies. For such policies, we can "learn" π_* by first learning θ_*.

5 IMO^3: Interactive Multi-Objective Off-Policy Optimization

Off-policy estimation and optimization in Section 4 assume that the utility parameters θ_* are known. Now we turn to *interactively* estimating θ_* by querying the designer for feedback on carefully selected policies over T rounds. Utility elicitation can be accomplished using a variety of query formats (e.g., value queries, bound queries, k-wise comparisons, critiques) and optimization criteria for selecting queries [Keeney and Raiffa, 1976; Boutilier, 2002; Boutilier, 2013].

5.1 Response Model

Following a common industrial practice (Section 3), we adopt a simple query format where we ask the designer to rate an objective value vector v corresponding to d objectives as "*acceptable*" or "*not acceptable*." We require a *response model* that relates this stochastic feedback to the designer's underlying utility for v. We adopt a *logistic response* model

$$\ell_{\theta_*}(v) = 1/(1 + \exp(-u_{\theta_*}(v))) \,, \tag{7}$$

where $u_{\theta_*}(v) = \theta_*^\top v$, and the designer responds "*acceptable*" with probability $\ell_{\theta_*}(v)$ and "*not acceptable*" otherwise. Roughly speaking, this can be understood as a designer's noisy feedback relative to some implicit baseline (e.g., the value vector of the production policy). Logistic response of this form arises frequently in modeling binary or k-wise discrete choice in econometrics, psychometrics, marketing, AI, and other fields [McFadden, 1974; Viappiani and Boutilier, 2010]; and lies at the heart of feedback mechanisms in much of the dueling bandits literature [Dudík *et al.*, 2015]. We defer the study of other types of feedback to future work.

5.2 Algorithm

Now we introduce IMO^3 for engaging the designer in solving the problem. We approach the problem as fixed-budget *best-arm identification (BAI)* [Karnin *et al.*, 2013], where we minimize the simple regret (2) in T rounds of interactions. At a high level, IMO^3 works as follows. In round $t \in [T]$, it selects a policy (arm) π_t and presents its off-policy estimated value vector $\hat{V}(\pi_t)$ to the designer. The designer responds with $Y_t \sim \text{Ber}\big(\ell_{\theta_*}(\hat{V}(\pi_t))\big)$, where $\text{Ber}(\mu)$ is a Bernoulli distribution with mean μ. After T rounds, IMO^3 computes the maximum likelihood estimate (MLE) $\hat{\theta}$ of θ_*, where $\hat{V}(\pi_t)$ serves as the feature vector for response Y_t. To incorporate other feedback and response models, only the MLE would change. Therefore, our algorithmic template is very general.

Algorithm 1 `IMO`^{3}

Input: Logging policy π_0, logged data $\mathcal{D}$, budget T, and pre-selection budget L

1: $\mathcal{W} \leftarrow \{\}$
2: **for** $i = 1, \ldots, L$ **do**
3: Sample θ_i from a unit ball in $\mathbb{R}^d$
4: $\pi_i \leftarrow \arg\max_{\pi \in \Pi} u_{\theta_i}\big(\hat{V}(\pi)\big)$
5: $\mathcal{W} \leftarrow \mathcal{W} + \{\pi_i\}$
6: $P_G(\mathcal{W}) \leftarrow$ G-optimal design over $\mathcal{W}$
7: **for** $t = 1, \ldots, T$ **do**
8: $\pi_t \sim P_G(\mathcal{W})$
9: Present $\hat{V}(\pi_t)$ to the designer and observe Y_t
10: $\hat{\theta} \leftarrow \mathrm{MLE}(\{\hat{V}(\pi_t), Y_t\}_{t=1}^T)$
11: Return $\tilde{\pi}_* \leftarrow \arg\max_{\pi \in \Pi} u_{\hat{\theta}}\big(\hat{V}(\pi)\big)$

To make `IMO`^{3} statistically efficient in identifying the optimal policy with limited budget, we must design a good distribution over policies to be presented to the designer. One challenge is that the policy space Π is continuous and infinite. To address this issue, we first discretize Π to a set $\mathcal{W}$ of L diverse policies, which are optimal under different random scalarizations. The other challenge is learning θ_* efficiently. We approach this as an optimal design problem [Wong, 1994]. Specifically, we use *G-optimality* to design a distribution over $\mathcal{W}$, from which we draw π_t in round t that minimizes the variance of the MLE $\hat{\theta}$. Since the design is variance minimizing, `IMO`^{3} chooses the final optimal policy $\tilde{\pi}_*$ solely based on the highest mean utility under $\hat{\theta}$. We experimented with more complex algorithm designs, where the distribution of π_t was adapted over time, analogous to sequential halving in BAI [Karnin *et al.*, 2013; Jamieson and Talwalkar, 2016]. However, none of these approaches improved `IMO`^{3} under small fixed budgets, and thus we focus on the non-adaptive algorithm.

We present `IMO`^{3} in Algorithm 1. In lines 1–5, the policy space Π is discretized into L policies $\mathcal{W}$. Any policy $\pi_i \in \mathcal{W}$ is optimal under its θ_i. Since θ_i are sampled uniformly from a unit ball, representing all scalarization directions, the policies π_i are diverse and allow us to learn about any θ_* efficiently. Note that we do not interact with the designer in this stage. In line 6, we compute the *G-optimal design* over $\mathcal{W}$, a distribution over $\mathcal{W}$ that minimizes the variance of the MLE $\hat{\theta}$ after T rounds. In lines 7–10, we interact with the designer over T rounds. In round $t \in [T]$, we draw π_t according to the G-optimal design, present its values $\hat{V}(\pi_t)$ to the designer, and receive feedback Y_t. In line 11, we compute the MLE $\hat{\theta}$ from all collected observations $\{\hat{V}(\pi_t), Y_t\}_{t=1}^T$. Finally, we use the estimated $\hat{\theta}$ to identify the optimal policy $\tilde{\pi}_*$ w.r.t. off-policy estimated values using an LP (Section 4.2).

5.3 Regret Analysis

We now analyze the simple regret of `IMO`^{3}, which is defined in (2). Due to space constraints, we focus on the IPS estimator and then discuss extensions to other estimators.

To state our regret bound, we first introduce some notation. Let $\mathcal{W} = \{\pi_i\}_{i=1}^L$ be the set of pre-selected policies in `IMO`^{3} and $\mathcal{V} = \{v_i\}_{i=1}^L$ be their estimated values, with $v_i = \hat{V}(\pi_i)$. Let $\hat{\pi}_* = \arg\max_{\pi \in \Pi} u_{\theta_*}\big(\hat{V}(\pi)\big)$ be the optimal policy under θ_* in logged data. Let $\hat{\pi}_* \in \mathcal{W}$ and $\pi_1 = \hat{\pi}_*$ without loss of generality. Let $\mu_i = v_i^\top \theta_* \in [0, 1]$ be the utility of policy π_i and $\Delta_i = \mu_1 - \mu_i$ be its gap. Let $\Delta_{\min} = \min_{i>1} \Delta_i$ be the minimum gap. Let $\alpha_* = \arg\min_{\alpha \in \Delta_{L-1}} g(\alpha)$ be the *G-optimal design* on $\mathcal{V}$, where $g(\alpha) = \max_{i \in [L]} v_i^\top G_\alpha^{-1} v_i$ and $G_\alpha = \sum_{i=1}^L \alpha_i v_i v_i^\top$. Let $h(\cdot)$ be the sigmoid function and $h'(\cdot)$ be its derivative.

Theorem 1. *Let $c_{\min}, \delta_1 > 0$ be chosen such that*

$$\min_{v \in \mathcal{V}} \min\{h'(v^\top \theta_*), h'(v^\top \hat{\theta})\} \geq c_{\min}$$

holds with probability at least $1 - \delta_1$. Then $R_T^{sim} \leq$

$$L \exp\left[-\frac{\Delta_{\min}^2 c_{\min}^2 T}{2g(\alpha_*)}\right] + 2||\theta_*||_2 \sqrt{\frac{dM^2 \log(2d/\delta_2)}{2N}} \quad (8)$$

holds with probability at least $1 - (\delta_1 + 2\delta_2)$, where d is the number of objectives, M is the tunable parameter in the IPS estimator, and N is the size of logged data.

The proof of Theorem 1 is in Appendix B. The regret bound decomposes into two terms. The first term is the regret of BAI w.r.t. *estimated policy values* and decreases with the amount of designer's feedback T. The second term reflects the error of the IPS estimator and decreases with data size N.

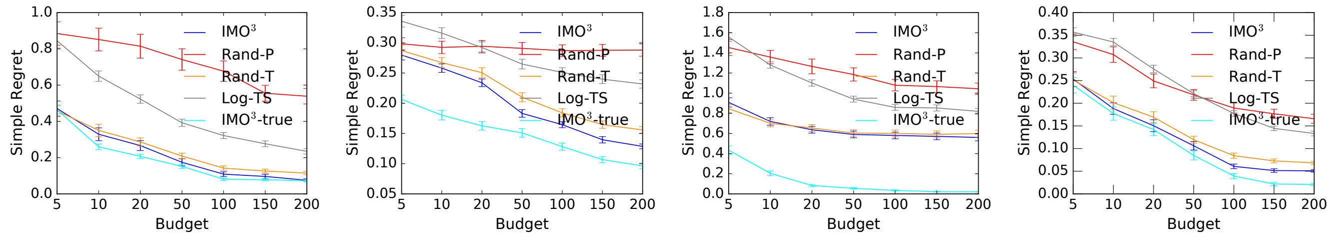
Specifically, the first term in (8) is $O(L \exp[-T])$. While it increases with the number of pre-selected policies L, it decreases exponentially with budget T. Therefore, even relatively small budget sizes of $T = O(\log L)$ lead to low simple regret. This is important, as large L may be needed to guarantee that the optimal policy under θ_* in logged data satisfies $\hat{\pi}_* \in \mathcal{W}$, a condition in our theorem. Regarding the other terms, $\Delta_{\min}^2 c_{\min}^2$ is a problem-specific constant and we minimize $g(\alpha_*)$ by design.

The second term in (8) decreases with data size N at an expected rate of $O(\sqrt{1/N})$. Now we discuss the errors for other estimators. For the DM estimator, this error depends on the quality of the reward model and cannot be directly analyzed. It could be large when the reward model is biased. For the DR estimator, it is unbiased if the reward model is unbiased or the propensity scores are correctly specified. If the reward model is unbiased, the error can be bounded the same as in the IPS estimator. Otherwise the error cannot be directly analyzed due to the bias of the reward model.

6 Experiments

In this section, we evaluate `IMO`^{3} on four MOO problems. We introduce the problems for evaluation in Section 6.1, describe several baseline methods in Section 6.2, and evaluate `IMO`^{3} vs. baselines from different perspectives in Section 6.3.

Due to space limit, we put the details of how to generate logged data for each problem in Appendix C. To simulate designer feedback, we sample the ground-truth scalarization $\theta_* \in \mathbb{R}^d$ from the unit ball, and sample responses from $\mathrm{Ber}\big(\ell(\hat{V}(\pi); \theta_*)\big)$, where $\hat{V}(\pi)$ is the off-policy estimated value vector presented to the designer. We generate feedback in the same way in all four problems.



(a) ZDT1. (b) Crashworthiness. (c) Stock investment. (d) Yahoo! news recommendation.

Figure 1: Simple regret of different algorithms for a fixed logged data size $N = 20,000$ and varying interaction budget. Each experiment is averaged over 10 logged data, 10 randomly selected θ_* and 5 runs under each combination of logged data and θ_*.

6.1 Multi-Objective Optimization Problems

ZDT1. The ZDT test suite [Zitzler *et al.*, 2000] is the most widely employed benchmark for MOO. We use ZDT1, the first problem in the test suite, a box-constrained n-dimensional two-objective problem, with objectives F_1 and F_2 defined as

$$F_1(x) = 5x_1, \quad F_2(x) = g(x)\left[1 - \sqrt{\frac{x_1}{g(x)}}\right], \tag{9}$$

$$\text{and } \; g(x) = 1 + \frac{9(\sum_{i=2}^{n} x_i)}{n-1},$$

where $x = (x_i)_{i=1}^n$ are variables and $x_i \in [0, 1], \forall i \in [n]$. We use $n = 5$ in our experiments, treating (x_4, x_5) as context, and perform optimization on $(x_i)_{i=1}^3$. We sample five combinations of (x_4, x_5) uniformly to create context set $\mathcal{X}$ and ten combinations of $(x_i)_{i=1}^3$ to create the action set $\mathcal{A}$.

Crashworthiness. This MOO problem is extracted from a real-world crashworthiness domain [de Carvalho *et al.*, 2018], where three objectives factor into the optimization of the crash-safety level of a vehicle. We refer to Sec. 2.1 of [de Carvalho *et al.*, 2018] for detailed objective functions and constraints. Five bounded decision variables $(x_i)_{i=1}^5$ represent the thickness of reinforced members around the car front. We use different combinations of the last two variables as contexts and the first three as actions. The rest settings are the same as for ZDT1.

Stock Investment. The stock investment problem is a widely studied real-world MOO problem [Liang and Qu, 2013], where we need to trade off returns and volatility of an investment strategy. We consider investing one dollar in a stock at the end of each day as an action and try to optimize the *relative gain* and *volatility* of this investment at the end of the next day. Specifically, the relative gain is the stock's closing price on the second day minus that on the first day, and we use the *absolute* difference as a measure of investment volatility. Our goal is to maximize the relative gain and minimize the volatility between two consecutive days of a one-dollar investment, on average. We use 48 popular stocks (see Appendix C for the full list) as the action set $\mathcal{A}$, and the four quarters of a year as the context set $\mathcal{X}$. We collect the closing stock prices from Yahoo Finance for the period Nov.1/2020–Nov.1/2021 for generating logged data.

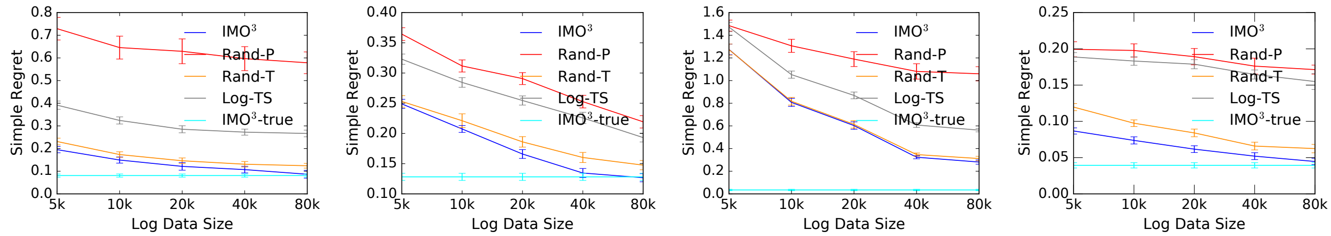
Yahoo! News Recommendation. This is a news article recommendation problem derived from the Yahoo! Today Module click log dataset (R6A). We consider two objectives to maximize, the *click through rate (CTR)* and *diversity* of the recommended articles. In the original dataset, each record contains the recommended article, the click event (0 or 1), the pool of candidate articles, and a 6-dimensional feature vector for each article in the pool. The recommended article is selected from the pool uniformly at random. We adopt the original click event in the logged dataset to measure CTR of the recommendation, and use the ℓ_2 distance between the recommended article's feature and the average feature vector in the pool to represent the diversity of this recommendation. For our experiments, we extract five different article pools as contexts and all logged records associated with them from the original data, resulting in 1,123,158 records in total. Each article pool has 20 candidates as actions.

6.2 Baselines

Random Policy (Rand-P). The *random policy* [Jamieson and Talwalkar, 2016] is a standard baseline in BAI, which selects a policy (arm) $\pi_t \in \Pi$ uniformly at random from the policy space in each round t. The off-policy value estimate $\hat{V}(\pi_t)$ is presented to the designer for feedback Y_t. After T rounds, the value estimates and their feedback are used to form the maximum likelihood estimate of θ_*, $\hat{\theta}$, which is used to solve (6) for the final identified policy.

Random Trade-off (Rand-T). Instead of sampling a random policy, Rand-T samples a trade-off vector θ_t uniformly at random from a d-dimensional unit ball, which is used to identify a policy π_t in each round by policy optimization in (6). The rest is the same as the Rand-P baseline.

Logistic Thompson Sampling (Log-TS). Many cumulative regret minimization algorithms with guarantees exist [Abeille and Lazaric, 2017; Kveton *et al.*, 2020]. We also consider a *cumulative-to-simple regret reduction* as a baseline. We adapt Thompson sampling (TS) for generalized linear bandits [Abeille and Lazaric, 2017; Kveton *et al.*, 2020] to the BAI problem, and output the "best" policy as the average of its selected policies. In each round t, we sample a trade-off vector θ_t from the current posterior over θ with Log-TS, which is used to identify a policy π_t in each round by policy optimization using (6). Then $\hat{V}(\pi_t)$ and feedback Y_t are used to update the posterior. The final output policy is the average of all policies selected in T rounds, $\tilde{\pi}_* = \sum_{t=1}^{T} \pi_t / T$. This reduction of Log-TS leads to a simple regret of $\hat{R}_T^{sim} = \tilde{O}(d^{\frac{3}{2}}\sqrt{T \log(1/\delta)})$, where $\tilde{O}$ stands for the big-O notation up to logarithmic factors in T. The proof is in Appendix C.



(a) ZDT1. (b) Crashworthiness. (c) Stock investment. (d) Yahoo! news recommendation.

Figure 2: Simple regret of different algorithms for a fixed interaction budget $T = 100$ and varying logged data size. Each experiment is averaged over 10 logged data, 10 randomly selected θ_* and 5 runs under each combination of logged data and θ_*.

$\texttt{IMO}^3$ with different value estimators. We fix the pre-selection budget $L = 500$, which requires no designer feedback. To assess the impact of off-policy estimated values on optimization performance, we test $\texttt{IMO}^3$ with its off-policy estimated values replaced by the true expected values (dubbed $\texttt{IMO}^3$-true). We use the IPS estimator by default. Experiments with the DM and DR estimators can be found in Appendix C.

6.3 Results and Analysis

For each of the problems, we first fix the size of the logged dataset and assess how simple regret (lower the better) varies with the interaction budget T. The results are shown in Figure 1. Each result is averaged over ten generated logged datasets, ten randomly sampled θ_*, and 5 repeated runs under each combination of logged data and θ_* (error bars represent standard error). We see that $\texttt{IMO}^3$ outperforms or performs comparably to our baselines in all four problems. While Rand-T is similar to the pre-selection phase of $\texttt{IMO}^3$ and performs relatively well, its exploration is less efficient and limited by the budget, and thus is worse than $\texttt{IMO}^3$. This illustrates the advantage of using G-optimal design with a sufficient number of pre-selected policies to query the designer for feedback. The gap between $\texttt{IMO}^3$ using estimated vs. true values is due to errors in value estimation—see the second term in our regret bound (Theorem 1). This term is invariant w.r.t. T, thus the gap remains relatively constant as T varies in our experiments.

We further study how the *amount* of logged data influences the simple regret of $\texttt{IMO}^3$. We fix $T = 100$, and vary the size of the logged dataset used for policy-value estimation. Intuitively, if the dataset is sufficient to provide an accurate value estimate for any policy, $\texttt{IMO}^3$ should perform similarly to directly using true values. Results in Figure 2 show that when the logged dataset is small, inaccurate value estimates cause algorithms that rely on off-policy estimates to perform poorly compared to using true values. As the size of the dataset increases, the decrease in value-estimation error allows $\texttt{IMO}^3$ to outperform the baselines by selecting the most effective policies for querying the designer. When the logged dataset is sufficiently large, more accurate value estimates ensure that $\texttt{IMO}^3$ converges to that of using true values.

7 Related Work

Drugan and Nowé [2013] is the first work to propose, analyze and experiment with a Pareto `UCB1` algorithm and a `UCB1` algorithm with a scalarized objective for MOO. Auer *et al.* [2016] formulate the problem of Pareto-frontier identification as a BAI problem. Thompson sampling in MOO is studied but not analyzed by Yahyaa and Manderick [2015]. Two recent works apply Gaussian process (GP) bandits to MOO. Paria *et al.* [2019] model the posterior of each objective function as a GP and minimize regret w.r.t. a known distribution of scalarization vectors. Zhang and Golovin [2020] show that this algorithm generates a set of points that maximize random hypervolume scalarization, an objective often used in practice. All above works are in the online setting, where the agent probes the environment to learn about its objective functions. Our setting is offline and the objective functions are estimated from logged data collected by some prior policy.

In terms of the motivation, the closest work to ours is that of Roijers *et al.* [2017], who treat online MOO as a two-stage problem, where the objective functions are estimated using initial interactions with the environment and the scalarization vector is then estimated via user interaction. Unlike our work, they do not propose a specific algorithm for their setting, but only adapt existing bandit algorithms based on learned utility functions. They also do not formulate their problem as off-policy optimization, and thus the process can be costly.

8 Conclusions

In this work, we study multi-objective optimization with unknown objective functions. We propose an interactive off-policy optimization algorithm for finding the optimal policy that achieves the desired trade-off among objectives. Specifically, we adapt off-policy estimators to evaluate policy values on all objectives, choose policies that effectively elicit designer's preferences, and learn the optimal policy using best arm identification. We prove upper bounds on the simple regret of our method and demonstrate its effectiveness with experiments on four MOO problems.

For future work, we plan to generalize (and analyze) our algorithm to more complex utility functions, other types of feedback and response models. We applied G-optimal design for BAI to provide theoretical guarantees—using other BAI algorithms for MOO is of interest.

Acknowledgments

This work is partially supported by the National Science Foundation under grant IIS-2128009, IIS-2007492 and IIS-1553568.

References

[Abeille and Lazaric, 2017] Marc Abeille and Alessandro Lazaric. Linear thompson sampling revisited. In *AISTATS*, 2017.

[Auer *et al.*, 2016] Peter Auer, Chao-Kai Chiang, Ronald Ortner, and Madalina M. Drugan. Pareto front identification from stochastic bandit feedback. In *AISTATS*, 2016.

[Boutilier, 2002] Craig Boutilier. A pomdp formulation of preference elicitation problems. In *Eighteenth National Conference on Artificial Intelligence*, 2002.

[Boutilier, 2013] Craig Boutilier. Computational decision support: Regret-based models for optimization and preference elicitation. In *Comparative Decision Making: Analysis and Support Across Disciplines and Applications*. 2013.

[Branke *et al.*, 2008] J. Branke, K. Deb, Kaisa Miettinen, and R. Slowinski. *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer-Verlag, 2008.

[Camerer, 2004] Colin F. Camerer. *Advances in Behavioral Economics*. 2004.

[de Carvalho *et al.*, 2018] Vinicius Renan de Carvalho, Jaime Simão, and Sichman. Solving real-world multi-objective engineering optimization problems with an election-based hyper-heuristic. 2018.

[Deaton and Cartwright, 2018] Angus Deaton and Nancy Cartwright. Understanding and misunderstanding randomized controlled trials. *Social Science & Medicine*, 2018.

[Drugan and Nowé, 2013] Madalina M. Drugan and Ann Nowé. Designing multi-objective multi-armed bandits algorithms: A study. In *IJCNN*, 2013.

[Dudik *et al.*, 2011] Miroslav Dudik, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. *ICML*, 2011.

[Dudík *et al.*, 2015] Miroslav Dudík, Katja Hofmann, Robert E. Schapire, Aleksandrs Slivkins, and Masrour Zoghi. Contextual dueling bandits. In *COLT*, 2015.

[Jamieson and Talwalkar, 2016] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *AISTATS*. PMLR, 2016.

[Karnin *et al.*, 2013] Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *ICML*, 2013.

[Keeney and Raiffa, 1976] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Wiley, 1976.

[Kohavi and Longbotham, 2011] Ron Kohavi and Roger Longbotham. Unexpected results in online controlled experiments. *SIGKDD Explor. Newsl.*, 2011.

[Kohavi *et al.*, 2009] Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M. Henne. Controlled experiments on the web: Survey and practical guide. *KDD*, 2009.

[Kveton *et al.*, 2020] Branislav Kveton, Manzil Zaheer, Csaba Szepesvari, Lihong Li, Mohammad Ghavamzadeh, and Craig Boutilier. Randomized exploration in generalized linear bandits. In *AISTATS*, 2020.

[Lambert and Pregibon, 2007] Diane Lambert and Daryl Pregibon. More bang for their bucks: assessing new features for online advertisers. *SIGKDD Explor.*, 2007.

[Lattimore and Szepesvári, 2020] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[Liang and Qu, 2013] J. J. Liang and B. Y. Qu. Large-scale portfolio optimization using multiobjective dynamic mutliswarm particle swarm optimizer. In *IEEE-SIS*, 2013.

[Mas-Colell *et al.*, 1995] Andreu Mas-Colell, Micheal D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[McFadden, 1974] Daniel McFadden. Conditional logit analysis of qualitative choice behavior. In *Frontiers in Econometrics*. 1974.

[Paria *et al.*, 2019] Biswajit Paria, Kirthevasan Kandasamy, and Barnabás Póczos. A flexible framework for multiobjective bayesian optimization using random scalarizations. In *UAI*, 2019.

[Roijers *et al.*, 2017] Diederik M. Roijers, Luisa M. Zintgraf, and Ann Nowé. Interactive thompson sampling for multiobjective multi-armed bandits. In *ADT*, 2017.

[Rosenbaum and Rubin, 1983] Paul R. Rosenbaum and Donald B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 1983.

[Strehl *et al.*, 2010] Alex Strehl, John Langford, Lihong Li, and Sham M Kakade. Learning from logged implicit exploration data. In *NeurIPS*, 2010.

[Swaminathan and Joachims, 2015] Adith Swaminathan and Thorsten Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. *ICML*, 2015.

[Tversky and Kahneman, 1974] Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 1974.

[Viappiani and Boutilier, 2010] Paolo Viappiani and Craig Boutilier. Optimal Bayesian recommendation sets and myopically optimal choice query sets. In *NeurIPS*, 2010.

[Wang *et al.*, 2021] Nan Wang, Branislav Kveton, and Maryam Karimzadehgan. Core: Capitalizing on rewards in bandit exploration. In *UAI*, 2021.

[Wong, 1994] Weng Kee Wong. Comparing robust properties of a, d, e and g-optimal designs. *Computational Statistics & Data Analysis*, 1994.

[Yahyaa and Manderick, 2015] Saba Q. Yahyaa and Bernard Manderick. Thompson sampling for multi-objective multiarmed bandits problem. In *ESANN*, 2015.

[Zhang and Golovin, 2020] Richard Zhang and Daniel Golovin. Random hypervolume scalarizations for provable multi-objective black box optimization. In *ICML*, 2020.

[Zitzler *et al.*, 2000] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 2000.